On TC^0 Lower Bounds for the Permanent

Jeff Kinne
Indiana State University
jkinne@cs.indstate.edu

September 5, 2012

Abstract

In this paper we consider the problem of proving lower bounds for the permanent. An ongoing line of research has shown super-polynomial lower bounds for slightly-non-uniform small-depth threshold and arithmetic circuits [All99, KP09, JS11, JS12]. We prove a new parameterized lower bound that includes each of the previous results as sub-cases. Our main result implies that the permanent does not have Boolean threshold circuits of the following kinds.

1. Depth $O(1)$, poly-log(n) bits of non-uniformity, and size $s(n)$ such that for all constants c, $s^{(c)}(n) < 2^n$. The size s must satisfy another technical condition that is true of functions normally dealt with (such as compositions of polynomials, logarithms, and exponentials).
2. Depth $o(\log \log n)$, poly-log(n) bits of non-uniformity, and size $n^{O(1)}$.
3. Depth $O(1)$, $n^{o(1)}$ bits of non-uniformity, and size $n^{O(1)}$.

Our proof yields a new "either or" hardness result. One instantiation is that either NP does not have polynomial-size constant-depth threshold circuits that use $n^{o(1)}$ bits of non-uniformity, or the permanent does not have polynomial-size general circuits.

1 Introduction

The Search for Hard Problems Computational complexity aims to determine the computational costs of solving important problems, requiring both upper bounds and lower bounds. Though many types of lower bounds have been difficult to prove, conjectured lower bounds have become central across complexity theory. As an example, consider the area of derandomization – the task of converting randomized algorithms into deterministic algorithms with as little loss in efficiency as possible. Work initiated by Nisan and Wigderson [NW94] gives conditions for derandomizing BPP, the set of problems decided by bounded-error randomized polynomial-time algorithms. If E, deterministic linear-exponential time, contains a problem requiring super-polynomial circuits then BPP is contained in subexponential time (SUBEXP) [BFNW93]. If E contains a problem requiring circuits of size $2^{\epsilon n}$ for some positive constant ϵ then BPP = P [IW97]. These results are called hardness versus randomness tradeoffs because randomness can be more efficiently removed by using stronger hardness assumptions.

The lower bound results that have been proved so far have generally been for very high complexity classes (e.g., exponential-time Merlin-Arthur protocols require super-polynomial size circuits [BFT98, MVW99]) or for restricted models of computation (e.g., lower bounds for

parity on constant-depth circuits [FSS84, Yao85, Hås87], lower bounds on monotone circuits [Raz85, Smo87]).

The long-term goal in proving lower bounds for restricted models is to prove lower bounds for increasingly more general models. The lower bounds for constant-depth circuits could be improved by allowing more powerful gates than the standard AND, OR, NOT. [Wil11] showed that non-uniform ACC^0 circuits – constant-depth circuits that may include MOD_m gates for arbitrary modulus m – of polynomial size cannot compute languages that are complete for nondeterministic exponential time (NEXP). Because MOD_m gates can be viewed as specialized majority gates, the next step in this direction is to prove lower bounds for constant-depth circuits with majority gates (TC^0 circuits).

Uniform TC^0 Lower Bounds The discussion above refers to proving lower bounds on *non-uniform* circuits – circuit families that consist of a different circuit for each input length n, with no requirement that the circuits are related to each other in any way. The task of proving super-polynomial lower bounds for non-uniform TC^0 circuits for languages in NEXP remains open. Progress has been made in proving lower bounds for *uniform* TC^0 circuits – circuit families where there exists a single Turing machine that can be used to reconstruct the circuit for any input length. The Turing machine that reconstructs the circuit should at a minimum run in polynomial time, but because P is believed to be more powerful than uniform TC^0 a stronger notion of uniformity is appropriate. One such strengthening is Dlogtime uniformity. A family of TC^0 circuits is *Dlogtime uniform* if there exists a Turing machine that correctly answers queries about connections in the circuits in time that is logarithmic in the size of the circuit. Dlogtime uniformity is the standard notion of uniformity for low circuit classes such at TC^0 (see [BIS90] for some discussion). We say a TC^0 circuit is uniform if it is Dlogtime uniform.

Polynomial-size uniform TC^0 circuits can be simulated in logarithmic space, so the space hierarchy theorem implies that PSPACE-complete languages cannot be computed by uniform TC^0 circuits of subexponential size. [All99] showed that the hard language can be lowered from PSPACE to the first level of the counting hierarchy at the expense of a slight loss in the size: languages complete for PP, such as language versions of the permanent, cannot be computed by uniform TC^0 circuits of size $s(n)$ if s is time-constructible and $s^{(c)}(n) < 2^n$ for all constants c, where $s^{(c)}$ denotes s composed with itself c times. [KP09] show a lower bound for threshold circuits with super-constant depth: languages complete for PP cannot be computed by uniform threshold circuits of depth $o(\log \log n)$ and polynomial size.

Non-Uniform TC^0 Lower Bounds While there has been progress on proving lower bounds for uniform TC^0 circuits, the ultimate question of proving super-polynomial lower bounds for non-uniform TC^0 circuits remains open. It has been observed that for every constant $k > 0$ the counting hierarchy contains languages that require general circuits of size n^k (see, e.g., [All96, KMS12, JS12]), and this applies to TC^0 circuits as well. Thus we have two types of lower bounds for TC^0 circuits in the counting hierarchy: fixed-polynomial size and fixed-polynomial non-uniformity, and large size but uniform. [JS11, JS12] show a lower bound that is intermediate between these two but for arithmetic circuits rather than Boolean threshold circuits: constant-depth constant-free arithmetic circuits of polynomial size and $n^{o(1)}$-succinctness cannot compute the permanent. [JS11] introduces the notion of succinctness as a way to interpolate between fully uniform and fully non-uniform circuits. A circuit family is $a(n)$-*succinct* if questions about connections in the circuit can be answered by a non-uniform circuit of size $a(n)$. Note that Dlogtime uniform polynomial-size circuits are necessarily poly-log-succinct, and non-uniform polynomial-size circuits are poly-succinct.

Many natural questions arise from the lower-bounds results for threshold circuits. Can the result of [JS12] be proved for Boolean threshold circuits (which are stronger than arithmetic

circuits in the sense that threshold circuits can simulate arithmetic circuits)? Can a tradeoff be proved between the amount of non-uniformity, circuit size, and depth in the lower bounds for threshold circuits? Our paper answers both questions in the affirmative, as we discuss in Section 1.1, and gives a unified proof of each of the previous results [All99, KP09, JS12]. The question which remains open is whether TC^0 lower bounds can be extended to simultaneous super-polynomial non-uniformity and size.

Permanent and Polynomial Identity Testing The line of research on TC^0 circuit lower bounds [All99, KP09, JS11, JS12] has focused on the permanent as the hard language. The primary property of the permanent that is used is the PP-completeness of the language version of the permanent. Beyond the fact that the proofs work for the permanent, proving lower bounds for the permanent is of interest because of connections to derandomization and the polynomial identity testing problem.

As mentioned already, strong enough lower bounds imply derandomization of BPP, but such lower bounds have been difficult to prove. To make progress, research has focused on derandomizing specific problems. In particular, much work has been done for polynomial identity testing (PIT) – for which there is a simple and efficient randomized algorithm while the best deterministic algorithms known require exponential time. Some results have given fast deterministic algorithms for restricted versions of PIT (for example, testing identities encoded by depth two or three arithmetic circuits), while other results have shown that lower bounds for restricted types of circuits yield deterministic identity testing procedures for restricted versions of PIT (see [AS09, SY10] for surveys of this work).

Of particular note to the current discussion is the result that exponential lower bounds for the permanent on constant-depth arithmetic circuits imply deterministic quasi-polynomial time algorithms for testing identities on constant-depth circuits that have degree poly-log(n) in each variable [DSY09]. Thus strong enough TC^0 lower bounds for the permanent directly imply improved PIT algorithms. This provides motivation to continue working towards non-uniform TC^0 lower bounds for the permanent, beyond the fact that TC^0 lower bounds are a logical next step after the ACC^0 lower bounds of [Wil11].

1.1 Our Results

Lower Bounds for Permanent Our main result is a new lower bound for small-depth threshold circuits computing the permanent, Theorem 1. Let L be any PP-hard language such that any language decidable in PPTIME($t(n)$) can be reduced to an instance of L of size $t(n)\cdot$poly-log($t(n)$) by a quasi-linear size uniform TC^0 reduction. By [Zan91], a paddable version of the 0-1 permanent satisfies this property. Let $s'^{(d)}$ denote a function s' composed with itself d times.

Theorem 1. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. Let $s'(n) = s(n^{O(1)})$, and $m = s(2^{O(n)})$. If $s'^{(d(m))}(n + a(m) +$ poly-log$(s(m))) < 2^n$ then L does not have depth $d(n) - O(1)$, $(a(n) -$ poly-log$(s(n)))$-succinct threshold circuits of size $s(n)/$ poly-log$(s(n))$.*

Each of the constants in the big-O and polylog terms in Theorem 1 are absolute constants independent of the functions s, a, and d.

We have the following corollary for the extreme cases of maximizing each of the three parameters – size, depth, and amount of non-uniformity – in Theorem 1.

Corollary 1. *The permanent does not have threshold circuits of the following kinds.*

1. *Depth $O(1)$, poly-log$(s(n))$-succinct, and size $s(n)$ such that $s(n)$ is non-decreasing, time-constructible and for all constants c, $s^{(c)}(n) < 2^n$ and $\log(s(s(2^{c \cdot n}))) = s^{(O(1))}(n)$.*
2. *Depth $o(\log \log n)$, poly-log(n)-succinct, and size $n^{O(1)}$.*
3. *Depth $O(1)$, $n^{o(1)}$-succinct, and size $n^{O(1)}$.*

Corollary 1 captures the main results from the previous works in this area as sub-cases. The main results of [All99] and [KP09] for the permanent correspond to Items 1 [1] and 2 of Corollary 1 but for uniform circuits. The main unconditional lower bounds for the permanent in [JS12] have the same parameters as Item 3 except they are for constant-depth constant-free arithmetic circuits rather than Boolean threshold circuits.

Lower Bounds for Permanent or Satisfiability We prove that Theorem 1 can be strengthened to imply that either NP is hard for succinct small-depth threshold circuits, or the permanent is hard for general circuits. In Theorem 2, SAT is the NP-complete language Boolean formula satisfiability. Any of the standard NP-complete languages could be used instead.

Theorem 2. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. Let $s'(n) = s(n^{O(1)})$, and $m = s(s(2^{O(n)}))$. If $s'^{(d(m))}(n + a(m) + \text{poly-log}(s(m))) < 2^n$ then either*

- SAT *does not have depth $d(n) - O(1)$, $(a(n) - \text{poly-log}(s(n)))$-succinct threshold circuits of size $s(n)/\text{poly-log}(s(n))$, or*
- *The permanent does not have non-uniform general circuits of size $s(n)/\text{poly-log}(s(n))$.*

Each item of Corollary 1 can also be stated in a similar way.

More Direct Proof of Known Result Our proof of Theorem 1 uses the fact that the exponential-time counting hierarchy contains a language that requires circuits of exponential size. The result has been stated for the setting of the standard counting hierarchy and circuits of polynomial size in a number of works – including [All96], [KMS12] and [JS12] – where it is derived by combining Toda's Theorem [Tod91a] and the fact that the polynomial hierarchy contains languages that require fixed-polynomial size circuits [Kan82].

We give a more direct proof of the result and state it for a wider range of parameters. The argument can be used to obtain a language with hardness up to the minimum of $2^n - 1$ and the maximal circuit complexity. For our definition of circuit size (string length of the circuit's description), the maximal circuit complexity is at least 2^n.

Theorem 3. *Let $h(n)$ be a time-constructible function such that for all n, $n \leq h(n) < 2^n$. There is a language L_{hard} in $\text{DTIME}^{\text{PP}}(\text{poly}(h(n)))$ that does not have circuits of size $h(n)$.*

Since separations for high resources imply separations for low resources, it will be optimal to set $h(n)$ as large as possible. We use the following in the proof of Theorem 1.

Corollary 2. *There is a language L_{hard} in $\text{DTIME}^{\text{PP}}(2^{O(n)})$ that does not have circuits of size $2^n - 1$.*

[1] The last condition on s in Item 1, that $\log(s(s(2^{c \cdot n}))) = s^{(O(1))}(n)$, holds for "normal" functions, those composed of logarithms, exponentials, and polynomials. The corresponding result of [All99] does not require this condition.

We point out that our direct proof of Theorem 3 obviates the need to use Toda's theorem in a result of [KMS12]. [KMS12] gives an alternate proof of a result of [KI04] that if polynomial identity testing can be derandomized, then either Boolean circuit lower bounds for NEXP or arithmetic circuit lower bounds for the permanent must follow. A number of proofs are known for these types of results [AM11], and the proof of [KMS12] gives the best-known tradeoff between the parameters. A further benefit of the [KMS12] proof is that it is more direct than other proofs, and using our proof of Theorem 3 simplifies their proof further.

We also use Theorem 3 to simplify the proof of the result of [Vin05] that for every constant $k > 0$ there is a language in PP requiring circuits of size n^k.

Remark We learned after publication of our paper that an identical proof of Theorem 3 and the result of [Vin05] is contained in [Aar06]. [Aar06] extends the results to apply to also quantum circuits.

1.2 Techniques

To see the structure of the proof of Theorem 1, we first give an outline for proving that constant-depth uniform threshold circuits of polynomial size cannot compute the permanent. We assume the permanent has uniform poly-size constant-depth threshold circuits and aim for a contradiction. We achieve the contradiction in two parts.

(i) We use the assumed easiness of the permanent to conclude that a non-uniformly hard language can be solved by large uniform small-depth threshold circuits. In particular, by the PP-completeness of the permanent and under the assumed easiness of the permanent, L_{hard} of Corollary 2, which is in E^{PP}, has uniform constant-depth threshold circuits of size $2^{O(n)}$.

(ii) Let C_{hard} be the circuit for L_{hard} at input length n from (i). By viewing the threshold gates within C_{hard} as questions about the permanent, we shrink the circuit as follows. The first level of threshold gates closest to the inputs in C_{hard} can be viewed as PP questions of size $\mathrm{poly}(n)$; using the assumed easiness of the permanent a circuit C_1 of size $\mathrm{poly}(n)$ can be used in place of the threshold gates on the first level. A similar argument shows that the second level of threshold gates reduce to PP questions of size $\mathrm{poly}(|C_1|)$, which can be replaced by a circuit of size $\mathrm{poly}(\mathrm{poly}(|C_1|))$ using the assumed easiness of the permanent. This process is repeated for each level of threshold gates in C_{hard}. If C_{hard} has depth d, we obtain a circuit of size $p^{(d)}(n)$ for some polynomial p after iterating for each level of threshold gates in C_{hard}.

The conclusion of (ii) is a contradiction – we have constructed a circuit of size $\mathrm{poly}(n)$ for computing C_{hard} although it should require size 2^n.

Parameterized Proof Theorem 1 follows the same strategy but with the size, depth, and succinctness of the assumed circuits for the permanent parameterized as $s(n)$, $d(n)$, and $a(n)$ respectively. Then the circuit C_{hard} is of size, depth, and succinctness $s(m)$, $d(m)$, and $O(a(m)+$ poly-log$(s(m)))$, for $m = s(2^{O(n)})$. The size of C_{hard} is $s(s(2^{O(n)}))$ rather than just $s(2^{O(n)})$ because the assumed easiness of the permanent is used twice to reduce E^{PP} to a threshold circuit. The term $a(m)+$poly-log$(s(m))$ appears in the inequality of Theorem 1 because the PP questions in (ii) for the threshold gates must refer to a particular gate in C_{hard} – which requires the $O(a(m)) +$ poly-log$(s(m))$ bits of succinctness for constructing C_{hard}. The circuit size s is composed with itself $d(m)$ times in the inequality because the process is iterated for each level of threshold gates in C_{hard}.

Permanent or NP For Theorem 2, we look more closely at how the easiness of the permanent is used. In (i) we use the fact that there is a hard language in E^{PP}. For Theorem 2 we instead use a hard language in E^{Σ_2}, meaning assuming NP is easy is enough to obtain the large threshold circuit C_{hard}. In (ii) we only use the assumption that the permanent has a small circuit – the depth and amount of succinctness do not matter. These two observations give Theorem 2.

Hardness of E^{PP} We give an argument for Theorem 3 that is more direct than arguments that have previously been given, showing that there is a language in $\mathrm{DTIME}^{\mathrm{PP}}(\mathrm{poly}(h(n)))$ that does not have circuits of size $h(n)$ for $n \leq h(n) < 2^n$. Consider input length n. The main idea is to pick an input, compute the output of all size $h(n)$ circuits on this input, and choose the output to differ from at least half; then repeat this on a new input, differing from at least half of the remaining size $h(n)$ circuits; continue for $h(n)+1$ iterations to differ from all circuits of size $h(n)$. $h(n)+1$ iterations are enough because for the definition of circuit size that we use (string length of the circuit's description) there are at most $2^{h(n)}$ circuits of size $h(n)$. The diagonalizing machine only needs to be able to determine the majority answer of $2^{h(n)}$ computations. In other words, the power of counting is needed, so that the appropriate output can be chosen using a PP oracle.

We point out that this diagonalization strategy has been used before, e.g., in [IKW02] to show that for every constant $k > 0$, EXP contains languages that require more than 2^{n^k} time and n^k bits of non-uniform advice.

Comparison with Previous Work Each of the previous works proving super-polynomial lower bounds for the permanent on small-depth threshold or arithmetic circuits [All99, KP09, JS12] includes a component similar to step (ii) above – the assumed easiness of the permanent is used to iteratively shrink a large threshold circuit. [All99] and [JS12] phrase that portion of their argument as collapsing the counting hierarchy under the assumed easiness of the permanent. This is equivalent to collapsing a large threshold circuit due to the equivalence between exponential-size uniform constant-depth threshold circuits and the counting hierarchy.

[All99] shows unconditionally that for s satisfying $s^{(O(1))} < 2^n$ the counting hierarchy contains a language that does not have uniform constant-depth threshold circuits of size $s(n)$. If the permanent has uniform constant-depth threshold circuits of size s, then the counting hierarchy collapses (in a way similar to our step (ii)) to size $s^{(O(1))}(n)$ uniform constant-depth threshold circuits – a contradiction if $s^{(O(1))}(n) < 2^n$.

[JS12] uses the collapse of the counting hierarchy under the assumed easiness of the permanent within a framework involving hitting sets for polynomials whose coefficients are computed by constant-depth arithmetic circuits. If the permanent is easy then there is a polynomial that avoids the hitting set and has coefficients computable in the counting hierarchy. The collapse of the counting hierarchy under the assumed easiness of the permanent then shows that the coefficients of the polynomial can be computed more efficiently than should be possible. The complete proof also uses machinery to translate between arithmetic and threshold circuits.

[KP09] uses an outline that is very similar to ours. If the permanent is easy then E has small-depth threshold circuits of size $2^{O(n)}$ and depth $o(\log n)$. These threshold circuits are then collapsed in a way that is similar to our step (ii) above, reaching a contradiction that E can be computed by subexponential size uniform threshold circuits (and thus in subexponential time).

Each of the earlier works uses a step similar to our step (ii) to contradict a known separation. Each work differs in the known separation that is contradicted, and the choice of separation to base the argument on effects some portions of the argument. The separations used by [All99] and [KP09] are uniform separations, meaning care must be taken to keep track of the uniformity

of the circuit that results from step (ii). By using a non-uniform separation, we do not need to keep track of the uniformity, resulting in a simpler argument. Using a non-uniform separation is also required for obtaining hardness against non-uniform circuits.

We have also stated our result as a tradeoff between the different parameters – size, depth, and non-uniformity – which previous works have not done.

1.3 Alternate Proof of Our Results

After completing our work, we learned that results equivalent to Corollary 1 were obtained independently by Chen and Kabanets [CK12]. The main results of [CK12] at first glance look slightly different than ours because the amount of non-uniformity in their statements is parameterized as a function of $\log s(n)$ rather than in terms of n, but an examination shows that our Theorem 1 implies Theorems 1.1, 1.2, and 1.3 of [CK12]. Not only are our results the same as [CK12], but the overall proof structure is similar. The main difference is that [CK12] uses L_{hard} resulting from the time hierarchy for threshold Turing machines with $o(n)$ bits of advice, whereas we use a non-uniformly hard language in the exponential time counting hierarchy. Using the different hard languages results in some differences between the two proofs.

2 Preliminaries

Complexity Classes We assume the reader is familiar with standard complexity classes and notions such as PP, PPTIME, TC^0, AC^0, P, EXP, and DTIME. We refer to standard texts such as [Gol08, AB09] for precise definitions and background.

Circuits In our results and proofs, all circuits are Boolean circuits. A *Boolean circuit* is a directed acyclic graph with each internal node labeled as an AND, OR, or NOT gate and with each root node labeled as either some input bit x_i or one of the constants 0 or 1. Without loss of generality, we assume NOT gates are pushed to the inputs (this can be done by at most doubling the size of the circuit), so that a circuit consists of only AND and OR gates, taking $x_1, \neg x_1, x_2, \neg x_2, ..., x_n, \neg x_n$ as input. One of the leaf nodes is labeled as the output of the circuit, and this output is computed in the natural way. The depth of the circuit is the length of the longest path from an input to the output.

A *threshold circuit* may additionally have majority gates of arbitrary fan-in. As majority gates can be used in place of AND and OR gates, without loss of generality all gates in a threshold circuit are majority gates.

We use the convention that the *size of a circuit* is the string length of its description. Thus the number of circuits of size n is at most 2^n. This makes the analysis cleaner than using the number of gates or wires and only effects results by polylogarithmic factors. Because there are exactly 2^{2^n} Boolean functions on n bits, it is immediate that there exists a language that requires circuits of size at least 2^n.

Succinct Circuits Our results concern a notion of non-uniformity termed *succinctness* that was introduced in [JS11]. Succinctness is a natural notion of non-uniformity for circuit classes that are only slightly non-uniform. A circuit family $\{C_i\}_{i\in\mathbb{N}}$ is $a(n)$-*succinct* if for each n, there is a circuit Q_n that is of size $a(n)$ and correctly answers queries about the connections in C_n. The standard notion of uniform constant-depth circuits is Dlogtime uniformity; a circuit family is *Dlogtime uniform* if there is a Turing machine that correctly answers queries about the connections in C_n and runs in time linear in its input length (and thus logarithmic in the size of the circuit). Note that Dlogtime uniform circuits are necessarily poly-log-succinct. It is

for this reason that poly-log terms appear frequently in our analysis – poly-log succinctness is the minimum needed to perform operations that Dlogtime-uniform circuits can perform.

Permanent and PP The only property of the permanent needed for our results is PP-hardness. [Zan91], building on [Val79], implies that any language in PPTIME(n) reduces to the 0-1 permanent with a quasi-linear size uniform AC^0 reduction, where quasi-linear means $n \cdot \text{poly-log}(n)$.

For the main part of the proof of Theorem 1, we use a different PP-complete language, $L_{\text{PP}} = \{x, M, 1^t |$ the probabilistic machine M runs in time at most t on input x and the majority of computation paths are accepting$\}$. The advantage of this language is that any PPTIME(n) language reduces to L_{PP} in linear time, so we can avoid polylog factors in the analysis by only reducing to the permanent at the very end of our proof. L_{PP} is contained in quasi-linear time PP, and by the result mentioned above reduces to instances of the permanent of quasi-linear size.

3 A Known Circuit Lower Bound

In this section we give an argument for Theorem 3 that is more direct than arguments that have previously been given. We refer to Sections 1.1 and 1.2 for discussion of previous proofs of this result.

Theorem 3 (restated). *Let $h(n)$ be a time-constructible function such that for all n, $n \leq h(n) < 2^n$. There is a language L_{hard} in $\text{DTIME}^{\text{PP}}(\text{poly}(h(n)))$ that does not have circuits of size $h(n)$.*

Proof. Let $x_1, ..., x_{h(n)+1}$ be the $h(n)+1$ lexicographically smallest inputs of length n. The PP language we use as oracle is

$$O = \{(1^n, j, b_1, ..., b_{h(n)+1}) |\ C(x_j) = b_j \text{ for at most } 1/2$$

$$\text{of the circuits } C \text{ of size } h(n) \text{ with } C(x_i) = b_i \text{ for all } 1 \leq i < j.\}$$

O can be decided in PP by a machine as follows. The machine guesses a circuit of size $h(n)$; if the circuit does not agree with one of the b_i between 1 and $j-1$ then the PP machine splits into two nondeterministic paths with one accepting and one rejecting; otherwise the PP machine accepts iff $C(x_j) \neq b_j$. Then there are at least half accepting paths iff at least half of the circuits in question disagree with b_j on x_j. As we can evaluate a circuit of size $h(n)$ in $\text{poly}(h(n))$ time, the running time for O is $\text{poly}(h(n))$, which is polynomial in the input length, so $O \in \text{PP}$.

L_{hard} is defined as follows. $L_{hard}(x_1) = O(1^n, 1, 0, 0, ..., 0)$, and already L_{hard} differs from at least half of the circuits of size $h(n)$. $L_{hard}(x_2) = O(1^n, L_{hard}(x_1), 1, 0, ..., 0)$. So now L_{hard} differs from at least 3/4 of the circuits of size $h(n)$. And so on. As there are at most $2^{h(n)}$ circuits of size $h(n)$, we will have differed from all in at most $h(n)+1$ steps. For inputs not in the set $\{x_1, ..., x_{h(n)+1}\}$ we can define L_{hard} arbitrarily (e.g., set it to 0). Notice that L_{hard} can be decided in $O((h(n))^2)$ time with access to the PP oracle O.

3.1 Lower Bound for PP

Theorem 3 can be used to simplify the proof of the result that for any constant $k > 0$, PP does not have circuits of size n^k [Vin05]. Suppose PP has polynomial size circuits (otherwise we have nothing to prove). By [LFKN92], $\text{PP} \subseteq \text{MA}$, and also $\text{P}^{\text{PP}} \subseteq \text{MA}$. It is known that $\text{MA} \subseteq \text{PP}$, so $\text{P}^{\text{PP}} = \text{MA} = \text{PP}$. By Theorem 3, for any $k > 0$ there is a language in P^{PP} that does not

have circuits of size n^k, and since $\mathrm{P}^{\mathrm{PP}} = \mathrm{PP}$ this language is in PP as well. We have that either PP does not have polynomial size circuits, or PP does not have circuits of size n^k.

The original proof of [Vin05] is similar to this proof. The main difference is that the earlier proof uses Toda's theorem [Tod91b] and circuit lower bounds in the polynomial hierarchy [Kan82] in place of Theorem 3.

4 Lower Bounds for Permanent

In this section we prove our main results, Theorem 1, Corollary 1, and Theorem 2. Theorem 1 is proved in Section 4.1 subject to two claims, which are proved in Sections 4.2 and 4.3. Theorem 2 is proved in Section 4.4

4.1 Proof of Theorem 1

Theorem 1 (restated). *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. Let $s'(n) = s(n^{O(1)})$, and $m = s(2^{O(n)})$. If $s'^{(d(m))}(n + a(m) + \text{poly-log}(s(m))) < 2^n$ then L does not have depth $d(n) - O(1)$, $(a(n) - \text{poly-log}(s(n)))$-succinct threshold circuits of size $s(n)/\text{poly-log}(s(n))$.*

To prove Theorem 1, we combine the hard language L_{hard} resulting from Corollary 2 with the following two claims. Let L_{PP} be the PP-complete language defined in Section 2

Claim 1. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. If L_{PP} has $a(n)$-succinct threshold circuits of depth $d(n)$ and size $s(n)$ then L_{hard} has threshold circuits of size $s(m)$, depth $d(m)$, and is $O(a(m)) + \text{poly-log}(s(m)))$-succinct, for $m = s(2^{O(n)})$.*

Proving Claim 1 amounts to plugging in the assumed circuit for L_{PP} into the E^{PP} computation of L_{hard}.

Claim 2. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. If L_{PP} has $a(n)$-succinct threshold circuits of depth $d(n)$ and size $s(n)$ then L_{hard} has size $s'^{(d(m))}(n + a(m) + \text{poly-log}(s(m)))$ circuits for $s'(n) = s(n^{O(1)})$ and $m = s(2^{O(n)})$.*

To prove Claim 2, we use the threshold circuit from Claim 1 and shrink it by using the easiness of L_{PP} to collapse the threshold gates iteratively. This is the step that is at the heart of all previous papers [All99, KP09, JS11, JS12] proving lower bounds for the permanent on small-depth threshold or arithmetic circuits.

If the size of the circuit for L_{hard} in Claim 2 is less than 2^n, we conclude that L_{PP} cannot have $a(n)$-succinct threshold circuits of depth $d(n)$ and size $s(n)$. The statement of Theorem 1 follows by the quasi-linear size uniform AC^0 reduction from L_{PP} to the permanent: if the permanent has depth $d(n)$ threshold circuits of size $s(n)$ and succinctness $a(n)$, then L_{PP} has depth $d(n)+O(1)$ threshold circuits of size $s(n)\,\text{poly-log}(s(n))$ and succinctness $a(n) + \text{poly-log}(s(n))$.

All that remains is to prove the claims.

4.2 Proof of Claim 1

We take the E^{PP} computation of L_{hard} of Corollary 2. First consider the PP oracle from the definition of L_{hard}. The oracle O in the proof of Theorem 3 is computable in polynomial

PPTIME, and the instances we need are of size $O(2^n)$. The oracle queries can thus be translated to queries to $L_{\rm PP}$ of size $N = 2^{O(n)}$ [2]. Given the assumed threshold circuits for $L_{\rm PP}$, the oracle queries can be decided by a depth $d(N)$ threshold circuit $C_{\rm PP}$ of size $s(N)$ and succinctness $a(N) + \text{poly-log}(s(N))$.

Deciding membership in L_{hard} amounts to querrying the oracle O on at most 2^n inputs. This gives an oracle circuit that makes exponentially many adaptive queries to O. In this circuit we replace each oracle gate with the circuit $C_{\rm PP}$, obtaining a single circuit deciding L_{hard} that is of size $\text{poly}(2^n \cdot s(N))$ that requires $a(N) + \text{poly-log}(s(N))$ bits of succinctness. This circuit can be viewed as a circuit value problem of size $m' = \text{poly}(2^n \cdot s(N))$. Because P $\subseteq$ PP and $L_{\rm PP}$ is complete for PP, this circuit value problem reduces to an instance of $L_{\rm PP}$ of size $m = O(m')$. Using the assumed easiness of $L_{\rm PP}$, such instances can be solved by threshold circuits of depth $d(m)$ and size $s(m)$. The amount of succinctness needed throughout the reductions is $a(N)+\text{poly-log}(s(N))+a(m)+\text{poly-log}(s(m))$, which is $O(a(m)+\text{poly-log}(s(m)))$. The statement of Claim 1 results from simplifying the expression for m to $s(2^{O(n)})$ using the fact that $s(n) \geq n$ and both s and a are non-decreasing.

4.3 Proof of Claim 2

Main Idea For each input length n, we aim to build a circuit for L_{hard} at input length n. With the assumed easiness of $L_{\rm PP}$ and using Claim 1, we have a threshold circuit C_{hard} for L_{hard} with size $s(m)$, depth $d(m)$, and succinctness $O(a(m) + \text{poly-log}(s(m)))$, for $m = s(2^{O(n)})$. The plan is to shrink C_{hard} by viewing the threshold gates as small PP questions and using the assumed easiness of $L_{\rm PP}$ to collapse the gates. We do this iteratively level by level in the circuit. The proof consists mostly of keeping track of the size of the circuit produced as a result of this process.

Iterative Shrinking of C_{hard} For each i, we define the following language.

$$L_i = \{(x, j) | \text{ on input } x \text{ of length } n, \text{ gate } j \text{ in } C_{hard} \text{ is at depth } i \text{ and outputs } 1\}$$

The value j is padded to n bits to ensure that each input length of L_i regards a single value of n. We iteratively construct circuits for input length $2n = |(x, y)|$ for $L_1, L_2, ..., L_{d(m)}$. For each i, we use the circuit constructed for L_i to build the circuit for L_{i+1}. The final circuit for $L_{d(m)}$ at length $2n$ corresponds to the output gate of C_{hard} for inputs x of length n.

First level of Threshold Gates First consider L_1, corresponding to the first level of threshold gates in C_{hard}. Given input (x, j) with x of length n, a PP machine determines the output of gate j as follows.

1. Use $O(a(m) + \text{poly-log}(s(m)))$ bits of advice as the succinctness for C_{hard} to verify that j is a gate at depth 1, and if not split into a rejecting and an accepting state.
2. Nondeterministically guess an input label k and use the advice from 1. as the succinctness for C_{hard} to verify k is an input to gate j; if not split into a rejecting and an accepting state.
3. Accept iff the input bit labeled by k is 1.

This PP computation has a majority of accepting computation paths iff j is a gate at depth 1 and the majority of the inputs to gate j are 1. The amount of time for the computation

[2] We can assume all queries are the same size because $L_{\rm PP}$ is paddable – queries of smaller length can be made longer to match the longest query.

is poly$(n + a(m) +$ poly-log$(s(m)))$, and we have used $O(a(m) +$ poly-log$(s(m)))$ bits of non-uniformity. This PP computation can be reduced to an instance of L_{PP} of size poly$(n + a(m) +$ poly-log$(s(m)))$. By the assumed easiness of L_{PP}, and including the $O(a(m) +$ poly-log$(s(m)))$ bits of non-uniformity, these instances are solved by a threshold circuit C_1 of size

$$S_1 = s(\text{poly}(n + a(m) + \text{poly-log}(s(m)))) + O(a(m) + \text{poly-log}(s(m))).$$

Because our ultimate goal is a non-uniform, arbitrary-depth circuit for L_{hard} we do not need to keep track of the depth and amount of succinctness in this circuit.

Level $i + 1$ of Threshold Gates Now consider L_{i+1} assuming we have a circuit C_i for L_i. Given an input (x, j), a PP machine can determine the correct output of gate j as follows.

1. Use $O(a(m) +$ poly-log$(s(m)))$ bits of advice as the succinctness for C_{hard} to verify that j is a gate at depth $i + 1$, and if not split into a rejecting and an accepting state.
2. Nondeterministically guess a gate label k and use the advice from 1. as the succinctness for C_{hard} to verify k is an input to gate j; if not split into a rejecting and an accepting state.
3. Accept iff C_i indicates that k outputs 1, namely if $C_i(x, k) = 1$.

This PP computation computes L_{i+1} just as in the case for L_1 above. The amount of time for the computation is poly$(n + a(m) +$ poly-log$(s(m)) + |C_i|)$, and we have used $O(a(m) +$ poly-log$(s(m))) + |C_i|$ bits of non-uniformity. This PP computation can be reduced to an instance of L_{PP} of size poly$(n + a(m) +$ poly-log$(s(m)) + |C_i|)$. By the assumed easiness of L_{PP}, letting $S_i = |C_i|$, and including the $O(a(m) +$ poly-log$(s(m))) + |C_i|$ bits of non-uniformity, we have a circuit C_{i+1} for L_{i+1} that is of size

$$S_{i+1} = s(\text{poly}(n + a(m) + \text{poly-log}(s(m)) + S_i)) + O(a(m) + \text{poly-log}(s(m))) + S_i.$$

Simplifying the Expression for the Circuit Size Let us simplify the formula for S_i. First, S_1 can be simplified as $S_1 = s(\text{poly}(n + a(m) + \text{poly-log}(s(m))))$ using the fact that s and a are non-decreasing and $s(M) \geq a(M)$. For similar reasons, S_{i+1} can be written as $s(\text{poly}(n + a(m) +$ poly-log$(s(m) + S_i)))$. Since s and a are non-decreasing, S_1, S_2, ..., S_i is non-decreasing so that $n + a(m) +$ poly-log$(s(m)) \leq S_i$ for each i. We can thus rewrite S_{i+1} as $s(\text{poly}(S_i))$. Letting $s'(M) = s(M^c)$ for large enough constant c, we have that $S_{i+1} = s'^{(i)}(n + a(m) +$ poly-log$(s(m)))$. C_{hard} is computed at level $d(m)$, so by a circuit of size $s'^{(d(m))}(n + a(m) +$ poly-log$(s(m)))$.

4.4 Proof of Theorem 2

In this section we observe that Theorem 1 can be strengthened by examining the proof more carefully, proving Theorem 2.

Theorem 2 (restated). *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. Let $s'(n) = s(n^{O(1)})$, and $m = s(s(2^{O(n)}))$. If $s'^{(d(m))}(n + a(m) +$ poly-log$(s(m))) < 2^n$ then either*

- SAT *does not have depth $d(n) - O(1)$, $(a(n) -$ poly-log$(s(n)))$-succinct threshold circuits of size $s(n)/$ poly-log$(s(n))$, or*
- *The permanent does not have non-uniform general circuits of size $s(n)/$ poly-log$(s(n))$.*

The easiness of the PP-hard language L_{PP} is used in the proof of Theorem 1 for two key purposes.

(i) Corollary 2 and Claim 1 show that if L_{PP} has small-depth threshold circuits, there is a hard language L_{hard} with large small-depth threshold circuits.

(ii) Claim 2 shows that if L_{PP} has small circuits, the circuit from (i) can be iteratively made smaller.

For step (i), we can replace L_{PP} by any language that, if assumed to have small-depth threshold circuits, implies a small-depth threshold circuit for a language with high circuit complexity. For example, we can use an NP-complete language and the following fact.

Theorem 4 ([Kan82, MVW99]). *There is a language L_{hard} in $\mathrm{DTIME}^{\Sigma_2^p}(2^{O(n)})$ that does not have circuits of size $2^n - 1$.*

Using the NP-complete language $L_{\mathrm{NP}} = \{(x, M, 1^t) |$ the nondeterministic machine M runs in time at most t on input x with an accepting path$\}$, Claim 1 becomes instead the following.

Claim 3. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. If L_{NP} has $a(n)$-succinct threshold circuits of depth $d(n)$ and size $s(n)$ then L_{hard} has threshold circuits of size $s(m)$, depth $d(m)$, and is $O(a(m)+\text{poly-log}(s(m)))$-succinct, for $m = s(s(2^{O(n)}))$.*

The change in the value of m is due to working in the third level of the exponential alternating hierarchy, whereas in Claim 1 the hard language was in the second level of the exponential counting hierarchy.

For step (ii), the proof only requires that the PP-hard language L_{PP} has small general circuits – the small-depth and succinctness restrictions are not used in the argument.

Combining these two observations, we have a result stating that if both (1) an NP-hard language has small succinct small-depth threshold circuits, and (2) a PP-hard language has small general circuits, then L_{hard} has small circuits. Specifically, we have the following claim in place of Claim 2. For conciseness we have assumed the same size for both L_{NP} and L_{PP}; a more general statement could be made that implies a tradeoff between the assumed circuit sizes for the two different languages.

Claim 4. *Let $s(n)$, $a(n)$, and $d(n)$ be non-decreasing functions with $s(n)$ time-constructible and $d(n), a(n) \leq s(n)$ for all n. If L_{NP} has $a(n)$-succinct threshold circuits of depth $d(n)$ and size $s(n)$ and L_{PP} has circuits of size $s(n)$ then L_{hard} has size $s'^{(d(m))}(n + a(m) + \text{poly-log}(s(m)))$ circuits for $s'(n) = s(n^{O(1)})$ and $m = s(s(2^{O(n)}))$.*

If the resulting circuit is of size less than 2^n, then the assumed circuits for either L_{NP} or L_{PP} must not exist. Including reductions from L_{NP} to SAT and from L_{PP} to the permanent results in the parameters of Theorem 2.

5 Discussion

In this section we discuss some possible extensions of the lower bounds for the permanent on small-depth threshold circuits.

Non-Uniformity A natural question is if our techniques allow the $n^{o(1)}$ amount of non-uniformity in Corollary 1 to be pushed any higher. It seems progress in this direction will need new ideas and/or a new framework. The framework used in this and previous papers all encounter a roughly inverse relationship between the size of circuits in the lower bound and the amount of non-uniformity that can be handled. In Theorem 1 hardness holds if the inequality

stated in the theorem holds. The inequality requires that $s(a(2^n)) < 2^n$ and more generally requires $a(m)$ to be an inverse of $s^{(d(m))}$. This arises in the proof due to the nature in which the assumed easiness of the permanent is used repeatedly in Claim 2, and a similar issue arises in earlier work in this area [All99, KP09, JS12].

Furthermore, the proofs of our main results relativize, but it is known that proving results with larger non-uniformity, say $\geq n$ bits, requires non-relativizing techniques. Thus to make progress we ought to look at utilizing techniques such as the interactive proofs for the permanent, random self-reducibility, and combinatorial properties of threshold circuits.

Average-Case Hardness Another direction that is natural to consider is whether an average-case hardness result can be proved for the permanent on constant-depth threshold circuits. This is an intriguing question because of the worst-case to average-case reductions known for the permanent. The standard random self-reduction views the permanent as a low-degree polynomial and uses polynomial interpolation, which can be done by uniform TC^0 circuits [HAB02]. But the reduction is randomized and in particular uses at least a linear amount of randomness. This could be translated into a linear amount of advice, but we have just seen our proof cannot handle this much advice.

Almost Everywhere Hardness Can it be shown that threshold circuits must fail to compute permanent and other PP-hard languages on almost every input length? Consider the argument given in Section 4 if the PP-hard language L_{PP} is assumed to be easy only on infinitely many input lengths. Theorem 3 and Corollary 2 go through with no problem, giving that L_{hard} is hard on infinitely many input lengths. But Claims 1 and 2 encounter difficulties because the argument uses the easiness of permanent on multiple different input lengths (lengths N and m from Claim 1 to obtain a large threshold circuit for L_{hard}, and roughly $d(m)$ input lengths in Claim 2 to shrink that circuit by looking at the threshold gates in the circuit). If the permanent is not easy on all these input lengths, the argument fails.

This situation is common to many other lower bounds that use indirect or multi-step arguments and has been discussed in [FS11].

Acknowledgments

This research was partially supported by Indiana State University, University Research Council grants #11-07 and #12-18. We thank Matt Anderson, Dieter van Melkebeek, and Dalibor Zelený for discussions that began this project, continued discussions since, and comments on early drafts of this work. We thank Matt in particular for observations that refined the statement of Theorem 2. We also thank the reviewers for comments and suggestions that improved the exposition of the paper.

References

[Aar06] Scott Aaronson. Oracles are subtle but not malicious. In *Proceedings of the IEEE Conference on Computational Complexity (CCC)*, pages 340–354, 2006.

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

[All96] Eric Allender. Circuit complexity before the dawn of the new millennium. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 1–18, 1996.

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

[AM11] Scott Aaronson and Dieter van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.

[AS09] Manindra Agrawal and Ramprasad Saptharishi. Classifying polynomials and identity testing. *Current Trends in Science, Platinum Jubilee Special*, pages 149–162, 2009.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[CK12] Ruiwen Chen and Valentine Kabanets. Lower bounds against weakly uniform circuits. In *Proceedings of the Annual International Computing and Combinatorics Conference (COCOON)*, pages 408–419, 2012.

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM Journal on Computing*, 39(4):1279–1293, 2009.

[FS11] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 569–580, Berlin, Heidelberg, 2011. Springer-Verlag.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Theory of Computing Systems*, 17:13–27, 1984.

[Gol08] Oded Goldreich. *Complexity Theory: A Conceptual Perspective*. Cambridge University Press, 2008.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002.

[Hås87] Johan Håstad. *Computational Limitations of Small-Depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

[JS11] Maurice Jansen and Rahul Santhanam. Permanent does not have succinct polynomial size arithmetic circuits of constant depth. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6755 of *Lecture Notes in Computer Science*, pages 724–735. Springer, 2011.

[JS12] Maurice Jansen and Rahul Santhanam. Marginal hitting sets imply super-polynomial lower bounds for permanent. In *Innovations in Theoretical Computer Science*, 2012.

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

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

[KMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization and circuit lower bounds. In *Computational Complexity*, volume 21, pages 3–61, 2012.

[KP09] Pascal Koiran and Sylvain Perifel. A superpolynomial lower bound on the size of uniform non-constant-depth threshold circuits for the permanent. In *Proceedings of the IEEE Conference on Computational Complexity (CCC)*, pages 35–40, 2009.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[MVW99] Peter Bro Miltersen, N. Variyam Vinodchandran, and Osamu Watanabe. Superpolynomial versus half-exponential circuit size in the exponential hierarchy. In *Proceedings of the Annual International Computing and Combinatorics Conference (COCOON)*, pages 210–220, 1999.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[Raz85] Alexander Razborov. Lower bounds on the monotone complexity of some Boolean function. *Dolk. Akad. SSSR*, 281(4):598–607 (in Russian), 1985. English translation in *Soviet Math. Dokl.* 31 (1985), 354–357.

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

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3–4):207–388, 2010.

[Tod91a] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tod91b] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

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

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theoretical Computer Science*, 347(1-2):415–418, 2005.

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

[Yao85] Andrew C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1985.

[Zan91] Viktoria Zanko. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2(1):77–82, 1991.