Electronic Colloquium on Computational Complexity, Report No. 59 (2012)

Uniform Circuits, Lower Bounds, and QBF Algorithms

Rahul Santhanam
University of Edinburgh

Ryan Williams
Stanford University

Abstract

We explore the relationships between circuit complexity, the complexity of generating circuits, and circuit-analysis algorithms. Our results can be roughly divided into three parts:

- **Lower Bounds Against Medium-Uniform Circuits.** Informally, a circuit class is "medium uniform" if it can be generated by an algorithmic process that is somewhat complex but not infeasible. We prove several unconditional lower bounds against medium uniform circuit classes, including
  - For every k, $\mathsf{P} \not\subseteq \mathsf{P}\text{-uniform}\mathsf{SIZE}(n^k)$. Namely, for any k, there is some language $L \in \mathsf{P}$ such that if size $O(n^k)$ circuits for L exist, they take super-polynomial time to generate.
  - For every k, $\mathsf{LOGSPACE}$ does not have $\mathsf{LOGSPACE}$-uniform branching programs of size n^k.
  - For every k, NP does not have $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform circuits of size n^k.
  - For every k, either P does not have non-uniform circuits of size n^k, or QBF (the language of true quantified Boolean formulae) does not have P-uniform branching programs of size $2^{n^{o(1)}}$.

  These lower bounds apply an indirect diagonalization argument which simulates a "medium uniform" class with a low-uniform class using small amount of non-uniformity.
- **Eliminating Non-Uniformity.** We complement these results by proving a "uniformization" lemma for NC^1, showing that *any* simulation of NC^1 in $\mathsf{ACC}^0/\text{poly}$ or $\mathsf{TC}^0/\text{poly}$ can be transformed into a uniform simulation using small advice. This lemma can be used to simplify part of the proof that faster SAT algorithms imply NEXP circuit lower bounds, and show that a nondeterministic $2^{n-\omega(\log n)}$-time algorithm for the following promise problem suffices for proving NEXP lower bounds against TC^0: *given a TC^0 circuit C of $n^{O(1)}$ size which is promised to be either unsatisfiable or have at least 2^{n-1} satisfying assignments, determine which is the case*.

  We also use this lemma to prove that if $\mathsf{NC}^1 \subset \mathsf{ACC}^0/\text{poly}$, then for all constants $k, c > 0$, the validity of quantified Boolean formulas (QBF) of size n^k on n variables can be decided in *deterministic* time $O(2^n/n^c)$.
- **The complexity of QBF.** Finally, we study the time complexity of QBF itself, and its application to lower bounds. As a partial converse to the above results, we show that if for each $k, c > 0$, the validity of quantified Boolean CNFs of size n^k with at most $k \log n$ alternations can be decided in time $2^n/n^c$, then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\text{poly}$.

  We also show that the exponential time complexities of quantified k-CNF and quantified (unrestricted) formulas are essentially identical. As a consequence, if quantified 3CNF formulas of n variables and $\text{poly}(n)$ size can be decided in $2^{n-n^{1/2+\varepsilon}}$ time deterministically (for some $\varepsilon > 0$) then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\text{poly}$. (Compare with the 3SAT problem, where 1.4^n-time deterministic algorithms are known.) This extends the recent connections of Williams between SAT algorithms and circuit lower bounds, to QBF algorithms.

ISSN 1433-8092

1 Introduction

In this paper, we explore the relationship between the complexity of circuits for solving problems, and the complexity of processes that can generate those circuits from scratch. In the non-uniform setting, we put no bounds on the complexity of generating circuits, and in this case it is extraordinarily difficult to prove lower bounds even against low-complexity circuits. In low-uniform[1] settings like DLOGTIME-uniformity, the circuits are extremely easy to construct – circuit complexity in this regime falls in line with standard machine-based complexity classes. The "medium-uniformity" settings, where the circuit-generating process is neither too easy nor too hard, are less well-understood, and studying them will help us better understand the two extremes.

Lower bounds by amplifying uniformity. In the first part of the paper, we prove a series of new lower bounds against "medium-uniform" circuits. The key insight behind the results is that under certain assumptions, medium-uniformity can sometimes be simulated with "low-uniformity" augmented with a small amount of advice. It is then possible to diagonalize against the low-uniformity small-advice simulation to derive a contradiction to the assumption.

The common theme of our results in the first part is that of proving lower bounds against notions of uniformity for which it is not possible to directly obtain lower bounds by diagonalization. Consider for example, the class of linear-size circuits. If a DLOGTIME-uniformity condition is imposed on the class, then it can be simulated in nearly-linear deterministic time, and we can easily diagonalize in P against it. However, for a notion of medium uniformity such as P-uniformity, it is not possible to directly diagonalize, as the notion of uniformity has more power than the diagonalizing language (which must lie in some fixed polynomial time bound). Our primary observation is that the uniformity condition can be made more succinct, and that this succinctness can then be used to give a low-uniform small-advice simulation.

Our first main result strengthens both the deterministic time hierarchy theorem and the classical result of Kannan [Kan82] that for any k, NP is not in P-uniformSIZE(n^k).

Theorem 1.1 *Let $k > 0$ be any constant.* $\mathsf{P} \not\subseteq \mathsf{P\text{-}uniformSIZE}(n^k)$

The idea in the proof of Theorem 1.1 can also be applied to smaller classes to get interesting separations. Note that the best non-uniform branching program lower bounds we know are $\Omega(n^2)$, using the Neciporuk method [Nec66]. However, if the branching programs are required to be LOGSPACE-uniform, we can prove:

Theorem 1.2 *For any k,* LOGSPACE *does not have* LOGSPACE*-uniform branching programs of size $O(n^k)$.*

We are also able to strengthen Kannan's lower bound for NP in a different direction, by *relaxing* the notion of uniformity used. To prove this result, we combine the uniformity trade-off idea used in the results above with ideas in an recent paper of Fortnow, Santhanam and Williams [FSW09] showing that fixed-polynomial size lower bounds can be "amplified".

Theorem 1.3 *Let $k \geq 1$ be any constant. Then* $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\mathsf{\text{-}uniformSIZE}(n^k)$.

It is still an open question whether QBF has P-uniform branching programs of polynomial size. We make progress on this question by showing that one of two alternatives holds: either P requires large non-uniform circuits, or QBF requires very large P-uniform branching programs.

[1] Note that somewhat counter-intuitively, the conventional meaning of "low-uniform" is *very uniform*

Theorem 1.4 *For each constant $k > 1$, P does not have non-uniform circuits of size n^k, or else QBF does not have* P-uniform *branching programs of size $2^{n^{o(1)}}$.*

Eliminating non-uniformity. The lower bounds above work by introducing a little non-uniformity into the process. In the second part of the paper, we study a situation where one can go in the opposite direction, and simulate non-uniform circuits by "medium-uniform" ones. We prove a lemma relating non-uniform circuits for NC^1 to subexponential-time uniform circuits for NC^1:

Lemma 1.1 *Suppose $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. For every $\varepsilon, k > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given any circuit C of size n and depth $k \log n$, prints an $O(k/\varepsilon)$-depth, $n^{O(k)}$-size $\mathcal{C}$-circuit that is equivalent to C.*

That is, a non-uniform inclusion of NC^1 in TC^0 implies *small-space uniform* (and hence subexponential-time uniform) TC^0 circuits for NC^1. This lemma simplifies and strengthens one of the main components of the proof that faster $\mathcal{C}$-SAT algorithms imply $\mathsf{NEXP} \not\subset \mathcal{C}/\text{poly}$ [Wil10, Wil11]. In particular, an intermediate result says that, if there are SAT algorithms running in $O(2^n/n^{10})$ time on all polynomial-size $\mathcal{C}$-circuits, and $\mathsf{NEXP} \subset \mathcal{C}/\text{poly}$, then there is a nondeterministic $o(2^n)$ time algorithm which generates $\mathcal{C}$-circuits equivalent to a given SUCCINCT 3SAT instance. By exploiting the structure of NC^1, Lemma 1.1 can be used to deterministically generate equivalent TC^0 (respectively, ACC) circuits in *subexponential* (2^{n^ε}) time, without assuming any algorithmic improvement on circuit satisfiability.

Moreover, Lemma 1.1 can be used to weaken the conditions necessary to prove lower bounds like $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$. In particular, we prove that a nondeterministic $2^{n-\omega(\log n)}$-time algorithm for the following promise problem suffices for proving $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$: *given a TC^0 circuit C of $n^{O(1)}$ size which is promised to be either unsatisfiable or have at least 2^{n-1} satisfying assignments, determine which is the case.*

Thus, while the *weakness* of medium uniformity was exploited in the first part of the paper to show lower bounds, here the *strength* of medium uniformity is exploited to show algorithmic results.

Another consequence of Lemma 1.1 is that, for simulations of NC^1 in smaller classes, non-uniformity can be simulated by low-uniformity with small advice:

Corollary 1.1 *For $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly} \iff$ for all $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^\varepsilon$.*

Another consequence of Lemma 1.1 is:

Theorem 1.5 *If $\mathsf{NC}^1 \subset \mathsf{ACC}/\text{poly}$ then for every $k, c > 0$, quantified Boolean formulas of n^k size and n variables can be solved in (deterministic) $O(2^n/n^c)$ time.*

Theorem 1.5 is significant because we obtain a faster *deterministic* QBF algorithm from a non-uniform assumption on NC^1 (previous arguments could only derive a nondeterministic algorithm for FORMULA SAT). With luck, Theorem 1.5 may be useful in a proof that $\mathsf{EXP} \not\subset \mathsf{ACC}/\text{poly}$. (To complete such a proof, we may also need a component which reduces every language in $\mathsf{TIME}[2^n]$ to quantified Boolean formulas of $n + O(\log n)$ variables and polynomial size, and runs in subexponential time. Assuming $\mathsf{EXP} \subset \mathsf{ACC}/\text{poly}$, it is plausible that such a reduction exists, since $\mathsf{EXP} = \mathsf{PSPACE} = \mathsf{MA}$ in this case.)

The complexity of QBF. Inspired by the second part, we further explore the prospects for proving non-uniform NC^1 lower bounds via faster QBF algorithms.

Define QB-kCNF, QB-CNF, QB-FORMULAS to be the quantified Boolean formula problem over k-CNF predicates, arbitrary CNF predicates, and formula predicates, respectively. In satisfiability problems, the choice of predicate can play a major role in the time complexity of the problem: for every k there is a $\delta < 1$ such that k-SAT is in $2^{\delta n}$ time, but no 1.999^n time algorithm is known for general CNF-SAT (the Strong Exponential Time Hypothesis posits that none exists). For FORMULA-SAT, there is no algorithm known to run in faster than $O(2^n/n)$ time for polynomial-size formulas. However, in the case of quantified Boolean formulas, the time complexities of QB-FORMULA and QB-kCNF are very tightly related:

Theorem 1.6 *If there an $\varepsilon > 0$ such that for all k, c,* QB-kCNF *of size n^c is in time 2^{n-n^ε}, then* QB-FORMULA *for n^c-size formulas is in time 2^{n-n^ε} (hence $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$ [Wil11]).*

Therefore, Quantified 3-CNFs are essentially *equivalent* in exponential time complexity to Quantified Boolean Formulas. This is very different from what we know for satisfiability. It is well-known that 3-SAT can be solved in less than $O(1.4^n)$ time [PPSZ98, DH09, Her11], and CNF-SAT on $\mathrm{poly}(n)$ clauses can be solved in $2^{n-n/O(\log n)}$ time [Sch05, CIP06]. However it is believed that CNF-SAT cannot be solved in $O(1.9^n)$ (this is implied by the Strong Exponential Time Hypothesis [IP01, CIP09]). Yet Formula SAT is not known to be in $O(2^n/n^{10})$ time (and if it were, then NEXP would not be in $\mathsf{NC}^1/\mathrm{poly}$ [Wil10, Wil11]). The proof of Theorem 1.6 yields the consequences:

Corollary 1.2 *If* QB-CNF *on n^k size instances with $k \log n$ alternations can be solved in $2^n/n^c$ time for every $c, k > 0$, then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$.*

Corollary 1.3 *If there is an $\varepsilon > 0$ such that* QB-3CNF *of size n^k can be solved in $2^{n-n^{1/2+\varepsilon}}$ time for all k, then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$.*

Therefore, even mild improvements on solving 3-CNF QBFs would imply NC^1 lower bounds.

1.1 Preliminaries and Notation

We assume basic knowledge of complexity theory [AB09]. For all complexity classes $\mathcal{C}$ defined as a class of polynomial-size circuits, we use $\mathcal{C}$ to denote the uniform version of the complexity class (with LOGTIME-uniformity being the default), and $\mathcal{C}/\mathrm{poly}$ to denote the non-uniform version. For instance, in this notation, we would say that prior work has established that $\mathsf{NEXP} \not\subset \mathsf{ACC}/\mathrm{poly}$ [Wil11]. This notation makes it clear when we are discussing uniform versus non-uniform classes.

Given a language L, L_n is the "slice" of L at length n, *i.e.*, $L_n = L \cap \{0,1\}^n$.

Given size function s and depth function d, $\mathsf{SIZE}(s)$ is the class of languages with circuits of size $O(s)$, $\mathsf{DEPTH}(d)$ the class of languages with circuits of depth d, and $\mathsf{SIZEDEPTH}(s,d)$ the class of languages which simultaneously have size $O(s)$ and depth d.

We need standard notions of uniformity for circuits. Given a class $\mathcal{C}$ of languages, a language L is said to have $\mathcal{C}$-uniform circuits of a certain size if there is a sequence of circuits of that size whose *direct connection language* is in $\mathcal{C}$. The direct connection language for a sequence of circuits $\{C_n\}$, where C_n is on n input bits, consists of all tuples of the form $\langle 1^n, g, h, r\rangle$, where g and h are indices of gates, r is the *type* of g (AND/OR/NOT/INPUT, and in case of INPUT, which of the n input bits g is, with an additional bit to specify whether g is the designated output gate), and h is a gate feeding in to g in case the type r is

not INPUT. Other encodings of the direct connection language are possible, but for the classes $\mathcal{C}$ we will consider, the encoding will not affect the result we prove.

By a description of a circuit C_n, we mean the adjacency list representation of the labelled DAG corresponding to C_n.

In one of our results, we also require the notion of a direct connection language for a branching program. This is defined in the same way as for a circuit, and at least for the notions of uniformity we use, the precise encoding will not matter.

For a uniform complexity class defined using machines or circuits, and given an advice length function a, we incorporate advice into the class in the standard way: the machines or circuits defining the class receive an additional advice input, which depends only on the input length n, and is of length at most $a(n)$.

2 Lower Bounds against Medium Uniformity

We will use a folklore result about a time hierarchy for deterministic time, where the lower bound holds against sublinear advice.

Proposition 1 *Let $d \geq 1$ and $d' > d$ be any constants. Then* $\mathsf{DTIME}(n^{d'}) \not\subseteq \mathsf{DTIME}(n^d)/o(n)$.

The following result simultaneously strengthens the time hierarchy theorem for deterministic time [HS65, HS66] and Kannan's result [Kan82] that for any fixed k, $\mathsf{NP} \not\subseteq \mathsf{P\text{-}uniformSIZE}(n^k)$.

Reminder of Theorem 1.1 *Let $k > 0$ be any constant.* $\mathsf{P} \not\subseteq \mathsf{P\text{-}uniformSIZE}(n^k)$

Proof of Theorem 1.1. Assume, to the contrary, that $\mathsf{P} \subseteq \mathsf{P\text{-}uniformSIZE}(n^k)$. Let $L \in \mathsf{P}$ be arbitrary. We will show that L can be simulated in a fixed deterministic time bound with small advice, which will yield a contradiction to Proposition 1.

Since $L \in \mathsf{P\text{-}uniformSIZE}(n^k)$ by assumption, there is a sequence of circuits $\{C_n\}$ for L, with each C_n having size at most cn^k for some constant c, such that the direct connection language L_{dc} of this sequence of circuits is in P. Now consider a "succinct" version L_{succ} of the language L_{dc} defined as follows. Let $Bin(n)$ be the binary representation of n. Then the tuple $\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle \in L_{succ}$ iff $\langle 1^n, g, h, r\rangle \in L_{dc}$.

We claim that $L_{succ} \in \mathsf{P}$. To decide L_{succ} in P, run the following procedure given an input y for L_{succ}. First, check if y can be parsed as a "valid" tuple $\langle z, g, h, r\rangle$, where $z = Bin(n)01^{\lceil n^{1/(3k)} \rceil}$ for some positive integer n, g and h are valid gate indices between 1 and cn^k, and r is a valid gate type. If this check fails, reject. Else simulate the polynomial-time machine deciding L_{dc} on $\langle 1^n, g, h, r\rangle$, accepting iff the machine accepts. The check on well-formedness can be done easily in polynomial time. The simulation of the polynomial-time machine for L_{dc} runs in time $\mathrm{poly}(n)$. Since the input length of y is at least $n^{1/3k}$, this is still polynomial time as a function of $|y|$. Clearly, the procedure accepts y iff L_{succ}.

Now, crucially, we use the assumption that $\mathsf{P} \subseteq \mathsf{P\text{-}uniformSIZE}(n^k)$ a second time. Since $L_{succ} \in \mathsf{P}$, there is a sequence $\{D_m\}$ of circuits for L_{succ} such that for each m, D_m has size $O(m^k)$. Given an integer n, let $m(n)$ be the least integer such the size of the tuple $\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle$ is at most $m(n)$ for any valid gate indices g and h for C_n and any valid gate type r. Using a standard encoding of tuples, we can assume, for large enough n, that $m(n) \leq n^{1/(2k)}$, since g, h, r can all be encoded with $O(\log n)$ bits each.

We now describe a simulation of L in time $O(n^{2k+2})$ with $o(n)$ bits of advice. Let M be an advice-taking machine which operates as follows. M receives an advice string of length $O(n^{1/2} \log n)$. It interprets this advice as a circuit D_m for the language L_{succ} on inputs of length $m = n^{1/(2k)}$. For each pair of gate indices g and h of C_n and each gate type r, M simulates the circuit D_m on $\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle$ to decide

whether gate h is an input to gate g and whether the type of gate g is r. Each such simulation can be done in time $O(n)$ since the circuit size is $O(n^{1/2})$. There are at most $O(n^{2k+1})$ such simulations that M performs, since there are at most that many triples $\langle g, h, r\rangle$, where g and h are gate indices of C_n and r is a gate type. Once all these simulations are performed, M can construct a description of the circuit C_n. It then simulates C_n on its input x to determine whether $x \in L$. This simulation can be done in time $O(n^{2k})$ since the circuit C_n is of size $O(n^k)$. The total time taken by M is $O(n^{2k+2})$, and M uses $O(n^{1/2} \log n)$ bits of advice. By our assumptions on C_n and D_m, the simulation is correct. Thus $L \in \mathsf{DTIME}(n^{2k+2})/O(n^{1/2} \log n)$.

Note that this is true for an *arbitrary* language $L \in \mathsf{P}$, hence we have that $\mathsf{P} \subseteq \mathsf{DTIME}(n^{2k+2})/O(n^{1/2})$. However, this is is contradiction to Proposition 1. □

A significant property of Theorem 1.1 is that the lower bound holds for a notion of uniformity which we cannot directly diagonalize against in polynomial time. Indeed, the following proposition shows that for each d, the class against which we show a lower bound contains a language that is not in $\mathsf{DTIME}(n^d)$.

Proposition 2 *For each $k \geq 1$ and $d > 0$, P-uniform$\mathsf{SIZE}(n^k) \not\subseteq \mathsf{DTIME}(n^d)$.*

Proof. The standard proof of the deterministic time hierarchy theorem [HS65, HS66] can be adapted to show that for each d, there is a unary language L which is in $\mathsf{DTIME}(n^{d+1})$ but not in $\mathsf{DTIME}(n^d)$. This unary language L can be recognized by P-uniform circuits of linear size – for each n, decide whether $1^n \in L$ in time $O(n^{d+1})$, outputting the trivial circuit which outputs the AND of its input bits if yes and the trivial circuit which outputs 0 on all inputs if no. □

The proof ideas of Theorem 1.1 can be adapted to prove lower bounds for other classes. We next show that there for each k, there are languages in NC which do not have NC-uniform formulas of size n^k, or indeed NC-uniform circuits of fixed polynomial size and fixed polylogarithmic depth. Note that the best-known formula size lower bound in NC against non-uniform formulas is $\Omega(n^{3-o(1)})$ [Has98].

We will require a hierarchy theorem for NC, which can again be shown using standard diagonalization.

Proposition 3 *For every k, $\mathsf{NC} \not\subseteq \mathsf{DLOGTIME}$-uniform$\mathsf{SIZEDEPTH}(n^k, (\log n)^k)/o(n)$.*

Theorem 2.1 *For any k, $\mathsf{NC} \not\subseteq \mathsf{NC}$-uniform$\mathsf{SIZEDEPTH}(n^k, (\log n)^k))$.*

Proof. Assume for a contradiction that there is a k such that $\mathsf{NC} \subseteq \mathsf{NC}$-uniform$\mathsf{SIZEDEPTH}(n^k, (\log n)^k)$. Let $L \in \mathsf{NC}$ be arbitrary. Let $\{C_n\}$ be a sequence of circuits of size $O(n^k)$ and depth $(\log n)^k$ solving L, and $L_{dc} \in \mathsf{NC}$ be the direct connection language of $\{C_n\}$. Define the succinct version L_{succ} of L_{dc} as in the proof of Theorem 1.1, with the same parameter $m(n)$. Observe that $L_{succ} \in \mathsf{NC}$ since checking whether a "succinct" tuple is valid, and then converting to a full tuple that can be offered as input to L_{dc} are both procedures that can be implemented in polylogarithmic depth. Hence L_{succ} has circuits of depth $O(m^k)$ and depth $O((\log m)^k)$, by assumption.

We now define $\mathsf{DLOGTIME}$-uniform$\mathsf{SIZEDEPTH}(n^{k'}, (\log n)^{k'})$ circuits taking $o(n)$ bits of advice which decide if $x \in L$, where k' is a fixed constant depending on k. The circuits interpret the advice as small-depth circuits for L_{succ} on inputs of length $m(n)$. The circuits simulate C_n on x implicitly, running the small-depth circuit for L_{succ} to retrieve any bit of C_n that is required. Since $m(n) \leq n^{1/(2k)}$, each run of the small-depth circuit for L_{succ} incurs a depth cost at most $O((\log n)^k)$ and size cost at most $n^{2/3}$. Simulating a circuit of depth $O((\log n)^k)$ and size $O(n^k)$ on an input can be done uniformly in size $O(n^{2k})$ and depth $O((\log n)^k)$. Because the circuit is being simulated implicitly, we incur an additional cost in size and depth, but the overall size is at most $O(n^{3k})$ and depth at most $O((\log n)^{k^2})$. Thus, by setting $k' = k^2$, we have the required simulation. But this contradicts Proposition 3, since L is arbitrary. □

Similarly, the following can be shown. We omit the proof because of its similarity to the previous ones.

Reminder of Theorem 1.2 *For any k,* LOGSPACE *does not have* LOGSPACE*-uniform branching programs of size $O(n^k)$.*

Theorem 1.1 improves Kannan's result that $\mathsf{NP} \not\subseteq \mathsf{P}\text{-uniform}\mathsf{SIZE}(n^k)$ by showing a better upper bound for the hard language, i.e., P rather than NP. We can improve his result in a different way by relaxing the uniformity condition instead to $\mathsf{P}_{||}^{\mathsf{NP}}$-uniformity. The main idea is to first relativize Theorem 1.1 to allow parallel access to an NP oracle both in the upper bound and in the uniformity bound, and then to strengthen the upper bound using an idea of Fortnow, Santhanam and Williams [FSW09].

Reminder of Theorem 1.3 *Let $k \geq 1$ be any constant. Then* $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\text{-uniform}\mathsf{SIZE}(n^k)$.

Proof of Theorem 1.3. First we show that $\mathsf{P}_{||}^{\mathsf{NP}} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\text{-uniform}\mathsf{SIZE}(n^k)$ in a completely analogous way to Theorem 1.1. Then we claim that if $\mathsf{P}_{||}^{\mathsf{NP}} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\text{-uniform}\mathsf{SIZE}(n^k)$, then $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\text{-uniform}\mathsf{SIZE}(n^{k-1})$. This result was shown without the uniformity conditions by Fortnow, Santhanam and Williams [FSW09]. An examination of their proof shows that a circuit C_n for any language in $\mathsf{P}_{||}^{\mathsf{NP}}$ can be constructed using fixed polynomial-time oracle access to circuits for two specific languages in NP. If the circuit sequences for these languages is each $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform, then so is the small circuit sequence for the $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform language, by converting the polynomial-time oracle machine to an oracle circuit and then substituting the circuits for the two oracles. Since k is arbitrary, we are done. □

For $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$, we do not get an unconditional lower bound, but rather a "gap result" in the style of Impagliazzo and Wigderson [IW01] or Buresh-Oppenheim and Santhanam [BOS06]. The result states that if we can diagonalize in E against an arbitrarily small exponential amount of advice, then we get lower bounds against E-uniform circuits of size close to the best possible. The main idea is to use the proof idea of Theorem 1.1 recursively.

Theorem 2.2 *If* $\mathsf{E} \not\subseteq \mathsf{DTIME}(2^{2n})/2^{\epsilon n}$ *for some $\epsilon > 0$, then* $\mathsf{E} \not\subseteq \mathsf{E}\text{-uniform}\mathsf{SIZE}(2^{\delta n})$ *for any $\delta < 1$.*

Proof. Let $\delta < 1$ be any constant. Assume that for any $L \in \mathsf{E}$, $L \in \mathsf{E}\text{-uniform}\mathsf{SIZE}(2^{\delta n})$. We will show that it follows that $L \in \mathsf{DTIME}(2^{2n})/2^{\epsilon n}$ for any constant $\epsilon > 0$.

We define a sequence of languages L_i as follows. $L_0 = L$. In general, L_i will be a "succinct" version of a connection language of circuits for L_{i-1}. The direct connection language we used before will not be succinct enough for our purposes, so we use instead what we call the indirect connection language L_{ic} of a sequence of circuits, where a tuple $\langle 1^n, g, i, b_1, b_2, r\rangle$ is in L_{ic} iff the gate with index g has type r, and moreover, if the type r is not INPUT, then if the bit $b_1 = 0$, the i'th bit of the index of the first input to g is b_2, and if the bit $b_1 = 1$, the i'th bit of the index of the second input to g is b_2. Essentially, L_{ic} encodes the adjacency list corresponding to the DAGs of the circuit sequence rather than the adjacency matrix. Note that for any sequence of circuits of size $2^{O(n)}$, $L_{ic} \in E$ iff $L_{dc} \in E$.

We now define L_1 more precisely. By assumption, there is a sequence of circuits $\{C_n\}$ for L such that C_n is of size $O(2^{\delta n})$ for each n, and the indirect connection language $L_{ic,0}$ of the sequence of circuits can be decided in E (since by assumption, the direct connection language can be decided in E). L_1 is the succinct version of $L_{ic,0}$ defined as follows: a tuple $\langle Bin(n), g, i, b_1, b_2, r\rangle$ belongs to L_1 iff the tuple $\langle 1^n, g, i, b_1, b_2, r\rangle$ belongs to $L_{ic,0}$. Note that since the gate index of g requires at least δn bits to describe, we can decide L_1 in E, hence by assumption, L_1 has E-uniform circuits of size $2^{\delta m}$.

Let $\{C_m^1\}$ be an E-uniform sequence of circuits of size $2^{\delta m}$ for L_1. As a function of n, the size of C_m^1 is at most $O(2^{\delta(\delta n + O(\log n))}) = O(2^{\delta^2 n}\text{poly}(n))$. Let $L_{ic,1}$ be the indirect connection language of the sequence $\{C_m^1\}$. We define L_2 to be the succinct version of $L_{ic,1}$ completely analogously to the previous paragraph.

Continuing in this way, we get a sequence of languages $L_1, L_2 \ldots$ such that L_k has E-uniform circuits of size $O(2^{\delta^k n}\text{poly}(n))$. Let k be such that $\delta^k < \epsilon$. Since $\delta < 1$, there exists such a k.

We now define a simulation of L in time $O(2^{2n})$ with $O(2^{\epsilon n})$ bits of advice. The advice is the description of a circuit for L_k. Given this description, we can recover in time $2^{(\delta^{k-1}+\delta^k+o(1))n}$ the description of a circuit for L_{k-1}. Again, from this description, we can recover in time $2^{(\delta^{k-1}+\delta^{k-2}+o(1))n}$ the description of a circuit for L_{k-2}. Continuing in this way, we can recover in total time $2^{(\delta+\delta^2+o(1))n} = O(2^{2n})$ the circuit C_n for L, whereupon we can run C_n on L to determine whether the input belongs to L or not.

□

Note that the conditional lower bound of Theorem 2.2 is close to best possible, as shown by the following easy result.

Proposition 4 E *has* E*-uniform circuits of size at most* $n2^n$.

Proof. For any language L in E, the truth table of L can be computed in linear exponential time, and from the truth table it is easy to compute canonical DNFs or CNFs of size at most $n2^n$ for L. □

The ideas above can also be used to say something about lower bounds for QBF. It is an open question whether QBF is in P-uniformNC. What we can show is that either P requires large non-uniform circuits, or QBF requires large P-uniform branching programs of exponential size. Note that the second disjunct is a substantially stronger statement than separating QBF from P-uniformNC.

The following proposition can be shown using a simple diagonalization.

Proposition 5 *For constructible space bounds* S_1 *and* S_2 *such that* $S_1 = \Omega(\log n)$ *and* $S_1 = o(S_2)$, $\text{SPACE}(S_2) \not\subseteq \text{SPACE}(S_1)/n$.

Reminder of Theorem 1.4 *For each constant* $k > 1$, P *does not have non-uniform circuits of size* n^k*, or else QBF does not have* P-uniform *branching programs of size* $2^{n^{o(1)}}$.

Proof of Theorem 1.4. Assume, to the contrary, that there is a constant k such that P has circuits of size n^k, and moreover that QBF has P-uniform branching programs of size $2^{n^{o(1)}}$. Under these assumptions, we will show a simulation of QBF in sublinear space with sublinear advice, and then derive a contradiction to the space hierarchy theorem.

Let $\{B_n\}$ be a sequence of P-uniform branching programs for QBF of size $2^{n^{o(1)}}$, and let $L_{dc} \in$ P be the direct connection language of this sequence. Consider the succinct version L_{succ} of L_{dc} defined as in the proof of Theorem 1.1. Since $L_{dc} \in$ P, $L_{succ} \in$ P, and hence L_{succ} has a sequence of circuits $\{D_m\}$ of size $O(m^k)$. We define a machine M accepting QBF in sublinear space with sublinear advice as follows: M treats its advice as a circuit D_m for L_{succ} on inputs of length $m(n)$, where $m(n) \leq n^{1/(2k)}$ is the same function as in the proof of Theorem 1.1. This circuit can be encoded using at most $n^{2/3}$ bits of advice. M implicitly generates parts of the branching program B_n as and when needed. It does this by repeatedly calling the circuit D_m on different inputs, and determining which state is the currently relevant one in the computation. It begins by cycling over all possible state inputs to D_m and checking which one has type "Start". Note that since B_n has size $2^{n^{o(1)}}$, the name of a state can be maintained using space $n^{o(1)}$. Each time B_n makes a transition, M figures out which state B_n transitions to, again calling the circuit D_m repeatedly to determine this. Note that since D_m has size $O(\sqrt{n})$, simulating D_m on any given input of length $m(n)$ can be done in space $O(n^{2/3})$. M continues the implicit simulation of B_n until an accept or reject state is reached in B_n, at which point it accordingly accepts or rejects.

The total space requirement of M is $O(n^{2/3})$, and M uses $O(n^{2/3})$ bits of advice. Therefore QBF is in $\mathsf{SPACE}(n^{2/3})/O(n^{2/3})$. Now, since QBF is complete for $\mathsf{SPACE}(n \cdot (\log n)^{O(1)})$ under quasilinear-time reductions, we have that $\mathsf{SPACE}(n \cdot (\log n)^{O(1)}) \subseteq \mathsf{SPACE}(n^{3/4})/n^{3/4}$ for large enough n, which is a contradiction to Proposition 5. □

3 A Uniformization Lemma For NC^1

We now turn to the problem of eliminating non-uniformity in low-complexity circuit classes. Recall the FORMULA EVAL problem: *given a formula F and input v to it, determine whether $F(v) = 1$*. Buss [Bus87] showed that FORMULA EVAL is complete under $\mathsf{DLOGTIME}$-reductions for $\mathsf{DLOGTIME}$-uniform NC^1. Hence FORMULA EVAL can be solved efficiently (in, for example, TC^0) iff $\mathsf{NC}^1 \subseteq \mathsf{TC}^0$.

Theorem 3.1 *Suppose $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. For every $\varepsilon > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given 1^n, prints an $O(1/\varepsilon)$-depth, $n^{O(1)}$-size $\mathcal{C}$-circuit that solves* FORMULA EVAL *on formulas of size n.*

The following lemma is an immediate corollary:

Reminder of Lemma 1.1 *Suppose $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. For every $\varepsilon, k > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given any circuit C of size n and depth $k \log n$, prints an $O(k/\varepsilon)$-depth, $n^{O(k)}$-size $\mathcal{C}$-circuit that is equivalent to C.*

The proof is inspired by Allender and Koucky [AK10] who showed that if $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, then the problem BALANCED FORMULA EVALUATION has $n^{1+\varepsilon}$ size $\mathcal{C}$-circuits, for $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. Rather than focusing on reducing circuit sizes, we focus on reducing non-uniformity.

Proof of Theorem 3.1. Assuming $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, let $k \geq 1$ be such that the FORMULA EVAL problem on formulas of size n has $\mathcal{C}$-circuits of n^k size. Buss [Bus87] showed that FORMULA EVAL can be solved in LOGTIME-uniform NC^1. Applying this algorithm, we can generate (in polynomial time) an n^c size formula $G(F, x)$ such that $G(F, x) = 1 \iff F(x) = 1$, for all formulas F of size n and all potential inputs x of length up to n. WLOG, G has depth at most $c \log n$, for some fixed $c \geq 1$.

Partition G into $t = n^{c-\varepsilon/k}$ subformulas $F_1, \ldots, F_t$ of at most $n^{\varepsilon/k}$ gates each. More precisely, we break the $c \log n$ levels of G into kc/ε groups, where each group contains $(\varepsilon/k) \log n$ adjacent levels. Each group consists of subformulas of depth $(\varepsilon/k) \log n$ and size at most $n^{\varepsilon/k}$. WLOG, we may assume each F_i has the same number of inputs (roughly $n^{\varepsilon/k}$).

Next, we "brute force" a small $\mathcal{C}$ circuit for small instances of FORMULA EVAL. Try all possible $\mathcal{C}$ circuits of size n^ε for FORMULA EVAL on all formula-input pairs of length up to $n^{\varepsilon/k}$. For each trial circuit T, we try all possible $2^{\tilde{O}(n^{\varepsilon/k})}$ formula-input pairs, and check that T correctly evaluates the input on the formula. By our choice of k, at least one T will pass this check on all of its inputs.

Once a suitable T has been found, we replace every subformula $F_i(y_1, \ldots, y_q)$ in G with the $\mathcal{C}$ circuit $T(F_i, y_1, \ldots, y_q)$. By our choice of T, the resulting circuit is equivalent to G, has size $O(n^{c-\varepsilon/k} \cdot n^\varepsilon) \leq O(n^{c+\varepsilon})$, and has depth dkc/ε, where d is the depth of T.

It is clear that the algorithm can run in $2^{O(n^\varepsilon)}$ time for any $\varepsilon > 0$. Furthermore, it can also be implemented to run in $O(n^\varepsilon)$ space: any desired bit of the n^c size formula G for FORMULA EVAL instance can be generated in LOGTIME, so the brute-force search for an n^ε size $\mathcal{C}$ circuit T equivalent to FORMULA EVAL can be carried out in $O(n^\varepsilon)$ space. Given T, the rest of the $\mathcal{C}$ can easily be generated in $O(n^\varepsilon)$ space by reading the appropriate bits from G. □

Note that, rather than brute-forcing the small $\mathcal{C}$ circuit for FORMULA EVALUATION, we could have simply provided it as advice. This implies:

Reminder of Corollary 1.1 *For $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly} \iff$ for all $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^{\varepsilon}$.*

Lemma 1.1 and Corollary 1.1 have consequences for lower bounds as well as algorithms. We first give a consequence for lower bounds, showing that either TC^0 computations cannot be speeded up in general using logarithmic depth and bounded fan-in, or NC^1 does not have non-uniform polynomial-size threshold circuits of bounded depth.

We need a hierarchy theorem for TC^0, which can be shown analogously to Proposition 1 and Proposition 3.

Proposition 6 *For any constants k and d and any $\epsilon < 1$, there is a language in TC^0 which cannot be decided by $\mathsf{DLOGTIME\text{-}uniform}\mathsf{TC}^0$ circuits of size n^k and depth d with n^{ϵ} bits of advice.*

Theorem 3.2 *At least one of the following holds:*

- *For all constants k, there is a language in TC^0 which does not have $\mathsf{DLOGTIME\text{-}uniform}$ circuits of depth $k \log n$.*
- $\mathsf{NC}^1 \not\subset \mathsf{TC}^0/\text{poly}$.

Proof. Assume that $\mathsf{NC}^1 \subset \mathsf{TC}^0/\text{poly}$ and that there is a constant k such that each language in TC^0 has $\mathsf{DLOGTIME\text{-}uniform}$ circuits of depth $k\log(n)$. We derive a contradiction.

Let L be an arbitrary language in TC^0. By the second assumption, L has $\mathsf{DLOGTIME\text{-}uniform}$ circuits of depth $k\log(n)$. From the first assumption and using Corollary 1.1 with $\epsilon = 1/(2k)$, we have that there exists constants c and d such that FORMULA EVAL can be decided by uniform threshold circuits of size m^c and depth d with m^{ϵ} bits of advice. This implies that any language with $\mathsf{DLOGTIME\text{-}uniform}$ circuits of depth $k\log(n)$ can be decided by uniform threshold circuits of size $O(n^{kc})$ and depth d with $O(n^{1/2})$ bits of advice, and hence so can L. Since L is an arbitrary language in TC^0, this contradicts Proposition 6. □

3.1 Algorithms From NC^1 Upper Bounds

We next derive some algorithmic consequences of Lemma 1.1.

Corollary 3.1 *If $\mathsf{NC}^1 \subset \mathsf{ACC}/\text{poly}$ then for all c, satisfiability of n^c size formulas with n variables can be computed (deterministically) in $O(2^{n-n^{\varepsilon}})$ time, for some $\varepsilon > 0$ depending on c.*

Proof. Given a formula F of size n^c, apply Lemma 1.1 to generate an equivalent $n^{O(c/\delta)}$ size ACC circuit of depth $O(1/\delta)$, in $2^{O(n^{\delta})}$ time, for some $\delta < 1$. Satisfiability of the ACC circuit can be determined in $2^{n-n^{\varepsilon}}$ time via an ACC-SAT algorithm [Wil11]. □

This corollary can be extended to conclude not just faster SAT algorithms but also faster QBF algorithms. Define QB-CNF, QB-CIRCUITS, QB-FORMULAS to be the quantified Boolean formula problem over CNF predicates, arbitrary circuit predicates, and formula predicates, respectively. In satisfiability problems, the choice of predicate can play a major role in the time complexity of the problem.

Reminder of Theorem 1.5 *If $\mathsf{NC}^1 \subset \mathsf{ACC}/\text{poly}$ then for every $k, c > 0$, QB-FORMULAS on formulas of n^k size and n variables can be solved in (deterministic) $O(2^n/n^c)$ time.*

Proof of Theorem 1.5. Given a formula F, we first use Lemma 1.1 to generate an equivalent ACC circuit A in subexponential time. Let A have n^q size. We determine the truth of the quantified circuit A as follows.

By trying all n^k possible settings to the last $k \log n$ quantified variables of the QBF, we can create a balanced formula T of size $O(n^k)$ and $k \log n$ depth, with copies of the ACC circuit A at its leaves, such that this overall circuit with $n - k \log n$ quantified variables is true if and only if the original quantified ACC circuit A is true. In particular, the ith level of T will have AND gates (resp., OR gates) if the ith quantified variable (out of the last $k \log n$ quantified variables) is universal (resp., existential). Each leaf of T represents an assignment on these last $k \log n$ variables, which is plugged into a copy of A and this copy is inserted at the appropriate leaf of T. The overall circuit has size $O(n^{k+q})$.

Next, we apply Lemma 1.1 to T. In $2^{O(n^\varepsilon)}$ time, we can generate an ACC circuit A' of $n^{O(k/\varepsilon)}$ size which equals T on all of its inputs. Replacing T with A' in the circuit, we now have a $n^{O(k/\varepsilon)+q}$-size circuit A'' composed of A' and copies of A, with only $n - k \log n$ inputs.

Using the ACC Circuit Evaluation algorithm of [Wil11], we can evaluate A'' on all of its $2^{n-k\log n}$ possible inputs in $O(2^n/n^{k-3} + n^{\text{poly}(k/\varepsilon,q,\log n)})$ time. Since k, ε, and q are constant, the running time is dominated by the first term. From the truth table of A'' we can determine whether the original quantified A is true in $O(2^n/n^{k-3})$ time. Setting k to be arbitrarily large, the theorem holds. □

3.2 Very Weak Derandomization For TC^0 Lower Bounds

Another consequence of Lemma 1.1 is that we can further weaken the algorithmic hypotheses needed to prove $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$. For $\mathcal{C} \in \{\mathsf{P}, \mathsf{NC}^1, \mathsf{TC}^0\}$, define DERANDOMIZE-$\mathcal{C}$ to be the following promise problem: *given a $\mathcal{C}$/poly circuit C which is promised to be either unsatisfiable or have at least 2^{n-1} satisfying assignments, determine which is the case*. We say that a nondeterministic algorithm A *solves* DERANDOMIZE-TC^0 if for all circuits C satisfying the promise,

- every computation path of $A(C)$ leads to one of three states: *reject*, *unsatisfiable*, or *satisfying*,
- at least one path of $A(C)$ is not a *reject*,
- if C is satisfiable then no path of $A(C)$ is *unsatisfiable*,
- if C is unsatisfiable then no path of $A(C)$ is *satisfiable*.

Theorem 3.3 *Suppose for all k, there is an $O(2^n/n^{10})$ time algorithm for solving* DERANDOMIZE-TC^0 *on all TC^0 circuits of n inputs, n^k size, and depth k. Then $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$.*

Proof. (Sketch) Using the succinct PCPs of Ben-Sasson *et al.* [BGH+05], Williams [Wil10] proved that if DERANDOMIZE-P on all n-input n^k-size circuits can be solved in $O(2^n/n^{10})$ time, then $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$. The construction of Ben-Sasson *et al.* in fact has the following property: given any $L \in \mathsf{NTIME}[2^n]$, they can generate (in polynomial time) an NC^1 circuit C with $n + O(\log n)$ inputs and n^c size for some universal c, such that

- if $x \in L$ then the truth table of C encodes a satisfiable k-CSP (for some fixed constant k),
- if $x \notin L$ then the truth table of C encodes a k-CSP that is at most $1/2$ satisfiable (namely, all assignments satisfy at most $1/2$ of the constraints).

Using the arguments of Williams [Wil10, Wil11], this can be used to show that if DERANDOMIZE-NC^1 on all n-input n^k-size circuits can be solved in $O(2^n/n^{10})$ time, then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\text{poly}$. Briefly, the idea is to assume that DERANDOMIZE-NC^1 has the $O(2^n/n^{10})$-time algorithm, and that $\mathsf{NEXP} \subset \mathsf{NC}^1/\text{poly}$, and use the two assumptions to nondeterministically simulate an arbitrary $L \in \mathsf{NTIME}[2^n]$ in $o(2^n)$ time

(a contradiction to the nondeterministic time hierarchy). This simulation of an arbitrary L can be done by constructing the NC^1 circuit C encoding a PCP for L on the input x, guessing an NC^1 circuit C' that encodes a proof to the PCP, then composing C and C' to form an NC^1 circuit D which is either unsatisfiable (the truth table of C' is a valid proof) or at most half the assignments are satisfying ($x \notin L$ and the truth table of C' is an invalid proof). Running an algorithm for DERANDOMIZE-NC^1 on D that takes $O(2^n/n^{10})$ time results in the contradiction to the nondeterministic time hierarchy.

Now, assume that $\mathsf{NEXP} \subset \mathsf{TC}^0/\text{poly}$ and that DERANDOMIZE-TC^0 can be solved in $O(2^n/n^{10})$ time, via some algorithm A. We wish to derive a contradiction. The first assumption, along with Lemma 1.1, implies that there is a deterministic subexponential time algorithm B which, when given an NC^1 circuit D, can generate an equivalent TC^0 circuit E that is only polynomially larger than D. Therefore, we can solve DERANDOMIZE-NC^1 in $O(2^n/n^{10})$ time as well, by simply applying the algorithm B to convert a given NC^1 circuit to a TC^0 one, then applying the algorithm A for DERANDOMIZE-TC^0. By the previous paragraph, this implies that $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\text{poly}$, a contradiction (as TC^0 is contained in NC^1). □

3.3 Quantified Formulas versus Quantified k-CNF

Finally, we establish a rather tight relationship between the time complexity of solving quantified formulas and that of solving quantified 3-CNF. We will show that the choice of predicate in a quantified problem essentially does not matter: if quantified 3-CNF is solvable in 1.9^n time then all quantified Boolean formulas are solvable in about 1.9^n time.

Definition 3.1 *A quantified Boolean formula ϕ has k* quantifier blocks *or k* alternations *if it has the form*

$$\phi = (Q_1\ x_1, \ldots, x_{t_1})(Q_2\ x_{t_1+1}, \ldots, x_{t_1+t_2}) \cdots (Q_k\ x_{t_1+\cdots+t_{k-1}+1}, \ldots, x_{t_1+\cdots+t_k})F,$$

where each $Q_i \in \{\exists, \forall\}$.

Theorem 3.4 *There is a polynomial time algorithm that takes any Boolean formula of n inputs and s size and outputs an equivalent* QB-CNF *instance of $n + O(\log s)$ variables, $O(s^4)$ clauses, and $O(\log n)$ quantifier blocks (alternations).*

Proof. We first do some general massaging of a given formula F, construed as a tree with interior nodes labeled by AND/OR gates, and leaves labelled by literals. We can assume WLOG that F has depth $c \log s$ where $c < 4$. (If this is not true, we can make it the case, at a cost of squaring the formula size, via a standard reduction.) Moreover, we may assume that the depth is even, and odd depths of F contain no AND gates (only OR gates, along with possibly $0 - 1$ constants and literals), while even depths contain no OR gates. (Enforcing this only increases the depth by a factor of two, and at most squares the size.) Finally, we can make the length of every path in F from the output gate to a literal *exactly* $d = 4 \log s$. (Suppose a path ends early at a literal ℓ; since $b \wedge b = b$ and $b \vee b = b$ for $b \in \{0, 1\}$, we can duplicate occurrences of ℓ to match the desired alternations of ANDs and ORs and the desired length of each path.) Notice that F now has exactly 2^d leaves.

Let d be the depth of F, and let $L = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$ be the set of literals of F. Define a mapping $\phi_F : \{0, 1\}^d \to L$ as follows. Given a d-bit string b, label every edge in F to a left child with a 0, and every edge to a right child with 1. Follow the path from the output gate (the root of the tree) by reading the bits of b from left to right, then output the literal at the leaf found. Note that by our above reductions on F, the map ϕ_F is a bijection.

The variables of our QB-CNF instance will be $x_1, \ldots, x_n$ along with new variables $y_1, \ldots, y_d$. For convenience, we use the notation $y_i^1 := y_i$ and $y_i^0 := \neg y_i$. The instance of QB-CNF includes all 2^d possible

clauses of the form

$$(y_1^{1-b_1} \vee \cdots \vee y_d^{1-b_d} \vee \phi_F(b_1, \ldots, b_d)),$$

over all possible vectors $(b_1, \ldots, b_d) \in \{0,1\}^d$. Noticing that

$$(y_1^{1-b_1} \vee \cdots \vee y_d^{1-b_d} \vee \phi_F(b_1, \ldots, b_d)) \equiv ((y_1^{b_1} \wedge \cdots \wedge y_d^{b_d}) \to \phi_F(b_1, \ldots, b_d)),$$

we have that (a) for every assignment to $(y_1, \ldots, y_d)$, at most one of the above clauses is not trivially satisfied, and (b) this remaining clause is satisfied iff the literal at the leaf defined by the root-to-leaf path $b_1 \cdots b_d$ is true. The final formula is then

$$(Q_1 x_1, \ldots, Q_n x_n)(\forall y_1)(\exists y_2) \cdots (\forall y_d)[C]$$

where C is the above collection of clauses and $Q_i \in \{\exists, \forall\}$ is the quantifier on variable x_i from the original quantified Boolean formula. □

Corollary 3.2 *There is a polynomial time reduction from* QB-FORMULA *instances of n inputs and s size to* QB-CNF *instances of $n + O(\log s)$ variables and $O(s^4)$ size.*

Corollary 3.3 *There is a polynomial time reduction from* FORMULA-SAT *instances of n inputs and s size to* QB-CNF *instances of $n + O(\log s)$ variables, $O(s^4)$ size, and $O(\log n)$ quantifier blocks.*

Corollary 3.4 *If* QB-CNF *on n^k size instances with $k \log n$ alternations can be solved in $2^n/n^c$ time for every $c, k > 0$, then* NEXP $\not\subset$ NC^1/poly.

Proof. Williams [Wil11] shows that if FORMULA-SAT instances of n inputs and n^k size can be solved in $O(2^n/n^{10})$ time for all k, then NEXP $\not\subset$ NC^1/poly. Hence the proof follows from Corollary 3.3. □

We can reduce from quantified CNF formulas to quantified k-CNF formulas, by applying a reduction of Calabro, Impagliazzo, and Paturi [CIP10].

Theorem 3.5 ([CIP10]) *For all k, there is a polynomial-time reduction from* CNF-SAT *with n variables and m clauses to* QB-kCNF *with $n + O(m^{1/(k-1)})$ variables,* poly(m, n) *clauses, and two quantifier blocks.*

Their proof easily extends to a reduction from QB-CNF with q quantifier blocks to QB-kCNF with $q + 1$ quantifier blocks, with the same increase in the number of variables. Combining Corollary 3.2 and Theorem 3.5, we find a close relation between the time complexity of QB-kCNF and QB-FORMULA:

Reminder of Theorem 1.6 *If there an $\varepsilon > 0$ such that for all k, c,* QB-kCNF *of size n^c is in time 2^{n-n^ε}, then* QB-FORMULA *for n^c-size formulas is in time 2^{n-n^ε} (hence* NEXP $\not\subset$ NC^1/poly *[Wil11]).*

We conclude with an algorithmic hypothesis for QB-3CNF, which follows from the above:

Corollary 3.5 *If* QB-3CNF *with* poly(n) *clauses and $O(\log n)$ alternations can be solved in $O(2^{n-n^{1/2+\varepsilon}})$ time for some $\varepsilon > 0$, then* NEXP $\not\subset$ NC^1/poly.

Many other interesting consequences can be derived along these lines. An intriguing question is whether one can find an algorithm for QB-3CNF with running time that is close to the hypothesis: can QB-3CNF with poly(n) clauses be solved in $2^{n-n^{1/2}}$ time?

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short PCPs verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005.

[BOS06] Joshua Buresh-Oppenheim and Rahul Santhanam. Making hard problems harder. In *Proceedings of 21st Annual IEEE Conference on Computational Complexity*, pages 73–87, 2006.

[Bus87] Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *STOC*, pages 123–131, 1987.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *Proc. IEEE Conference on Computational Complexity*, pages 252–260, 2006.

[CIP09] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Proc. International Workshop on Parameterized and Exact Computation*, 2009.

[CIP10] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. On the exact complexity of evaluating quantified k-CNF. In *Parameterized and Exact Computation (IPEC)*, pages 50–59, 2010.

[DH09] Evgeny Dantsin and Edward A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*, pages 403–424. IOS Press, 2009.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed polynomial size circuit bounds. In *Proceedings of 24th Annual IEEE Conference on Computational Complexity*, pages 19–26, 2009.

[Has98] Johan Hastad. The shrinkage exponent of de Morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

[Her11] Timon Hertli. 3-SAT faster and simpler - Unique-SAT bounds for PPSZ hold in general. In *FOCS*, pages 277–284, 2011.

[HS65] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.

[HS66] Frederick Hennie and Richard Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

[Nec66] Eduard Neciporuk. On a boolean function. *Doklady of the Academy of the USSR*, 169(4):765–766, 1966.

[PPSZ98] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005. (See also FOCS'98).

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

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 231–240, 2010.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of 26th Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de