# On Uniformity and Circuit Lower Bounds

**Citation for published version:**
Santhanam, R & Williams, R 2014, 'On Uniformity and Circuit Lower Bounds', *Computational complexity*, vol. 23, no. 2, pp. 177-205. https://doi.org/10.1007/s00037-014-0087-y

**Digital Object Identifier (DOI):**
10.1007/s00037-014-0087-y

**Link:**
Link to publication record in Edinburgh Research Explorer

**Document Version:**
Publisher's PDF, also known as Version of record

**Published In:**
Computational complexity

comput. complex. **23** (2014), 177 – 205
© Springer Basel 2014
1016-3328/14/020177-29
*published online* May 16, 2014
DOI 10.1007/s00037-014-0087-y

computational complexity

# ON UNIFORMITY AND CIRCUIT LOWER BOUNDS

Rahul Santhanam and Ryan Williams

**Abstract.** We explore relationships between circuit complexity, the complexity of generating circuits, and algorithms for analyzing circuits. Our results can be divided into two parts:

1. *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 (stronger than LOGTIME) but not infeasible. Using a new kind of indirect diagonalization argument, we prove several new unconditional lower bounds against medium-uniform circuit classes, including:

- For all $k$, $\mathsf{P}$ is not contained in $\mathsf{P}$-uniform $\mathsf{SIZE}(n^k)$. That is, for all $k$, there is a language $L_k \in \mathsf{P}$ that does not have $O(n^k)$-size circuits constructible in polynomial time. This improves Kannan's lower bound from 1982 that $\mathsf{NP}$ is not in $\mathsf{P}$-uniform $\mathsf{SIZE}(n^k)$ for any fixed $k$.
- For all $k$, $\mathsf{NP}$ is not in $\mathsf{P}^{\mathsf{NP}}_{||}$-uniform $\mathsf{SIZE}(n^k)$. This also improves Kannan's theorem, but in a different way: the uniformity condition on the circuits is stronger than that on the language itself.
- For all $k$, $\mathsf{LOGSPACE}$ does not have $\mathsf{LOGSPACE}$-uniform branching programs of size $n^k$.

2. *Eliminating non-uniformity and (non-uniform) circuit lower bounds.* We complement these results by showing how to convert any potential simulation of $\mathsf{LOGTIME}$-uniform $\mathsf{NC}^1$ in $\mathsf{ACC}^0$/poly or $\mathsf{TC}^0$/poly into a medium-uniform simulation using small advice. This lemma can be used to simplify the proof that faster SAT algorithms imply $\mathsf{NEXP}$ circuit lower bounds and leads to the following new connection:

- Consider the following task: *given a* $\mathsf{TC}^0$ *circuit* $C$ *of* $n^{O(1)}$ *size, output* yes *when* $C$ *is unsatisfiable, and output* no *when* $C$ *has at least* $2^{n-2}$ *satisfying assignments.* (Behavior on other inputs can be arbitrary.) Clearly, this problem can be solved efficiently using randomness. If this problem can be solved deterministically in $2^{n-\omega(\log n)}$ time, then $\mathsf{NEXP} \not\subset \mathsf{TC}^0$/poly.

Another application is to derandomize randomized $\mathsf{TC}^0$ simulations of $\mathsf{NC}^1$ on almost all inputs:

- Suppose $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. Then, for every $\varepsilon > 0$ and every language $L$ in $\mathsf{NC}^1$, there is a LOGTIME-uniform $\mathsf{TC}^0$ circuit family of polynomial size recognizing a language $L'$ such that $L$ and $L'$ differ on at most $2^{n^\varepsilon}$ inputs of length $n$, for all $n$.

**Keywords.** Circuit complexity, uniformity, lower bounds, satisfiability algorithms.

**Subject classification.** 68Q15: Complexity classes, 68Q17: Computational difficulty of problems.

## 1. Introduction

An important problem at the interface of machine-based computational complexity and circuit complexity is to understand the relationship between the complexity of circuits for solving problems, and the complexity of algorithmic processes that can generate those circuits from scratch. In the non-uniform setting, we put no computable bounds on the complexity of generating circuits, and in this case, it is extraordinarily difficult to prove lower bounds even against "simple" circuit classes. In low-uniform[1] settings like LOGTIME uniformity (Barrington *et al.* 1990; Buss 1987), 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, is a middle ground that is less understood, and it is quite plausible that exploring it will help us better understand the two extremes. In this paper, we study this middle ground and make concrete progress that may translate to further progress on low-uniform and non-uniform results in the future.

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

**Lower bounds by amplifying uniformity.** In the first part of the paper, we prove new lower bounds against "medium-uniform" circuits. Our key insight is that if we assume that a "medium complexity" class has small medium-uniform circuits, this assumption can be applied in multiple ways: not only is it applicable to a language $L$ in the medium complexity class, but it is also applicable to another (medium complexity) language encoding *circuits* for $L$. Applying the hypothesis multiple times allows us to simulate medium-uniformity with a very small amount of non-uniform advice and diagonalize against the small-advice simulation to derive a contradiction.

This strategy leads to new 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 $\mathsf{LOGTIME}$-uniformity condition is imposed on the class, then it can be simulated in nearly-linear deterministic time, and we can easily diagonalize to find a function in (for example) $n^2$ time that does not have such circuits. However, suppose we wish to find a function in $\mathsf{P}$ that does not have small circuits constructible via a "medium" uniformity notion, such as $\mathsf{P}$-uniformity. Then, the notion of uniformity (which allows for an arbitrary polynomial-time bound) can be more powerful than the function itself (which must lie in some fixed polynomial time bound), so it is no longer possible to directly diagonalize. Nevertheless, we can "indirectly" diagonalize, by reducing the assumption that $\mathsf{P}$ has small $\mathsf{P}$-uniform circuits to another time hierarchy result.

To describe our results, let us first set up some notation informally (definitions are in Section 1.1). Given a class $\mathcal{C}$ of languages, recall a language $L$ is said to have $\mathcal{C}$-$\mathsf{uniform}$ circuits of size $s(n)$ if there is a size-$s(n)$ circuit family $\{D_n\}$ such that the description of $D_n$ is computable in $\mathcal{C}$. (There are several possible choices about what "description" means; in this paper, our notions of uniformity will be so powerful that these choices are all essentially equivalent.) Our first main result strengthens both the deterministic time hierarchy theorem and the result of Kannan (1982) that for any $k$, $\mathsf{NP}$ is not in $\mathsf{P}$-$\mathsf{uniform}$ $\mathsf{SIZE}(n^k)$.[2]

[2] Kannan's work (Kannan 1982) is primarily known for proving that $\Sigma_2\mathsf{P} \not\subset$

THEOREM 1.1. *For every $k$,* $\mathsf{P} \not\subset \mathsf{P}$-$\mathsf{uniform\ SIZE}(n^k)$.

That is, for all $k$, there is an $L \in \mathsf{P}$ such that any algorithm generating $n^k$-size circuits for $L$ must run in super-polynomial time. (Of course, the common belief is that for all $k$, there is an $L \in \mathsf{P}$ that does not have $n^k$-size circuits *at all*, but this is an extremely hard problem: resolving it would imply $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$, for example.) The ideas in the proof of Theorem 1.1 can also be applied to smaller classes. Note that the best non-uniform branching program size lower bounds known have the form $\Omega(n^2/\mathrm{poly}(\log n))$ (Nečiporuk 1966). However, for branching programs constructible in logarithmic space, we can prove superpolynomial lower bounds:

THEOREM 1.2. *For every $k$,* $\mathsf{LOGSPACE}$ *does not have* $\mathsf{LOGSPACE}$-*uniform branching programs of size $O(n^k)$.*

We are also able to strengthen Kannan's lower bound for $\mathsf{NP}$ 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 of Fortnow *et al.* (2009) to show that fixed polynomial-size lower bounds can be "amplified."

THEOREM 1.3. *For every $k$,* $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform\ SIZE}(n^k)$.

**Reducing non-uniformity for "simple" circuit classes.** The above lower bounds work by simulating medium-uniform circuits with low-uniformity and a little advice. In the second part of the paper, we study situations where *non-uniform* circuits can be simulated by "medium-uniform" ones with a little advice. We show how to translate non-uniform constant-depth circuits for $\mathsf{NC}^1$ into *subexponential-time uniform* constant-depth circuits for $\mathsf{NC}^1$:

Footnote 2 continued
$\mathsf{SIZE}(n^k)$ for all constants $k$. The statement $\mathsf{NP} \not\subset \mathsf{P}$-$\mathsf{uniform\ SIZE}(n^k)$ is a corollary, because its negation implies that $\mathsf{P} = \mathsf{NP} \subset \mathsf{SIZE}(n^k)$, hence $\Sigma_2\mathsf{P} = \mathsf{NP} \subset \mathsf{SIZE}(n^k)$, contradicting the lower bound for $\Sigma_2\mathsf{P}$.

Lemma 1.4. *Suppose* $\mathsf{NC}^1$ *is contained in* $\mathcal{C}$/poly, *where* $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$.[3] *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 $\mathsf{NC}^1$ in $\mathsf{TC}^0$ implies small-space uniform (and hence subexponential-time uniform) $\mathsf{TC}^0$ circuits for $\mathsf{NC}^1$. We remark that Lemma 1.4 holds for any constant-depth circuit class over any basis (with arbitrary fan-in): $\mathsf{ACC}^0$ and $\mathsf{TC}^0$ just happen to be the most popular such classes.

An interesting consequence of Lemma 1.4 is that, for simulations of $\mathsf{NC}^1$ in smaller classes, non-uniformity can be simulated by low-uniformity with small advice:

Corollary 1.5. *For* $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}$/poly *if and only if for all* $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^{\varepsilon}$.

Another consequence of Lemma 1.4 is that *randomized* simulations based on constant-depth circuits can be meaningfully derandomized, assuming they are powerful enough. For a ($\mathsf{LOGTIME}$-uniform) complexity class $\mathcal{C}$, we define $\mathsf{BP}\mathcal{C}$ to be its randomized version with two-sided error in the standard way.

Theorem 1.6. *Suppose* $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. *Then, for every* $\varepsilon > 0$ *and every language* $L$ *in* $\mathsf{NC}^1$, *there is a* ($\mathsf{LOGTIME}$ *uniform*) $\mathsf{TC}^0$ *circuit family of polynomial size recognizing a language* $L'$ *such that* $L$ *and* $L'$ *differ on at most* $2^{n^{\varepsilon}}$ *inputs of length* $n$, *for all* $n$.

That is, if $\mathsf{NC}^1$ can be solved with uniform probabilistic $\mathsf{TC}^0$ circuits, then the circuits can be made deterministic while preserving uniformity, at the expense of making errors on a small fraction of inputs. This is somewhat surprising, and it is reasonable to think that Theorem 1.6 may be applied to prove lower

[3] In this paper, the default assumption (unless otherwise specified) is that a complexity class $\mathcal{C}$ defined with respect to some polynomial-size class of circuits is $\mathsf{LOGTIME}$-$\mathsf{uniform}$ , and we use $\mathcal{C}$/poly to denote the non-uniform version of the class. This notation makes it clear when we are discussing uniform versus non-uniform classes.

bounds against $\mathsf{BPTC}^0$ in the future. (Currently, $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPTC}^0$ is open!) The idea is to combine the ideas of Lemma 1.4 with prior work of Goldreich & Wigderson (2002), who show how to derandomize algorithms in generic settings when the random bits used are both "small" with respect to the input length and "oblivious" to the input.

Lemma 1.4 can also be applied to simplify and strengthen a key component of the proof that faster $\mathcal{C}$-SAT algorithms imply $\mathsf{NEXP} \not\subset \mathcal{C}/\text{poly}$ (for many circuit classes $\mathcal{C}$) (Williams 2010, 2011). In particular, a necessary intermediate theorem says that if (a) there are SAT algorithms for all polynomial-size $\mathcal{C}$-circuits with $n$ inputs running in $O(2^n/n^{10})$ time and (b) $\mathsf{NEXP} \subset \mathcal{C}/\text{poly}$, then there is a nondeterministic $o(2^n)$ time algorithm that, given an unrestricted circuit $D$ of polynomial size, can generate a polynomial-sized $\mathcal{C}$-circuit equivalent to $D$.[4] By exploiting the structure of $\mathsf{NC}^1$, Lemma 1.4 can be used to deterministically generate an equivalent $\mathsf{TC}^0$ (respectively, $\mathsf{ACC}$) circuit from a given $\mathsf{NC}^1$ circuit in *subexponential* ($2^{n^{\varepsilon}}$) time, without assuming any algorithmic improvement on circuit satisfiability.

Finally, we weaken the conditions necessary to prove lower bounds like $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$. Consider the following problem.

DERANDOMIZE-$\mathsf{TC}^0$: *given a* $\mathsf{TC}^0$ *circuit* $C$ *of* $n^{O(1)}$ *size, output* yes *when* $C$ *is unsatisfiable, and output* no *when* $C$ *has at least* $2^{n-2}$ *satisfying assignments.* (Behavior on other inputs can be arbitrary.)

This problem can be trivially solved with high probability, by simply trying random assignments. We prove that a *deterministic* $2^{n-\omega(\log n)}$ time algorithm for the problem is sufficient for $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$:

THEOREM 1.7. *Suppose for all* $k$, *there is an* $O(2^n/n^k)$ *time deterministic algorithm for solving* DERANDOMIZE-$\mathsf{TC}^0$ *on all* $\mathsf{TC}^0$ *circuits of* $n$ *inputs,* $n^k$ *size, and depth* $k$. *Then* $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$.

[4] Recently, Jahanjou *et al.* (2013) gave a succinct $\mathsf{AC}^0$ version of the Cook-Levin theorem, which also simplifies the proof of the original SAT-to-circuit-lower-bounds connection, but does not imply Lemma 1.4.

Goldreich and Meir (personal communication) have observed that the existing framework for proving $\mathsf{NEXP}$ circuit lower bounds from SAT algorithms also extends to derandomization problems which seem much "simpler" than SAT: for example, $2^{n-\omega(\log n)}$ time algorithms for *approximating* the acceptance probability of a given circuit to within 1/10 is already enough to yield $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$. The above theorem strengthens their observation even further: we may focus on the case of distinguishing unsatisfiable circuits from "very satisfiable" circuits.

**1.1. Preliminaries and notation.** We assume basic knowledge of computational complexity (Arora & Barak 2009). Along with that, we will need standard notions of uniformity for circuits. The *direct connection language* for a sequence of circuits $C = \{C_n\}$, where $C_n$ is on $n$ input bits, is the language $L_C$ consisting 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 of course possible, but for the large classes $\mathcal{C}$ we will consider, this encoding will not affect the results.

Given a class $\mathcal{C}$ of languages, a language $L$ is said to have $\mathcal{C}$-$\mathsf{uniform}$ *circuits of size* $s(n)$ if there is a size-$s(n)$ circuit family $\{C_n\}$ such that its direct connection language $L_C$ is computable in $\mathcal{C}$. By a *description of a circuit* $C_n$, we mean the list of tuples in $L_C$ corresponding to gates in $C_n$. A language $L$ is said to have $\mathsf{LOGTIME}$-$\mathsf{uniform}$ *circuits of size* $s(n)$ if there is a size-$s(n)$ circuit family $\{C_n\}$ such that the "compressed" direct connection language

$$L'_C = \{\langle n, g, h, r\rangle \mid \langle 1^n, g, h, r\rangle \in L_C\}$$

is computable in *linear time*. (Note that a linear time algorithm for $L'_C$ runs in time logarithmic in $n$ and the size of the circuit.)

Given a size function $s : \mathbb{N} \to \mathbb{N}$ and depth function $d : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}(s)$ is defined to be the class of languages with *non-uniform* circuits of size $O(s(n))$, $\mathsf{DEPTH}(d)$ the class of languages with non-uniform circuits of depth $d(n)$, and $\mathsf{SIZEDEPTH}(s, d)$ the class of languages which simultaneously have size $O(s(n))$ and depth $d(n)$ non-uniform circuits. (The class $\mathsf{LOGTIME\text{-}uniform\ SIZEDEPTH}$ $(s,d)$ has the usual interpretation: The "compressed" direct connection language for such a family of such circuits is in linear time.)

For more robust circuit complexity classes $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0, \mathsf{NC}^1, \mathsf{NC}\}$ defined with polynomial-size circuits, we use $\mathcal{C}$ to denote the uniform version of the complexity class (with LOGTIME uniformity being the default, unless otherwise specified), 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 $\mathsf{NEXP} \not\subset \mathsf{ACC}/\mathrm{poly}$ (Williams 2011). 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.*, $L_n = L \cap \{0,1\}^n$.

In one of our results, we also require the notion of a direct connection language for a branching program. This is defined in analogously as for a circuit, and 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 $a : \mathbb{N} \to \mathbb{N}$, 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 2.1. $\mathsf{DTIME}(n^{d+1}) \not\subseteq \mathsf{DTIME}(n^d)/n$, *for all* $d \geq 1$.

Proof. Let $\{M_i\}$ be a list of machines running in time $n^d$. We will construct a machine $M'$ running in $n^{d+1}$ time that differs from every $M_i$ and infinite sequence of advice strings $\{a_n\}$ where $|a_n| \leq n$: given an input $x$, $M'(x)$ simulates $M_{|x|}(x)$ augmented with advice string $x'$, where $x'$ is the first $|a_{|x|}|$ bits of $x$. □

The following result simultaneously strengthens the hierarchy theorem that $\mathsf{P} \not\subset \mathsf{DTIME}(n^k)$ for every fixed $k$ (Hartmanis &

Stearns 1965; Hennie & Stearns 1966) and Kannan's result (Kannan 1982) that $\mathsf{NP} \not\subseteq \mathsf{P\text{-}uniform\ SIZE}(n^k)$ for every fixed $k$.

**Reminder of Theorem 1.1** $\mathsf{P} \not\subseteq \mathsf{P\text{-}uniform\ SIZE}(n^k)$, *for all* $k$.

Proof (Theorem 1.1). Assume $\mathsf{P} \subseteq \mathsf{P\text{-}uniform\ SIZE}(n^k)$. Let $L \in \mathsf{P}$ be arbitrary. We will show that $L$ can be simulated in a fixed deterministic time bound with $o(n)$ advice, which will yield a contradiction to Proposition 2.1.

By assumption, $L \in \mathsf{P\text{-}uniform\ SIZE}(n^k)$, so there is a circuit family $\{C_n\}$ for $L$ of size at most $c \cdot n^k$ for some constant $c$. Furthermore, by $\mathsf{P}$-uniformity, the direct connection language $L_{dc}$ for $\{C_n\}$ (see Section 1.1 for the definition) is in $\mathsf{P}$. We consider a "succinct" version $L_{succ}$ of the language $L_{dc}$, defined as follows. Letting $Bin(n)$ be the binary representation of $n$, define

$$L_{succ} = \{\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle \mid \langle 1^n, g, h, r\rangle \in L_{dc}\}.$$

Intuitively, $L_{succ}$ is an "unpadded" version of $L_{dc}$.

Observe that $L_{succ} \in \mathsf{P}$. Given an input $y$ for $L_{succ}$, our polynomial-time algorithm first checks 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 $c \cdot n^k$ and $r$ is a valid gate type. If this check fails, *reject*. Otherwise, the algorithm runs the polynomial-time machine deciding $L_{dc}$ on $\langle 1^n, g, h, r\rangle$, and *accepts* if and only if this machine accepts. Note that this algorithm for $L_{succ}$ runs in time polynomial in $|y|$, since we only simulate the machine for $L_{dc}$ when $n^{1/(3k)} \leq |y| \leq n$ and the machine for $L_{dc}$ runs in time polynomial in $n$.

Now, we apply the assumption $\mathsf{P} \subseteq \mathsf{P\text{-}uniform\ SIZE}(n^k)$ for a second time. Since $L_{succ} \in \mathsf{P}$, there is a circuit family $\{D_m\}$ of $O(m^k)$ size for $L_{succ}$. 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. (Note that this step in fact only requires the assumption $\mathsf{P} \subset \mathsf{SIZE}(n^k)$.)

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. On input $x$ of length $n$, $M$ 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 $m(n) \leq n^{1/(2k)}$. For every possible pair of gate indices $g$ and $h$ of $C_n$ and every possible 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 $O(n)$, as the size of $D_m$ 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 relevant triples $\langle g, h, r \rangle$. Once all these simulations are performed, $M$ has a full description of the circuit $C_n$. It then simulates $C_n$ on $x$ and accepts if and only if $C_n(x)$ outputs 1. 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, $L \in \mathsf{DTIME}(n^{2k+2})/O(n^{1/2} \log n)$.

However, as $L \in \mathsf{P}$ was chosen to be *arbitrary*, we have $\mathsf{P} \subseteq \mathsf{DTIME}(n^{2k+2})/O(n^{1/2} \log n)$, which contradicts Proposition 2.1. □

Note that for every $L \in \mathsf{P}$, there is *some* $k$ such that $L \in$ $\mathsf{P\text{-}uniform\ SIZE}(n^k)$; Theorem 1.1 shows that for *every* $k$, there are languages that not in $\mathsf{P\text{-}uniform\ SIZE}(n^k)$.

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 we prove a lower bound against contains a language that is not in $\mathsf{DTIME}(n^d)$.

PROPOSITION 2.2. *For every* $d \geq 1$, *there is a language in* $\mathsf{P\text{-}uniform\ SIZE}(O(n))$ *that is not in* $\mathsf{DTIME}(n^d)$.

PROOF. The standard proof of the deterministic time hierarchy theorem (Hartmanis & Stearns 1965; Hennie & Stearns 1966) 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 $\mathsf{P}$-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 $\mathsf{NC}$ which do not have $\mathsf{NC}$-uniform formulas of size $n^k$, or indeed $\mathsf{NC}$-uniform circuits of fixed polynomial size and fixed polylogarithmic depth. Note that the best-known formula size lower bound in $\mathsf{NC}$ against non-uniform formulas is $\Omega(n^{3-o(1)})$ (Håstad 1998).

We will require a hierarchy theorem for $\mathsf{NC}$, which can again be shown using standard diagonalization.

PROPOSITION 2.3. *For every $k$, $\mathsf{NC}$ is not contained in $\mathsf{LOGTIME}$-uniform $\mathsf{SIZEDEPTH}(n^k, (\log n)^k)/o(n)$.*

PROOF. Ruzzo (1981) characterized $\mathsf{LOGTIME}$-uniform $\mathsf{SIZEDEPTH}(n^{O(1)}, (\log n)^k)$ as the class of languages decided by alternating Turing machines operating simultaneously in time $O((\log n)^k)$ and space $O(\log n)$. His simulation extends to the following: There is a universal constant $c$ such that every language in $\mathsf{LOGTIME}$-uniform $\mathsf{SIZEDEPTH}(n^k, (\log n)^k)$ can be decided by an alternating machine using $O((\log n)^k)$ time and $c \cdot k \log n + o(\log n)$ space. (Note the $\mathsf{LOGTIME}$ machine for the direct connection language runs in time *linear* in the gate indices; by known simulations of time with alternations (Dymond & Tompa 1985), this machine can be simulated in $O(\log n/ \log\log n)$ alternating time and space.)

The result then follows using the same proof as of Proposition 2.1, but applied to simultaneously time and space bounded alternating Turing machines. □

THEOREM 2.4. *For every $k \geq 1$, $\mathsf{NC}$ is not contained in $\mathsf{NC}$-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.

Now, we will define a family of LOGTIME-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 2.3, 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 old result that $\mathsf{NP} \not\subseteq$ P-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 *et al.* (2009).

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

Proof. First we claim that $\mathsf{P}_{||}^{\mathsf{NP}}$ is not contained in $\mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform}$ $\mathsf{SIZE}(n^k)$; the proof is completely analogous to that of Theorem 1.1. Then, we claim that $\mathsf{P}_{||}^{\mathsf{NP}} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform}$ $\mathsf{SIZE}(n^k)$ implies that $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform}$ $\mathsf{SIZE}(n^{k-1})$. This result was shown without the uniformity conditions by Fortnow *et al.* (2009). 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 $\mathsf{NP}$. If each of the circuit sequences for these languages is $\mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform}$ then so is the small circuit sequence for the $\mathsf{P}_{||}^{\mathsf{NP}}$-$\mathsf{uniform}$ 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 & Wigderson (2001) or Buresh-Oppenheim & Santhanam (2006). The result states that if we can diagonalize in $\mathsf{E}$ against an arbitrarily small exponential amount of advice, then we get lower bounds against $\mathsf{E}$-uniform circuits of size close to the best possible. The main idea is to use the proof idea of Theorem 1.1 recursively.

Theorem 2.5. *If* $\mathsf{E} \not\subseteq \mathsf{DTIME}(2^{2n})/2^{\epsilon n}$ *for some* $\epsilon > 0$, *then* $\mathsf{E} \not\subseteq$ $\mathsf{E}$-$\mathsf{uniform}$ $\mathsf{SIZE}(2^{\delta n})$ *for any* $\delta < 1$.

Proof. Let $\delta < 1$ be any constant. Assume that for any $L \in \mathsf{E}$, $L \in \mathsf{E}$-$\mathsf{uniform}$ $\mathsf{SIZE}(2^{\delta n})$. 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 $b_2$. For technical reasons, we also require that all gate indices $G$ of the circuit are encoded with the same number of bits. 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 $\mathsf{E}$ (since by assumption, the direct connection language can be decided in $\mathsf{E}$). $L_1$ 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 $\delta n$ bits to describe (by definition), we can decide $L_1$ in time $2^{O(n)}$. Hence by assumption, $L_1$ has $\mathsf{E}$-uniform circuits of size $2^{\delta m}$, where $m$ is the length of the input.

Let $\{C^1_m\}$ be an $\mathsf{E}$-uniform sequence of circuits of size $2^{\delta m}$ for $L_1$. As a function of $n$, the size of $C^1_m$ 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 $\{C^1_m\}$. 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 $\mathsf{E}$-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 2.5 is close to best possible, as shown by the following easy result.

Proposition 2.6. *$\mathsf{E}$ has $\mathsf{E}$-uniform circuits of size at most $n2^n$.*

Proof. For any language $L$ in $\mathsf{E}$, 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 $L$. □

## 3. A uniformization lemma for $\mathsf{NC}^1$

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, determines whether $F(v) = 1$.* Buss (1987) showed that Formula Eval is complete under $\mathsf{LOGTIME}$-reductions for $\mathsf{NC}^1$.

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 1.4** *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 & Koucký (2010) 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. Recall that Formula Eval can be solved in (logtime uniform) $\mathsf{NC}^1$. 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 $kc/\varepsilon$ 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 $n^{\varepsilon}$ 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 $dkc/\varepsilon$, 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 $n^{\varepsilon}$ 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 1.5** *For* $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly} \iff$ *for all* $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^{\varepsilon}$.

Note that when the direct connection language of the $\mathcal{C}$-circuit of size $O(n^{\varepsilon})$ is provided as advice in the proof of Theorem 3.1, the resulting $\mathcal{C}$-circuits for Formula Evaluation are LOGTIME-uniform (given the advice).

Lemma 1.4 and Corollary 1.5 have consequences for lower bounds as well as algorithms. We first give a consequence for lower bounds, showing that either $\mathsf{TC}^0$ computations cannot be speeded up in general using logarithmic depth and bounded fan-in, or $\mathsf{NC}^1$ does not have non-uniform polynomial-size threshold circuits of bounded depth.

We need a hierarchy theorem for $\mathsf{TC}^0$, which can be shown analogously to Propositions 2.1 and 2.3.

PROPOSITION 3.2. *For every constant $k$ and $d$ and all $\epsilon < 1$, there is a language in $\mathsf{TC}^0$ which cannot be decided by (logtime uniform) $\mathsf{TC}^0$ circuits of size $n^k$ and depth $d$ with $n^\epsilon$ bits of advice.*

PROOF. The result follows using the proof idea of Proposition 2.1, and applying it to the threshold Turing machine characterization of $\mathsf{TC}^0$ (Allender 1999; Parberry & Schnitger 1988). □

THEOREM 3.3. *At least one of the following holds:*

- *For all constants $k$, there is a language in $\mathsf{TC}^0$ which does not have $\mathsf{LOGTIME}$-$\mathsf{uniform}$ 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 $\mathsf{TC}^0$ has $\mathsf{LOGTIME}$ -$\mathsf{uniform}$ circuits of depth $k \log(n)$. We derive a contradiction.

Let $L \in \mathsf{TC}^0$. By the second assumption, $L$ has $\mathsf{LOGTIME}$ -$\mathsf{uniform}$ circuits of depth $k \log(n)$. From the first assumption and using Corollary 1.5 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 $m^\epsilon$ bits of advice on inputs of length $m$. This implies that any language with $\mathsf{LOGTIME}$-$\mathsf{uniform}$ 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 $\mathsf{TC}^0$, this contradicts Proposition 3.2. □

We can also use Lemma 1.4 to derive algorithmic consequences of $\mathsf{NC}^1 \subset \mathsf{ACC}/\mathrm{poly}$. Practically anything computable in subexponential time on $\mathsf{ACC}$ circuits can be extended to $\mathsf{NC}^1$ circuits, under the assumption. For instance:

Corollary 3.4. *If* $\mathsf{NC}^1 \subset \mathsf{ACC}/\mathrm{poly}$ *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 1.4 to generate an equivalent $n^{O(c/\delta)}$ size $\mathsf{ACC}$ circuit of depth $O(1/\delta)$, in $2^{O(n^{\delta})}$ time, for some $\delta < 1$. Satisfiability of the $\mathsf{ACC}$ circuit can be determined in $2^{n-n^{\varepsilon}}$ time via an $\mathsf{ACC}$-SAT algorithm (Williams 2011). □

**3.1. Derandomizing $\mathsf{TC}^0$ by assuming randomized $\mathsf{TC}^0$ is powerful.** Next, we prove that if $\mathsf{NC}^1$ can be simulated in randomized $\mathsf{TC}^0$, then we can derive a non-trivial *deterministic* $\mathsf{TC}^0$ simulation of $\mathsf{NC}^1$ as well.

**Reminder of Theorem 1.6** *Suppose* $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. *Then for every $\varepsilon > 0$ and every language $L$ in* $\mathsf{NC}^1$, *there is a (*$\mathsf{LOGTIME}$ *uniform)* $\mathsf{TC}^0$ *circuit family of polynomial size recognizing a language $L'$ such that $L$ and $L'$ differ on at most $2^{n^{\varepsilon}}$ inputs of length $n$, for all $n$.*

Proof. Assume $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. It follows that $\mathsf{NC}^1 \subset \mathsf{TC}^0/\mathrm{poly}$, and therefore by the arguments of Lemma 1.4, by providing $n^{\varepsilon}$ advice (namely a small $\mathsf{TC}^0$ circuit for evaluating arbitrary formulas of size $n^{\varepsilon/k}$) we can translate any $n^c$ size $\mathsf{NC}^1$ circuit into a $n^{O(c)}$ size $\mathsf{TC}^0$ circuit of depth $O(1/\varepsilon)$, in polynomial time.

However, the assumption that $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$ yields more: rather than $n^{\varepsilon}$ bits of *non-uniform* advice, the inclusion provides a LOGTIME-uniform $\mathsf{TC}^0$ circuit $C(F, x, r)$ of size $n^{\varepsilon}$, which takes a formula $F$ of size $n^{\varepsilon/k}$, an input $x$ of size $n^{\varepsilon/k}$ and at most $n^{\varepsilon}$ bits of randomness $r$ as input, such that for every $F$ and $x$,

$$\Pr_{r\in\{0,1\}^{n^{\varepsilon}}}[C(F, x, r) = F(x)] > 3/4.$$

We first need to amplify the success probability of the circuit $C$. Let $t$ be a parameter and define a new $\mathsf{TC}^0$ circuit $C'(F, x, r_1, \ldots, r_t)$ which takes the MAJORITY of $C(F, r_i)$ for $i = 1, \ldots t$. By standard probabilistic arguments, for every $F$ and $x$ we have

$$\Pr_{r_1,\ldots,r_t \in \{0,1\}^{n^\varepsilon}}[C'(F, x, r_1, \ldots, r_t) = F(x)] > 1 - 2^{\Omega(t)}.$$

Choose $t = d \cdot n^\varepsilon$ for sufficiently large $d$, so that the probability of agreement is greater than $1 - 1/2^{3n^\varepsilon}$. Then, by a union bound, we have that random choices of $r_1, \ldots, r_t$ are simultaneously good for *all* $F$ and $x$ of at most $n^{\varepsilon/k}$ size, i.e.,

$$\Pr_{r_i \in \{0,1\}^{n^\varepsilon}}[(\forall\ F, x)\ C'(F, x, r_1, \ldots, r_t) = F(x)] > 1 - 2^{n^\varepsilon}.$$

Now, for a given $\mathsf{NC}^1$ circuit $N$ of size $n^c$, after converting $N$ into a formula $F_N$ of size $S = n^{O(c)}$, there will be at most $2S/n^{\varepsilon/k}$ subformulas of $F_N$, each of $n^{\varepsilon/k}$ size, which need to be accurately modeled by the $\mathsf{TC}^0$ circuit $C'$. Replace each $n^{\varepsilon/k}$-size subformula $F'(x')$ of $F_N$ with $C'(F', x', r_1, \ldots, r_t)$, and let $D$ be the $\mathsf{TC}^0$ circuit that results from this replacement. Since $C'$ succeeds against *all* formulas $F$ and inputs $x$ of size at most $n^{\varepsilon/k}$ with high probability, we conclude that in fact our circuit $D$ works for all inputs (of length $n$) with high probability, i.e.,

$$\Pr_{r_1,\ldots,r_t \in \{0,1\}^{n^\varepsilon}}[(\forall\ x \text{ of length } n)\ D(x, r_1, \ldots, r_t) = N(x)] > 1 - 2^{n^\varepsilon}.$$

Therefore, using only $d \cdot n^{2\varepsilon}$ random bits $r_1, \ldots, r_t$, we can efficiently construct a $\mathsf{TC}^0$ circuit $D$ that agrees with $N$ on a given input. That is, we are in a situation where short, sublinear-length, and randomly chosen "advice" (chosen *prior* to receiving the input $x$) succeeds against all inputs $x$ simultaneously, with high probability.

This is precisely the situation described in a paper of (Goldreich & Wigderson 2002) who prove that in such situations, one can generically provide, for every $\varepsilon > 0$, a *deterministic* simulation which is successful on all but $2^{n^\varepsilon}$ inputs of length $n$, by extracting additional randomness from the input itself. In more detail, say that a function $E_n : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ is a *$k$-extractor* if, for every random variable $X$ over $\{0,1\}^n$ that puts probability mass at most $1/2^k$ on all inputs,[5] the random variable $E(X, U_m)$ has statistical distance at most $1/10$ from $U_\ell$ (where $U_q$ denotes the uniform distribution on $\{0,1\}^q$). Given a randomized algorithm $R$ using $\ell$ bits of randomness, and an extractor $E_n$ which takes $n$ bits and $e \log n$ bits and outputs $\ell$ bits, Goldreich and Wigderson's deterministic simulation of $R$ is simply:

> Given an input $x$ of length $n$, output the majority value of $R(x, E_n(x, r))$ over all binary strings $r$ of length $e \log n$.

Goldreich & Wigderson (2002, Theorem 3) prove that, when we use a family of functions

$$\{E_n : \{0,1\}^n \times \{0,1\}^{e \log n} \to \{0,1\}^{\ell(n)}\}_n$$

such that $E_n$ is a $k(n)$-extractor for all $n$, the above algorithm agrees with $R$ on all but $2^{k(n)}$ of the $n$-bit inputs. To complete the proof, it suffices for us to exhibit an extractor family $\{E_n\}$ that is computable in $\mathsf{LOGTIME}$-uniform $\mathsf{TC}^0$ and has $k(n) = \ell(n)^c$ for a fixed constant $c$. Then, our final uniform $\mathsf{TC}^0$ simulation of $C$ will compute the MAJORITY value of $D(x, E_{|x|}(x, r))$ over all $O(n^e)$ random seeds $r$. In our case, the amount of randomness needed can be made $\ell(n) = n^\varepsilon$ for any desired $\varepsilon > 0$, so this simulation will err on at most $2^{k(n)} \leq 2^{n^{c\varepsilon}}$ inputs for a fixed $c$ and arbitrarily small $\varepsilon > 0$.

Finally, we observe that such extractors do exist: by using Theorem 4.6 in Viola (2005), the extractors corresponding to Impagliazzo-Wigderson pseudorandom generators provided by Theorem 5 in Trevisan (2001) are computable in $\mathsf{LOGTIME}$-uniform $\mathsf{TC}^0$. □

**3.2. Very weak derandomization for $\mathsf{TC}^0$ lower bounds.** Lemma 1.4 also has bearing on the emerging connections between circuit satisfiability algorithms and circuit lower bounds. We can

[5] More formally, for all $x \in \{0,1\}^n$, $\Pr[X = x] \leq 1/2^k$.

give a much simpler proof that faster $\mathsf{TC}^0$ SAT algorithms imply $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$ (originally proved in Williams 2011 with a more involved argument):

Theorem 3.5. *Suppose for all $k$, there is an $O(2^n/n^{10})$ time algorithm for solving satisfiability of $\mathsf{TC}^0$ circuits with $n$ inputs, $n^k$ size, and depth $k$. Then $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$.*

Proof. In their work on succinct PCPs, Ben-Sasson *et al.* (2005) also gave a very efficient proof of the Cook-Levin theorem, showing that for any $L \in \mathsf{NTIME}[2^n]$ and instance $x$ of length $n$, one can generate (in poly($n$) time) an $\mathsf{NC}^1$ circuit $C_x$ with $n + c\log n$ inputs and $n^c$ size for a universal $c < 10$, such that $x \in L$ if and only if the truth table of $C_x$ encodes a satisfiable constraint satisfaction problem, where each constraint has $O(1)$ variables.

Williams (2010) (Theorem 3.4) applies this result to prove that if $\mathsf{NC}^1$ circuit satisfiability on all $n$-input $n^k$-size circuits is solvable in $O(2^n/n^{10})$ time (for all $k$), then $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$. In brief, the proof assumes that the circuit SAT algorithm exists and that $\mathsf{NEXP} \subset \mathsf{NC}^1/\mathrm{poly}$, and uses the two assumptions to nondeterministically simulate an arbitrary $L \in \mathsf{NTIME}[2^n]$ in nondeterministic $o(2^n)$ time (a contradiction to the nondeterministic time hierarchy).[6] This simulation of an arbitrary $L$ can be done by constructing the aforementioned $\mathsf{NC}^1$ circuit $C_x$ on the input $x$, guessing an $\mathsf{NC}^1$ circuit $C'$ that encodes a satisfying assignment (i.e., a witness) to the constraint satisfaction problem, then composing $C_x$ and $C'$ to form an $\mathsf{NC}^1$ circuit $D$ which is unsatisfiable if and only if the truth table of $C'$ is a satisfying assignment. An $\mathsf{NC}^1$ circuit SAT algorithm that takes $O(2^{n+c\log n}/(n + c\log n)^{10}) = o(2^n)$ time results in a contradiction.

Assume now that $\mathsf{NEXP} \subset \mathsf{TC}^0/\mathrm{poly}$ and we have an algorithm $A$ for $\mathsf{TC}^0$ circuit satisfiability according to the hypothesis of the theorem. We wish to derive a contradiction. The first assumption,

[6] Technically speaking, Theorem 3.4 in Williams (2010) only shows the consequence $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{NC}^1/\mathrm{poly}$, but this can be easily improved to $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$, by extending work of Impagliazzo *et al.* (2002) to show that $\mathsf{NEXP} \subset \mathsf{NC}^1/\mathrm{poly}$ implies every problem in $\mathsf{NEXP}$ has witnesses that can be encoded as truth tables of $\mathsf{NC}^1$ circuits.

along with Lemma 1.4, implies there is a deterministic $2^{O(n^{\varepsilon})}$-time algorithm $B_{\varepsilon}$ which, given an arbitrary $\mathsf{NC}^1$ circuit $D$, can generate an equivalent $\mathsf{TC}^0$ circuit $E$ that is only polynomially larger than $D$, where the degree of this polynomial is linear in $1/\varepsilon$. Therefore, we can solve $\mathsf{NC}^1$ circuit SAT in $O(2^n/n^{10})$ time as well, by applying the algorithm $B_{\varepsilon}$ to convert a given $\mathsf{NC}^1$ circuit into $\mathsf{TC}^0$ for some $\varepsilon < 1$, then applying algorithm $A$ for $\mathsf{TC}^0$ circuit SAT. By the previous paragraph, this implies that $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\text{poly}$, a contradiction (as $\mathsf{TC}^0$ is contained in $\mathsf{NC}^1$). □

Recently, Jahanjou *et al.* (2013) showed that the result of Ben-Sasson *et al.* cited above can be implemented with the $\mathsf{NC}^1$ circuits replaced by $\mathsf{AC}^0$ *circuits*. This stronger result also implies Theorem 3.5.

Using succinct PCPs instead of a succinct Cook-Levin reduction, we can also show that very weak derandomization of $\mathsf{TC}^0$ suffices for such lower bounds. For $\mathcal{C} \in \{\mathsf{P}, \mathsf{NC}^1, \mathsf{TC}^0\}$, define Derandomize-$\mathcal{C}$ to be the problem:

> *Given a* $\mathcal{C}/\text{poly}$ *circuit* $C$*, output* yes *when* $C$ *is unsatisfiable and* no *when* $C$ *has at least* $2^{n-2}$ *satisfying assignments, with arbitrary behavior otherwise.*

(We choose this version of the problem rather than the version where $C$ has either a high fraction of satisfying assignments or a small fraction, as that problem is in $\mathsf{PromiseBPP}$ while the problem is "merely" in $\mathsf{PromiseRP}$.)

We say that a nondeterministic algorithm $A$ *solves* Derandomize-$\mathcal{C}$ if for every infinite family of circuits $\{C\}$ from the class,

- every computation path of $A(C)$ leads to one of three possible final states: *don't know*, *yes*, or *no*,
- at least one path of $A(C)$ does not lead to *don't know*,
- if $C$ has at least $2^{n-2}$ satisfying assignments, then no path of $A(C)$ leads to *yes*, and
- if $C$ is unsatisfiable, then no path of $A(C)$ leads to *no*.

**Reminder of Theorem 1.7** *Suppose for all $k$, there is an $O(2^n/n^k)$ time algorithm for solving* Derandomize-$\mathsf{TC}^0$ *on all* $\mathsf{TC}^0$ *circuits of $n$ inputs, $n^k$ size, and depth $k$. Then* $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\text{poly}$.

Proof. The proof is by contradiction, as in Theorem 3.5. Briefly, we first follow Williams (2010) and apply a succinct version of the PCP theorem proved by Ben-Sasson *et al.* (2005) to argue how to reduce the question of proving $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ to that of solving Derandomize-$\mathsf{P}$ faster. That reduction involves guessing a poly-size circuit $D$ encoding an alleged *witness* for an $\mathsf{NTIME}[2^n]$ computation, then using the succinct PCP and the faster Derandomize-$\mathsf{P}$ algorithm to verify the witness in nondeterministic $o(2^n)$ time. In order to replace $\mathsf{P}$ with $\mathsf{TC}^0$¡, we have to guess $D$ and guess a "helper" $\mathsf{TC}^0$ circuit $D'$ which is intended to give output information on every gate of $D$. Verifying $D'$ is correct will imply that $D$ is correct. We then show how to reduce the problem of verifying $D'$ to calls to Derandomize-$\mathsf{TC}^0$.

The succinct PCP work of Ben-Sasson *et al.* (2005) also shows that for any $L \in \mathsf{NTIME}[2^n]$ and input $x$ of length $n$, one can generate (in poly$(n)$ time) an $\mathsf{NC}^1$ circuit $C_x$ with $n + c\log n$ inputs and $n^c$ size such that:

- if $x \in L$, then the truth table of $C_x$ encodes a satisfiable CSP, where each constraint has poly$(n)$ variables, and

- if $x \notin L$, then the truth table of $C_x$ encodes such a CSP, where every variable assignment satisfies at most 10% of the constraints.

(Contrast this with the Cook-Levin result used in Theorem 3.5, which only distinguishes satisfiable and unsatisfiable instances, but each constraint has only $O(1)$ variables.) This can be applied to show that if Derandomize-$\mathsf{P}$ on all $n$-input $n^k$-size circuits can be solved in $O(2^n/n^{10})$ time, then $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ (Williams 2010). An improvement of Ben-Sasson *et al.*'s construction given by Mie (2009) yields succinct PCPs with only $O(1)$ queries (instead of poly$(n)$). That is, given any $L \in \mathsf{NTIME}[2^n]$ and instance $x$ of length $n$, one can generate (in poly($n$) time) a circuit $C_x$ with $n + c \log n$ inputs and $n^c$ size[7] such that

- if $x \in L$, then the truth table of $C_x$ encodes a satisfiable $k$-CSP (for some constant $k$), and
- if $x \notin L$, then the truth table of $C_x$ encodes a $k$-CSP such that every variable assignment satisfies at most 10% of the constraints.

Say that a circuit is *at most $\rho$-satisfiable* (respectively, at least $\rho$-satisfiable) if it has at most (at least) $\rho \cdot 2^n$ satisfying assignments. Analogously to the proof of Theorem 1.3 in Williams (2010), Mie's PCPs can be used to show that a nondeterministic $o(2^n)$-time algorithm that can distinguish unsatisfiable circuits from circuits which are at least 9/10-satisfiable would imply $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$.

Let us first outline that proof. We construct a new nondeterministic algorithm for $L$, which given an instance $x$ guesses a $|x|^{O(1)}$-size circuit $D$ intended to encode a satisfying assignment $S$ to the $k$-CSP encoded by $C_x$. (Assuming $\mathsf{NEXP} \subset \mathsf{P}/\text{poly}$, such a circuit $D$ exists when $x \in L$.) The algorithm verifies the guessed $D$ by constructing a circuit $C'$ of polynomially larger size which contains copies of $C_x$ and $k$ copies of $D$, such that $C'(i) = 0$ if and only if the $i$th constraint of the $k$-CSP is satisfied by the assignment $S$. (This construction appears in Williams (2010) and is straightforward, so we will not repeat it here.) By properties of the PCP, this circuit $C'$ is either:

- unsatisfiable (i.e., all constraints of the $k$-CSP are satisfied) or
- at least 9/10-satisfiable (i.e., every variable assignment satisfies at most 10% of the constraints).

[7] We believe that the circuit $C_x$ in Mie's construction can be made to have $O(\log n)$ depth as well, which would simplify our argument: we could assume that $C_x$ is $\mathsf{NC}^1$, then apply the strategy of Theorem 3.5. Indeed, an earlier version of this work (Santhanam & Williams 2012) states that such a construction follows from Ben-Sasson *et al.*. However, verifying this is indeed true is quite technically involved, so we leave it as an interesting open problem here and provide an alternative argument. It appears that, very recently, this open problem has been resolved (Ben-Sasson & Viola 2014).

With an $O(2^n/n^{10})$ time algorithm that can distinguish the two cases, $N$ can recognize $L$ in nondeterministic $o(2^n)$ time; this is a contradiction.

We now extend the above argument to show that a slightly faster algorithm for DERANDOMIZE-$\mathsf{TC}^0$ is enough to yield a contradiction to $\mathsf{NEXP} \not\subset \mathsf{TC}^0$/poly. The argument has similarities to those found in Williams (2011), so we shall keep the discussion at a high level, emphasizing those places where the proof differs.

Assuming $\mathsf{P} \subset \mathsf{TC}^0$/poly, the generated circuits $C'$ in the above argument have equivalent $\mathsf{TC}^0$ circuits $D$ of polynomially larger size. Just as above, we will construct a nondeterministic $o(2^n)$ time algorithm $N$ recognizing an arbitrary $L \in \mathsf{NTIME}[2^n]$. The algorithm $N$ begins by guessing a circuit $D$ as described above, constructing the circuit $C'$, then guessing a $\mathsf{TC}^0$ circuit $D'(x, i)$ which prints the output of the $i$th gate of $C'(x)$, where $i$ ranges from 1 to the size of $C'$. (It is easy to see that if $\mathsf{P} \subset \mathsf{TC}^0$/poly, then such a $D'$ also exists.) Next, $N$ uses polynomially many copies of $D'$ to construct a $\mathsf{TC}^0$ circuit $E$ (of poly$(n)$ size) such that $E(x) = 0$ if and only if for all gates $i$ of $C'$, $D'(x, i)$ is *consistent*: that is, if $i_1$ and $i_2$ are the two gate indices whose outputs are the inputs to gate $i$, then $E(x, i)$ outputs 0 if and only if the output value $D'(x, i)$ is consistent with the input values $D'(x, i_1)$ and $D'(x, i_2)$, and the gate type of $i$. (The construction of $E$ is analogous to Lemma 3.1 in Williams 2011.) Letting $i^\star$ be the index of the output gate of $C'$, we have that $E(x) = 0$ implies $D'(x, i^\star) = C'(x)$.

Let $A$ be a nondeterministic algorithm solving DERANDOMIZE-$\mathsf{TC}^0$, as defined earlier (i.e., $A$ outputs *no* on some computation path if a given $\mathsf{TC}^0$ circuit is at least 1/4-satisfiable, and *yes* on some path if the circuit is unsatisfiable, but never outputs *yes* on some path and *no* on another path). Our nondeterministic $N$ runs $A$ on $E$ and *rejects* if $A$ returns *no.* Next, $N$ runs $A$ on the circuit $E'(x) := D'(x, i^\star)$ and *accepts* if and only if $A$ returns *yes.* This concludes the description of $N$.

To see that the algorithm $N$ correctly recognizes $L$, first suppose $x \in L$. Then, there is a circuit $D$ such that the resulting circuit $C'$ is unsatisfiable. If $N$ guesses $D$ and a $\mathsf{TC}^0$ circuit $D'$ consistent with $C'$, then $E$ is unsatisfiable and $A$ returns *yes* on

$E$. The circuit $E'$ must be unsatisfiable as well, so the nondeterministic $A$ returns *yes* on $E'$ and $N$ accepts on some path.

For the other case, suppose $x \notin L$. Then regardless of the $D$ that is guessed, the resulting circuit $C'$ is at least 9/10-satisfiable. $N$ guesses some $\mathsf{TC}^0$ circuit $D'$ and constructs $E$. If $A$ returns *no* on $E$, then the computation path rejects. Otherwise, if $A$ returns *yes* on $E$, then by assumption, $E$ is at most 1/4-satisfiable, so we have $E(x) = 1$ on at most 1/4 of all possible $x$. Therefore, $C'(x) = D'(x, i^\star)$ on at least 3/4 of all $x$. In the worst case, $D'(x, i^\star) = 1$ on at least $(9/10 - 1/4)$ of the possible inputs $x$. Therefore, $E'$ is at least $(9/10 - 1/4)$-satisfiable, the algorithm $A$ returns *no* on $E'$, and $N$ rejects on these computation paths as well. □

## Acknowledgements

We thank Eli Ben-Sasson for useful discussions on the nice properties of known succinct PCPs. R.S. was supported in part by the 2013 ERC Consolidator Grant ALUnif: "Algorithms and Lower Bounds: A Unified Approach." R.W. was supported in part by a David Morgenthaler II Faculty Fellowship at Stanford, and NSF CCF-1212372. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

## References

Eric Allender (1999). The Permanent Requires Large Uniform Threshold Circuits. *Chicago Journal of Theoretical Computer Science.*

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

Sanjeev Arora & Boaz Barak (2009). *Computational Complexity - A Modern Approach.* Cambridge University Press.

David Barrington, Neil Immerman & Howard Straubing (1990). On Uniformity within $NC^1$. *Journal of Computer and System Sciences* **41**.

Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan & Salil Vadhan (2005). Short PCPs verifiable in polylogarithmic time. In *Proceedings of Twentieth Annual IEEE Conference on Computational Complexity*, 120–134.

Eli Ben-Sasson & Emanuele Viola (2014). Short PCPs with projection queries. *Electronic Colloquium on Computational Complexity (ECCC)* **21**, 17.

Joshua Buresh-Oppenheim & Rahul Santhanam (2006). Making Hard Problems Harder. In *Proceedings of 21st Annual IEEE Conference on Computational Complexity*, 73–87.

Samuel R. Buss (1987). The Boolean formula value problem is in ALOGTIME. In *STOC*, 123–131.

Patrick W. Dymond & Martin Tompa (1985). Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences* **30**(2), 149–161.

Lance Fortnow, Rahul Santhanam & Ryan Williams (2009). Fixed Polynomial Size Circuit Bounds. In *Proceedings of 24th Annual IEEE Conference on Computational Complexity*, 19–26.

Oded Goldreich & Avi Wigderson (2002). Derandomization that is Rarely Wrong from Short Advice that Is Typically Good. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, 209–223.

Juris Hartmanis & Richard Stearns (1965). On the Computational Complexity of Algorithms. *Trans. Amer. Math. Soc. (AMS)* **117**, 285–306.

Johan Håstad (1998). The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM Journal on Computing* **27**(1), 48–64.

Frederick Hennie & Richard Stearns (1966). Two-Tape Simulation of Multitape Turing Machines. *Journal of the ACM* **13**(4), 533–546.

Russell Impagliazzo, Valentine Kabanets & Avi Wigderson (2002). In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences* **65**(4), 672–694.

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

Hamidreza Jahanjou, Eric Miles & Emanuele Viola (2013). Local reductions. *CoRR* **abs/1311.3171**.

Ravi Kannan (1982). Circuit-Size Lower Bounds and Non-Reducibility to Sparse Sets. *Information and Control* **55**(1), 40–56.

Thilo Mie (2009). Short PCPPs verifiable in polylogarithmic time with O(1) queries. *Annals of Mathematics and Artificial Intelligence* **56**(3–4), 313–338.

Eduard Nečiporuk (1966). On a Boolean Function. *Doklady of the Academy of the USSR* **169**(4), 765–766.

Ian Parberry & Georg Schnitger (1988). Parallel Computation with Threshold Functions. *Journal of Computer and System Sciences* **36**(3), 278–302.

Walter Ruzzo (1981). On Uniform Circuit Complexity. *Journal of Computer and System Sciences* **22**(3), 365–383.

Rahul Santhanam & Ryan Williams (2012). Uniform Circuits, Lower Bounds, and QBF Algorithms. *Electronic Colloquium on Computational Complexity (ECCC)* **19**, 59.

Luca Trevisan (2001). Extractors and pseudorandom generators. *Journal of the ACM* **48**(4), 860–879. URL http://doi.acm.org/10.1145/502090.502099.

Emanuele Viola (2005). The Complexity of Constructing Pseudorandom Generators from Hard Functions. *Computational Complexity* **13**(3), 147–188.

Ryan Williams (2010). Improving Exhaustive Search Implies Superpolynomial Lower Bounds. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, 231–240.

Ryan Williams (2011). Non-uniform ACC Circuit Lower Bounds. In *Proceedings of 26th Annual IEEE Conference on Computational Complexity*, 115–125.

Manuscript received 24 June 2013

Rahul Santhanam
School of Informatics,
University of Edinburgh,
Edinburgh, UK.
rsanthan@inf.ed.ac.uk

Ryan Williams
Computer Science Department,
Stanford University,
Stanford, CA, USA.
rrwilliams@gmail.com