Electronic Colloquium on Computational Complexity, Report No. 135 (2011)

Stronger Lower Bounds and Randomness-Hardness Trade-Offs Using Associated Algebraic Complexity Classes

Maurice Jansen[*]

Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh
maurice.julien.jansen@gmail.com

Rahul Santhanam[†]

Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh
rsanthan@inf.ed.ac.uk

October 9, 2011

Abstract

We associate to each Boolean language complexity class $\mathcal{C}$ the algebraic class a$\cdot\mathcal{C}$ consisting of families of polynomials $\{f_n\}$ for which the evaluation problem over $\mathbb{Z}$ is in $\mathcal{C}$. We prove the following lower bound and randomness-to-hardness results:

1. If polynomial identity testing (PIT) is in NSUBEXP then a$\cdot$NEXP does not have *poly* size constant-free arithmetic circuits.
2. a$\cdot$NEXP$^{\mathrm{RP}}$ does not have *poly* size constant-free arithmetic circuits.
3. For every fixed k, a$\cdot$MA does not have arithmetic circuits of size n^k.

Items 1 and 2 strengthen two results due to Kabanets and Impagliazzo [6]. The third item improves a lower bound due to Santhanam [11].

We consider the special case low-PIT of identity testing for (constant-free) arithmetic circuits with low formal degree, and give improved hardness-to-randomness trade-offs that apply to this case.

Combining our results for both directions of the hardness-randomness connection, we demonstrate a case where derandomization of PIT and proving lower bounds are *equivalent*. Namely, we show that low-PIT $\in$ i.o-NTIME$[2^{n^{o(1)}}]/n^{o(1)}$ if and only if there exists a family of multilinear polynomials in a$\cdot$NE$/lin$ that requires constant-free arithmetic circuits of super-polynomial size and formal degree.

[*]Supported by EPSRC Grant H05068X/1
[†]Supported in part by EPSRC Grant H05068X/1

ISSN 1433-8092

1 Introduction

In this paper we study the arithmetic circuit complexity of families of multivariate polynomials $\{f_n\}$ in terms of the computational hardness of the underlying evaluation problem. Towards this end we associate to each Boolean language complexity class $\mathcal{C}$ the class a$\cdot\mathcal{C}$ consisting of all families of polynomials $\{f_n\}$ with integer coefficients, such that given an integer input tuple x to f_n, an integer i and a bit b, it can be decided within the resources of the class $\mathcal{C}$ whether the ith bit of $f_n(x)$ equals b. We restrict the number of variables, the degree, and the bit size of coefficients of such families to be polynomially bounded in n (See Section 2 for the formal definition). We note that a similar notion was suggested by Koiran and Perifel [9].

One of our main motivations is to find an elegant way to state "hybrid" results involving Boolean and arithmetic circuit lower bounds, such as the trade-offs of Kabanets and Impagliazzo [6] or the lower bound of Santhanam [11]. These are examples where people, perhaps unknowingly, have been proving lower bounds and randomness-hardness tradeoffs geared towards associated algebraic classes, while in our opinion lacking the proper language to describe, and consequently interpret, the results. The a$\cdot\mathcal{C}$ notions provides a language for succinctly expressing these results, and leads to natural questions for making improvements. Consequently we have strengthened several important results from the literature.

A prime example of the above situation is the celebrated theorem by Kabanets and Impagliazzo [6], which says that if polynomial identity testing (PIT) is in NSUBEXP, then either NEXP $\not\subseteq$ P/*poly*, or permanent does not have poly-size arithmetic circuits. PIT is the problem of deciding for a given arithmetic circuit Φ whether it computes the zero polynomial. We refer to a recent survey by Saxena [12] for more on this problem. The quoted theorem tells us that derandomization of polynomial identity testing yields lower bounds of *some* sort. However, it doesn't tell us whether these will be Boolean lower bounds or arithmetic lower bounds. We make the observation (proof in Appendix A) that the theorem by Kabanets and Impagliazzo is equivalent to the statement PIT $\in$ NSUBEXP $\Rightarrow$ a$\cdot$NEXP/*lin* $\not\subseteq$ ASIZE$'$(*poly*). Here ASIZE$'$(*poly*) denotes the class of polynomial families $\{f_n\}$ computable by constant-free arithmetic circuit of size *poly*(n) using addition, multiplication, and division by constants (See Section 2). Hence, to answer the above question, putting PIT in NSUBEXP gives *arithmetic* lower bounds for families of polynomials that can be evaluated in NEXP with linear advice. A natural improvement to the result would be to drop the linear advice. We show that this can indeed be done[1], resulting in the following stronger theorem:

Theorem 1. PIT $\in$ NSUBEXP $\Rightarrow$ a$\cdot$NEXP $\not\subseteq$ ASIZE$'$(*poly*).

In the above, on the right hand side, the associated algebraic class gives us a measure of the explicitness of the lower bound. We have improved this explicitness from evaluable in NEXP/*lin* down to evaluable in NEXP. As it is generally undesirable to have non-uniform dependencies appearing in the explicitness measure of a lower bound, the main significance of our result is that we have managed to remove the non-uniformity.

Similar to Theorem 1 we observe that Theorem 5.2 of Ref. [6], which states that either NEXP$^{\mathrm{RP}}$ $\not\subset$ P/*poly* or Permanent does not have poly-size arithmetic circuits, is equivalent to the statement that a$\cdot$NEXP$^{\mathrm{RP}}$/*lin* $\not\subseteq$ ASIZE$'$(*poly*) (See Appendix A). We also improve the explicitness of this lower bound and obtain that

Theorem 2. a$\cdot$NEXP$^{\mathrm{RP}}$ $\not\subseteq$ ASIZE$'$(*poly*).

[1]An obvious way to do this would be to 'just' show that a$\cdot$NEXP $\subseteq$ ASIZE$'$(*poly*) $\Rightarrow$ a$\cdot$NEXP/*lin* $\subseteq$ ASIZE$'$(*poly*). It is not clear whether this is true, cf. Appendix A.

Furthermore, we improve a theorem by Santhanam [11] which states that for every k, either MA $\not\subseteq$ SIZE(n^k), or there exists a family polynomial $\{f_n\}$, whose graph is decidable in MA, that is not in ASIZE(n^k). We show the following stronger result:

Theorem 3. *For every k,* a·MA $\not\subseteq$ ASIZE(n^k)*.*

The above results demonstrate the usefulness of the a·$\mathcal{C}$ notion. There are further reasons why the notion is worth exploring. It gives a way of bringing uniformity into the algebraic complexity setting. Note that traditional algebraic complexity classes such as VP and VNP are inherently non-uniform. We also feel the notion could facilitate more interactions of techniques from structural complexity and algebraic complexity. Given how few lower bound techniques we have available, and given the well-known barriers such as natural proofs and algebrization to finding new ones, we need to make the best use of the ones we have. The recent lower bounds success of Williams [17] is an instructive example of how known techniques from different domains can be combined to give an interesting new result.

In general, one might ask for any known separation $\mathcal{C} \not\subseteq \mathcal{D}$ in the Boolean world whether it can be strengthened to show that a·$\mathcal{C} \not\subseteq$ a·$\mathcal{D}$. Note that this would indeed be a strengthening as $\mathcal{C} \subseteq \mathcal{D}$ trivially implies a·$\mathcal{C} \subseteq$ a·$\mathcal{D}$. Arithmetic analogues of time hierarchy results, eg., a·DTIME[n^2] $\subsetneq$ a·DTIME[n^3] and a·NTIME[n^2] $\subsetneq$ a·DTIME[n^3] can be proved quite easily using the fact that the separation can be witnessed by a unary language. However, we don't know whether arithmetic analogues of results such as Williams' lower bound hold. There could be a connection between proving the arithmetic analogue of a Boolean result and whether the techniques used to prove the Boolean result algebrize in the sense of Aaronson and Wigderson [1]. We have not properly explored this yet.

One of the advantages of using associated algebraic classes is that this enables us to derive tighter hardness-randomness trade-offs. This is especially striking for the case of the low-formal-degree polynomial identity testing problem (low-PIT). We define low-PIT as the special case of PIT for circuits Φ whose formal degree $deg(\Phi)$ is less than or equal to the size $|\Phi|$. Formal degree is a syntactic notion, easily computed for a circuit (See Section 2). Examples of types of circuits that automatically satisfy the degree restriction $deg(\Phi) \leq |\Phi|$ are formulas and skew-circuits, the latter being equivalent to algebraic branching programs. This makes low-PIT an important special case of the general problem. We show that we can specialize Theorem 1 to obtain the following low-degree version:

Theorem 4. *low*-PIT $\in$ NSUBEXP $\Rightarrow$ a·NEXP $\not\subseteq$ ASIZEDEG$'$(*poly*)*.*

In the above ASIZEDEG$'$(*poly*) is the class of families of polynomials $\{f_n\}$ computable by constant-free arithmetic circuits of size $poly(n)$ and formal degree $poly(n)$ (The 'prime' indicates that we allow a single division by a previously computed constant at the output gate).

For the special case of low-PIT, we also make progress on trade-offs that go in the opposite direction. Namely, we show that derandomization can be achieved under weaker hardness assumptions than was known previously. For example using our techniques we can prove the following theorem:

Theorem 5. *Suppose there exists a family* $\{p_n\} \in$ ml·NEXP *with* $\{p_n\} \not\in$ i.o-ASIZEDEG$'$($n^{e(n)}$)*, where $e(n)$ is a monotone non-decreasing time constructible function with $e(n) = \omega(1)$. Then low*-PIT $\in$ NTIME[$2^{n^{o(1)}}$]*.*

In the above, ml·NEXP is the subclass of a·NEXP consisting of all families $\{f_n\}$, where each f_n is multilinear. The key improvement[2] that we make here over the techniques of Ref. [6], is

[2]See some remarks about the difference in Section 2.

that we can work with ASIZEDEG$'$-hardness instead of ASIZE$'$-hardness in case we only need to cater for low-PIT. To achieve such improved trade-offs, we prove a so-called root extraction lemma (Lemma 5) that is formal-degree efficient. This lemma, which is of independent interest, is subsequently combined with the framework of Ref. [6]. As an additional twist, we start with a hardness assumption in terms of an associated algebraic class.

Finally, combining our results for both directions of the hardness vs. randomness connection, the work of this paper culminates with the following theorem, which demonstrates a setting where derandomization of PIT and proving lower bounds are *formally equivalent*:

Theorem 6. *There exists a family* $\{p_n\} \in \mathrm{ml{\cdot}NE}/lin$ *with* $\{p_n\} \notin \mathrm{ASIZEDEG}'(s(n))$*, for* $s(n) = n^{\omega(1)}$ *if and only if low-*$\mathrm{PIT} \in \text{i.o-NTIME}[2^{r(n)}]/r(n)$*, for* $r(n) = n^{o(1)}$.

Our paper is the first in the literature where for a specific setting an equivalence is established between proving lower bounds and derandomization of PIT. In this the associated algebraic classes play a central role, which we offer as further evidence of the importance of this notion.

2 Preliminaries

Let $\mathrm{NSUBEXP} = \cap_\epsilon \mathrm{NTIME}[2^{n^\epsilon}]$. We define SIZE$(s(n))$ to be the class of all languages in $\{0,1\}^*$ computable by Boolean circuits of size $s(n)$. A (division-free) arithmetic circuit over some field $\mathbb{F}$ and a set of variables $X = \{x_1, x_2, \ldots, x_n\}$ is given by a labeled directed acyclic graph. Nodes of in-degree zero are labeled with elements of $X \cup \mathbb{F}$. Other nodes are labeled by $+$ or $\times$. To each node, a.k.a. gate, we can associate a polynomial $\in \mathbb{F}[X]$, defined inductively in the obvious way. If constant-labels are restricted to be in $\{-1, 0, 1\}$ the circuit is called constant-free. For the size of an arithmetic circuit we count the number of wires. We define ASIZE$(s(n))$ to be the class of all families of polynomials $\{f_n\}$ with integer coefficients that have constant-free arithmetic circuits of size $s(n)$. We let ASIZE$'(s(n))$ be the class obtained from ASIZE$(s(n)$ by allowing one single division at the output gate by an integer $a \neq 0$, where a has been computed by the circuit. We remark that due to a result by Strassen [14] on avoidance of divisions, cf. Theorem 2.17 and Corollary 3.9 in [6], a family of polynomials $\{f_n\}$ of $poly(n)$ degree can be computed by an arithmetic circuit of $poly(n)$ size *with arbitrary use of division gates* iff $\{f_n\} \in$ ASIZE$'(poly(n))$. We define "infinitely often" versions of these classes in the obvious way. For example i.o-ASIZE$(s(n))$ is the class of families $\{f_n\}$ such that for infinitely many n, f_n can be computed by a size $s(n)$ circuit.

ASIZEDEG$(s(n))$ is obtained from ASIZE$(s(n))$ by adding the restriction that formal degree of the circuit is bounded by $s(n)$ as well. Formal degree is defined inductively as follows. For input gates, regardless of their label, formal degree is 1. Formal degree of an addition gate is taken to be the maximum of the formal degree of its inputs. For multiplication gates one takes the sum of formal degrees of its inputs. We define the class ASIZEDEG$'(poly)$ to be the class of families of polynomials $\{f_n\}$ with integer coefficients such that there exist families $\{g_n\}$ and $\{c_n \in \mathbb{Z}\}$ in ASIZEDEG$(poly)$ with $f_n = g_n/c_n$, for each n.

Families in ASIZE$(poly)$ can have super-polynomial degree, e.g. x^{2^n} can be computed with $n-1$ repeated multiplications. We like to point out that in general for a family $\{f_n\} \in$ ASIZE$(poly)$ with $deg(f_n) = n^{O(1)}$ *it is not known* whether $\{f_n\} \in$ ASIZEDEG$(poly)$. In particular it is a fallacy to think the well-known trick of computing degree components separately at every gate in the circuit proves this, as was pointed[3] out by Bürgisser [3], cf. [8]. Namely, this construction requires a model where arbitrary constants can be used by the circuit at unit

[3]The class ASIZEDEG$(poly)$ is known as VP^0 in the literature, whereas ASIZE$(poly)$ corresponds to families of polynomials with τ-complexity $poly(n)$, cf. Ref. [9].

cost. A similar remark can be made for the classes ASIZEDEG$'(poly)$ and ASIZE$'(poly)$, in which case it is perhaps more obvious that computing components separately does not help, since one of the given circuits only computes a constant.

We define the language corresponding to the polynomial identity testing problem PIT $= \{\Phi : \Phi$ is a division-free constant-free arithmetic circuit such that $\Phi \equiv 0\}$. Similarly, we define low-PIT to be the following language

$$\{\Phi : \Phi \text{ is a division-free constant-free arithmetic circuit of formal degree} \leq |\Phi| \text{ and } \Phi \equiv 0\}.$$

We make use of the well-known Schwartz-Zippel-deMillo-Lipton Lemma.

Lemma 1 ([4, 13, 18])**.** *Let A be an arbitrary nonempty subset of the field $\mathbb{F}$. Then for any nonzero polynomial $f \in \mathbb{F}[X]$ of degree d, $\Pr[f(a_1, a_2, \ldots, a_n) = 0] \leq \frac{d}{|A|}$, where the a_i's are picked independently and uniformly at random from A.*

We use several easily proved propositions.

Proposition 1. *A constant-free division-free arithmetic circuit of size s and formal degree d without variables computes an integer constant of absolute value at most 2^{ds}.*

By hard-wiring inputs we obtain the following corollary:

Corollary 1. *For a constant-free division-free arithmetic circuit of size s and formal degree d computing a polynomial $f(x_1, x_2, \ldots, x_n)$, if we evaluate f on integers $a_1, \ldots, a_n$ of at most B bits, then $|f(a_1, a_2, \ldots, a_n)| \leq 2^{O(dsB^2)}$.*

Proposition 2 (Multilinear Extension over $\mathbb{Z}$)**.** *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function. Define $F(x_1, \ldots, x_n) = \sum_{a \in \{0,1\}^n} f(a) \prod_{i \in [n]} (1 - x_i + a_i(2x_i - 1))$. Then F is a multilinear polynomial with integer coefficients that coincides with f on $\{0,1\}^n$. Furthermore, F is the unique polynomial with these properties.*

We now get to the central definition of this paper.

Definition 1. *Let $\mathcal{C}$ be a language complexity class. Corresponding to $\mathcal{C}$ we have the associated algebraic class* a$\cdot\mathcal{C}$ *which is given by the collection of all polynomial families $\{f_n\}$ defined in $m(n) = n^{O(1)}$ variables of degree poly(n) having integer coefficients of poly(n) bit size such that the evaluation language*

$$E(\{f_n\}) := \{(1^n, a_1, a_2, \ldots, a_{m(n)}, i, b) : \textit{the ith bit of } f_n(a_1, a_2, \ldots, a_{m(n)}) \textit{ equals } b\} \in \mathcal{C},$$

where i, $a_1, a_2, \ldots, a_{m(n) \in \mathbb{Z}}$ are given in binary. We denote the subclass of a$\cdot\mathcal{C}$ *consisting of families of multilinear polynomials by* ml$\cdot\mathcal{C}$.

Typically for a complexity class $\mathcal{C}$ we will have the complementation property that a$\cdot\mathcal{C}$ = a$\cdot(\mathcal{C} \cap \text{co}\mathcal{C})$. This is due to the inclusion of the bit b in the definition of the evaluation language. We have the following property in particular:

Proposition 3. a$\cdot$NEXP = a$\cdot$(NEXP $\cap$ coNEXP).

Namely, given a NEXP-machine M deciding $E(\{f_n\})$ for some family of polynomials $\{f_n\}$, one can simulate M on inputs $(1^n, a_1, a_2, \ldots, a_{m(n)}, i, b)$ and $(1^n, a_1, a_2, \ldots, a_{m(n)}, i, 1-b)$. In these nondeterministic simulations one finds at most one of them accepting, and it is guaranteed there exists at least one such path. For all paths where an accept is found, the machine knows exactly whether $(1^n, a_1, a_2, \ldots, a_{m(n)}, i, b) \in E(\{f_n\})$. This means that we have a nondeterministic exponential time *flag machine* for computing the characteristic function of $E(\{f_n\})$, which implies that $E(\{f_n\}) \in$ NEXP $\cap$ coNEXP. Recall that a flag machine sets a flag bit and produces output. If the flag is 0 this means on the given path no output is produced. If the flag is 1 it signals the output is valid. To compute a function, all flag = 1 paths must produce the same value, and there must be at least one such path.

3 Improved Lower Bounds from Derandomization of PIT

In this section, in order to avoid ambiguity we use a new variable N to indicate the input length for Boolean complexity classes. For example, $\Sigma_4\mathrm{TIME}[N]$ is the class of all languages decidable by Σ_4-machines in time $O(N)$ for inputs of size N. We will prove Theorem 2, and the following theorem (which implies Theorem 1):

Theorem 7. $\mathrm{PIT} \in \mathrm{NSUBEXP} \Rightarrow \mathrm{ml{\cdot}NEXP} \not\subseteq \mathrm{ASIZE}'(\mathit{poly})$.

We first establish fixed polynomial size arithmetic circuit lower bounds for a·PH.

Theorem 8. *For any fixed k, there exists $\{f_n\} \in \mathrm{a{\cdot}PH}$ with $\{f_n\} \not\in \text{i.o-}\mathrm{ASIZE}'(n^k)$. Furthermore, each f_n is multilinear in n variables, has degree $3k$ and has coefficients in $\{0,1\}$.*

Proof. For simplicity we show a fixed size lower bound in terms of ASIZE instead of ASIZE$'$. The proof is easily modified to yield the more general statement. There are $2^{O(n^{2k})}$ arithmetic circuits of size at most n^k. Consider the class $\mathcal{F}$ of homogeneous multilinear polynomials in n variables of degree $3k$ with $0,1$ coefficients. Then $|\mathcal{F}| = 2^{\binom{n}{3k}}$. Hence there exists $f_n \in \mathcal{F}$ that is not in $\mathrm{ASIZE}(n^k)$. Our goal is to find it 'in PH'.

Let $\mathcal{C}$ be the class of arithmetic circuits corresponding to $\mathcal{F}$, where we just represent in the $\Sigma\Pi$-form, i.e. a sum of monomials. We can fix some representation of $\mathcal{C}$ by strings of length $O(n^{3k})$. Our goals is to find the lexicographically least circuit $\Phi \in \mathcal{C}$ such that *for all* arithmetic circuits Ψ of size n^k, $\Phi - \Psi \not\equiv 0$. Define the language L to consist of tuples $(1^n, <\Phi>)$ with the property that for all circuits Ψ of size n^k, $\Phi - \Psi \not\equiv 0$, where $<\Phi>$ to denotes the string encoding of Φ. Checking $\Phi - \Psi \not\equiv 0$ is a coRP predicate. This implies that L is in $\mathrm{coNP}^{\mathrm{RP}}$. On input 1^n, using binary search and making *existential* queries to L, one can find the lexicographically least Φ of size $O(n^{3k})$ such that $(1^n, <\Phi>) \in L$ in $\mathrm{FP}^{\mathrm{NP}^{\mathrm{coRP}^{\mathrm{RP}}}}$. Define f_n to be the polynomial computed by this Φ. Once the sum of monomials representations of f_n is known, evaluations is *poly*-time computable for integer inputs. Hence we obtain that $E(\{f_n\}) \in \mathrm{PH}$. □

From the proof of Theorem 8 we can conclude that the following lemma is true:

Lemma 2. *There exists a constant $c_1 \in \mathbb{N}$, such that for any $k \geq 1$, there exists $\{f_n\} \in$ ml$\cdot$ $\Sigma_4\mathrm{TIME}[N^{c_1 k}]$ with $\{f_n\} \not\in \text{i.o-}\mathrm{ASIZE}'(n^k)$.*

Namely, to describe an algorithm for $E(\{f_n\})$, consider an input $(1^n, a_1, \ldots, a_n, i, b)$ of size N. The proof of Theorem 8 shows that we can first find in $\Sigma_4\mathrm{TIME}[\mathit{poly}(n^{3k})]$ a sum-of-monomials description of a polynomial f_n of size $O(n^{3k})$ that requires size n^k. After that we evaluate $f(a_1, \ldots, a_n)$, which can be done in time $\mathit{poly}(N^{3k})$ given this simple representation of f_n. We get that the total overhead for deciding $E(\{f_n\})$ is $\Sigma_4\mathrm{TIME}[\mathit{poly}(N^k)]$. One now easily derives the following lemma:

Lemma 3. *There exists a constant $c_2 \in \mathbb{N}$, such that for any $k \geq 1$, there exists $\{f_n\} \in$ ml$\cdot\mathrm{DTIME}^{\mathit{0,1\text{-}Perm}[1]}[N^{c_2 k}]$ with $\{f_n\} \not\in \text{i.o-}\mathrm{ASIZE}'(n^k)$.*

Proof. By Toda's theorem [15] and Valiant's Completeness result [16], we know that there exists an absolute constant $b \in \mathbb{N}$ so that for every $k \in \mathbb{N}$, $\Sigma_4\mathrm{TIME}[N^k] \subseteq \mathrm{DTIME}^{0,1\text{-Perm}[1]}[N^{bk}]$. Let c_1 be the constant given by Lemma 2. We get that $\Sigma_4\mathrm{TIME}[N^{c_1 k}] \subseteq \mathrm{DTIME}^{0,1\text{-Perm}[1]}[N^{bc_1 k}]$. Hence the lemma holds for $c_2 = bc_1$. □

We use the following lemma by Kinne, van Melkebeek and Shaltiel (See Appendix B for the change from ASIZE to ASIZE'):

Lemma 4 (Claim 5 in [7])**.** *There exists a constant d such that the following holds for any functions $a(\cdot)$ and $t(\cdot)$ with $a(\cdot)$ time-constructible and $t(\cdot)$ monotone. If* $\mathrm{PIT} \in \mathrm{NTIME}(t(N))$ *and* $\{per_n\} \in \mathrm{ASIZE}'(a(n))$*, then* $\mathrm{DTIME}^{\mathit{0,1\text{-}Perm}[1]}[N] \subseteq \mathrm{NTIME}[t(N \cdot \log^d N \cdot a(\sqrt{N}))]$.

We are now ready to prove Theorem 7.

Proof. (Theorem 7) We are done if $\{per_n\} \not\in \mathrm{ASIZE}'(poly)$, so assume that $\mathrm{ASIZE}'(\{per_n\}) \leq n^\ell$, for $\ell \in \mathbb{N}$. Consider arbitrary $k \geq 1$. Combining Lemmas 3 and 4, we obtain that for any monotone function $t(\cdot)$, if $\mathrm{PIT} \in \mathrm{NTIME}[t(N)]$, then $\mathrm{ml{\cdot}NTIME}[t(N^{c_2 k} \cdot \log^d N^{c_2 k} \cdot N^{c_2 \ell k/2})] \not\subseteq \mathrm{ASIZE}'(n^k)$. As we are assuming that $\mathrm{PIT} \in \mathrm{NSUBEXP}$, if we apply this with $t(N) = 2^{N^\epsilon}$, for small enough ϵ, we get that $\mathrm{ml{\cdot}NTIME}[2^N] \not\subseteq \mathrm{ASIZE}'(n^k)$. Since k was arbitrary, we get that $\mathrm{ml{\cdot}NTIME}[2^N] \not\subseteq \mathrm{ASIZE}'(poly)$, which implies that $\mathrm{ml{\cdot}NEXP} \not\subseteq \mathrm{ASIZE}'(poly)$. □

Next we move on to the proof of Theorem 2.

Proof. (Theorem 2). Suppose that $\mathrm{a{\cdot}NEXP}^{\mathrm{RP}} \subseteq \mathrm{ASIZE}'(poly)$. Then $\mathrm{a{\cdot}EXP} \subseteq \mathrm{ASIZE}'(poly)$. We claim that this implies that $\mathrm{EXP} \subseteq \mathrm{SIZE}(poly)$. Let $L \in \mathrm{EXP}$ be any language. We will show that $L \in \mathrm{SIZE}(poly)$. Since we can evaluate multilinear extensions (Proposition 2) of characteristic functions of EXP languages within EXP itself, we get $\{F_n\}$ in a·EXP, where F_n is the multilinear extension of χ_L on $\{0,1\}^n$. We get that $\{F_n\} \in \mathrm{ASIZE}'(poly)$. This means that we have constant-free (division-free) arithmetic circuits Φ_1 and Φ_2 of size at most $p(n) = n^{O(1)}$, such that Φ_2 does not contain variables and computes some nonzero constant $c \in \mathbb{Z}$. Furthermore, if Φ_1 computes G_n then it holds that $G_n = c \cdot F_n$. For input $a \in \{0,1\}^n$, $F_n(a) \in \{0,1\}$, which means for such inputs $G_n(a) \in \{0, c\}$. We want to evaluate Φ_1 modulo some prime number q that does not divide c. This will tell us $\chi_L(a)$. We have that $|c| \leq 2^{2^{p(n)}}$ due to Proposition 1. This means that c has at most $2^{p(n)}$ prime factors. Hence, using the Prime Number Theorem there exists a prime number q of $p(n)^2$ bits, provided n is large enough, that does not divide c. As our task is to show only the non-uniform upper bound $L \in \mathrm{SIZE}(poly)$, mere existence of this number q suffices for our purposes, as we can hardcode it into the Boolean circuit simulating Φ_1 and Φ_2. Hence $\mathrm{EXP} \subseteq \mathrm{SIZE}(poly)$.

Babai, Fortnow, Lund [2] prove that $\mathrm{EXP} \subseteq \mathrm{SIZE}(poly) \Rightarrow \mathrm{EXP} = \mathrm{MA}$. So we get that $\mathrm{EXP} = \mathrm{MA}$. Also, because easily $\{per_n\} \in \mathrm{a{\cdot}NEXP}^{\mathrm{RP}}$, we have that $\{per_n\} \in \mathrm{ASIZE}'(poly)$. This implies that $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}^{\mathrm{RP}}$, cf. Lemma 5.3 in Ref. [6]. By Toda's Theorem [15], $\mathrm{MA} \subseteq \mathrm{P}^{\#\mathrm{P}}$. Hence we obtain that $\mathrm{EXP} = \mathrm{MA} \subseteq \mathrm{NP}^{\mathrm{RP}}$. By padding this implies that $\mathrm{EEXP} \subseteq \mathrm{NEXP}^{\mathrm{RP}}$. Hence $\mathrm{a{\cdot}EEXP} \subseteq \mathrm{a{\cdot}NEXP}^{\mathrm{RP}} \subseteq \mathrm{ASIZE}'(poly)$. This is a contradiction. One can easily deduce that $\mathrm{a{\cdot}EEXP} \not\subseteq \mathrm{i.o\text{-}ASIZE}'(n^{\log n})$ by observing that Lemma 2 also holds if we allow k to depend on n as $k(n) = \lceil \log n \rceil$. □

We can specialize Lemma 4 so that we replace the condition "$\mathrm{PIT} \in \mathrm{NTIME}(t(N))$ and $\{per_n\} \in \mathrm{ASIZE}'(a(n))$" by "low-$\mathrm{PIT} \in \mathrm{NTIME}(t(N))$ and $\{per_n\} \in \mathrm{ASIZEDEG}'(a(n))$" (See Appendix B). This yields the following theorem (which implies Theorem 4):

Theorem 9. *low*-$\mathrm{PIT} \in \mathrm{NSUBEXP} \Rightarrow \mathrm{ml{\cdot}NEXP} \not\subseteq \mathrm{ASIZEDEG}'(poly)$.

4 Stronger Fixed Size Lower Bounds for MA

As the result we aim to strengthen puts somewhat different constraints on constants appearing in arithmetic circuits compared to what we have seen so far, we make the following provisional definition. Let $\mathrm{ASIZE}^{\mathrm{free}}(s(n))$ denote the class obtained from $\mathrm{ASIZE}(s(n))$ by granting the underlying circuits arbitrary constant labels $\in \mathbb{Z}$. Similar to Theorem 8 we have the following theorem.

Theorem 10. $\forall k \exists \{f_n\} \in$ a$\cdot$PH *with* $\{f_n\} \notin$ i.o-ASIZE$^{\textit{free}}(n^k)$.

Proof. Consider the set of all arithmetic circuits of circuits of size n^k with n inputs with constant labels in $\mathbb{Z}$. Evaluating such circuits modulo 2, this defines a class $\mathcal{R}$ of at most $2^{O(n^{2k})}$ Boolean functions $\{0,1\}^n \to \{0,1\}$. Let $\mathcal{B}$ be the class of all Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $f(a) = 0$, for all a where the number of 1s is not equal to $3k$. Then $|\mathcal{B}| = 2^{\binom{n}{3k}}$. Hence there exist $f_n \in \mathcal{B}/\mathcal{R}$. We can represent element of $\mathcal{B}$ by strings of length $\binom{n}{3k}$. Similarly as in the proof of Theorem 8, one can find $f_n \in \mathcal{B}/\mathcal{R}$. Letting $\mathcal{C}$ be the class of arithmetic circuits of size n^k with constants labels $\in \{0,1\}$, our goal for this is the following:

- Find the lexicographically least $f_n \in \mathcal{B}$ such that *for all* $\Psi \in \mathcal{C}$ *there exists* $a \in \{0,1\}^n$ such that $\Psi(a) \bmod 2 \neq f(a)$.

We conclude that we can compute f_n in FP$^{\text{PH}}$. Applying Proposition 2, we take F_n to be the multilinear extension of f_n. We have that $\{F_n\} \notin$ ASIZE$^{\text{free}}(n^k)$. Namely, if this were not true, then reducing modulo 2, we obtain a circuit in $\mathcal{C}$ that coincides with F_n on $\{0,1\}^n$, but for such inputs F_n coincides with f_n. Finally, as f_n can be computed in FP$^{\text{PH}}$, and we only have $\binom{n}{3k} = O(n^{3k})$ terms in the outermost sum in the definition of F_n, we can easily evaluate F_n for integer inputs in FP$^{\text{PH}}$. This implies that $E(\{F_n\}) \in$ P$^{\text{PH}} \subseteq$ PH. $\square$

We want to strengthen Theorem 4 of [11], which we can reformulate it in our terminology as follows:

Theorem 11 ([11])**.** *For every k, either 1)* MA $\not\subseteq$ SIZE(n^k)*, or 2)* a$\cdot$MA $\not\subseteq$ ASIZE$^{\textit{free}}(n^k)$.

We will show that for every k, the second item holds by itself. Let us briefly remark on a technical issue related to this reformulation. For $\{f_n\}$, where f_n is a integer polynomial over n variables, Ref. [11] uses the notion $Gh(\{f_n\}) = \{(\vec{x}, v) | f_n(\vec{x}) = v\}$, and proves that for every k, either MA $\notin$ SIZE(n^k), or there exists $\{f_n\} \notin$ ASIZE$^{\text{free}}(n^k)$ with $Gh(\{f_n\}) \in$ MA. We prefer to work with the evaluation language $E(\{f_n\})$ instead of $Gh(\{f_n\})$. One can observe that the argument we give to strengthen Theorem 11 can be easily modified to work with $Gh(\cdot)$ instead. Consider the following proposition:

Proposition 4. *If* $\{per_n\} \in$ ASIZE$^{\textit{free}}(\textit{poly})$*, then 1) 0,1-permanent of an $n \times n$ matrix over $\mathbb{Z}$ can be computed with poly(n) size Boolean circuits, and 2)* PH $\subseteq$ MA.

For the above, it is argued in Ref. [11], proof of Theorem 4, that the first item follows from $\{per_n\} \in$ ASIZE$^{\text{free}}(\textit{poly})$, and that the second item follows from the first. The following theorem implies Theorem 3 from the introduction:

Theorem 12. *For any fixed k, there exists* $\{f_n\} \in$ a$\cdot$MA/ASIZE$^{\textit{free}}(n^k)$.

Proof. We show that Item 2 of Theorem 11 holds by itself. For this, we indicate how the proof of Theorem 4 in Ref. [11] must be modified. This proof conditions on the predicate $\{\text{per}_n\} \in$ ASIZE$^{\text{free}}(\textit{poly})$. If this is not true, the proof there can easily be modified to use $E(\cdot)$ instead of $Gh(\cdot)$, which then yields the statement of the theorem. Otherwise, suppose that $\{\text{per}_n\} \in$ ASIZE$^{\text{free}}(\textit{poly})$. By Proposition 4 we have that PH $\subseteq$ MA. The latter implies that a$\cdot$PH $\subseteq$ a$\cdot$MA. Hence in this case Item 2 holds also, due to Theorem 10. $\square$

5 A Characterization of Derandomization for low-PIT

We will use the algebraic hardness-to-randomness framework of Ref. [6]. The refinement that we make here is to show that it suffices to start with a weaker[4] ASIZEDEG$'$-hardness assumption rather than ASIZE$'$-hardness, in case we only need to cater for low-PIT.

For a polynomial $f(x,y) \in \mathbb{F}[x_1,\ldots,x_n,y]$ and $p(x) \in \mathbb{F}[x_1,\ldots x_n]$, $f_{|y=p}$ denotes the polynomial obtained by substituting p for y in f. We will also write this polynomial as $f(x,p)$. In case $f_{|y=p} = 0$, we say that p is a *root* of f for y. The following is our degree-efficient root extraction lemma:

Lemma 5. *Suppose that $f \in \mathbb{Z}[x_1,\ldots,x_n,y]$ is a nonzero polynomial computed by a division-free constant free arithmetic circuit of size s and formal degree D. Suppose that $p \in \mathbb{Z}[x_1,\ldots,x_n]$ is a root of f for y. Then there exist constant-free division-free arithmetic circuits Φ_1 and Φ_2 of size and formal degree bounded by poly(n,s,D,L) such that the following are true:*

1. *Φ_1 computes a polynomial $q \in \mathbb{Z}[x_1,\ldots,x_n]$.*
2. *Φ_2 does not contain variables. It computes a nonzero constant $c \in \mathbb{Z}$.*
3. *It holds that $c \cdot p = q$.*
4. *L bounds the maximum bit size of $p(x)$ on $\{0,1,\ldots,d_p d_f\}^n$, where d_f and d_p are the degrees of f and p, respectively.*

The proof of the above lemma follows by analyzing the degree blow-up in the root extraction method of Ref. [5]. As this procedure involves Newton iteration it is a priori not at all clear that formal-degrees are well-behaved, but this turns out to be true.

5.1 Proof of Lemma 5

We use the following proposition for the proof of Lemma 5:

Proposition 5. *Let $f(x_1,\ldots,x_n,y)$ be a polynomial of degree d_f that is computed by a constant-free division-free arithmetic circuit Φ of size s and formal-degree d. Then $\frac{\partial^r f}{\partial^r y}$ can be computed by a constant-free division-free arithmetic circuit of size $O(sd_f^2 + rd_f)$ and formal degree $d + d_f + r$.*

Proof. We sketch the proof. First write $f = \sum_{i=0}^{d_f} y^i c_i(x_1,\ldots,x_n)$. By computing degree components separately for all gates of Φ, we obtain an arithmetic circuit of size $O(sd_f^2)$ and formal degree at most d computing $c_0, c_1, \ldots, c_{d_f}$ separately. Then we compute $y, y^2, \ldots, y^{d_f - r}$ within size and degree $O(d_f)$. We compute for each $i \geq r$, $i(i-1)\ldots i-r+1)$ within size $O(rd_f)$ and degree r. We then put together a circuit for $\frac{\partial^r f}{\partial^r y} = \sum_{i\geq r}^{d_f} i(i-1)\ldots(i-r+1)y^{i-r}c_i(x_1,\ldots,x_n)$ that is of size $O(sd_f^2 + rd_f)$ and formal degree at most $d + d_f + r$. $\square$

For a polynomial $f \in \mathbb{F}[X]$, we denote by $f_{=i}$, or if it is clear from the context simply f_i, the homogeneous component of degree i. We also use the notation $f_{\leq i}$, which is defined as $f_0 + f_1 + \ldots + f_i$. We have the following lemma:

Lemma 6 (Gauss). *Let $f \in \mathbb{F}[X,y]$ be a nonzero polynomial, and let $p \in \mathbb{F}[X]$ be a root of f for y. Then $p - y$ is an irreducible factor of f in the ring $\mathbb{F}[X,y]$.*

In the above situation, the *multiplicity* of the root p is defined to be the largest number m such that $(p-y)^m$ divides f. We use the following lemma:

[4]See Section 2 for some remarks pertaining to these measures when dealing with families of *poly*-degree.

Lemma 7 ([5]). *Let $f \in \mathbb{F}[X, y]$ and let $f'(x, y) := \frac{\partial f}{\partial y}$. Let $p \in \mathbb{F}[X]$ be a root of f for y, and assume that $\xi_0 := f'(0, p(0)) \neq 0$. Then $\forall k \geq 1$ it holds that $p_{\leq k+1} = p_{\leq k} - \frac{1}{\xi_0} \cdot f(x, p_{\leq k})_{=k+1}$.*

We now give the proof of Lemma 5. Let $\mathcal{M}$ be the multiplicity of the root p in f. One first reduces to the case where $\mathcal{M} = 1$ by taking partial derivatives. Namely, as shown in Ref. [5], if $f = (y - p)^r \cdot g$, with g not divisible by $(y - p)$, then p is a root of multiplicity one the *nonzero* polynomial $\frac{\partial^{r-1} f}{\partial y}$. This blows up the size and formal degree of the circuit we are trying to extract from to $O(sd_f^2 + \mathcal{M}d_f)$ and $D + d_f + \mathcal{M}$, respectively, due to Proposition 5. Note that $\mathcal{M}$ can be bounded by d_f. We conclude that wlog. we may assume that $\mathcal{M} = 1$.

Let $f' = \frac{\partial f}{\partial y}$. We have that $f'(x, p) \not\equiv 0$. Namely, $f = (y - p) \cdot g$ with g not divisible by $(y - p)$. So $f' = g + (y - p) \cdot \frac{\partial g}{\partial y}$. Hence $f'(x, p) = g(x, p) \not\equiv 0$ by Lemma 6.

One next reduces to the case $f'(0, p(0)) \neq 0$ by making a coordinate shift. Let $F = f(x + a_0, y)$ and $P = p(x + a_0)$, where $a_0 \in \mathbb{Z}^n$ is such that $f'(a_0, p(a_0)) \neq 0$. Note we can find such a_0 as $f'(x, p) \not\equiv 0$. let $F' = \frac{\partial F}{\partial y}$. Then P is a root of F and $F'(0, P(0)) = f'(a_0, p(a_0)) \neq 0$. We have that $deg(f'(x, p)) \leq d_p d_f$. Hence by Lemma 1 we can find $a_0 \in \{0, 1, \ldots, d_p d_f\}^n$ such that $f'(a_0, p(a_0)) \neq 0$. This means entries of a_0 are $O(\log d_p d_f))$ many bits . So we can compute all n entries by going over the binary expansion by a constant-free arithmetic circuit of size $O(n \log d_p d_f))$. This is an inconsequential blow-up. One does the root extraction for P from F, and then apply the reverse coordinate shift to obtain p. Also the latter can be done at ignorable cost as it only add $O(n \log d_p d_f))$ circuitry.

Let L' bounds the maximum bit size of $f'(x, p(x))$ and $p(x)$ on $\{0, 1, \ldots, d_p d_f\}^n$, where $f' = \frac{\partial f}{\partial y}$. We can conclude that for the rest of the proof we may assume that $\xi_0 := f'(0, p(0)) \neq 0$, where we have that the bit size of ξ_0 and $p(0)$ is bounded by L'. This means that ξ_0 and $p(0)$ can be computed by a constant-free (division-free) arithmetic circuit of size and formal degree $O(L')$.

Let Φ be a constant-free division-free arithmetic circuit of size s computing f. We will first construct an arithmetic circuit Ψ with gates in $\{+, \times\}$ that computes p that using constants $\in \{-1, 0, 1\} \cup \{-\frac{1}{\xi_0}\}$. The constant $-\frac{1}{\xi_0}$ will be used in Ψ only at multiplications gates to compute a division by $-\xi_0$. In a second step we remove divisions by $-\xi_0$ to finally obtain Φ_1 and Φ_2.

By Lemma 7, $p_i = -\frac{1}{\xi_0} f(x, p_{\leq i-1})_i$, for each $i > 0$. We use this equation to compute $p_0, p_1, \ldots, p_{d_p}$ in stages, using a copy of Φ each time. Wlog. we assume there is a single gate h in Φ that carries the label y. Let g be the output gate of Φ.

To start the construction, at stage $i = 0$ we must compute p_0. This is the constant term of p, i.e. $p(0)$. We observed above that we have a constant-free arithmetic circuit of size and formal degree $O(L')$ computing $p(0)$.

We now describe stage $i > 0$. In the circuit constructed so far there are gates $G_0, G_1, \ldots, G_{i-1}$ computing $p_0, p_1, \ldots, p_{i-1}$, respectively. We will use a copy $\overline{\Phi}_i$ of Φ, but where for each gate we compute the homogeneous components up to degree i separately. This means that for each gate v in Φ there exists a corresponding gate $v_{i,j}$ in $\overline{\Phi}_i$ for all $j \in \{0, 1, \ldots, i\}$. In other words, we use a naming scheme where the first index specifies which copy, and the second index specifies the degree. Wlog. we assume all addition and multiplication gates in Φ have fan-in two. We add circuitry to Ψ as follows. For each v in Φ, excluding g and h,

- if v is an addition gate with inputs u and w, then in $\overline{\Phi}_i$, $v_{i,j}$ is an addition gate with inputs $u_{i,j}$ and $w_{i,j}$, for all $j \in \{0, 1, \ldots, i\}$.

- if v is an multiplication gate with inputs u and w, then in $\overline{\Phi}_i$, for all $j \in \{0, 1, \ldots, i\}$, we add gates computing $u_{i,\ell} w_{i,m}$ for all $\ell + m = j$. Then we make $v_{i,j}$ an addition gate so it computes $v_{i,j} = \sum_{t=0}^{j} u_{i,t} w_{i,j-t}$.

We also do the following:

- For the gate h (which is labeled by y) in Φ, for each $0 \leq j \leq i-1$, the gate $h_{i,j}$ in $\overline{\Phi}_i$ is an addition gate with dummy input zero and input from G_j. The gate $h_{i,i}$ is a input gate labeled with the constant zero. This effectively achieves setting $y := p_{\leq i-1}$.
- For the output gate g in Φ, we have that the corresponding gate $g_{i,i}$ computes $f(x, p_{\leq i-1})_i$. We add one more multiplication gate, named G_i, that multiplies with $-\frac{1}{\xi_0}$ (we allocate globally one constant gate labeled $-\frac{1}{\xi_0}$). We have that G_i computes p_i.

The above is repeated up to stage $i = d_p$, at which point the construction of Ψ is completed by adding a single addition gates which computes $p_0 + p_1 + \ldots + p_{d_p}$. We will analyze the formal degrees of nodes added by the construction. Let $D' = D \cdot D''$, where $D'' = O(L')$ is the formal degree of the circuit computing $p(0)$. Let $v^1, v^2, \ldots, v^s$ be a topological sort of the gates of Φ, where $v^1 = h$ (the unique node with label y) and $v^s = g$ (the output gate). Thus for each $k \in [s]$, we have for $1 \leq i \leq d_p$ and $0 \leq j \leq i$ a corresponding node $v^k_{i,j}$ in Ψ. In what follows, for a node v^k in Φ, $deg(v^k)$ denotes the formal degree of v^k in Φ, whereas $deg(v^k_{i,j})$ denotes formal degree of node $v^k_{i,j}$ in Ψ.

Claim 1.

$$deg(v^k_{i,j}) \leq \begin{cases} \alpha(i,k)D'j & \textit{if } j \geq 1 \\ D' & \textit{if } j = 0 \end{cases},$$

for all $1 \leq k \leq s$ *and* $i \geq \max(1, j)$*, where* $\alpha(i,k) = is + k$*.*

Proof. We prove this claim by simultaneous induction on j and the value of the function $\alpha(i,k)$.

Base Case $j = 0$. Nodes of the form $v^1_{i,0}$ are addition gates taking input from the gate G_0 that computes $p(0)$ together with a dummy constant zero gate. The degree of G_0 is bounded by D''. Nodes of the form $v^k_{i,0}$ for $k \neq 1$ compute a constant that is equal to the output of the gate v^k in $\Phi(0, p(0))$. Hence we can assume that the formal degree of such a node is bounded by $D \cdot D''$, as D bounds the formal degree in $\Phi(0, y)$.

Base Case $\alpha(i,k) = s + 1$. The lowest value $\alpha(i,k)$ can obtain is $s+1$. This happens for $i = 1$ and $k = 1$. The node $v^1_{1,j}$ corresponds to the input gate labeled with y. In case $j = 0$ it receives input from G_0, which means formal degree is bounded by D''. In case $j = 1$ it is a constant zero gate. In both cases Claim 1 holds.

Induction Case $j > 0$.

Consider an arbitrary node $v^k_{i,j}$ in Ψ with $j > 0$. We now make a case distinction bases on the type of the corresponding gate v^k in Φ. We will apply the induction hypothesis whenever we have lowered the value of j or have decreased the value of the function $\alpha(i,k)$.

Subcase 1: v^k is a constant labeled gate. Then $v^k_{i,j}$ is labeled by a constant also, and thus has degree at most 1.

Subcase 2: v^k is a x-labeled gate. In this case $v^k_{i,j}$ is labeled with x if $j = 1$ and labeled with 0 otherwise. Hence degree is at most 1.

Subcase 3: v^k is the y-labeled gate, i.e. $k = 1$. In case $i = j$ then $v^k_{i,j}$ is an input gate labeled with the constant 0, and the bound of Claim 1 clearly holds. For $j < i$, by the construction $deg(v^k_{i,j}) = deg(G_j) = 1 + deg(v^s_{j,j})$. Note that $\alpha(j,s) = (j+1)s \leq is < is + k = \alpha(i,k)$. We have lowered the value of α and thus can apply the induction hypothesis to get $deg(v^k_{i,j}) \leq 1 + (j+1)sD'j$. We thus need to check that $1 + (j+1)sD'j \leq (is+k)D'j$. This holds iff $\frac{1}{D'j} + (j+1)s \leq (is+k)$. This is true as $j+1 \leq i$, $k = 1$ and $D'j \geq 1$.

Subcase 4: v^k is an addition gate. Say $v^k = +(v^\ell, v^m)$, where obviously $\ell, m < k$. Then $v^k_{i,j} = +(v^\ell_{i,j}, v^m_{i,j})$. Hence $deg(v^k_{i,j}) = \max(deg(v^\ell_{i,j}), deg(v^m_{i,j}))$. Note that we can apply for the latter the induction hypothesis based on the value of α. We get $deg(v^k_{i,j}) \leq \max((is + \ell)D'j, (is+m)D'j) \leq (is+k)D'j$.

Subcase 5: v^k is a multiplication gate. Say $v^k = \times(v^\ell, v^m)$, where $\ell, m < k$. Then, per abuse of notation, we have that $v^k_{i,j} = \sum_{t=0}^{j} v^\ell_{i,t} \cdot v^m_{i,j-t}$. The degree of $v^k_{i,j}$ is the maximum degree of $deg(v^\ell_{i,t} \cdot v^m_{i,j-t})$, where the latter (abusive) notation is intended to denote the formal degree of the intermediate multiplication gate allocated in the construction which takes inputs from $v^\ell_{i,t}$ and $v^m_{i,j-t}$. Similarly, by writing $v^k_{i,j} = \sum_{t=0}^{j} v^\ell_{i,t} \cdot v^m_{i,j-t}$ we mean that gate $v^k_{i,j}$ is a sum gate taking inputs from all such intermediate multiplication gates. We consider the terms of this expression separately. We treat the case $t = 0$ and the symmetrical case $t = j$ first. We have that $deg(v^\ell_{i,0} \cdot v^m_{i,j}) = deg(v^\ell_{i,0}) + deg(v^m_{i,j})$. Note that by induction on j, $deg(v^\ell_{i,0}) \leq D'$ and that by induction on the value of α, $deg(v^m_{i,j}) \leq (is+m)D'j$. So $deg(v^\ell_{i,0} \cdot v^m_{i,j}) \leq D' + (is+m)D'j$. We have that $D' + (is+m)D'j \leq (is+k)D'j$ iff $\frac{1}{j} + (is+m) \leq (is+k)$. The latter holds as $m \leq k-1$ and $j \geq 1$.

Now consider case where $0 < t < j$. In this case we apply the induction hypothesis on j for both terms. Hence $deg(v^\ell_{i,t} \cdot v^m_{i,j-t}) = deg(v^\ell_{i,t}) + deg(v^m_{i,j-t}) \leq (is+\ell)D't + (is+m)D'(j-t) < (is+k)D't + (is+k)D'(j-t) = (is+k)D'j$.

We have verified Claim 1. □

By Claim 1 we can conclude that the formal degree of Ψ is at most $d_p(d_p+1)sD' = poly(s, D, L', d_p)$. Taking into account the size blow-up due to preprocessing we get that p can be compute by an arithmetic circuit of formal degree $poly(n, s, D, L', d_p, d_f)$. This is $poly(n, s, D, L', d_f)$ as $d_p \leq d_f$. The latter circuit still uses the constant $\frac{-1}{\xi_0}$ at multiplication gates, which we will treat next. Wlog. we can ignore the minus sign as we can compute $\frac{-1}{\xi_0}$ from $\frac{1}{\xi_0}$ with inconsequential increase in size and degree. So we assume our task is to remove a single gate with label $\frac{1}{\xi_0}$. First we replace the label $\frac{1}{\xi_0}$ by a new variable z to get a circuit Ψ' computing a polynomial $q(x,z)$ such that $q(x, \frac{1}{\xi_0}) = p$. Let D''' be the formal degree of Ψ', then $D''' = poly(n, s, D, L', d_f)$. Let $s' = poly(n, s, D, L', d_f)$ be the size of Ψ'. We can write $q = \sum_{i=0}^{D'''} z^i c_i(x_1, \ldots, x_n)$. Hence $\xi_0^{D'''} \cdot p = \sum_{i=0}^{D'''} (\xi_0)^{D'''-i} c_i(x_1, \ldots, x_n)$. Recall ξ_0 can be computed within size and formal degree $O(L')$. Hence the integer $\xi_0^{D'''}$ can be computed by an arithmetic circuit of size $O(L' + D''')$ and formal degree $O(D'''L')$. This is the circuit Φ_2 as mentioned in the statement of the lemma. We can construct an arithmetic circuit from Ψ' computing $c_0, c_1, \ldots, c_{D'''}$ separately with size $O(s'(D''')^2)$ and formal degree D'''. We can compute $\xi_0, \xi_0^2, \ldots, \xi_0^{D'''}$ separately in size $O(L' + D''')$ and formal degree $O(D'''L')$. From this we construct the circuit Φ_1 computing $\sum_{i=0}^{D'''} (\xi_0)^{D'''-i} c_i(x_1, \ldots, x_n)$. We may conclude that size an formal degree of both Φ_1 and Φ_2 are $poly(n, s, D, L', d_f)$.

To finish the proof, we show that we can replace the parameter L' by L. By Proposition 5, we have that f' can be computed by a constant-free circuit of size $O(sd_f^2)$ and formal degree $D + d_f + 1$. By Corollary 1 we get that $f'(x, p(x))$ has bit size $O(sd_f^2(D + d_f + 1)\max(L, \log d_f d_p)^2)$ on $\{0, 1, \ldots, d_f d_p\}^n$. The bit size of $p(x)$ on $\{0, 1, \ldots, d_f d_p\}$ is bounded by L by definition. hence we get that $L' = poly(s, d_f, d_p, D, L) = poly(s, d_f, D, L)$. Finally, we can also drop d_f in the statement of the bound as $d_f \leq D$. This completes the proof of Lemma 5. □

5.2 Hardness-Randomness Trade-Offs

We need the following lemma:

Lemma 8 (Nisan-Wigderson Design [10]). *Let n, m be integers with $n < 2^m$. There exists a family of sets $S_1, S_2, \ldots, S_n \subseteq [\ell]$, such that 1) $\ell = O(m^2/\log n)$, 2) For each i, $|S_i| = m$, and*

3) For every $i \neq j$, $|S_i \cap S_j| \leq \log n$. Furthermore, the above family of sets can be computed deterministically in time poly$(n, 2^\ell)$.

Define NW^p as follows. For parameters ℓ, m, n, construct the set system $S_1, S_2, \ldots, S_n$ as in Lemma 8. Then for $a_1, a_2, \ldots, a_\ell \in \mathbb{F}$, and a polynomial p in m variables, $NW^p(a) = (p(a_{|S_1}), p(a_{|S_2}), \ldots, p(a_{|S_n}))$. The following lemma is proved using a hybrid argument, cf. Lemma 7.6 in [6]:

Lemma 9. *Let n and m be integers with $n < 2^m$ and $m < n$. Suppose we are given a nonzero polynomial $f \in \mathbb{Z}[y_1, \ldots, y_n]$ of degree d_f and a multilinear polynomial $p \in \mathbb{Z}[x_1, \ldots, x_m]$ with coefficients of bit size at most m^e, for some integer constant $e \geq 1$. Assume that f can be computed by a division free constant-free arithmetic circuit of size s and formal degree D. Let $S \subseteq \mathbb{Z}$ be any set of size $|S| > d_f m$, and let ℓ be given by Lemma 8. Suppose that $f(NW^p(a)) = 0$ for all $a \in S^\ell$. Then there exists $q \in \mathbb{Z}[x_1, \ldots, x_m]$ and $c \in \mathbb{Z}/\{0\}$ such that $p = q/c$, where q and c can be computed by constant-free division-free arithmetic circuits of size and formal degree poly(n, m^e, s, D).*

Proof. Define

$$\begin{aligned} f_0 &= f(y_1, y_2, \ldots, y_n) \\ f_1 &= f(p(x_{|S_1}), y_2, \ldots, y_n) \\ &\vdots \\ f_n &= f(p(x_{|S_1}), p(x_{|S_2}), \ldots, p(x_{|S_n})). \end{aligned}$$

Note that f_n is a polynomial of degree at most $d_f m$. By assumption $f_n(a) = 0$ for all $a \in S^\ell$. Hence by Lemma 1 we have that $f_n \equiv 0$. On the other hand $f_0 \not\equiv 0$. Let i be the smallest number such that $f_i \not\equiv 0$ and $f_{i+1} \equiv 0$. Consider the polynomial f_i:

$$f_i = f(p(x_{|S_1}), p(x_{|S_2}), \ldots, p(x_{|S_i}), y_{i+1}, \ldots, y_n)$$

We want to set x-variables not in S_{i+1} and $y_{i+2}, \ldots, y_n$ to certain integer values, while keeping the polynomial nonzero, and furthermore argue this new polynomial can be computed by a small constant-free arithmetic circuit. By Lemma 1 there exists values in $\{0, 1, \ldots, d_f\}$ for removing the y-variables, while leaving the polynomial nonzero. We can compute these values with a constant-free circuit of size at most $O(d_f)$ and formal degree 1. Also by Lemma 1 we get that there exist values in $\{0, 1, \ldots, d_f m\}$ for removing x-variables not in S_{i+1}. Consider $p(x_{|S_j})$ for $j \leq i$. After fixing values not in S_{i+1} we obtain a multilinear polynomial in at most $\log n$ variables. We can bound the absolute value of any coefficient of a monomial in this polynomial by $(d_f m)^m 2^m 2^{m^e}$, i.e. $O(m^e(\log d_f + \log m))$ bits. We can conclude that after fixing, the polynomial resulting from $p(x_{|S_j})$ can be computed by a constant-free arithmetic circuit of size $O(n \log n \cdot m^e(\log d_f + \log m))$ and formal degree $O(m^e(\log d_f + \log m) \cdot \log n)$. Putting everything together, we conclude that there exists a nonzero polynomial $g(x_{|S_{i+1}}, y_{i+1})$ that can be computed by a constant-free arithmetic circuit of size $O(s + d_f + n^2 \log n \cdot m^e(\log d_f + \log m))$ and formal degree $O(D \cdot m^e(\log d_f + \log m) \log n)$ such that $p(x_{|S_{i+1}})$ is a root for y_{i+1} of g. We now apply root extraction Lemma 5. The bit size of p on $\{0, 1, \ldots, d_f m\}$ can be bound by $O(m^e(\log d_f + \log m))$. We obtain arithmetic circuits Φ_1 and Φ_2 computing a polynomial q and a nonzero integer c such that $p \cdot c = q$. The size and formal degree of Φ_1 and Φ_2 are bounded by poly$(n, m^e, d_f, s, D) =$ poly(n, m^e, s, D). □

Our first trade-off is as follows:

Theorem 13. ml$\cdot$NEXP $\not\subseteq$ ASIZEDEG$'(poly(n)) \Rightarrow$ *low*-PIT $\in \bigcap_{\epsilon>0}$ i.o-NTIME$[2^{N^{\epsilon}}]$.

Proof. Consider a family $\{p_m\} \in$ ml$\cdot$NEXP that is not in ASIZEDEG$'(poly)$. By reindexing we can assume wlog. that p_m is defined over m variables. Let e be such that coefficients of p_m are at most m^e bits. We have that for every k, there exist infinitely many m such that p_m cannot be written as $p_m = f_m/c_m$, where f_m and $c_m \in \mathbb{Z}/\{0\}$ are computed by constant-free division-free arithmetic circuits of size and formal degree at most m^k. The $m \in \mathbb{Z}$ that satisfy this property we call the *good indexes for k*. We use the fact that a$\cdot$NEXP = a$\cdot$(NEXP $\cap$ coNEXP). This means that we there exists a constant d and a nondeterministic flag machine M running in time $2^{(n')^d}$ for inputs of size n' that can compute the characteristic function of $E(\{p_m\})$ on a given input, cf. Proposition 3.

Let c_0 be an absolute constant that bounds the overhead of Lemma 9, in the sense that for the case $n = s = D$ we can write an upper bound of $n^{c_0}m^{ec_0}$ for the bound $poly(n, m^e, s, D)$ given by the lemma. We will describe an i.o-NSUBEXP algorithm for low-PIT. Let Φ be a constant-free (division-free) arithmetic circuit of size N computing f. First we check that the formal degree of Φ is bounded by N, if not reject.

Let $m = \lfloor N^{1/r} \rfloor$, where r is chosen arbitrarily large. We claim that for infinitely many input lengths N the following test property holds: for every constant-free arithmetic circuit Ψ of size N, $\Psi \equiv 0 \Leftrightarrow (\forall a \in S^{\ell}), \Psi(NW^{p_m}(a)) = 0$, where $S = [Nm+1]$ with $\ell = O(m^2/\log N)$ taken according to Lemma 8. This follows from Lemma 9. Namely, let $k = c_0(r+e)$ and let $\mathcal{M}$ be the set of good indexes for k. Then $\mathcal{M}$ is an infinite set. Consider input lengths N of the form $N = (N')^r$, where $N' \in \mathcal{M}$. For such N, we set $m = N'$. The test property can only be violated if for some Ψ of size N we have that $\Psi \not\equiv 0$, while $(\forall a \in S^{\ell}), \Psi(NW^{p_m}(a)) = 0$. By Lemma 9 we obtain that p_m can be written as $p_m = f_m/c_m$, for f_m and $c_m \in \mathbb{Z}/\{0\}$ that are computed by constant-free arithmetic circuits of size and formal degree at most $N^{c_0}m^{c_0 e} = (m)^{c_0(r+e)} = m^k$. We know the latter does not hold for $m \in \mathcal{M}$.

We continue the description of the algorithm. We produce a set H to sample Φ with, namely we take H to be the output of $NW^{p_m}(\cdot)$ on S^{ℓ}. We have that $|H| = (Nm+1)^{\ell} = 2^{O(N^{2/r}\log N)}$. We do simulations of the machine M for $E(\{p_m\})$ to get all the bits of all the evaluations of $p_m(\cdot)$ on S^{ℓ}. As p_m is multilinear with coefficients of bit length at most m^e, we can bound the bit size of any such evaluation by $O(m^e \log(Nm))$. This means that the inputs $(1^m, a_1, \ldots, a_m, i, b)$ that we simulate M on, have bit size $O(m \log(Nm))$ (recall that i is given in binary). The simulation for a single such input thus costs NTIME$[2^{O(m^d \log^d(Nm))}]$ = NTIME$[2^{O(N^{d/r}\log^d N)}]$. To get all bits of an evaluation for a single element in H therefore takes at most NTIME$[O(m^e \log(Nm)) \cdot 2^{O(N^{d/r}\log^d N)}]$, which we can bound as NTIME$[2^{O(N^{d/r}\log^d N)}]$. To construct the entire set H we can use the same asymptotic time bound assuming wlog. that $d \geq 2$.

If during the process of obtaining all the bits we obtain a flag bit set to 0, we reject. This means that on every path where we pass this check, we have obtained a hitting set, unless N is an input length where the test property is not satisfied. On these paths, we continue to verify deterministically that $f(h) = 0$ for all $h \in H$. If yes, then we accept, else reject. By our previous remarks, for infinitely many N, this correctly decides whether $\Phi \equiv 0$.

Let us consider the cost of evaluation of Φ on elements of H. For $a \in S^{\ell}$ and subset S_j in the Nisan-Wigderson design, the bit size of $p_m(a_{|S_j})$ is $O(m^e \log(Nm))$. By Corollary 1 this means that the absolute value of any gate of Φ for input $NW^{p_m}(a)$ is at most $2^{O(N^2 m^{2e} \log^2(Nm))} = 2^{N^{O(1)}}$. Thus intermediate values can be represented by $poly(N)$ bits. We conclude that evaluation of Φ on a single element of the test set H cost time $poly(N)$. We can conclude the entire cost of our test algorithm is NTIME$[2^{O(N^{d/r}\log^d N)}]$. As r can be chosen arbitrarily large and d is an absolute constant not depending on r, we conclude that low-PIT $\in \bigcap_{\epsilon>0}$ i.o-NTIME$[2^{N^{\epsilon}}]$. $\square$

5.2.1 Proof of Theorem 5

The following corollary to Lemma 9 follows straightforwardly:

Corollary 2. *Let $e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be an monotone non-decreasing function with $e(n) = \omega(1)$ and $n^{e(n)} < 2^n$. Suppose that $\{p_n\}$ is a family of multilinear polynomials in n variables with coefficients of bit size at most $n^{e'}$ for some integer e', such that* $\mathrm{ASIZEDEG}'(p_n) > n^{e(n)}$*. Then there exists an absolute constant $c > 0$ such that for any division-free constant-free arithmetic circuit Φ of size n with $deg(\Phi) \leq n$, if we take m such that $m^{e(m)-e'c} > n^c$ and let ℓ be given by Lemma 8, then for all large enough n, $\Phi \equiv 0 \Leftrightarrow (\forall a \in S^\ell), \Phi(NW^{p_m}(a)) = 0$, where $S = [nm + 1]$.*

We will describe an NSUBEXP algorithm for low-PIT. Let Φ be an arithmetic circuit of size N, and let f be the polynomial computed by it. First we check that the formal degree of Φ is bounded by N, if not reject. Else, consider the given family $\{p_m\} \in$ ml·NEXP. By reindexing we may assume wlog. that p_m is defined over m variables. Let $e' \geq 1$ be such that p_m has coefficients of bit size at most $m^{e'}$.

We use the fact (Proposition 3) that a·NEXP = a·(NEXP ∩ coNEXP). This means that we have a nondeterministic flag machine M running in time $2^{(n')^d}$ for inputs of size n' that can compute the characteristic function of $E(\{p_m\})$, where d is an absolute constant.

Let c be the constant given by Corollary 2. Let $m = \min\{m : m^{e(m)-ce'} > N^c\}$. An easy argument shows that $m = N^{o(1)}$. We can compute m by linear search in time $poly(N)$ by repeatedly computing $r(i) := i^{e(i)-e'c}$ for $i = 1, 2, \ldots$, until the first $r(i) > N^c$ is found. There will be $N^{o(1)}$ iterations, each taking $poly(N, e(N)) = poly(N)$ time. By Corollary 2, there exists N_0 such that for all $N \geq N_0$ the following test property holds: $\Phi \equiv 0 \Leftrightarrow (\forall a \in S^\ell), \Phi(NW^{p_m}(a)) = 0$, where $S = [Nm+1]$ with $\ell = O(m^2/\log N)$ taken according to Lemma 8. We can hardcode the threshold N_0 in our algorithm, and perform a brute-force check for $N < N_0$. For $N \geq N_0$ we operate as follows. We produce a set H to sample Φ with, namely take H to be the output of $NW^{p_m}(\cdot)$ on S^ℓ. We have that $|H| = (Nm + 1)^\ell = 2^{N^{o(1)}}$.

We do simulations of the machine M for $E(\{p_m\})$ to get all the bits of all the evaluations of $p_m(\cdot)$ on S^ℓ. As p_m is multilinear with coefficient of at most $m^{e'}$ many bits, we can bound the bit size of any such evaluation by $O(m^{e'} \log(Nm))$. This means that the inputs $(1^m, a_1, \ldots, a_m, i, b)$ that we simulate M on, have bit size $O(m \log(Nm))$ (recall i is given in binary). The simulation for a single such input thus costs $\mathrm{NTIME}[2^{O(m^d \log^d(Nm))}] = \mathrm{NTIME}[2^{N^{o(1)}}]$. To get all bits of an evaluation for a single element in H therefore takes at most $\mathrm{NTIME}[O(m^{e'} \log(Nm)) \cdot 2^{N^{o(1)}}] = \mathrm{NTIME}[2^{N^{o(1)}}]$. To construct the entire set H thus takes $\mathrm{NTIME}[2^{N^{o(1)}}]$.

If during the process of obtaining all the bits we obtain a flag bit set to 0, we reject. This means that if on every path where we pass this check, we have obtained a hitting set, provided N is large enough. On the path where we pass this check, we continue to verify deterministically that $f(h) = 0$ for all $h \in H$. If yes, then we accept, else reject. By our previous remarks, for infinitely many N, this correctly decides whether $\Phi \equiv 0$.

Let us consider the cost of evaluation of Φ on elements of H. For $a \in S^\ell$ and subset S_j in the Nisan-Wigderson design, the bit size of $p_m(a_{|S_j})$ by $O(m^{e'} \log(Nm))$. By Corollary 1 this means that the absolute value of any gate of Φ for input $NW^{p_m}(a)$ is at most $2^{O(N^2 m^{2e'} \log^2(Nm))} = 2^{N^{O(1)}}$. Thus intermediate values can be represented by $poly(N)$ bits. We conclude that evaluation of Φ on a single element of the test set H cost time $poly(N)$. We can conclude the entire cost of our test algorithm is $NTIME[2^{N^{o(1)}}]$. □

5.2.2 Proof of Theorem 6

We first prove the hardness-to-randomness direction. We use the following straightforward corollary to Lemma 9:

Corollary 3. *Let $s(n) = n^{\omega(1)}$ be a function. Suppose that $\{p_n\}$ is a family of multilinear polynomials in n variables with coefficients of bit size at most $n^{e'}$ for some integer e', such that p_n cannot be written as q_n/c_n for $c_n \in \mathbb{Z}\backslash\{0\}$ for any q_n and c_n computed by constant-free arithmetic circuits of size $s(n)$. Then there exists an absolute constant $c > 0$ such that for any division-free constant-free arithmetic circuit Φ of size n with $deg(\Phi) \leq n$, if we take m such that $s(m) \cdot m^{-e'c} > n^c$ and let ℓ be given by Lemma 8, then for all large enough n, $\Phi \equiv 0 \Leftrightarrow (\forall a \in S^\ell), \Phi(NW^{p_m}(a)) = 0$, where $S = [nm+1]$.*

We will describe an i.o-NTIME$[2^{n^{o(1)}}]/n^{o(1)}$ algorithm for low-PIT. Let Φ be an arithmetic circuit of size N, and let f be the polynomial computed by it. First we check that the formal degree of Φ is bounded by N, if not reject. Else, consider the given family $\{p_m\}$. By reindexing we may assume wlog. that p_m is defined over m variables. Let $e' \geq 1$ be such that p_m has coefficients of bit size at most $m^{e'}$. We have that for infinitely many m, p_m has ASIZEDEG$'$-hardness larger than $s(m)$, where $s(m) = m^{\omega(1)}$. The m that have this property we call *good*.

We use the complementation property for ml·NE/*lin*, cf. Proposition 3 and the comment thereafter. This means that we have a nondeterministic flag machine M running in time $2^{O(n')}$ with $O(n')$ bits of advice for inputs of size n' that can compute the characteristic function of $E(\{p_m\})$ Let c be the constant given by Corollary 2. For input size N the algorithm receives two strings of advice α and β. First, if there exists a good m_0 such that $s(m_0)(m_0)^{-ce'} \in [N^c, (N+1)^c]$, then $\alpha = 1^{m_0}$. If there is no such m_0, then α is set to the empty string. A simple argument shows that $|\alpha| = N^{o(1)}$. For the second piece of advice β we obtain the advice M needs so we can complete the simulations which we describe below (we will analyze this in more detail there).

In case the algorithm receives the empty string for α, it halts and rejects. Otherwise, we set $m = m_0$. Note that as N^c is a strict monotone increasing function it must be that for infinitely many N we obtain a good m_0 as advice. By Corollary 2, provided N is large enough, the following test property holds: $\Phi \equiv 0 \Leftrightarrow (\forall a \in S^\ell), \Phi(NW^{p_m}(a)) = 0$, where $S = [Nm+1]$ with $\ell = O(m^2/\log N)$ taken according to Lemma 8.

Let us continue the description of the algorithm. We produce a set H to sample Φ with, namely take H to be the output of $NW^{p_m}(\cdot)$ on S^ℓ. We have that $|H| = (Nm+1)^\ell = 2^{N^{o(1)}}$.

We do simulations of the machine M for $E(\{p_m\})$ to get all the bits of all the evaluations of $p_m(\cdot)$ on S^ℓ. As p_m is multilinear with coefficient of at most $m^{e'}$ many bits, we can bound the bit size of any such evaluation by $O(m^{e'}\log(Nm))$. This means that the inputs $(1^m, a_1, \ldots, a_m, i, b)$ that we simulate M on, have bit size $O(m\log(Nm))$ (recall i is given in binary). For the string β we give the advice that M needs for all input lengths up to this maximum bit size, which is $O(m^2\log^2(Nm)) = N^{o(1)}$ in total. Given such advice, the simulation for a single such input thus costs NTIME$[2^{O(m\log(Nm))}]$ = NTIME$[2^{N^{o(1)}}]$. To get all bits of an evaluation for a single element in H therefore takes at most NTIME$[O(m^{e'}\log(Nm))\cdot 2^{N^{o(1)}}]$ = NTIME$[2^{N^{o(1)}}]$ with the same amount of advice. We conclude that we can construct the entire set H in NTIME$[2^{N^{o(1)}}]$ with $N^{o(1)}$ advice.

If during the process of obtaining all the bits we obtain a flag bit set to 0, we reject. This means that if on every path where we pass this check, we have obtained a hitting set, provided N is large enough. On the path where we pass this check, we continue to verify deterministically that $f(h) = 0$ for all $h \in H$. If yes, then we accept, else reject. By our previous remarks, for infinitely many N, this correctly decides whether $\Phi \equiv 0$.

Let us consider the cost of evaluation of Φ on elements of H. For $a \in S^\ell$ and subset S_j in the Nisan-Wigderson design, the bit size of $p_m(a_{|S_j})$ by $O(m^{e'}\log(Nm))$. By Corollary 1 this means that the absolute value of any gate of Φ for input $NW^{p_m}(a)$ is at most $2^{O(N^2 m^{2e'}\log^2(Nm))} = 2^{N^{O(1)}}$. Thus intermediate values can be represented by $poly(N)$ bits. We conclude that evaluation of Φ on a single element of the test set H cost time $poly(N)$. We can conclude the entire cost of our test algorithm is $NTIME[2^{N^{o(1)}}]$ with $N^{o(1)}$ advice, and that for infinitely many input lengths N the algorithm is correctly decides low-PIT. This concludes the proof of the hardness-to-randomness direction.

Next we prove the converse direction. Suppose that low-PIT $\in$ i.o-NTIME$[2^{n^{r(n)}}]/r(n)$ for some $r(n) = n^{e(n)}$, where $e(n) = o(1)$. We assume wlog that $e(n)$ is monotonically decreasing. We are done if $\{per_n\} \notin$ ASIZEDEG$'(poly)$, so assume that $\{per_n\}$ has size and formal degree upper-bounded by n^ℓ, for $\ell \in \mathbb{N}$. Let $k(n) = O(\log(n))$ be a monotonically increasing slowly growing function of n, which we shall specify more carefully later.

Using the proof strategy of Lemma 3 we know that when $n^{k(n)}$ is time-constructible there exists a constant c_2 and a deterministic Turing machine M_0 running in time $n^{c_2 k(n)}$ making queries to $0,1$-permanent, so that M_0 on input 1^n can construct the description of a multilinear polynomial f_n in n variables of degree $3k(n)$ that, for all but finitely many n, cannot be written as $f_n = g_n/c_n$ for polynomial g_n and integer constant c_n computable by size $n^{k(n)}$ constant-free circuits. We will not be able to ensure constructibility of $n^{k(n)}$ - instead, we give M_0 advice which specifies $k(n)$. Wlog. M_0 makes queries that are all of the same length m^2 to per$_m$ with $m(n) = n^{c_3 k(n)}$ for some integer constant c_3.

Also we have a machine M_1 running in NTIME$[2^{r(n)}]/r(n)$ for deciding low-PIT. This machine correctly decides instances of low-PIT for infinitely many input lengths. Using the ideas in the proof of Lemma 4, we can show that for the function $R(m) = \lceil m^{l+1} \cdot \log^d m \rceil$, we can reduce testing a candidate circuit for per_m of size m^l to a low-PIT instance of size $R(m)$. Call a length n *good* if there exists $\ell_0 \in [R(n^{c_3 k(n)}), R((n+1)^{c_3 k(n)})]$ such that M_1 is correct for instances of size ℓ_0. We choose $k(n)$ so that $R(N^{c_3 k(n)})^{e(n)} < N$. For instance, choosing $k(n) = 1/(c_3(l+2)e(n))$ would suffice. Since $e(n) = o(1)$ and monotonically decreasing, we have that $k(n) = \omega(1)$ and monotonically increasing, but without loss of generality we can assume $e(n)$ is large enough that $k(n) = O(\log(n))$.

We construct the following nondeterministic Turing machine M for deciding the evaluation language of a family of polynomials $\{h_n\}$. The polynomial h_n will be f_n for n good, and 0 for n bad. On input $(1^n, a_1, a_2, \ldots, a_n, i, b)$ of size N the machine requires four types of advice. It receives an N-bit string indicating for all lengths in $[N]$ whether they are good or not. If there are no good lengths in this input the rest of the advice can be arbitrary. Otherwise, for the largest $n_0 \in [N]$ that is good we give in binary the good length ℓ_0 in $[R(n_0^{c_3 k(n_0)}), R((n_0+1)^{c_3 k(n_0)})]$ and we give a string γ of $(\ell_0)^{e(\ell_0)}$ bits of advice the machine M_1 needs to correctly run at the input length ℓ_0. The fourth type of advice is simply an exhaustive list of numbers n between 1 and N such $k(n) > k(n-1)$. Since $k(N) = O(\log(N))$, the list has size $O(\log(N))$ and moreover each number on the list can be represented with $O(\log(N))$ bits.

The machine M operates as follows. By looking at the first advice string, if n is not a good length it rejects. Otherwise, M guesses an arithmetic circuit Φ for per$_{m_0}$, where $m_0 = n_0^{c_3 k(n_0)}$ of size m_0^ℓ. M now nondeterministically simulates M_1 with the advice string γ. On paths where a reject is found, M rejects also. On other paths it simulates M_0 on 1^n. When M_0 needs to know the value of $k(n)$, it can figure this out from the list given as the fourth type of advice. Whenever a query is made to per$_{m(n)}$, let's say to a matrix A, it pads A to a matrix A' of dimension m_0, simply by adding 1's along the diagonal and 0's everywhere else. It evaluates $\phi(A')$ to answer the query. This way M obtain the description of a multilinear polynomial g_n, which it then evaluates on its input.

We show that M operates in time $2^{O(N)}$ with linear advice, and that it is the evaluation machine for a family of polynomials $\{h_n\}$ such that $\{h_n\} \notin \text{ASIZEDEG}'(poly)$. For the time and advice bounds, note that the time taken is $2^{O(N)}$ by choice of $k(n)$ and the time bound on M_0. Also each of the four types of advice has at most linear length - the first is an N-bit string, the second is $O(\log(N))$ bits long, and the third is $O(N)$ bits long by choice of $k(n)$ and since $e(n)$ is monotonically decreasing, and the last is $O(\log^2(N))$ bits long.

Next, we argue that M is an evaluation machine for the family of polynomials $\{h_n\}$ defined by setting $h_n = f_n$ when n is good, and $h_n = 0$ otherwise. Note that $\{h_n\} \notin \text{ASIZEDEG}'(n^{k(n)})$ because for all but finitely many n, f_n does not have arithmetic circuits of size and formal degree bounded by $n^{k(n)}$. When n is bad, by the way we defined our advice strings, M correctly rejects. If n is good, for any N which is a bit length for evaluation of h_n on certain inputs, the advice will indicate that there is at least one good length (since n is good). By the definition of goodness of lengths, M will run M_1 on a circuit for which M_1 returns the correct answer, and hence we get that on accepting paths of M_1 (which are guaranteed to exist on a correct guess ϕ for a circuit for Permanent), by specializing the proof of Lemma 4 to low-PIT, the machine M_0 indeed produces a description of f_n. Thus h_n agrees with f_n on good n, as desired. This concludes the proof, as we get $n^{k(n)}$ size lower bounds for h_n, and $n^{k(n)} = n^{\omega(1)}$ by choice of $k(n)$. □

References

[1] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1:1–54, 2009.

[2] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Addendum in vol. 2 of same journal.

[3] P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory.* Springer Verlag, 2000.

[4] R. DeMillo and R. Lipton. A probabilistic remark on algebraic program testing. *Inf. Proc. Lett.*, 7:193–195, 1978.

[5] M. Jansen. Extracting roots of arithmetic circuits by adapting numerical methods. In *Proc. 2nd Symp. on Innovations in Computer Science*, 2011.

[6] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity testing means proving circuit lower bounds. *Computational Complexity*, 13(1–2):1–44, 2004.

[7] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2010.

[8] P. Koiran. Shallow circuits with high powered inputs. In *Proc. 2nd Symp. on Innovations in Computer Science*, 2011.

[9] P. Koiran and S. Perifel. Interpolation in Valiant's theory. *Computational Complexity*, 20(1):1–20, 2011.

[10] N. Nisan and A. Wigderson. Hardness versus randomness. *J. Comp. Sys. Sci.*, 49:149–167, 1994.

[11] Rahul Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

[12] N. Saxena. Progress of polynomial identity testing. Technical Report ECCC TR09-101, Electronic Colloquium in Computational Complexity, 2009.

[13] J.T. Schwartz. Fast probabilistic algorithms for polynomial identities. *J. Assn. Comp. Mach.*, 27:701–717, 1980.

[14] V. Strassen. Vermeidung von divisionen. *Journal für die Reine und Angewandte Mathematik*, 264:182–202, 1973.

[15] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20:865–877, 1991.

[16] L. Valiant. The complexity of computing the permanent. *Theor. Comp. Sci.*, 8:189–201, 1979.

[17] R. Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of 26th IEEE Conference on Computational Complexity*, 2011.

[18] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation (EUROSAM '79)*, volume 72 of *Lect. Notes in Comp. Sci.*, pages 216–226. Springer Verlag, 1979.

A Characterization Results

Recall again that there exists a family of poly-size arithmetic circuits over $\mathbb{Q}$ with arbitrary use of divisions that compute $\{per_n\}$ iff $\{per_n\} \in \text{ASIZE}'(poly)$. We also remark that in Ref. [6] arithmetic circuits can have arbitrary constants, but that the bit size of constants is counted towards the size. Within the polynomial size regime this is equivalent to the constant-free model. Thus we can state Theorem 4.1 and Theorem 5.2 of Ref. [6] as follows:

Theorem 14 (Theorem 4.1 in [6]). *PIT $\in$ NSUBEXP $\Rightarrow$ NEXP $\not\subseteq$ P/poly, or $\{per_n\} \notin \text{ASIZE}'(poly)$.*

Theorem 15 (Theorem 5.2 in [6]). [5] *$\text{NEXP}^{\text{RP}} \not\subseteq$ P/poly or $\{per_n\} \notin \text{ASIZE}'(poly)$.*

We use Lemma 3.16 in Ref. [9], which can be adapted[6] as follows:

Theorem 16 ([9]). *$\{per_n\} \in \text{ASIZE}'(poly) \Rightarrow \text{a}\cdot\text{CH}/poly \subseteq \text{ASIZE}'(poly(n))$.*

We first give a characterization for the r.h.s. of Theorem 14.

Theorem 17. *$\text{a}\cdot\text{NEXP}/lin \not\subseteq \text{ASIZE}'(poly) \Leftrightarrow \text{NEXP} \not\subseteq \text{P}/poly$, or $\{per_n\} \notin \text{ASIZE}'(poly)$.*

Proof. We first prove the left-to-right direction. We argue by the contrapositive. Suppose that NEXP $\subseteq$ P/*poly* and $\{per_n\} \in \text{ASIZE}'(poly)$. By Theorem 16, the latter implies that $\text{a}\cdot\text{CH}/poly \subseteq \text{ASIZE}'(poly)$. We use that NEXP $\subseteq$ P/*poly* $\Rightarrow$ NEXP/*lin* $\subseteq$ P/*poly*. Hence $\text{a}\cdot\text{NEXP}/lin \subseteq \text{a}\cdot\text{P}/poly \subseteq \text{a}\cdot\text{CH}/poly \subseteq \text{ASIZE}'(poly)$.

Next we prove the right-to-left direction. Suppose that $\text{a}\cdot\text{NEXP}/lin \subseteq \text{ASIZE}'(poly)$. As $\{per_n\}$ is easily seen to be in $\text{a}\cdot\text{NEXP}/lin$ we immediately have that $\{per_n\} \in \text{ASIZE}'(poly)$.

[5]The original statement would be for ASIZE, but this can be strengthened to ASIZE′.

[6]Aside from a change in language, the statement here differs in that we work with ASIZE′ instead of ASIZE, but this can be dealt with.

Now consider an arbitrary $L \in$ NEXP. Let N be the corresponding NEXP-machine. Using Proposition 2, let F_n be the multilinear extension of the characteristic function χ_L of L on $\{0,1\}^n$.

We claim that $\{F_n\} \in$ a·NEXP/*lin*. This is done as following given $(1^n, a_1, \ldots, a_n, i, b)$ as input we require as advice $|L \cap \{0,1\}^n|$ in binary, which is linear in the input size. We then simulate the nondeterministic machine N for all inputs in $\{0,1\}^n$. On paths where we see less than $|L \cap \{0,1\}^n|$ accepts, we reject. On all other paths we know exactly the function χ_L. We can now use the formula of Proposition 2 for F_n to compute $F(a_1, a_2, \ldots, a_n)$ in deterministic exponential time. This proves the claim.

We get that $\{F_n\} \in$ ASIZE$'$(*poly*). This means that we have constant-free (division-free) arithmetic circuits Φ_1 and Φ_2 of size at most $p(n) = n^{O(1)}$, such that Φ_2 does not contain variables and computes some nonzero constant $c \in \mathbb{Z}$. Furthermore, if Φ_1 computes G_n then it holds that $G_n = c \cdot F_n$. For input $a \in \{0,1\}^n$, $F_n(a) \in \{0,1\}$, which means for such inputs $G_n(a) \in \{0, c\}$. We want to evaluate Φ_1 modulo some prime number q that does not divide c. This will tell us $\chi_L(a)$. We have that $|c| \leq 2^{2^{p(n)}}$ due to Proposition 1. This means that c has at most $2^{p(n)}$ prime factors. Hence, also using the Prime Number Theorem there exists a prime number q of $p(n)^2$ bits, provided n is large enough, that does not divide c. As our task is to show only the non-uniform upper bound $L \in$ P/*poly*, mere existence of this number q suffices for our purposes. We conclude that there exist *poly* size Boolean circuits for χ_L. We have shown that NEXP $\subseteq$ P/*poly*. $\square$

As a corollary to Theorem 17 we get that Theorem 14 is equivalent to the statement PIT $\in$ NSUBEXP $\Rightarrow$ a·NEXP/*lin* $\not\subseteq$ ASIZE$'$(*poly*). Incidentally, one obvious way to prove the stronger statement PIT $\in$ NSUBEXP $\Rightarrow$ a·NEXP $\not\subseteq$ ASIZE$'$(*poly*), would be to 'just' show that a·NEXP $\subseteq$ ASIZE$'$(*poly*) $\Rightarrow$ a·NEXP/*lin* $\subseteq$ ASIZE$'$(*poly*). In spite of the striking resemblance to the know fact that NEXP $\subseteq$ SIZE(*poly*) $\Rightarrow$ NEXP/*poly* $\subseteq$ SIZE(*poly*), it is not clear whether this implication is true. To prove the latter, one simply makes the advice part of the input, giving some language in NEXP, which by assumption is in P/*poly*. Hardcoding the right advice one obtains poly-size circuits for the original NEXP/*poly* language. For the former however, when making advice part of the input, the resulting language does not necessarily correspond to the an evaluation language $E(\{g_n\})$ of a family of polynomials $\{g_n\}$. It seems hard to enforce this semantic condition.

We observe that also Theorem 15 can be characterized in our framework. Namely, by relativizing the proof of Theorem 17 one can conclude that the statement of this theorem is equivalent to the following lower bound:

Theorem 18. a·NEXP$^{\mathrm{RP}}$/*lin* $\not\subseteq$ ASIZE$'$(*poly*).

B Specialization to Low-Degree

Let us first say some words about adapting Lemma 4 to use ASIZE$'$ instead of ASIZE. We use the notation of the proof in Ref. [7]. There one crucial idea is to guess a candidate circuit for per_m of size $a(m)$, and to check whether the candidate is correct, which can be done, as by the self-reducibility of per_m this reduces to a PIT instance of size $m^2 \cdot \log^d m \cdot a(m)$. For the adaption, the idea is to guess two constant-free arithmetic circuits Φ_1 and Φ_2 of size $a(m)$, where Φ_2 does not contain variables, and to verify that $\Phi_1 \equiv \Phi_2 \cdot per_m$. One can still use a self-reducibility property to reduce this check to a single PIT instance, cf. Theorem 3.10 in [6]. This only changes the 'bottom-most' equation in the chain of self-reductions. The proof of the lemma then goes through without a problem.

Next we show that we can derive a low-PIT version of Theorem 7. Observe this only requires a specialization of Lemma 4, where we replace the condition "PIT $\in$ NTIME($t(N)$) and $\{per_n\} \in$ ASIZE$'(a(n))$" by "low-PIT $\in$ NTIME($t(N)$) and $\{per_n\} \in$ ASIZEDEG$'(a(n))$". For the modified proof one guesses two constant-free circuits Φ_1 and Φ_2, where Φ_2 does not contain variables, of size and formal degree at most $a(m)$. Formal degree is computed easily and $a(\cdot)$ is time-constructible, so we can reject for guesses where the formal degree is larger than $a(m)$. The guesses are candidates for satisfying $\Phi_1 \equiv \Phi_2 \cdot per_m$. Invoking the self-reducibility property, as was done in the above, increases formal degree to an inconsequential extent, Namely, checking that $\Phi_1 \equiv \mathrm{per}_m \cdot \Phi_2$ reduces to checking the identity of an constant-free arithmetic circuit Ψ of size at most $m^2 \cdot \log^d m \cdot a(m)$ and formal degree at most $(m+1)a(m)$. If needed, we use some padding to ensure that $deg(\Psi) \leq |\Psi|$, and hence we can use our low-PIT algorithm to correctly decide whether $\Psi \equiv 0$. Consequently, the proof of the lemma goes through for the modification "low-PIT $\in$ NTIME($t(N)$) and $\{per_n\} \in$ ASIZEDEG$'(a(n))$". This then yields Theorem 9.