Electronic Colloquium on Computational Complexity, Report No. 7 (2012)

Lower Bounds against Weakly Uniform Circuits

Ruiwen Chen* Valentine Kabanets†

January 28, 2012

Abstract

A family of Boolean circuits $\{C_n\}_{n \geqslant 0}$ is called $\gamma(n)$-*weakly uniform* if there is a polynomial-time algorithm for deciding the direct-connection language of every C_n, given *advice* of size $\gamma(n)$. This is a relaxation of the usual notion of uniformity, which allows one to interpolate between complete uniformity (when $\gamma(n) = 0$) and complete non-uniformity (when $\gamma(n) > |C_n|$). Weak uniformity is essentially equivalent to *succinctness* introduced by Jansen and Santhanam [JS11].

Our main result is that PERMANENT is not computable by polynomial-size $n^{o(1)}$-weakly uniform TC^0 circuits. This strengthens the results by Allender [All99] (for *uniform* TC^0) and by Jansen and Santhanam [JS11] (for weakly uniform *arithmetic* circuits of constant depth). Our approach is quite general, and can be used to extend to the "weakly uniform" setting all currently known circuit lower bounds proved for the "uniform" setting. For example, we show that PERMANENT is not computable by polynomial-size $(\log n)^{O(1)}$-weakly uniform threshold circuits of depth $o(\log \log n)$, generalizing the result by Koiran and Perifel [KP09].

1 Introduction

Understanding the power and limitation of efficient algorithms is the major goal of complexity theory, with the "P vs. NP" problem being the most famous open question in the area. While proving that no NP-complete problem has a uniform polynomial-time algorithm would suffice for separating P and NP, a considerable amount of effort was put into the more ambitious goal of trying to show that no NP-complete problem can be decided by even a *nonuniform* family of polynomial-size Boolean circuits.

More generally, an important goal in complexity theory has been to prove strong (exponential or super-polynomial) circuit lower bounds for "natural" computational problems that may come from complexity classes larger than NP, e.g., the class NEXP of languages decidable in nondeterministic exponential time. By the counting argument of Shannon [Sha49], a randomly chosen n-variate Boolean function requires circuits of exponential size. However, the best currently known circuit lower bounds for *explicit* problems are only linear for NP problems [LR01, IM02], and polynomial for problems in the polynomial-time hierarchy PH [Kan82].

To make progress, researchers introduced various restrictions on the circuit classes. In particular, for Boolean circuits of *constant* depth, with NOT and unbounded fan-in AND and OR gates (AC^0 circuits), exponential lower bounds are known for the PARITY function [FSS84, Yao85, Hås86]. For constant-depth circuits that additionally have (unbounded fan-in) MOD_p gates, one also needs

*School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada; ruiwenc@sfu.ca
†School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada; kabanets@cs.sfu.ca

ISSN 1433-8092

exponential size to compute the MOD_q function, for any two distinct primes p and q [Raz87, Smo87]. With little progress for decades, Williams [Wil11] has recently shown that a problem in NEXP is not computable by polynomial-size ACC^0 circuits, which are constant-depth circuits with NOT gates and unbounded fan-in AND, OR and MOD_m gates, for any integer $m > 1$. However, no lower bounds are known for the class TC^0 of constant-depth threshold circuits with unbounded fan-in majority gates.[1]

To make more progress, another restriction has been added: *uniformity* of circuits. Roughly speaking, a circuit family is called uniform if there is an efficient algorithm that can construct any circuit from the family. There are two natural variations of this idea. One can ask for an algorithm that outputs the entire circuit in time polynomial in the circuit size; this notion of uniformity is known as P-uniformity. In the more restricted notion, one asks for an algorithm that describes the local structure of the circuit: given two gate names, such an algorithm determines if one gate is the input to the other gate, as well as determines the types of the gates, in time linear (or polynomial) in the input size (which is logarithmic or polylogarithmic time in the size of the circuit described by the algorithm); such an algorithm is said to decide the *direct-connection language* of the given circuit. This restricted notion is called DLOGTIME-(or POLYLOGTIME-)uniformity [Ruz81, AG94]. We will use the notion of POLYLOGTIME-uniformity by default, and, for brevity, will omit the word POLYLOGTIME.

It is easy to show (by diagonalization) that, for any fixed exponential function $s(n) = 2^{n^c}$ for a constant $c \geqslant 1$, there is a language in EXP (deterministic exponential time) that is not computable by a uniform (even P-uniform) family of Boolean $s(n)$-size circuits.[2] Similarly, as observed in [All99], a PSPACE-complete language requires exponential-size uniform TC^0 circuits. For the smaller complexity class $\#\mathsf{P} \subseteq \mathsf{PSPACE}$, Allender and Gore [AG94] showed PERMANENT (which is complete for #P [Val79]) is not computable by uniform ACC^0 circuits of sub-exponential size. Later, Allender [All99] proved that PERMANENT cannot be computed by uniform TC^0 circuits of size $s(n)$ for any function s such that, for all k, $s^{(k)}(n) = o(2^n)$ (where $s^{(k)}$ means the function s composed with itself k times). Finally, Koiran and Perifel [KP09] extended this result to show that PERMANENT is not computed by polynomial-size uniform threshold circuits of depth $o(\log \log n)$.

Recently, Jansen and Santhanam [JS11] have proposed a natural relaxation of uniformity, termed *succinctness*, which allows one to interpolate between non-uniformity and uniformity. According to [JS11], a family of $s(n)$-size circuits $\{C_n\}$ is succinct if the direct-connection language of C_n is decided by some circuit of size $s(n)^{o(1)}$. In other words, while there may not be an efficient algorithm for describing the local structure of a given $s(n)$-size circuit C_n, the local structure of C_n can be described by a *non-uniform* circuit of size $s(n)^{o(1)}$. Note that if we allow the non-uniform circuit to be of size $s(n)$, then the family of circuits $\{C_n\}$ would be completely non-uniform. So, intuitively, the restriction to the size $s(n)^{o(1)}$ makes the notion of succinctness close to that of non-uniformity.

The main result of [JS11] is that PERMANENT does not have succinct polynomial-size *arithmetic* circuits of constant depth, where arithmetic circuits have unbounded fan-in addition and multiplication gates and operate over integers. While relaxing the notion of uniformity, [JS11] were only able to prove a lower bound for the *weaker* circuit class, as polynomial-size constant-depth arith-

[1] A plausible explanation of this "barrier" is given by the "natural proofs" framework of [RR97], who argue it is hard to prove lower bounds against the circuit classes that are powerful enough to implement cryptography.

[2] Unlike the nonuniform setting, where every n-variate Boolean function is computable by a circuit of size about $2^n/n$ [Lup58], *uniform* circuit lower bounds can be bigger than 2^n.

metic circuits can be simulated by polynomial-size TC^0 circuits. A natural next step was to prove a super-polynomial lower bound for PERMANENT against succinct TC^0 circuits. This is achieved in the present paper.

1.1 Our main results

We improve upon [JS11] by showing that PERMANENT does not have succinct polynomial-size TC^0 circuits. In addition to strengthening the main result from [JS11], we also give a simpler proof. Our argument is quite general and allows us to extend to the "succinct" setting all previously known uniform circuit lower bounds of [AG94, All99, KP09].

Recall that the direct-connection language for a circuit describes the local structure of the circuit; more precise definitions will be given in the next section. For a function $\alpha : \mathbb{N} \to \mathbb{N}$, we say that a circuit family $\{C_n\}$ of size $s(n)$ is *α-weakly uniform* if the direct-connection language L_{dc} of $\{C_n\}$ is decided by a polynomial-time algorithm that, in addition to the input of L_{dc} of size $m \in O(\log s(n))$, has an advice string of size $\alpha(m)$; the advice string just depends on the input size m. The notion of α-weakly uniform is essentially equivalent to the notion of α-succinct introduced in [JS11]; see the next section for more details.

We will call a circuit family subexp-*weakly uniform* if it is α-weakly uniform for $\alpha(m) \in 2^{o(m)}$. Similarly, we call a circuit family poly-*weakly uniform* if it is α-weakly uniform for $\alpha(m) \in m^{O(1)}$. Observe that for $m = O(\log s)$, we have $2^{o(m)} = s^{o(1)}$ and $m^{O(1)} = \mathsf{poly} \log s$.

Our main results are the following. First, we strengthen the lower bound of [JS11].

Theorem 1.1. PERMANENT *is not computable by* subexp-*weakly uniform* poly-*size* TC^0 *circuits.*

Let us call a function $s(n)$ *sub-subexponential* if, for any constant $k > 0$, we have that the k-wise composition $s^{(k)}(n) \leqslant 2^{n^{o(1)}}$. We use $\mathsf{subsubexp}$ to denote the class of all sub-subexponential functions $s(n)$. We extend a result of [All99] to the "weakly-uniform" setting.

Theorem 1.2. PERMANENT *is not computable by* poly-*weakly uniform* $\mathsf{subsubexp}$-*size* TC^0 *circuits.*

Finally, we extend the result of [KP09].

Theorem 1.3. PERMANENT *is not computable by* poly-*weakly uniform* poly-*size threshold circuits of depth* $o(\log \log n)$.

Table 1 below summarizes these results, and refers to the statements later in the paper where these results are proved.

Table 1: Lower bounds for PERMANENT against α-weakly uniform threshold circuits.

$\alpha(m) = \alpha(O(\log s(n)))$	Depth $d(n)$	Size $s(n)$	Theorem
$2^{o(m)} = n^{o(1)}$	$O(1)$	$n^{O(1)}$	Theorem 4.6
$m^{O(1)} = (\log s(n))^{O(1)} \leqslant n^{o(1)}$	$O(1)$	$s^{(k)}(n) \leqslant 2^{n^{o(1)}}$	Theorem 4.8
$m^{O(1)} = (\log n)^{O(1)}$	$o(\log \log n)$	$n^{O(1)}$	Theorem 4.10

1.2 Our techniques

At the high level, we use the method of *indirect diagonalization*:

- assuming PERMANENT is easy and using diagonalization, we first show the existence of a "hard" language in a certain complexity class $\mathcal{C}$ (the counting hierarchy, to be defined below);
- assuming PERMANENT is easy, we show that the above "hard" language is actually "easy" (as the easiness of PERMANENT collapses the counting hierarchy), which is a contradiction.

In more detail, we first extend the well-known correspondence between uniform TC^0 and alternating polylog-time Turing machines (that use majority states) to the weakly uniform setting, by considering alternating Turing machines with *advice*. To construct the desired "hard" language, we use diagonalization against such alternating Turing machines with advice. The assumed easiness of PERMANENT is used to argue two things about the constructed "hard" language L_{hard}:

1. L_{hard} is in fact "hard" for a much more powerful class $\mathcal{A}$ of algorithms;
2. L_{hard} is decided by a "simple" algorithm A.

The contradiction ensues since the algorithm A turns out to be from the class $\mathcal{A}$.

1.3 Relation to the previous work

A similar indirect-diagonalization strategy was used (explicitly or implicitly) in all previous papers showing uniform or weakly uniform circuit lower bounds for PERMANENT [AG94, All99, KP09, JS11]. Our approach is most closely related to that of [All99, KP09]. The main difference is that we work in the weakly uniform setting, which means that we need to handle a certain amount of non-uniform advice. To that end, we have adapted the method of indirect diagonalization, making it modular (as outlined above) and sufficiently general to work also in the setting with advice. Due to this generality of our proof argument, we are able to extend the afore-mentioned lower bounds from the uniform setting to the weakly uniform setting.

The approach adopted by [JS11] goes via the well-known connection between derandomization and circuit lower bounds (cf. [HS82, KI04, Agr05]). Since the authors of [JS11] work with the algebraic problem of Polynomial Identity Testing (given an arithmetic circuit computing some polynomial over integers, decide if the polynomial is identically zero), their final lower bounds are also in the algebraic setting: for weakly uniform arithmetic constant-depth circuits. By making the diagonalization arguments in [JS11] more explicit (along the lines of [All99]), we are able to get the lower bound for weakly uniform Boolean (TC^0) circuits, thereby both strengthening the results and simplifying the proofs from [JS11].

The remainder of the paper. We give the necessary background in Section 2. Section 3 provides the details of our indirect diagonalization method. This set-up is then used in Section 4 to prove our main results (Theorems 1.1–1.3 above). We give other weakly uniform circuit lower bounds in Section 5. We give concluding remarks in Section 6.

2 Preliminaries

For details on the basic complexity notions, we refer to [AB09].

2.1 Circuits

Recall that a *Boolean circuit* C_n on n inputs $x_1, \ldots, x_n$ is a directed acyclic graph with a single output gate (the node of out-degree 0), n nodes of in-degree 0 (input gates labeled $x_1, \ldots, x_n$), and internal nodes of in-degree 2 (for AND and OR gates) or 1 (for NOT gates). The *size* of the circuit C_n is defined to be the number of gates, and is denoted by $|C_n|$. For a function $s : \mathbb{N} \to \mathbb{N}$ and a circuit family $\{C_n\}_{n \geqslant 0}$, we say that the circuit family is in $\mathsf{SIZE}(s)$, if for all sufficiently large n we have $|C_n| \leqslant s(n)$.

The *depth* of a circuit C_n is defined to be the length of a longest path from some input gate to the output gate. We will be talking about constant-depth circuits, in which case we allow all gates (other than the NOT gates) to have unbounded fan-in. In addition to AND and OR, we may have other types of gates: MAJ (which is 1 iff more than half of its inputs are 1), or MOD_m gate for some integer $m > 0$ (which is 1 iff the integer sum of the inputs is divisible by m).

AC^0 circuits are constant-depth Boolean circuits with NOT gates and unbounded fan-in AND and OR gates. ACC^0 circuits are constant-depth Boolean circuits with unbounded fan-in AND, OR and MOD_m gates for some positive integer m. Finally, TC^0 circuits are constant-depth Boolean circuits with unbounded fan-in AND, OR and MAJ (or threshold) gates. For a function $s : \mathbb{N} \to \mathbb{N}$ and a circuit type $\mathcal{C} \in \{\mathsf{AC}^0, \mathsf{ACC}^0, \mathsf{TC}^0\}$, we denote by $\mathcal{C}(s)$ the class of families of $s(n)$-size n-input circuits of type $\mathcal{C}$. When $s(n)$ is a polynomial in n, we may drop it and simply write $\mathcal{C}$ to denote the class of polynomial-size $\mathcal{C}$-circuits. Finally, we drop the superscript 0 in $\mathsf{AC}^0, \mathsf{ACC}^0$, and TC^0, when we want to talk about the corresponding type of circuits where the depth $d(n)$ may be a function of the input size n.

2.2 Weakly uniform circuit families

Following [Ruz81, AG94], we define the *direct connection language* of a circuit family $\{C_n\}$ as

$$L_{dc} = \{(n, g, h) \colon g = h \text{ and } g \text{ is a gate in } C_n, \text{ or } g \neq h \text{ and } h \text{ is an input to } g \text{ in } C_n\},$$

where n is in binary representation, and g and h are binary strings encoding the gate types and names. The *type* of a gate could be constant 0 or 1, Boolean logic gate NOT, AND, or OR, majority gate MAJ, modulo gate MOD_m for some integer m, or input $x_1, x_2, \ldots, x_n$. For a circuit family of size $s(n)$, we need $c_0 \log s(n)$ bits to encode (n, g, h), where c_0 is a constant at most 4.

A circuit family $\{C_n\}$ is *uniform* [BIS90, AG94] if its direct connection language is decidable in time polynomial in its input length $|(n, g, h)|$. This condition was referred to as $\mathsf{POLYLOGTIME}$-uniformity in [AG94]. It is a more relaxed notion than the usual $\mathsf{DLOGTIME}$-uniformity [BIS90], which requires that the direct connection language be decided in linear time.

Following [JS11], for a time-constructible function $\alpha : \mathbb{N} \to \mathbb{N}$, we say that a circuit family $\{C_n\}$ of size $s(n)$ is *α-succinct* if its direct connection language L_{dc} is in $\mathsf{SIZE}(\alpha)$; i.e., L_{dc} has (non-uniform) Boolean circuits of size $\alpha(m)$, where $m = c_0 \log s(n)$ is the input size for L_{dc}. Trivially, for $\alpha(m) \geqslant 2^m$, every circuit family is α-succinct. The notion becomes nontrivial when $\alpha(m) \ll 2^m/m$. We will use $\alpha(m) = 2^{o(m)}$ (slightly succinct) and $\alpha(m) = m^{O(1)}$ (highly succinct).

We recall the definition of Turing machines with advice from [KL82]. Given functions $t \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\alpha \colon \mathbb{N} \to \mathbb{N}$, we say that a language L is in $\mathsf{DTIME}(t)/\alpha$, if there is a deterministic Turing machine M and a sequence of advice strings $\{a_n\}$ of length $\alpha(n)$ such that, for any $x \in \{0,1\}^n$, $M(x, a_n)$ decides whether $x \in L$ in time $t(n, \alpha(n))$. If the function $t(n, m)$ is upper-bounded by a polynomial in $n + m$, we say that $L \in \mathsf{P}/\alpha$.

Definition 2.1. A circuit family $\{C_n\}$ of size $s(n)$ is *α-weakly uniform* if its direct connection language is decided in P/α; recall that the input size for the direct-connection language describing C_n is $m = c_0 \log s(n)$, and so the size of the advice string needed in this case is $\alpha(c_0 \log s(n))$.

The two notions are closely related.

Lemma 2.2. *In the notation above, $\alpha(m)$-succinctness implies $\alpha(m)\log\alpha(m)$-weak uniformity, and conversely, $\alpha(m)$-weak uniformity implies $(\alpha(m)+m)^{O(1)}$-succinctness.*

Proof sketch. A Boolean circuit of size s can be represented by a binary string of size $O(s \log s)$; and a Turing machine running in time t can be simulated by a circuit family of size $O(t \log t)$. $\square$

The notion of weak uniformity (succinctness) interpolates between full uniformity on one end and full non-uniformity on the other end. For example, 0-weak uniformity is the same as uniformity. On the other hand, α-weak uniformity for $\alpha(m) \geqslant 2^m$ is the same as non-uniformity. For that reason, we will assume that the function α in "α-weakly uniform" is such that $0 \leqslant \alpha(m) \leqslant 2^m$.

Definition 2.3. We say a circuit family $\{C_n\}$ is subexp-*weakly uniform* if it is α-weakly uniform for $\alpha(m) \in 2^{o(m)}$; similarly, we say $\{C_n\}$ is poly-*weakly uniform* if it is α-weakly uniform for $\alpha(m) \in m^{O(1)}$.

2.3 Alternating Turing machines

Following [CKS81, PS86, AG94], an *alternating Turing machine (ATM)* is a nondeterministic Turing machine with two kinds of states: universal states and existential states. In the usual definition of an ATM, each configuration has either zero or two successor configurations; configurations with no successors, which are called *leaves*, are halting configurations; a configuration in universal (existential) state is accepting iff all (at least one) of its successors are accepting. We also consider the generalized ATMs where each configuration has an unbounded number of successors, obtained by replacing a subtree of "bounded branching" configurations by a single configuration. We assume an ATM has random access to the input.

A *threshold Turing machine* is an ATM with majority (MAJ) states; a configuration in a *majority state* may have an unbounded number of successors, and it is accepting iff more than half of its successors are accepting. We denote by $\mathsf{Th}_{d(n)}\mathsf{TIME}(t(n))$ the class of languages accepted by threshold Turing machines having at most $d(n)$ alternations and running in time $O(t(n))$. Note that the class $\mathsf{Th}_{d(n)}\mathsf{TIME}(t(n))$ is closed under complement, since the negation of majority is simply the majority of negations.

Recall that a language A is in PP ($\mathsf{C_=P}$) if there is a nondeterministic polynomial-time Turing machine M such that $x \in A$ iff the number of accepting paths of M on input x is greater than (equal to) the number of rejecting paths. The *counting hierarchy*, studied in [Wag86, Tor91], is defined as $\mathsf{CH} = \cup_{d \geqslant 0} \mathsf{CH}_d$, where $\mathsf{CH}_0 = \mathsf{P}$ and $\mathsf{CH}_{d+1} = \mathsf{PP}^{\mathsf{CH}_d}$. This definition is unchanged if we replace PP with $\mathsf{C_=P}$. The counting hierarchy can be equivalently defined via threshold Turing machines: $\mathsf{CH}_d = \mathsf{Th}_d\mathsf{TIME}(n^{O(1)})$.

Alternating Turing machines can be also equipped with modulo states MOD_m for some fixed m; a MOD_m configuration is accepting iff the number of its accepting successors is 0 modulo m. We denote by $\mathsf{Mod}_{d(n)}\mathsf{TIME}(t(n))$ the class of languages decided by ATMs with MOD_m states for some fixed $m > 0$ dependent on the language, making at most $d(n)$ alternations and running in time $O(t(n))$. Following [GKR$^+$95, All99], we denote by ModPH the class $\cup_{d \geqslant 0} \mathsf{Mod}_d\mathsf{TIME}(n^{O(1)})$.

In general, on different inputs, an ATM may follow computation paths with different sequences of alternations; however, by introducing dummy states, it is always possible to transform the machine into an equivalent machine such that all computation paths on inputs of the same size will follow the same sequence of alternations, whereas the number of alternations and the running time will change only by a constant factor; see [AG94] for details.

2.4 Weak uniformity vs. alternating Turing machines with advice

It is well-known that uniform $\mathsf{AC}^0(2^{\mathsf{poly}(n)})$ corresponds to the polynomial-time hierarchy PH [FSS84]. There are similar correspondences between uniform $\mathsf{ACC}^0(2^{\mathsf{poly}(n)})$ and ModPH [GKR$^+$95, AG94], as well as between uniform $\mathsf{TC}^0(2^{\mathsf{poly}(n)})$ and the counting hierarchy CH [GKR$^+$95, AG94]; see Table 2 below for the summary. More precisely, for time constructible $t(n)$ such that $t(n) = \Omega(\log n)$,

$$\cup_{d\geqslant 0}\mathsf{Th}_d\mathsf{TIME}(\mathsf{poly}(t(n))) = \text{uniform } \mathsf{TC}^0(2^{\mathsf{poly}(t(n))}),$$

$$\cup_{d\geqslant 0}\mathsf{Mod}_d\mathsf{TIME}(\mathsf{poly}(t(n))) = \text{uniform } \mathsf{ACC}^0(2^{\mathsf{poly}(t(n))}).$$

Table 2: Correspondence between hierarchies and uniform circuit classes.

Alternation	Hierarchy	Circuits	Reference
$\exists, \forall$	PH	uniform AC^0	[FSS84]
$\exists, \forall, \mathrm{MOD}_2, \mathrm{MOD}_3, \dots$	ModPH	uniform ACC^0	[GKR$^+$95, AG94]
$\exists, \forall, \mathrm{MAJ}$	CH	uniform TC^0	[PS86, BIS90]

The following gives the correspondence between weakly uniform threshold circuits and threshold Turing machines with advice.

Lemma 2.4. *Let L be any language decided by a family of α-weakly uniform $d(n)$-depth threshold circuits of size $s(n)$. Then L is decidable by a threshold Turing machine with $d'(n) = 3d(n) + 2$ alternations, taking advice of length $\alpha(m)$ for $m = c_0 \log s(n)$, and running in time $t(n) = d'(n) \cdot \mathsf{poly}(m + \alpha(m))$.*

Proof. The proof follows directly from [AG94] in which ACC^0 circuits are considered. Let $\{C_n\}$ be the circuit family deciding L. Its direct connection language L_{dc} is accepted by some Turing machine U, on input size $m = c_0 \log s(n)$, taking advice a_m of size $\alpha(m)$ and running in time $\mathsf{poly}(m + \alpha(m))$. We will construct a threshold Turing machine M which takes advice and decides L. For any input x of length n, machine M takes advice $b_n \equiv a_m$, and does the following:

- ($\exists$) guess gate g of C_n, and check that U accepts (n, g, g), i.e., g is a gate in C_n;
- ($\forall$) guess gate h and check that U rejects (n, h, g), i.e., g is the output;
- Call Eval(g), which is a recursive procedure defined below.

The procedure Eval(g) is as follows:

- ($\exists$) If g is an OR gate, then guess its input h; if U rejects (n, g, h) then reject, otherwise call Eval(h).

- (∀) If g is an AND gate, then guess its input h; if U rejects (n, g, h) then accept, otherwise call Eval(h).
- (MAJ) If g is a MAJ gate, then guess its input h and a bit $b \in \{0, 1\}$; if U rejects (n, g, h), then accept when $b = 1$ and reject when $b = 0$, otherwise call Eval(h).
- If g is a constant gate, then accept iff it is 1.
- If g is an input, then accept iff the corresponding input bit is 1.

It is easy to verify that M with advice b_n accepts x iff $C_n(x) = 1$. The number of alternations that M takes on any computation path is at most $d(n)+2$. However, each path may follow a different sequence of states. To resolve this, we replace each state on each path by a sequence of three states $(\exists, \forall, \text{MAJ})$, where two of them are dummy. This gives a machine with each computation path following the same alternations, and the total number of alternations is at most $3d(n) + 2$. The access to inputs is only at the last step of each computation path (corresponding to the bottom level of the circuit).

At each alternation, the machine simulates U and runs in time $\mathsf{poly}(m + \alpha(m))$. Therefore, the total running time is bounded by $d'(n) \cdot \mathsf{poly}(m + \alpha(m))$. □

Similar to Lemma 2.4, we have the following correspondence between weakly uniform ACC circuits and alternating Turing machines with modulo states.

Lemma 2.5. *Let L be any language decided by a family of α-weakly uniform $d(n)$-depth ACC circuits of size $s(n)$ with MOD_r gates, for some integer $r > 0$. Then L is decidable by an alternating Turing machine with MOD_r states and $d'(n) = O(d(n))$ alternations, taking advice of length $\alpha(m)$ where $m = c_0 \log s(n)$, and running in time $d'(n) \cdot \mathsf{poly}(m + \alpha(m))$.*

3 Indirect diagonalization

Here we establish the components needed for our indirect diagonalization, as outlined in the Introduction. First, in Section 3.1, we give a diagonalization argument against alternating Turing machines with advice, getting a language in the counting hierarchy CH that is "hard" against weakly uniform TC^0 circuits of certain size. Then, in Section 3.2, using the assumption that a canonical P-complete problem has small weakly uniform TC^0 circuits, we conclude that the "hard" language given by our diagonalization step is actually hard for a stronger class of algorithms: weakly uniform Boolean circuits of some size s' *without any depth restriction*. Finally, in Section 3.3, using the assumption that PERMANENT has small weakly uniform TC^0 circuits, we show that CH collapses, and our assumed hard language is in fact decidable by weakly uniform s'-size Boolean circuits, which is a contradiction. (Our actual argument is more general: we consider threshold circuits of not necessarily constant depth $d(n)$, and non-constant levels of the counting hierarchy.)

3.1 Diagonalization against alternating Turing machines with advice

Lemma 3.1. *For any time-constructible functions $\alpha, d, t, T : \mathbb{N} \to \mathbb{N}$ such that $t(n) \log t(n) = o(T(n))$ and $\alpha(n) \in o(n)$, there exists a language $D \in \mathsf{Th}_{d(n)}\mathsf{TIME}(T(n))$ which is not decided by threshold Turing machines with $d(n)$ alternation running in time $t(n)$ and taking advice of length $\alpha(n)$.*

Proof. The proof is by diagonalization. Define the language D consisting of those inputs x of length n that have the form $x = (M, y)$ (using some pairing function) such that the threshold TM M with advice y, where $|y| = \alpha(n)$, rejects input (M, y) in time $t(n)$ using at most $d(n)$ alternations. Language D is decided in $\mathsf{Th}_{d(n)}\mathsf{TIME}(T(n))$ by simulating M and flipping the result.

For contradiction, suppose that D is decided by some threshold Turing machine M_0 with $d(n)$ alternations taking advice $\{a_n\}$ of size $\alpha(n)$. Consider the input (M_0, a_n) with $|M_0| = n - \alpha(n)$; we assume that each TM has infinitely many equivalent descriptions (by padding), and so for large enough n, there must exist such a description of size $n - \alpha(n)$. By the definition of D, we have (M_0, a_n) is in D iff M_0 with advice a_n rejects it; but this contradicts the assumption that M_0 with advice $\{a_n\}$ decides D. $\square$

Lemma 3.2. *For any time-constructible functions $\alpha, d, t, T : \mathbb{N} \to \mathbb{N}$ such that $t(n) \log t(n) = o(T(n))$ and $\alpha(n) \in o(n)$ and any integer $m > 1$, there exists a language $D \in \mathsf{Mod}_{d(n)+1}\mathsf{TIME}(T(n))$ which is not decided by alternating Turing machines with MOD$_m$ states and $d(n)$ alternation running in time $t(n)$ and taking advice of length $\alpha(n)$.*

Proof sketch. The proof is similar to the proof of Lemma 3.1, except that when flipping the result, the negation can be simulated by a MOD$_m$ state, using the identity $\neg x = \mathrm{MOD}_m(x)$. $\square$

3.2 If P is easy

Let L_0 be a P-complete language under uniform projections (functions computable by uniform Boolean circuits with NOT gates only, without any AND or OR gates). For example, the standard P-complete set $\{(M, x, 1^t) : M \text{ accepts } x \text{ in time } t\}$ works.

Lemma 3.3. *Suppose L_0 is decided by a family of α-weakly uniform $d(n)$-depth threshold circuits of size $s(n)$. Then, for any time-constructible function $t(n) \geqslant n$ and $0 \leqslant \beta(m) \leqslant 2^m$, every language L in β-weakly uniform $\mathsf{SIZE}(t(n))$ is decided by $\mu(n)$-weakly uniform $d(\mathsf{poly}(t(n)))$-depth threshold circuits of size $s'(n) = s(\mathsf{poly}(t(n)))$ on n inputs, where $\mu(n) = \alpha(c_0 \log s'(n)) + \beta(c_0 \log t(n))$.*

Proof. Let U be an advice-taking algorithm deciding the direct-connection language for the $t(n)$-size circuits for L. For any string y of length $\beta(m)$ for $m = c_0 \log t(n)$, we can run U with the advice y to construct some circuit C^y of size $t(n)$ on n inputs. We can construct the circuit C^y in time at most $\mathsf{poly}(t(n))$, and then evaluate it in time $O(t(n))$ on any given input of size n.

Consider the language $L' = \{(x, y, 1^{t(n)}) \mid |x| = n,\ |y| = \beta(m),\ C^y(x) = 1\}$. By the above, we have $L' \in \mathsf{P}$. Hence, by assumption, L' is decided by an α-weakly uniform $d(l)$-depth threshold circuits of size $s(l)$, where $l = |(x, y, 1^{t(n)})| \leqslant \mathsf{poly}(t(n))$. To get a circuit for L, we simply use as y the advice of size $\beta(m)$ needed for the direct-connection language of the $t(n)$-size circuits for L. Overall, we need $\alpha(c_0 \log s(l)) + \beta(m)$ amount of advice to decide L by weakly uniform $d(\mathsf{poly}(t(n)))$-depth threshold circuits of size $s(\mathsf{poly}(t(n)))$. $\square$

3.3 If Permanent is easy

Since Permanent is hard for the first level of the counting hierarchy CH, assuming that Permanent is "easy" implies the collapse of CH (see, e.g., [All99]). It was observed in [KP09] that is also possible to collapse super-constant levels of CH, under the same assumption. Below we argue the collapse of super-constant levels of CH under the assumption that Permanent has "small" weakly uniform circuits.

We use the notation $f \circ g$ to denote the composition of the functions f and g, and the notation $f^{(i)}$ is used to denote the composition of f with itself for i times; we use the convention that $f^{(0)}$ is the identity function.

Lemma 3.4. *Suppose that* Permanent *is in γ-weakly uniform $\mathsf{SIZE}(s(n))$, for some $\gamma(m) \leqslant 2^{o(m)}$. For every $d(n) \leqslant n^{o(1)}$, every language A in $\mathsf{Th}_{d(n)}\mathsf{TIME}(\mathsf{poly})$ is also in $(2d(n) \cdot \gamma)$-weakly uniform $\mathsf{SIZE}((s \circ q)^{(d(n)+1)}(n))$, for some polynomial q dependent on A.*

Proof. Consider an arbitrary n. Let $d = d(n)$. The language A restricted to inputs of size n has a threshold circuit C of depth d and size $2^{\mathsf{poly}(n)}$ such that the direct-connection language of C is decided by a polynomial-time Turing machine M (where M is determined by the language A). More precisely, we identify the gates of the circuit with the configurations of the given threshold TM for A; the output gate is the initial configuration; leaf (input) gates are halting configurations; deciding if one gate is an input to the other gate is deciding if one configuration follows from the other according to our threshold TM, and so can be done in polynomial time (dependent on A); finally, given a halting configuration, we can decide if it is accepting or rejecting also in polynomial time (dependent on A).

For a gate g of C, we denote by C_g the subcircuit of C that determines the value of the gate g. We say that g is at depth i, for $1 \leqslant i \leqslant d$, if the circuit C_g is of depth i. Note that each gate at depth $i \geqslant 1$ is a majority gate.

For every $0 \leqslant i \leqslant d$, let B_i be a circuit that, given $x \in \{0,1\}^n$ and a gate g at depth i, outputs the value $C_g(x)$.

Claim 3.5. *There are polynomials q and q' dependent on A such that, for each $0 \leqslant i \leqslant d$, there are $2i\gamma$-weakly uniform circuits B_i of size $(s \circ q)^{(i)} \circ q'$.*

Proof. We argue by induction on i. For $i = 0$, to compute $B_0(x, g)$, we need to decide if the halting configuration g of our threshold TM for A on input x is accepting or not; by definition, this can be done by the TM M in deterministic polynomial time. Hence, B_0 can be decided by a completely uniform circuit of size at most $q'(n)$ for some polynomial q' dependent on the running time of M.

Assume that we have the claim for i. Let s' be the size of the γ'-weakly uniform circuit B_i, where $s' \leqslant (s \circ q)^{(i)} \circ q'$ and $\gamma' \leqslant 2i\gamma$. Consider the following Turing machine N:

> "On input $z = (x, g, U, y, 1^{s'/2})$, where $|x| = n$, g is a gate of C, $|U| = \gamma(c_0 \log s')$, $|y| = \gamma'(c_0 \log s')$, interpret U as a Turing machine that takes advice y to decide the direct-connection language of some circuit D of size s' on inputs of length $|(x, g)|$. Construct the circuit D using U and y, where to evaluate U on a given input we simulate U for at most s' steps. Enter the MAJ state. Nondeterministically guess a gate h of C and a bit $b \in \{0,1\}$. If h is not an input gate for g, then accept if $b = 1$ and reject if $b = 0$; otherwise, accept if $D(x, h) = 1$ and reject if $D(x, h) = 0$."

If U is a polynomial-time TM, then each simulation of U on a given input takes time $\mathsf{poly}(c_0 \log s' + \gamma'(c_0 \log s'))$, which is less than s' by our assumptions that $\gamma(m) \leqslant 2^{o(m)}$ and $d \leqslant (s')^{o(1)}$. Thus, to evaluate U on a particular input, it suffices to simulate U for at most s' steps, which is independent of what the actual polynomial time bound of U is. It follows that we can construct the circuit D (given U and y) in time $p(s')$, where p is a polynomial that does not depend on U. Also, to decide if h is an input gate to g, we use the polynomial-time TM M. We conclude that N is a PP machine which runs in some polynomial time (dependent on A). Since Permanent is PP-hard [Val79], we have a uniform reduction mapping z (an input to N) to an instance of PERMANENT of size $q(|z|)$, for some polynomial q (dependent on A).

By our assumption on the easiness of PERMANENT, we get that the language of N is decided by γ-weakly uniform circuits C_N of size at most $s'' = s(q(s'))$. If we plug in for U and y the actual Turing machine description and the advice needed to decide the direct-connection language of B_i, we get from C_N the circuit B_{i+1}. Note that the direct-connection language of this circuit B_{i+1} is decided in polynomial time (using the algorithm for direct-connection language of C_N) given the advice needed for C_N plus the advice needed to describe U and y. The total advice size is at most $\gamma(c_0 \log s'') + \gamma(c_0 \log s'') + \gamma'(c_0 \log s') \leqslant 2(i+1)\gamma(c_0 \log s'')$. □

Finally, we take the circuit B_d and use it to evaluate $A(x)$ by computing the value $B_d(x, g)$ where g is the output gate of C, which can be efficiently constructed (since this is just the initial configuration of our threshold TM for A on input x). By fixing g to be the output gate of C, we get the circuit for A which is $2d\gamma$-weakly uniform of size at most $(s \circ q)^{(d)}(r(n))$, where the polynomial r depends on the language A. Upper-bounding the polynomial r by $(s \circ q)$ yields the result. □

4 Proofs of the main results

Here we use the technical tools from the previous section in order to prove our main results. Recall that L_0 is the P-complete language defined earlier.

4.1 Proof of Theorem 1.1

First, assuming L_0 is easy, we construct a hard language in CH.

Lemma 4.1. *Suppose L_0 is in* subexp*-weakly uniform* TC^0 *of depth d. Then, for a constant d' dependent on d, there is a language $L_{diag} \in \mathsf{CH}_{d'}$ which is not in* subexp*-weakly uniform* $\mathsf{SIZE}(\mathsf{poly})$.

Proof. Let $\alpha(m) \in 2^{o(m)}$ be such that L_0 is in α-weakly uniform TC^0 of depth d. Consider an arbitrary language L in β-weakly uniform $\mathsf{SIZE}(\mathsf{poly})$, for an arbitrary $\beta(m) \in 2^{o(m)}$. By Lemma 3.3, L has $\mu(n)$-weakly uniform threshold circuits of depth d and polynomial size, where $\mu(n) = \alpha(O(\log n)) + \beta(O(\log n)) \leqslant n^{o(1)}$. By Lemma 2.4, we have that L is decided by a threshold Turing machine with $d' = O(d)$ alternations, taking advice of length $\mu(n) \leqslant n^{o(1)} \leqslant n/\log^2 n$, and running in time $d' \cdot \mathsf{poly}(O(\log n) + n^{o(1)}) \leqslant n^{o(1)} \leqslant n/\log^2 n$. We conclude that *every* language in subexp-weakly uniform $\mathsf{SIZE}(\mathsf{poly})$ is also decided by some threshold Turing machine in time $n/\log^2 n$, using d' alternations and advice of size $n/\log^2 n$.

Using Lemma 3.1, define L_{diag} to be the language in $\mathsf{Th}_{d'}\mathsf{TIME}(n)$ which is not decidable by any threshold Turing machine in time $n/\log^2 n$, using d' alternations and advice of size $n/\log^2 n$. It follows that L_{diag} is different from every language in subexp-weakly uniform $\mathsf{SIZE}(\mathsf{poly})$. □

Next, assuming PERMANENT is easy, we show that every language in CH is easy.

Lemma 4.2. *If* PERMANENT *is in* subexp*-weakly uniform* $\mathsf{SIZE}(\mathsf{poly})$*, then every language in* CH *is in* subexp*-weakly uniform* $\mathsf{SIZE}(\mathsf{poly})$.

Proof. The proof is immediate by Lemma 3.4. □

We now show that L_0 and PERMANENT cannot both be easy.

Theorem 4.3. *At least one of the following must be false:*

1. L_0 *is in* subexp*-weakly uniform* TC^0*;*
2. PERMANENT *is in* subexp*-weakly uniform* $\mathsf{SIZE}(\mathsf{poly})$*.*

Proof. The proof is immediate by Lemmas 4.1 and 4.2. □

To unify the two items in Theorem 4.3, we use the following result.

Lemma 4.4 ([Val79, AG94])**.** *For every language* $L \in \mathsf{P}$*, there are uniform* AC^0*-computable function* M *(mapping a binary string to a polynomial-size Boolean matrix) and Boolean function* f *such that, for every* x*, we have* $x \in L$ *iff* $f(\text{PERMANENT}(M(x)) = 1$*.*

This lemma immediately yields the following.

Corollary 4.5. *If* PERMANENT *has* α*-weakly uniform* $d(n)$*-depth threshold circuits of size* $s(n)$*, then* L_0 *has* α*-weakly uniform* $(d(n^{O(1)}) + O(1))$*-depth threshold circuits of size* $s(n^{O(1)})$*.*

Now we prove Theorem 1.1, which we re-state below.

Theorem 4.6. PERMANENT *is not in* subexp*-weakly uniform* TC^0*.*

Proof. Otherwise, by Corollary 4.5, both claims in Theorem 4.3 would hold, which is impossible. □

4.2 Proof of Theorem 1.2

Recall that a function $r(n)$ is sub-subexponential if, for every constant $k > 0$, $r^{(k)}(n) \leqslant 2^{n^{o(1)}}$. Also recall that $\mathsf{subsubexp}$ denotes the class of all sub-subexponential functions $r(n)$. Below, we will use the simple fact that, for every constant $k > 0$, the composition of k sub-subexponential functions is also sub-subexponential.

Lemma 4.7. *Suppose* L_0 *is in* poly*-weakly uniform* $\mathsf{TC}^0(\mathsf{subsubexp})$ *of depth* d*. Then, for a constant* $d' = O(d)$*, there is a language* $L_{diag} \in \mathsf{CH}_{d'}$ *which is not in* poly*-weakly uniform* $\mathsf{SIZE}(\mathsf{subsubexp})$*.*

Proof. The proof is similar to that of Lemma 4.1. Let $\alpha(m) \in \mathsf{poly}(m)$ and $s(n) \in \mathsf{subsubexp}$ be such that L_0 is in α-weakly uniform d-depth $\mathsf{TC}^0(s(n))$.

Consider an arbitrary language L in β-weakly uniform $\mathsf{SIZE}(t(n))$, for arbitrary $\beta(m) \in \mathsf{poly}(m)$ and $t(n) \in \mathsf{subsubexp}$. By Lemma 3.3, L is in $\mu(n)$-weakly uniform d-depth $\mathsf{TC}^0(s'(n))$, where $s'(n) = s(\mathsf{poly}(t(n)))$ and $\mu(n) = \alpha(c_0 \log s'(n)) + \beta(c_0 \log t(n)) \leqslant n^{o(1)}$ (since s' and t are sub-subexponential). By Lemma 2.4, we have that L is decided by a threshold Turing machine with $d' = O(d)$ alternations, taking advice of length $\mu(n) \leqslant n^{o(1)} \leqslant n/\log^2 n$, and running in time $d' \cdot \mathsf{poly}(c_0 \log s'(n) + \alpha(c_0 \log s'(n))) \leqslant n^{o(1)} \leqslant n/\log^2 n$. We conclude that *every* language in poly-weakly uniform $\mathsf{SIZE}(\mathsf{subsubexp})$ is also decided by some threshold Turing machine in time $n/\log^2 n$, using d' alternations and advice of size $n/\log^2 n$.

Using Lemma 3.1, define L_{diag} to be the language in $\mathsf{Th}_{d'}\mathsf{TIME}(n)$ which is not decidable by any threshold Turing machine in time $n/\log^2 n$, using d' alternations and advice of size $n/\log^2 n$. It follows that L_{diag} is different from every language in poly-weakly uniform $\mathsf{SIZE}(\mathsf{subsubexp})$. □

Now we are ready to prove Theorem 1.2, which we re-state below.

Theorem 4.8. Permanent *is not in* poly*-weakly uniform* $\mathsf{TC}^0(\mathsf{subsubexp})$.

Proof. Suppose that, for some $\alpha(m) \in \mathsf{poly}(m)$ and $s(n) \in \mathsf{subsubexp}$, Permanent is in α-weakly uniform $\mathsf{TC}^0(s(n))$; this also implies that Permanent is in α-weakly uniform $\mathsf{SIZE}(\mathsf{poly}(s(n)))$. By Corollary 4.5, L_0 is in α-weakly uniform $\mathsf{TC}^0(\mathsf{poly}(s(n)))$, and so, by Lemma 4.7, there is a language $L_{diag} \in \mathsf{CH}$ which is not in poly-weakly uniform $\mathsf{SIZE}(\mathsf{subsubexp})$. But, by Lemma 3.4, every language L in CH is in poly-weakly uniform $\mathsf{SIZE}(\mathsf{subsubexp})$. A contradiction. □

4.3 Proof of Theorem 1.3

Lemma 4.9. *Suppose* L_0 *is computable by* poly*-weakly uniform* poly*-size threshold circuits of depth* $o(\log\log n)$. *Then there is a language* $L_{diag} \in \mathsf{Th}_{\log\log n}\mathsf{TIME}(n)$ *which is not computable by* poly*-weakly uniform* $\mathsf{SIZE}(n^{\mathsf{poly}(\log n)})$.

Proof. Let $\alpha(m) \in \mathsf{poly}(m)$, $s(n) \in \mathsf{poly}(n)$, and $d(n) \in o(\log\log n)$ be such that L_0 is computable by α-weakly uniform $d(n)$-depth threshold circuits of size $s(n)$.

Consider an arbitrary language L in β-weakly uniform $\mathsf{SIZE}(t(n))$, for arbitrary $\beta(m) \in \mathsf{poly}(m)$ and $t(n) \in n^{\mathsf{poly}(\log n)}$. By Lemma 3.3, L is in $\mu(n)$-weakly uniform $d'(n)$-depth threshold circuits of size $s'(n)$, where $d'(n) = d(\mathsf{poly}(t(n))) \leqslant o(\log\log n)$, $s'(n) = s(\mathsf{poly}(t(n))) \leqslant n^{\mathsf{poly}(\log n)}$, and $\mu(n) = \alpha(c_0 \log s'(n)) + \beta(c_0 \log t(n)) \leqslant \mathsf{poly}(\log n)$.

By Lemma 2.4, we have that L is decided by a threshold Turing machine with at most $O(d'(n)) < \log\log n$ alternations, taking advice of length $\mu(n) \leqslant n^{o(1)} \leqslant n/\log^2 n$, and running in time $O(d'(n)) \cdot \mathsf{poly}(c_0 \log s'(n) + \alpha(c_0 \log s'(n))) \leqslant n^{o(1)} \leqslant n/\log^2 n$. We conclude that *every* language in poly-weakly uniform $\mathsf{SIZE}(n^{\mathsf{poly}(\log n)})$ is also decided by some threshold Turing machine in time $n/\log^2 n$, using $\log\log n$ alternations and advice of size $n/\log^2 n$.

Using Lemma 3.1, define L_{diag} to be the language in $\mathsf{Th}_{\log\log n}\mathsf{TIME}(n)$ which is not decidable by any threshold Turing machine in time $n/\log^2 n$, using $\log\log n$ alternations and advice of size $n/\log^2 n$. It follows that L_{diag} is the required language. □

Now we prove Theorem 1.3, re-stated below.

Theorem 4.10. Permanent *is not computable by* poly*-weakly uniform* poly*-size threshold circuits of depth* $o(\log\log n)$.

Proof. Assume otherwise. Then Permanent is also in poly-weakly uniform $\mathsf{SIZE}(\mathsf{poly})$, and so, by Lemma 3.4, every language in $\mathsf{Th}_{\log\log n}\mathsf{TIME}(n)$ is in poly-weakly uniform $\mathsf{SIZE}(n^{\mathsf{poly}(\log n)})$. On the other hand, by Corollary 4.5, L_0 is computable by poly-weakly uniform threshold circuits of poly-size and depth $o(\log\log n)$, and so, by Lemma 4.9, there is a language $L_{diag} \in \mathsf{Th}_{\log\log n}\mathsf{TIME}(n)$ such that L_{diag} is not in poly-weakly uniform $\mathsf{SIZE}(n^{\mathsf{poly}(\log n)})$. A contradiction. □

5 Other lower bounds

Here we use diagonalization against advice classes to prove exponential lower bounds for weakly uniform circuits of both constant and unbounded depth.

5.1 Lower bounds for ACC^0 and AC^0

The following result generalizes the result in [AG94] on uniform ACC^0 circuits.

Theorem 5.1. *Permanent is not in* poly-*weakly uniform* $\mathsf{ACC}^0(2^{n^{o(1)}})$.

Proof. It is shown in [BT94, AG94] that every language L in uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$ is also decidable by uniform depth-two circuits of related size $s'(n) \in 2^{n^{o(1)}}$ where *(i)* the bottom level consists of AND gates of fan-in $(\log s'(n))^{O(1)}$, and *(ii)* the top level is a symmetric gate (whose value depends only on the number of inputs that evaluate to one). Using this fact as well as the #P-completeness of Permanent [Val79], Allender and Gore [AG94] argue that L is in $\mathsf{DTIME}(n^9)^{\text{Permanent}[1]}$ (with a single oracle query to Permanent). This result can be easily generalized to the case when L has weakly uniform circuits. That is, for $\alpha(m) = m^{O(1)}$, any language in α-weakly uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$ is also in $\mathsf{DTIME}(n^9)^{\text{Permanent}[1]}/\gamma(n)$ for some $\gamma(n) = n^{o(1)}$.

For the sake of contradiction, suppose that Permanent is in α-weakly uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$. Consider a language $L \in \mathsf{DTIME}(n^{10})^{\text{Permanent}[1]}$ which is not in $\mathsf{DTIME}(n^9)^{\text{Permanent}[1]}/n^{o(1)}$; the existence of such an L is easy to argue by diagonalization (similarly to the proof of Lemma 3.1). Let M be the corresponding oracle machine deciding L. Consider the following languages:

$$L' = \{(x, y)\colon M \text{ uses } y \text{ as the answer of the oracle query and accepts } x\},$$

$$L'' = \{(x, i)\colon \text{the } i\text{th bit of the oracle query made by } M \text{ on input } x \text{ is } 1\}.$$

Clearly, both L' and L'' are in P. Since P is reducible to Permanent via uniform AC^0 reduction, we get that both L' and L'' are in α-weakly uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$. To construct circuits for L, on any input x, we use the circuit for L'' to construct the oracle query, use the circuit for Permanent to answer the query, and then use the circuit for L' to decide whether $x \in L$. Since L', L'' and Permanent all have α-weakly uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$ circuits, the resulting circuit is also in α-weakly uniform $\mathsf{ACC}^0(2^{n^{o(1)}})$. This implies that L is in $\mathsf{DTIME}(n^9)^{\text{Permanent}[1]}/n^{o(1)}$. A contradiction. □

We note that one can also show a lower bound for NP against weakly uniform AC^0 circuits.

Theorem 5.2. NP *is not in* poly-*weakly uniform* $\mathsf{AC}^0(\mathsf{subsubexp})$.

Proof. The proof is analogous to that of Theorem 4.8, by replacing Permanent with Sat, CH with PH, and threshold circuits with Boolean circuits. □

Note, however, that this lower bound is weaker than the well-known result that Parity requires *exponential-size non-uniform* AC^0 circuits [Hås86].

5.2 Lower bounds for general circuits

We use the following diagonalization result.

Lemma 5.3 ([HM95, Pol06])**.** *For any constants c and d,* $\mathsf{EXP} \not\subseteq \mathsf{DTIME}(2^{n^d})/n^c$, *and* $\mathsf{PSPACE} \not\subseteq \mathsf{DSPACE}(n^d)/n^c$.

Proof sketch. We will construct a language $L_{diag} \in \mathsf{EXP}$ which is different from every language in $\mathsf{DTIME}(2^{n^d})/n^c$. Let M be an arbitrary machine which takes advice of length n^c and runs in time 2^{n^d}. Fix the input length $n = 2^{|M|}$, where $|M|$ is the length of the binary description of M. Let $x_1, \ldots, x_{n^c}$ be distinct inputs in $\{0,1\}^n$. We use these n^c inputs to diagonalize against all advice of length n^c. On input x_1, we enumerate all possible advice in $\{0,1\}^{n^c}$, simulate M with each advice on x_1, and accept iff the majority rejects; then we delete all advice for which the output is in the majority. We then repeat the process on $x_2, \ldots, x_{n^c}$, but only check the advice that are not deleted by the previous inputs. In this way, we have L_{diag} is different from the language decided by M with any advice of length n^c. To diagonalize against all machines of the same length as M, we need only $n^c \cdot 2^{|M|} \ll 2^n$ inputs. It is obvious that the constructed diagonal language is in EXP.

The proof for PSPACE is similar. The only difference is that we do not have enough space to write down the exponential number of advice strings. Instead, on input x, we need to check whether each advice string is already diagonalized against by inputs that are lexicographically smaller than x, and then compute the majority output over all remaining advice strings. All this can be done in polynomial space. □

Theorem 5.4. EXP *is not in* poly*-weakly uniform* $\mathsf{SIZE}(2^{n^{o(1)}})$.

Proof. Let L be an arbitrary language in poly-weakly uniform $\mathsf{SIZE}(2^{n^{o(1)}})$. For any input length n, given advice of length $\mathsf{poly}(\log 2^{n^{o(1)}}) \leqslant n^{o(1)}$, we can construct a circuit for L of size $2^{n^{o(1)}}$ in time at most $2^{n^{o(1)}}$, and evaluate it on any given input of size n in time at most $2^{n^{o(1)}}$. Thus, $L \in \mathsf{DTIME}(2^{n^{o(1)}})/n^{o(1)}$.

Using Lemma 5.3, construct $L_{diag} \in \mathsf{EXP}$ which is not in $\mathsf{DTIME}(2^n)/n$. By the above, this L_{diag} is not in poly-weakly uniform $\mathsf{SIZE}(2^{n^{o(1)}})$. □

Recall that a Boolean circuit is called a *formula* if the underlying DAG is a tree (i.e., the fan-out of each gate is at most 1). We denote by $\mathsf{FSIZE}(s(n))$ the class of families of Boolean formulas of size $s(n)$. We use a modified definition of the the direct-connection language for bounded fan-in formulas with AND, OR, and NOT gates: we assume that, for any given gate in the formula, we can determine in polynomial time who its parent gate is, and who its left and right input gates are.

Lynch [Lyn77] gave a log-space algorithm for the Boolean formula evaluation problem, which can be adapted to work also in the case of input formulas given by the direct connection language (instead of the usual infix notation).

Lemma 5.5 (implicit in [Lyn77])**.** *Let $\{F_n\}$ be a uniform family of Boolean formulas of size $s(n)$. There is a* $\mathsf{poly}(\log s(n))$*-space algorithm that, on input x of length n, computes $F_n(x)$.*

Proof sketch. The input formula can be viewed as a tree, where each node has at most two children, and the evaluation algorithm will traverse the tree following specific rules. We assume that the formula is well-formed, which can be verified in $\mathsf{poly}(\log s(n))$-space.

The traversal starts from the left-most leaf, which can be identified in $\mathsf{poly}(\log s(n))$-space. Then, we traverse the tree such that, for each node A, *(i)* when we arrive at A from its left child, we either go to its parent (if the value of the left child fixes the value of A), or go to its right child and continue traversing the tree; *(ii)* when we arrive at A from its right child, we go directly to A's parent (the value of A is now determined by the value of the right child, as we know the left child has already been visited). The final node in this traversal is the root, which has no parent.

The traversal is in $\mathsf{poly}(\log s(n))$-space since we only need to remember the current node of the tree (and the direct-connection language is decided in time, and hence also in space, at most $\mathsf{poly}(\log s(n)))$. $\square$

We have the following.

Theorem 5.6. PSPACE *is not in* poly*-weakly uniform* $\mathsf{FSIZE}(2^{n^{o(1)}})$.

Proof. Let L be an arbitrary language decided by a family $\{F_n\}$ of poly-weakly uniform Boolean formulas of size $2^{n^{o(1)}}$; its direct connection language is decided in deterministic time $n^{o(1)}$ with advice of size $n^{o(1)}$. Using Lemma 5.5 (generalized in the straightforward way to handle weakly uniform formulas), we get that L can be decided in $\mathsf{DSPACE}(n^{o(1)})/n^{o(1)}$. Appealing to Lemma 5.3 completes the proof. $\square$

6 Conclusion

We have shown how to use indirect diagonalization to prove lower bounds against weakly uniform circuit classes. In particular, we have proved that PERMANENT cannot be computed by polynomial-size TC^0 circuits that are only slightly uniform (whose direct-connection language can be efficiently computed using sublinear amount of advice). We have also extended to the weakly uniform setting other circuit lower bounds that were previously known for the case of uniform circuits.

One obvious open problem is to improve the TC^0 circuit lower bound for PERMANENT to be exponential, which is not known even for the uniform case. Another problem is to get super-polynomial uniform TC^0 lower bounds for a language from a complexity class below $\#\mathsf{P}$ (e.g., PH). Strongly exponential lower bounds even against uniform AC^0 would be very interesting. One natural problem is to prove a better lower bound against *uniform* AC^0 (say for PERMANENT) than the known non-uniform AC^0 lower bound for PARITY.

References

[AB09] S. Arora and B. Barak. *Complexity theory: a modern approach*. Cambridge University Press, New York, 2009.

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

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the Twenty-Fifth Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 92–105, 2005.

[All99] E. Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, 1999.

[BIS90] D.A.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[BT94] R. Beigel and J. Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[CKS81] A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[FSS84] M. Furst, J.B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[GKR$^+$95] F. Green, J. Köbler, K.W. Regan, T. Schwentick, and J. Toran. The power of the middle bit of a #P function. *Journal of Computer and System Sciences*, 50:456–467, 1995.

[Hås86] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[HM95] S. Homer and S. Mocas. Nonuniform lower bounds for exponential time classes. In *Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science*, pages 159–168. Springer Verlag, 1995.

[HS82] J. Heintz and C.-P. Schnorr. Testing polynomials which are easy to compute. *L'Enseignement Mathématique*, 30:237–254, 1982.

[IM02] K. Iwama and H. Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *Proceedings of the Twenty-Seventh International Symposium on Mathematical Foundations of Computer Science*, pages 353–364. Springer Verlag, 2002.

[JS11] M.J. Jansen and R. Santhanam. Permanent does not have succinct polynomial size arithmetic circuits of constant depth. In *Proceedings of the Thirty-Eighth International Colloquium on Automata, Languages, and Programming, Part I*, pages 724–735, 2011.

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

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

[KL82] R.M. Karp and R.J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(3-4):191–209, 1982.

[KP09] P. Koiran and S. Perifel. A superpolynomial lower bound on the size of uniform non-constant-depth threshold circuits for the permanent. In *Proceedings of the Twenty-Fourth Annual IEEE Conference on Computational Complexity*, pages 35–40, 2009.

[LR01] O. Lachish and R. Raz. Explicit lower bound of $4.5n - o(n)$ for boolean circuits. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 399–408, 2001.

[Lup58] O.B. Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958. English translation in *Soviet Mathematics Doklady*.

[Lyn77] N.A. Lynch. Log space recognition and translation of parenthesis languages. *Journal of the Association for Computing Machinery*, 24:583–590, 1977.

[Pol06] C. Pollett. Languages to diagonalize against advice classes. *Computational Complexity*, 14:341–361, 2006.

[PS86] I. Parberry and G. Schnitger. Parallel computation with threshold functions. In *Proceedings of the First Annual IEEE Conference on Structure in Complexity Theory*, pages 272–290, 1986.

[Raz87] A.A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41:333–338, 1987.

[RR97] A.A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[Ruz81] W.L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[Sha49] C.E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 77–82, 1987.

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the Association for Computing Machinery*, 38:752–773, 1991.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Wag86] K.W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

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

[Yao85] A.C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the Twenty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de