Electronic Colloquium on Computational Complexity, Report No. 207 (2026)

Towards an Interesting VPSPACE-complete Problem

Marco Carmosino* Nikhil Gupta† Ilya Volkovich‡

Abstract

We introduce a variant of a polynomial family called $\widetilde{\mathcal{TQBF}}$ obtained from 'arithmetization' of the popular PSPACE-complete problem - TQBF. This polynomial family has several nice characteristics such as: $\widetilde{\mathcal{TQBF}} \in \mathsf{VPSPACE}_b$, where $\mathsf{VPSPACE}_b$ is the 'bounded' algebraic version of PSPACE, it is *self-testable*, it captures the computational power of PSPACE and others. Although the first version of $\widetilde{\mathcal{TQBF}}$ appeared in an earlier work of Carmosino et al. (RANDOM, 2015), we believe that our presentation is cleaner and simpler.

Building on that, we construct another polynomial family, $\mathcal{TP} \in \mathsf{VPSPACE}_b$, by mixing $\widetilde{\mathcal{TQBF}}$ and $\mathcal{PERM}$, the family of the Permanent polynomial. While we are unable to prove that $\mathcal{TP}$ is $\mathsf{VPSPACE}_b$-complete, we show that it has many traits of $\mathsf{VPSPACE}_b$-completeness as well as several important consequences in computational complexity, which are listed below.

- We show that if $\mathcal{TP}$ can be computed by circuits from a circuit class $\mathcal{C} \subseteq \mathsf{VNP}$ then $\mathsf{VPSPACE}_b \subseteq \mathcal{C}$.
- We also conclude that if $\mathcal{C}$ has a black-box PIT algorithm that uses sub-polynomial space, then $\mathcal{TP}$ cannot be computed by polynomial-size arithmetic circuits from $\mathcal{C}$.
- Finally, we prove a version of a Karp-Lipton style collapse theorem by showing that if $\widetilde{\mathcal{TQBF}}$ has "small" arithmetic circuits then PSPACE collapses to NP with a PIT oracle (i.e. $\mathsf{PSPACE} \subseteq \mathsf{NP}^{\mathsf{PIT}}$).

The second result makes a partial progress towards the resolution of an open problem posed in a survey by Shpilka & Yehudayoff (Foundations and Trends in Theoretical Computer Science, 2010). As a corollary, we give an "inconsistent triad" of PIT and circuit lower bounds, similar to the one given by Kabanets and Impagliazzo (Computational Complexity, 2004). Finally, we note that Malod gave complete polynomial families for VPSPACE, the 'unbounded' algebraic version of PSPACE (Foundations of Computation Theory, 2011).

AI Usage Statement

This paper was written without the use of ChatGPT, large language models (LLMs), or other generative AI tools.

*IBM Email: mlc@ibm.com

†Department of Computer Science and Engineering, Indraprastha Institute of Information Technology (IIIT) Delhi. Email: nikhil@iiitd.ac.in.

‡Computer Science Department, Boston College, Chestnut Hill, MA. Email: ilya.volkovich@bc.edu

ISSN 1433-8092

1 Introduction

Computational complexity aims to understand relationships between various complexity classes. Some of the well-studied classes in the literature are P, NP and PSPACE. In the seminal work of [Val79], Valiant defined the classes VP and VNP as the algebraic analogues of P and NP, respectively, conjecturing that $\mathsf{VP} \neq \mathsf{VNP}$. As in the case of the P vs NP problem, it follows from the definition that $\mathsf{VP} \subseteq \mathsf{VNP}$. Yet, it has since been a major open question to separate these two classes. However, we have a definite answer to this question in the monotone and multilinear settings. In particular, monotone VP was shown to be a strict set of monotone VNP [Yeh19, Sri20]. On the other hand, in the multilinear world, $\mathsf{VP} = \mathsf{VNP}$ [MST16].

A standard way to approach the questions of the kind $\mathsf{VP} \stackrel{?}{=} \mathsf{VNP}$ is by identifying the relevant "complete" problems. Indeed, in the same paper [Val79], it was also shown that the Permanent is VNP-complete under the appropriate notion of reduction. By contrast, VP has a very simple definition as the class of all (families of) polynomials computed by arithmetic circuits of polynomial degree and size. Given that, the question of separating VP from VNP can be rephrased as a natural question of proving circuit lower bounds:

"Show that the Permanent cannot be computed by polynomial-size arithmetic circuits."

As a subsequent step in mirroring the traditional Boolean complexity classes, in [KP09], Koiran and Perifel introduced the class $\mathsf{VPSPACE}$ as the algebraic analogue of PSPACE. Indeed, $\mathsf{VPSPACE}$ can be thought of as a class of polynomials whose coefficients can be computed in polynomial space. The natural corresponding question is to study the relationship between $\mathsf{VP}, \mathsf{VNP}$ and $\mathsf{VPSPACE}$. Although not immediately, it follows from the previous results (e.g. [Bür00]) that $\mathsf{VP} \subseteq \mathsf{VNP} \subseteq \mathsf{VPSPACE}$. Yet, the question of proper containment amongst $\mathsf{VPSPACE}$ and the other classes lacks interest, if posed in its simple form. In particular, while the degrees of the polynomials in VP and VNP are polynomially-bounded, the definition of $\mathsf{VPSPACE}$ allows arbitrary (i.e. exponentially-large) degrees. Hence, clearly, $\mathsf{VP} \subseteq \mathsf{VNP} \subsetneq \mathsf{VPSPACE}$. To level the playing field, Koiran and Perifel also introduced the class $\mathsf{VPSPACE}_b$ - the bounded version of $\mathsf{VPSPACE}$ - and established the containment $\mathsf{VP} \subseteq \mathsf{VNP} \subseteq \mathsf{VPSPACE}_b$. As a (potential) alternative approach, they also considered the class VP_{nb} introduced by Malod [Mal03, Mal07] as the unbounded version of VP, and have shown that one can "transform" a polynomial from $\mathsf{VPSPACE}$ to $\mathsf{VPSPACE}_b$ (and vice versa), using Kronecker's transformation. Based on that, they established that $\mathsf{VP} = \mathsf{VPSPACE}_b \iff \mathsf{VP}_{nb} = \mathsf{VPSPACE}$. As a conclusion, it is necessary and sufficient to focus on the "bounded" regime - i.e. $\mathsf{VPSPACE}_b$. It was recently shown in [CGT24] that monotone VNP is strictly contained in monotone $\mathsf{VPSPACE}_b$.

1.1 Polynomial Identity Testing

A very important problem in computational complexity is *polynomial identity testing* (PIT), where we want to test if a multivariate polynomial given either as an arithmetic circuit or via a black-box access [1] is identically zero. A randomized polynomial-time black-box PIT algorithm is known due to the DeMillo-Lipton-Schwartz-Zippel Lemma [DL78, Zip79, Sch80]. A deterministic polynomial-time PIT algorithm is conjectured to exist, yet obtaining it seems far from our reach currently. This is a very important open question, and such a result would also imply strong lower bounds in computational complexity theory [HS80, KI04, Agr05]. One can observe that designing a deterministic polynomial-time black-box PIT for a class of arithmetic circuits $\mathcal{C}$ is equivalent to constructing a polynomial-size *hitting set* (Definition 2.21) for that same class.

PIT has been extensively studied in the last two decades. Efficient deterministic PIT algorithms are known for many interesting classes of constant-depth arithmetic circuits [KS01, SS12, SS13, ASSS16, SV18, PS21, DDS21, LST24, BSV26], constant-read arithmetic formulas [SV15, AvMV15, MRS16, MV18, BGV23], read-once oblivious algebraic branching programs (ROABPs) [FS12, FSS14, AGKS15, GKS17] and other restricted classes. PIT has also been used in many important algorithmic results such as: primality testing [AB03, AKS04], checking whether a graph has a perfect matching [Lov79, KUW86, MVV87, ST17] and others. These connections, together with the strong relationships between PIT and lower bounds, and the quest for derandomization, put PIT on the front burner in theoretical computer science. For more details, we direct the interested readers to various surveys on the topic [SY10, Sax09, Sax14, DG24].

[1] A black-box of a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ outputs $f(\mathbf{a})$, given a point $\mathbf{a} \in \mathbb{F}^n$ as an input, in a unit time.

1.2 Reconstruction of Arithmetic Circuits

Arithmetic circuit reconstruction is the following problem - given black-box access to a size-s arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, output an arithmetic circuit C' computing f. It is another very significant problem studied in computational complexity. Reconstruction of arithmetic circuits is an algebraic analogue of exact learning of Boolean functions introduced in [Ang88]. Two important paradigms of exact learning Boolean functions are: learning with membership queries and learning with equivalence queries. In [Vol16], it was shown that for polynomials these notions are equivalent. In particular, a reconstruction algorithm for an arithmetic class $\mathcal{C}$ subsumes the black-box PIT algorithm for that same class which, in turn, can be used to simulate equivalence queries. However, unlike the latter, we do not have a randomized polynomial-time reconstruction algorithm for polynomial-size arithmetic circuits.

Connections between circuit reconstruction and lower bounds are also known [FK09, Vol16]. The reconstruction of arithmetic circuits has been studied in the literature in the worst-case and average-case settings. We have efficient reconstruction worst-case algorithms for some special sub-classes of constant-depth circuits [BOT88, KS01, KS06, Shp09, GKL12, Sin16, Vol17, Sin22, BSV21, BSV25, SS25, SS26]. On the other hand, efficient average-case reconstruction algorithms are known for more general sub-classes of constant-depth circuits [KS19, GKS20, BGKS22], algebraic branching programs [KNST17], arithmetic formulas [GKQ14, GST23]. For more details, we refer the interested readers to [SY10, Vol16].

1.3 Motivation

Given the long-standing difficulty in achieving any advancement in the problem of proving (unconditional) super-polynomial lower bounds on the Permanent, it was suggested to seek connections between this and other long-standing open problems in algebraic complexity such as PIT and circuit reconstruction. The survey of [SY10] lists the following open problems:

Open Problem 17: *"Assume that a circuit class $\mathcal{C}$ has a PIT algorithm that runs in time $\mathrm{poly}(s)$ for circuits of size s. Does the permanent require super-polynomial circuits from $\mathcal{C}$?"*

Open Problem 32: *"Prove that if an arithmetic circuit class $\mathcal{C}$ is learnable then the permanent does not have polynomial-size circuits from $\mathcal{C}$."*

1.4 Our Results

We consider the polynomial family $\widetilde{\mathcal{TQBF}} \triangleq \left\{\widetilde{\mathrm{TQBF}}_n : n \in \mathbb{N}\right\}$ in $\mathsf{VPSPACE}_b$, where for every $n \in \mathbb{N}$ the corresponding polynomial $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is in $O(n^4)$ variables and is of degree $O(n^4)$. The polynomial $\widetilde{\mathrm{TQBF}}_n$ is obtained by a 'careful' arithmetization of the popular PSPACE-complete language TQBF. The first version of the polynomial family $\widetilde{\mathcal{TQBF}}$ was introduced in the PhD thesis of Carmosino [Car19], and was inspired by the work of Trevisan and Vadhan [TV07]. In this work, we refine the arithmetization of TQBF and present a cleaner version of $\widetilde{\mathcal{TQBF}}$. The detailed construction of $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is given in Section 3. We show that $\widetilde{\mathcal{TQBF}}$ satisfies many traits of $\mathsf{VPSPACE}_b$-completeness, and we conjecture that it is $\mathsf{VPSPACE}_b$-complete. For convenience, we also define the family $\widetilde{\mathcal{TQBF}'}$ as a projection of $\widetilde{\mathcal{TQBF}}$. Formally: $\widetilde{\mathrm{TQBF}'}_n(\mathbf{y}) \triangleq \widetilde{\mathrm{TQBF}}_n(\mathbf{0}, \mathbf{y}, \mathbf{z} \setminus \{z_{1,m}\} = \mathbf{0}, z_{1,m} = 1)$ i.e., $\widetilde{\mathrm{TQBF}'}_n$ is obtained by substituting all $\mathbf{x}$ variables and all $\mathbf{z} \setminus \{z_{1,m}\}$ variables equal to 0 and $z_{1,m}$ equal to 1 in $\widetilde{\mathrm{TQBF}}_n$. Here $\widetilde{\mathrm{TQBF}'}_n(\mathbf{y})$ is a polynomial in $8n^3$ variables.

Subsequently, we also introduce the polynomial family $\mathcal{TP} := \{\mathrm{TP}_n : n \in \mathbb{N}\}$ which is a mix of $\widetilde{\mathcal{TQBF}}$ and $\mathcal{PERM}$. Formally, for every n : $\mathrm{TP}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}) := \widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}) + \mathrm{Perm}_n(\mathbf{u})$. It follows that $\mathcal{TP} \in \mathsf{VPSPACE}_b$. See Section 5 for the formal definition and more details on $\mathcal{TP}$.

Remark. In this section, we recall several results from prior work to set up the context, presenting them in the form of lemmas. In particular, Lemmas 1.1, 1.2, and 1.3 are taken from the *previously* existing literature.

1.4.1 Completeness

Our first set of results establishes completeness properties of the constructed polynomial families. We begin by showing that the polynomial families $\widetilde{\mathcal{TQBF}}$ and $\widetilde{\mathcal{TQBF}'}$ capture the computational power of PSPACE as the graph [2] of $\widetilde{\mathcal{TQBF}'}$ (and hence of $\widetilde{\mathcal{TQBF}}$) is PSPACE-complete.

Theorem 1. *The language* $L_{\mathrm{TQBF}'} \triangleq \left\{ y \in \{0,1\}^* \;\middle|\; |y| = 8n^3 \wedge \widetilde{\mathrm{TQBF}'}_n(y) = 1 \right\}$ *is* PSPACE*-complete.*

Remark: We remark that as in the case of the TQBF problem, the reduction itself can be implemented by Boolean formulas [3].

Given that, our next result showcases the completeness property of $\mathcal{TP}$ with respect to containment in any subclass of VNP! This is a partial progress towards understanding whether $\mathcal{TP}$ is $\mathsf{VPSPACE}_b$-complete (see the first open question in Section 6).

Theorem 2. *For any* [4] *circuit class* $\mathcal{C} \subseteq \mathsf{VNP}$*:* $\mathcal{TP} \in \mathcal{C}$ *if and only if* $\mathsf{VPSPACE}_b \subseteq \mathcal{C}$*.*

1.4.2 Lower Bounds

Our next set of results can be viewed as conditional and unconditional lower bounds for $\mathcal{TP}$. In particular, an immediate consequence of Theorem 2 is a "quantifier swap" of a previous result showing that a space-efficient black-box PIT algorithm implies circuit lower bounds.

Lemma 1.1 ([HS80, Agr05], [vzG93]; see also [SY10])**.** *Suppose there is a sub-polynomial* [5] *space black-box* PIT *algorithm for a circuit class* $\mathcal{C}$*. Then* $\mathsf{VPSPACE}_b$ *does not have polynomial-size circuits from* $\mathcal{C}$*.*

[HS80, Agr05] show how to transform a **given**, space-efficient, hitting set for a class $\mathcal{C}$ into a polynomial in $\mathsf{VPSPACE}_b$ that is "hard" for that same $\mathcal{C}$ (i.e. does not have polynomial-size circuits from $\mathcal{C}$). And while this transformation is "explicit" (i.e. constructive) the "hard" polynomial depends on the particular hitting set at hand. Roughly speaking, this can be stated as: "for *any* hitting set $\mathcal{H}$ for the class $\mathcal{C}$, *there exists* a hard polynomial for $\mathcal{C}$ that depends on $\mathcal{H}$". As was mentioned above, our next corollary allows to swap the order of the quantifiers to obtain that: "$\mathcal{TP}$ is a hard polynomial for a class $\mathcal{C}$, as long as there exist *any* hitting set $\mathcal{H}$ for $\mathcal{C}$"!

Corollary 3. *Suppose there is a sub-polynomial-space black-box* PIT *algorithm for a circuit class* $\mathcal{C} \subseteq \mathsf{VNP}$*. Then* $\mathcal{TP}$ *cannot be computed by polynomial-size circuits from* $\mathcal{C}$*.*

Our result makes partial progress on an open problem raised in [SY10] (See **Open Problem 17** in Section 1.3). Indeed, the statement of our result shares the same general flavour: instead of the Permanent, the resulting lower bound is for $\mathcal{TP}$. It is also worth noting that the family $\mathcal{TP}$ in our result could be replaced with any family that is complete for $\mathsf{VPSPACE}_b$. In this sense, the corollary provides supporting evidence for the completeness of $\mathcal{TP}$. At the same time, as mentioned previously, Malod [Mal11] has constructed families that are complete for $\mathsf{VPSPACE}$. While it is possible to transform a $\mathsf{VPSPACE}$-complete family into one that is complete for $\mathsf{VPSPACE}_b$, using techniques from [KP09], this transformation is non-trivial, and the resulting $\mathsf{VPSPACE}_b$ family may not retain a "natural" structure. On the algorithmic side, we note that the vast majority of the existing PIT algorithms operate in sub-polynomial space (see [BGNV26]). Furthermore, many of them require only logarithmic space! We list some important classes of arithmetic circuits admitting sub-polynomial space black-box PIT algorithms.

Circuit Class	**Work**
Multilinear constant-read formulas	[AvMV15, MV18]
Depth-3 formulas with constant transcendental degree	[ASSS16]
Sub-linear-width ROABPs	[GKS17]

Table 1: List of circuit classes having sub-polynomial space PIT algorithms

[2] One can think of the $L_{\mathrm{TQBF}'}$ as the graph.
[3] In fact, the reduction can be implemented in a much smaller complexity class such as logarithmic space.
[4] We make the standard assumption that $\mathcal{C}$ is closed under substitutions of constants $\{0,1\}$, which holds for all known classes.
[5] That is, a PIT algorithm that uses $s^{o(1)}$ amount of work space for circuits of size s from $\mathcal{C}$.

Corollary 4. *$\mathcal{TP}$ cannot be computed by polynomial-size Depth-3 formulas with constant transcendental degree, and sub-linear-width ROABPs - with polynomially-bounded constants.*

We remark that $\mathcal{TP}$ cannot be computed by any multilinear formula since the polynomial $\mathrm{TP}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u})$ is not a multilinear polynomial.

Constant-free and Constant-bounded Classes. In order to better understand the relationships among existing algebraic complexity classes, prior works (e.g., [Bür00, KP09, Mal03, Poi08, Mal11, MR13, CT23]) have considered modified or restricted variants of these classes. In [Mal03], Malod introduced *constant-free* circuits, in which the only allowed constants are $\{-1, 0, 1\}$. One can observe that such circuits can only compute polynomials with integer coefficients. This notion naturally extends to classes. For instance: VP^0 (resp., VNP^0, $\mathsf{VPSPACE}_b^0$) denotes the constant-free variant of VP (resp., VNP, $\mathsf{VPSPACE}_b$) in which the constants appearing in the circuits are restricted to $\{-1, 0, 1\}$. To level the playing field further we also study a finer-grained, *constant-bounded* variant of VP. That is, we denote by $\mathsf{VP}^{\mathrm{poly}}$ the variant of VP where the *bit complexity* of the circuit constants is **polynomially-bounded**. As a result, the circuit constants may be at most **exponential** in the input size.

It is immediate from the definition that $\mathsf{VP}^0 \subseteq \mathsf{VP}^{\mathrm{poly}}$. In fact, $\mathsf{VP}^{\mathrm{poly}}$ is a strict super class of VP^0 since it contains, for example, the family $x_1 + x_2 + \ldots + x_n + \frac{1}{2}$ whereas VP^0 only contains polynomials with integer coefficients. (For more details, see Definition 2.11)

1.4.3 Characterization of $\mathsf{VPSPACE}_b$ and Inconsistent Triads

In [KP09], Koiran & Perifel have shown the following characterization/transfer theorem for $\mathsf{VPSPACE}_b$.

Lemma 1.2 (Proposition 4.1 of [KP09])**.** *If $\mathsf{PSPACE/poly} = \mathsf{P/poly}$ and $\mathsf{VNP} = \mathsf{VP}$ then $\mathsf{VPSPACE}_b = \mathsf{VP}$. The converse holds, assuming the Generalized Riemann Hypothesis (GRH).*

It has since remained an open problem to establish an **unconditional** equivalence that does not rely on the GRH. While we do not resolve this problem entirely, we make partial progress by establishing an unconditional "constant-free" version of the characterization. Our proof can be seen as a corollary of the proof of Theorem 2. We remark that, to the best of our knowledge, it is currently unclear how to extend the original proof of [KP09] to the constant-free setting without assuming the GRH.

Theorem 5. *Suppose $\mathsf{VPSPACE}_b^0 = \mathsf{VP}^0$ (resp., $\subseteq \mathsf{VP}^{\mathrm{poly}}$). Then $\mathsf{PSPACE/poly} = \mathsf{P/poly}$ and $\mathsf{VNP}^0 = \mathsf{VP}^0$ (resp., $\subseteq \mathsf{VP}^{\mathrm{poly}}$).*

We note that the both assumptions $\mathsf{VPSPACE}_b^0 = \mathsf{VP}^0$ and $\mathsf{VPSPACE}_b^0 \subseteq \mathsf{VP}^{\mathrm{poly}}$ are stronger than $\mathsf{VPSPACE}_b = \mathsf{VP}$, as each implies the latter. Subsequently, as another corollary, we obtain an inconsistent triad, similar to the one in [KI04].

Corollary 6. *Let $\mathcal{C}$ be an arbitrary circuit class. The following assumptions cannot hold simultaneously.*

1. $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$.
2. *The Permanent is computable by arithmetic circuits from $\mathcal{C}$.*
3. *There is a sub-polynomial space black-box PIT algorithm for $\mathcal{C}$.*

For reference, the following was shown in [KI04].

Lemma 1.3 (The inconsistent triad from [KI04])**.** *The following cannot hold simultaneously.*

1. $\mathsf{NEXP} \subseteq \mathsf{P/poly}$.
2. *The Permanent is computable by polynomial-size arithmetic circuits (i.e. $\mathcal{PERM} \in \mathsf{VP}^{\mathrm{poly}}$).*
3. *There is a sub-exponential time PIT algorithm.*

We note that the two triads are incomparable: while the first assumption is weakened, the third assumption is strengthened. In addition, our result holds for arbitrary classes of circuits $\mathcal{C}$ even **outside** of VP. On the other hand, to the best of our knowledge, the result of [KI04] only holds for "sufficiently rich" classes like arithmetic circuits/formulas/ABPs[6]. A proof sketch of Corollary 6 is given in Section 5.3.

[6]The reason being that the assumed PIT algorithm should work not only for circuits from $\mathcal{C}$ but also for sums of such circuits.

1.4.4 Self-Testability and Karp-Lipton Style Collapse

Our final set of results emphasizes an important property of $\widetilde{\mathcal{TQBF}}$ - self-testability. Roughly speaking, a polynomial P is *self-testable* if given a circuit C that purports to compute P one can decide if C indeed computes P, in polynomial-time, given oracle access to PIT. Efficient self-testability algorithms are also known for two important polynomial families not known to be in VP: the Permanent [Lip89, Mul10], and the Nisan-Wigderson design polynomial [GS19].

Theorem 7 (Informal)**.** *$\widetilde{\mathcal{TQBF}}$ is self-testable.*

The self-testability of $\widetilde{\mathcal{TQBF}}$ is formally stated in Theorem 4.7. In fact, we prove a somewhat weaker version of self-testability. Nonetheless it allows us to prove a Karp-Lipton style collapse theorem. More specifically, we show that PSPACE collapses to $\mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$, where $\mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$ is the set of languages decided by non-deterministic polynomial-time Turing machine with oracle access to a PIT algorithm for $\mathcal{C}$. Note that the collapse of the *Boolean* complexity class PSPACE is based on an assumption about the *arithmetic* complexity of $\widetilde{\mathcal{TQBF}}$.

Corollary 8. *Let $\mathcal{C} \subseteq \mathsf{VP}^{\mathrm{poly}}$ be a class satisfying some mild technical conditions. Then:*

1. *If $\widetilde{\mathcal{TQBF}} \in \mathcal{C}$ then $\mathsf{PSPACE} \subseteq \mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$.*
2. *If, in addition, $\mathcal{C}$ admits a deterministic or randomized reconstruction algorithm, then $\mathsf{PSPACE} = \mathsf{P}$ or $\mathsf{PSPACE} = \mathsf{BPP}$, respectively.*
3. *In contrast, if a sub-polynomial space reconstruction algorithm for $\mathcal{C}$ exists, then $\mathcal{TP} \notin \mathcal{C}$.*

Remark 1. The first result in the corollary should be contrasted with a result of [BFL91] which states that $\mathsf{PSPACE} \subseteq \mathsf{P/poly} \implies \mathsf{PSPACE} \subseteq \mathsf{MA}$. Indeed, by Lemma 4.5: $\widetilde{\mathcal{TQBF}} \in \mathsf{VP}^{\mathrm{poly}} \implies \mathsf{PSPACE} \subseteq \mathsf{P/poly}$. So when put together we obtain: $\widetilde{\mathcal{TQBF}} \in \mathsf{VP}^{\mathrm{poly}} \implies \mathsf{PSPACE} \subseteq \mathsf{P/poly} \implies \mathsf{PSPACE} \subseteq \mathsf{MA}$. On the other hand, for any $\mathcal{C} \subseteq \mathsf{VP}$ we have that $\mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}} \subseteq \mathsf{NP}^{\mathsf{BPP}} \subseteq \mathsf{MA}$.

Remark 2. We remark that the polynomial families $\mathcal{TP}$ and $\widetilde{\mathcal{TQBF}}$ can be defined over any field $\mathbb{F}$ (finite or otherwise) and all of the above results continue to hold over arbitrary fields, with the appropriate modifications. Specifically, the corresponding complexity classes become $\mathsf{VP}_{\mathbb{F}}$, $\mathsf{VNP}_{\mathbb{F}}$, and $\mathsf{VPSPACE}_{\mathbb{F},b}$, respectively.

1.5 Proof Overview

The main results mentioned above follow from various properties of the polynomial families $\widetilde{\mathcal{TQBF}}$ and $\mathcal{TP}$. These properties are artefacts of the nice recursive structure of the polynomial $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$. This polynomial is obtained by 'arithmetizing' and 'linearizing' the Boolean problem TQ-3-CNF (Definition 2.3), a special case of TQBF, an important PSPACE-complete problem. Since the proofs hinge on the structure of $\widetilde{\mathrm{TQBF}}_n$, we first present a high-level overview of this polynomial.

Construction of $\widetilde{\mathrm{TQBF}}_n$. In Section 3, we give the formal definition of the polynomial $\widetilde{\mathrm{TQBF}}_n$. This polynomial is described in terms of 'slices'. To obtain the first slice, we need a *universal 3-CNF* $\Phi_n(\mathbf{x}, \mathbf{y})$, i.e., for every n-variate 3-CNF $\varphi(\mathbf{x})$, there exists an assignment $\mathbf{b} \in \{0,1\}^{|\mathbf{y}|}$ such that $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x}, \mathbf{b})$. Let $\Psi := Q_1 x_1 \cdots Q_n x_n \Phi_n(\mathbf{x}, \mathbf{y})$ be the universal TQ-3-CNF, where Q_1 is an existential quantifier and other quantifiers are alternating.

The construction of $\widetilde{\mathrm{TQBF}}_n$ starts with 'arithmetizing' $\Phi_n(\mathbf{x}, \mathbf{y})$ to obtain the polynomial $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ over a field. We need one more ingredient, which is the arithmetization of the quantifiers $\forall$ and $\exists$ (see Definition 3.1). The arithmetization of a quantifier Q_i is denoted as $\widehat{Q}_i$. The first slice of $\widetilde{\mathrm{TQBF}}_n$ is $\widehat{Q}_n \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$, i.e., it is obtained by applying $\widehat{Q}_n$ to $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$. It follows from the definition of $\widehat{Q}_n$ that the variable x_n does not appear in $\widehat{Q}_n \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$. Now, the intuition says that the next slice should be obtained by arithmetizing the quantifier Q_{n-1}, i.e., applying $\widehat{Q}_{n-1}$ to the polynomial $\widehat{Q}_n \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$, and so on. Unfortunately, the construction is not that simple, and the main reason for deviating from this nice intuitive idea is that

the degree of $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ doubles with arithmetization of every quantifier, as a result we might end up with a polynomial with exponential degree in $\mathbf{y}$ variables! We resolve this issue by 'linearizing' the polynomial $\widehat{Q}_n\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ with respect to every variable appearing in this polynomial one by one. After that, we apply $\widehat{Q}_{n-1}$ to this polynomial, linearize all the variables appearing in it, and then apply $\widehat{Q}_{n-2}$, and so on. In this manner, we create $O(n^4)$ polynomials, which we call the *slices* of $\widetilde{\text{TQBF}}_n$. Then, we multiply every slice with a new variable, and the summation of all these terms results in $\widetilde{\text{TQBF}}_n$. The structure of $\widetilde{\text{TQBF}}_n$ is given in Equation (5) in Section 3. We denote the family of these polynomials by $\widetilde{\mathcal{TQBF}}$. Using this family, we define another polynomial family called $\mathcal{TP}$, which is a combination of $\widetilde{\text{TQBF}}$ and $\mathcal{PERM}$, the family of the permanent polynomial. Our construction implies $\mathcal{TP} \in \mathsf{VPSPACE}_b$.

Now, we present the proof ideas of our results. We classify them thematically as was done in Section 1.4.

1.5.1 Completeness - Proof idea of Theorem 1 and Theorem 2

Proof idea of Theorem 1. In this theorem, we show that the language $L_{\text{TQBF}'}$ defined in Theorem 1 is PSPACE-complete. To show $L_{\text{TQBF}'} \in \mathsf{PSPACE}$, we prove that the polynomial family $\widetilde{\mathcal{TQBF}}$ is in (uniform) $\mathsf{VPSPACE}_b$ in Lemma 4.1. In particular, we show that the polynomial $\widetilde{\text{TQBF}}_n$ can be computed by a PSPACE-uniform arithmetic circuit of depth $\text{poly}(n)$ (the size can be exponential in n). The nice recursive structure of TQBF_n given in Section 3 plays an important role in proving this.

After this, the PSPACE-hardness of $L_{\text{TQBF}'}$ is shown by proving that TQ-3-CNF $\leq_p L_{\text{TQBF}'}$, where $\leq_p$ denotes polynomial-time many-to-one reducibility. TQ-3-CNF (Definition 2.3) is a well-known PSPACE-complete language. Let Ψ be an arbitrary quantified 3-CNF in n variables. We prove that given Ψ, we can compute an assignment $\mathbf{b}_\Psi \in \{0,1\}^{8n^3}$ in polynomial time such that $\Psi \in$ TQ-3-CNF if and only if $\mathbf{b}_\Psi \in L_{\text{TQBF}'}$. At the core of this sits the argument where we show how different slices of $\text{TQBF}_n(\mathbf{x},\mathbf{b}_\Psi)$ agree with different layers of quantifiers in Ψ over the Boolean hypercube. This is proven in Lemma 4.3. The complete proof of this theorem is given in Section 4.2.

Proof of Theorem 2. As the main part of this result, we prove that $\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$ if and only if $\mathsf{VPSPACE}_b = \mathsf{VNP}$ (see Theorem 5.8). As $\mathsf{VNP} \subseteq \mathsf{VPSPACE}_b$ and as $\widetilde{\mathcal{TQBF}} \in \mathsf{VPSPACE}_b$, one direction is trivial. For the other direction, suppose $\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$. Let $\mathbf{f} = \{f_n\}$ be an arbitrary family in $\mathsf{VPSPACE}_b$. Then, by Definition 2.17, we can express $f_n(\mathbf{x})$ as

$$f_n(\mathbf{x}) = \sum_{\mathbf{e}\in\{0,\ldots,p(n)\}^{p(n)}} \left(c_{\mathbf{f}}(n,\mathbf{e},0) \sum_{i=1}^{p(n)} 2^{i-1} \cdot c_{\mathbf{f}}(n,\mathbf{e},i) \cdot \mathbf{x}^{\mathbf{e}} \right),$$

where $c_{\mathbf{f}}(n,\mathbf{e},0)$ and $c_{\mathbf{f}}(n,\mathbf{e},i) \in \{0,1\}$ can be computed in PSPACE. Using the PSPACE-completeness of $L_{\text{TQBF}'}$, we show that there exists $m = \text{poly}(n)$ and a tuple of $|\mathbf{y}|$ many polynomial-size Boolean formulas φ_f such that $c_{\mathbf{f}}(n,\mathbf{e},i) = 1$ if and only if $\text{TQBF}'_m(\varphi_f(n,\mathbf{e},i)) = 1$ for every $i \in \{0,\ldots,p(n)\}$. As $\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$ we get that the coefficient of every monomial $\mathbf{x}^{\mathbf{e}}$ of $f_n(\mathbf{x})$ in the above equation is a projection of a VNP polynomial. It follows from here that $\mathbf{f} \in \mathsf{VNP}$. This result is then extended to $\mathcal{TP}$ and holds for any sub-class of VNP. The complete proof of this theorem is given in Section 5.2.

1.5.2 Lower Bounds - Proof Idea of Corollary 3

By Lemma 1.1: $\mathsf{VPSPACE}_b$ does not have polynomial-size circuits from $\mathcal{C}$. By the contrapositive of Theorem 2, neither does $\mathcal{TP}$.

1.5.3 Characterization of $\mathsf{VPSPACE}_b$ and Inconsistent Triads - Proof Idea of Theorem 5

In Theorem 5, we show that $\mathsf{PSPACE/poly} = \mathsf{P/poly}$ and $\mathsf{VNP}^0 = \mathcal{C}$ if and only if $\mathsf{VPSPACE}_b^0 = \mathcal{C}$ for every $\mathcal{C} \subseteq \mathsf{VP}^{\text{poly}}$. This strengthens a result in [KP09] (see Proposition 5.3) for $\mathsf{VP}^{\text{poly}}$ as we no longer need the GRH, which was crucial for their result. For Theorem 2 we "skip" the step that shows that $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ and, requires the GRH, and use our result that $\widetilde{\text{TQBF}} \in \mathsf{VNP}$ iff $\mathsf{VPSPACE}_b = \mathsf{VNP}$ (Theorem 5.8).

1.5.4 Self-Testability and Karp-Lipton Style Collapse - Proof Ideas of Theorem 7 and Corollary 8

Proof idea of Theorem 7. A formal version of this theorem is given in Theorem 4.7. In this theorem, we give a deterministic polynomial-time algorithm with oracle access to PIT for $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$, such that this algorithm takes input a circuit $\mathsf{C} \in \mathcal{C}$ and decides if C computes $\widetilde{\mathrm{TQBF}}_n$. The algorithm tests whether all 'appropriate projections' of C compute all slices of $\widetilde{\mathrm{TQBF}}_n$. In this result, the algorithm starts by testing the first slice, then uses the structure of arithmetization of Q_n to test the computation of the next slice, and so on. The most important thing here is that if the algorithm has succeeded in testing that appropriate projections of C compute all the slices of $\widetilde{\mathrm{TQBF}}_n$ up to a particular index, then that means that all these slices are in $\mathcal{C}$ and the next slice is in $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$. Now, the algorithm uses the PIT for $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$ to test whether a suitable projection of C computes this slice correctly. In this manner, we attain the self-testability of $\widetilde{\mathrm{TQBF}}_n$. We want to emphasize that one of the most important properties of $\widetilde{\mathrm{TQBF}}_n$ for this result is that the degrees of all the slices of $\widetilde{\mathrm{TQBF}}_n$ are $\mathrm{poly}(n)$. The formal proof of Theorem 7 is given in Section 4.4.

Proof idea of Corollary 8. This corollary says that if $\mathsf{VPSPACE}_b$ has polynomial-size arithmetic circuits from a class $\mathcal{C}$ satisfying some properties then $\mathsf{PSPACE} = \mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$. The idea is to first guess a candidate circuit $\mathsf{C} \in \mathcal{C}$ for $\widetilde{\mathcal{TQBF}}_n$, for the appropriate value of n, and then use the self-testability result above (i.e. Theorem 7) to verify that C indeed computes $\widetilde{\mathcal{TQBF}}_n$. Given that, we can compute a circuit C' for $\widetilde{\mathcal{TQBF}'}_n$ as a projection of $\widetilde{\mathcal{TQBF}}_n$. The result then follows from Theorem 1.

Given an efficient reconstruction algorithm, instead "guessing" the circuit, we learn it slice-by-slice using the a DP-like update rule for every slice. This, essentially, follows from "downward self-reducibility" structure behind the self-testability. Corollary 8 is formally proved in Section 4.5.

1.6 Comparison With Previous Results

1. **Comparison with [Poi08].** In [Poi08], Poizat gave an alternative, purely algebraic, definition for $\mathsf{VPSPACE}$ via arithmetic circuits endowed with an additional type of gate called "projection" gate. Given that equivalent definition, any family of universal circuits with projection gates is, by definition, complete for $\mathsf{VPSPACE}$. Yet, to the best of our knowledge, no existing construction of such a family has the desired "nice" properties like self-testability, efficient 'simulation' of PSPACE, Karp-Lipton style collapse etc. Moreover, while one can easily convert a $\mathsf{VPSPACE}$-complete family into a $\mathsf{VPSPACE}_b$-complete family using the inverse Kronecker's transformation, this transformation may further compromise any existing structure.

2. **Comparison with [Mal11].** In [Mal11], Malod introduced a few families of $\mathsf{VPSPACE}$-complete polynomials. These families were obtained by giving yet another alternative characterization of $\mathsf{VPSPACE}$ via 'succinct' algebraic branching programs and arithmetic circuits. To the best our knowledge, none of these families appear to satisfy any of the "nice" properties outlined above.

3. **Comparison with [Car19].** The first version of the polynomial $\widetilde{\mathrm{TQBF}}$ appeared in the PhD thesis of Marco Carmosino [Car19][7], which was inspired by [TV07]. We believe that our presentation of $\widetilde{\mathrm{TQBF}}$ is cleaner and simpler. The variant given by Carmosino satisfied *downwards self-reducibility*. We feel that this makes the construction of the polynomial family relatively more involved. On the other hand, our polynomial family satisfies a weaker property *self-testability*, which is sufficient to get the results mentioned in this paper. Also, we introduced the polynomial family $\mathcal{TP}$, which helped us obtain the majority of our results.

4. It follows from the results of [LFKN90, Sha92, BFL91] that if $\mathsf{P}^{\#\mathsf{P}}$ or PSPACE has polynomial-size Boolean circuits, then the respective class collapses to MA. In [KI04] this collapse was slightly improved by showing that if the Permament, a complete problem for $\#\mathsf{P}$, has polynomial-size *arithmetic* circuits then $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{PIT}}$. Corollary 8 can be seen as an algebraic counterpart of collapse of PSPACE.

[7] Previously in a conference version [CIKK15].

Additionally, our results apply to any arithmetic circuit class satisfying 'mild' technical conditions, whether the collapse of [KI04] only applies to 'sufficiently' rich classes (like general circuits or formulas) since it requires 'strong' closure properties.

Organization

The rest of the document is organized as follows. We compare our results with other related results in Section 1.6. After that, in Section 2, we present some preliminary results. Then, we present the construction of the polynomial families $\widetilde{\mathcal{TQBF}}$ in Section 3. In Section 4, we prove important properties of $\widetilde{\mathcal{TQBF}}$, which comprises many of our results. Section 5 is devoted to the polynomial family $\mathcal{TP}$ and its properties, which cover our remaining results. At last, we present some important open questions in Section 6.

2 Preliminaries

Notations. $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers. For a non-zero natural number n, $[n]$ refers to the set $\{1, \ldots, n\}$ and for $m \in \mathbb{N}, m < n$, $[m, n] := \{m, m+1, \ldots, n\}$. Sets of variables are represented by $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}$ and tuples over the underlying field are denoted as $\mathbf{a}, \mathbf{b}, \mathbf{c}$ etc. The set of n-variate polynomials over a field $\mathbb{F}$ is represented by $\mathbb{F}[x_1, x_2, \ldots, x_n]$. For $f \in \mathbb{F}[x_1, x_2, \ldots, x_n], i \in [n]$, and $a \in \mathbb{F}$, $f|_{x_i=a}$ denotes the polynomial obtained by substituting $x_i = a$ in f. Similarly, if $\varphi(\mathbf{x})$ is an n-variate Boolean formula, $x_i \in \mathbf{x}$ and $a \in \{0, 1\}$, then $\varphi|_{x_i=a}$ is obtained by substituting $x_i = a$ in φ.

2.1 Boolean Complexity

2.1.1 Uniform Complexity

Definition 2.1 (The class DSPACE)**.** *Let $s : \mathbb{N} \to \mathbb{N}$. A language L is said to be in* DSPACE$[s(n)]$ *if there exists a deterministic Turing machine M such that for every $w \in \{0,1\}^*$, $w \in L$ if and only if $M(w) = 1$. In addition, M decides w using $O(s(|w|))$ space.*

Definition 2.2 (PSPACE)**.** *The class* PSPACE *is defined as* PSPACE $= \bigcup_{k \in \mathbb{N}}$ DSPACE(n^k).

We now recall the definition of the 'True Quantified Boolean Formula' TQBF and its version TQ-3-CNF.

Definition 2.3 (TQBF , TQ-3-CNF)**.** *A quantified Boolean formula (QBF) looks like $Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \varphi(x_1, \ldots, x_n)$, where $Q_i \in \{\forall, \exists\}$ for every $i \in [n]$ and $\varphi(x_1, \ldots, x_n)$ is a quantifier-free Boolean formula. Then,* TQBF *is the set of all QBFs which are true.* [8] TQ-3-CNF *is a special case of* TQBF *when φ is a 3-*CNF*.*

Both TQBF and TQ-3-CNF are well-known PSPACE-complete problems under many-to-one, polynomial-time reductions. Furthermore, these reduction can be implemented using Boolean formulas[9]. A proof of this can be found in [AB09].

2.1.2 Non-uniform Complexity

Definition 2.4 (P/poly)**.** *A language L is in* P/poly*, if there exists a polynomial function $p : \mathbb{N} \to \mathbb{N}$, a deterministic Turing machine $M(w, y)$ and a sequence of strings $\{\alpha_n\}_{n \in \mathbb{N}}$ called advice, where $\alpha_n \in \{0,1\}^{p(n)}$ such that for every $w \in \{0,1\}^*$: $w \in L$ if and only if $M(w, \alpha_{|w|}) = 1$, and M runs for at most $O(p(|w|))$ steps, given $\alpha_{|w|}$.*

An equivalent definition of P/poly is the set of languages that are decided by polynomial-sized families of Boolean circuits. See [AB09] for more details.

Definition 2.5. (DSPACE$[p(n)]/q(n)$)*. Let $p, q : \mathbb{N} \to \mathbb{N}$ be functions. A language L is in* DSPACE$[p(n)]/q(n)$*, if there exist a deterministic Turing machine $M(w, y)$ and a sequence of strings $\{\alpha_n\}_{n \in \mathbb{N}}$ called* advice*, where $\alpha_n \in \{0,1\}^{q(n)}$ such that for every $w \in \{0,1\}^*$: $w \in L$ if and only if $M(w, \alpha_{|w|}) = 1$ and M uses at most $O(p(|w|))$ amount of work space, given $\alpha_{|w|}$.*

[8] Note that every QBF is either True or False as it does not have any free variable.

[9] In fact, both reductions can be implemented in a much smaller complexity class such as logarithmic space.

The following definition gives a non-uniform variant of the class PSPACE.

Definition 2.6 (PSPACE/poly)**.** *The class* PSPACE/poly *is defined as follows*

$$\mathsf{PSPACE/poly} = \bigcup_{\text{polynomial functions } p,q} \mathsf{DSPACE}[p(n)]/q(n).$$

The following uniform to non-uniform transfer lemma is considered a folklore result. For the sake of completeness, we include its proof in Section A.1 of the Appendix.

Lemma 2.7. $\mathsf{PSPACE/poly} \subseteq \mathsf{P/poly}$ *if and only if* $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$.

2.2 Algebraic Complexity

Definition 2.8 (Arithmetic circuit and formula)**.** *An* arithmetic circuit C *over a field* $\mathbb{F}$ *is a directed acyclic graph, where every leaf node is labelled by either a variable or an* $\mathbb{F}$*-constant, and every other node is labelled by either* $+$ *or* $\times$ *operation. The computation in* C *happens in the bottom-up manner as follows: A leaf node computes its label and a node* v *in* C *labelled by* $\otimes \in \{+, \times\}$ *and having children* $u_1, \ldots, u_m$ *computes the polynomial* $P_v = P_{u_1} \otimes \cdots \otimes P_{u_m}$*, where* $P_v, P_{u_1}, \ldots, P_{u_m}$ *are polynomials computed by* $v, u_1, \ldots, u_m$*, respectively. The output of the node having out-degree equal to zero (or the root node) is called the polynomial computed by* C*. The size and depth of* C *are defined to be the number of edges in* C *and the length of the longest leaf-to-root path in* C*, respectively.*

If the underlying graph of C *is a tree, then it is said to be an* arithmetic formula*.*

Definition 2.9 (Syntactic Degree)**.** *Let* C *be an arithmetic circuit over a field* $\mathbb{F}$*. Then, the* syntactic degree *of any gate* v*, denoted* $\deg(v)$ *is defined inductively as follows: if* v *is labelled by an* $\mathbb{F}$*-constant then* $\deg(v) := 0$ *and if it is a variable then* $\deg(v) := 1$*. Otherwise,* v *is an internal node, and suppose it has children* $u_1, \ldots, u_m$*. If* v *is labelled by* $+$ *then* $\deg(v) := \max\{\deg(u_1), \ldots, \deg(u_m)\}$*, and if it is labelled by* $\times$ *then* $\deg(v) := \sum_{i=1}^{m} \deg(u_i)$*. The syntactic degree of* C *is defined as the syntactic degree of the root node of* C*.*

Remark: The "actual" i.e. *semantic degree* of a gate, on the other hand, is defined to be the degree of the polynomial computed by that gate. Indeed, one can view the problem of polynomial identity testing as trying to bridge between the syntactic and the semantic degrees of a given circuit. It is easy to prove the following using the inductive definition of the syntactic degree of a gate of an arithmetic circuit.

Observation 2.10. *There exists an algorithm that given an arithmetic circuit* C *as an input, outputs its* ***syntactic*** *degree, in time polynomial in the size of* C*.*

Proof Idea. Regard the circuit as straight-line program and use a table-based dynamic programming solution. The base cases are the leaves. Update the degree of each gate v using the ruled outline in the definition. □

2.3 Valiant's Classes and Related Problems

We recall the definitions of some important classes of polynomials over a field. The first two classes VP and VNP are algebraic analogues of P and NP, respectively, and were defined in a seminal work by Valiant [Val79].

Definition 2.11 ($\mathsf{VP}, \mathsf{VP}^0$ and $\mathsf{VP}^{\text{poly}}$)**.** *A family of polynomials* $\{f_n\}$ *over a field* $\mathbb{F}$ *is said to be in* VP *if there exists a polynomial function* $p : \mathbb{N} \to \mathbb{N}$ *such that for every* $n \in \mathbb{N}$*,* f_n *has degree and number of variables at most* $p(n)$*, and is computed by an arithmetic circuit of size* $p(n)$*.* VP^0 *is a variant of* VP *in which the allowed constants appearing in the circuits are only* $\{-1, 0, 1\}$*.* $\mathsf{VP}^{\text{poly}}$ *is (another) variant of* VP *in which, for every* $n \in \mathbb{N}$ *the bit complexity of the constants appearing in the* ***circuit that computes*** f_n *is at most* $p(n)$*.*

In other words, $\mathsf{VP}^{\text{poly}}$ is a version of VP where the *bit-size* of the **circuit constants** is polynomially-bounded, hence the constants themselves may be at most exponential in the input size. The class VP^0 was defined by Malod in [Mal03] (also, see [Bür24]) and is also referred to as "constant-free" arithmetic circuits.

Although it may initially appear that any constant in $\mathsf{VP}^{\mathrm{poly}}$ could be simulated by a VP^0 circuit, this is not the case: VP^0 does not permit rational numbers (over $\mathbb{C}$ or $\mathbb{R}$) nor constants from extension fields (e.g., $\mathbb{F}_{p^k}$ for $k \geq 2$).

One of the main motivations for studying $\mathsf{VP}^{\mathrm{poly}}$ and VP^0 instead of VP is that allowing arbitrary constants can make the model unreasonably powerful, a concern highlighted in several works, including [Val82, Bür00]. In addition, VP^0 and $\mathsf{VP}^{\mathrm{poly}}$ enable a more fine-grained comparison of algebraic complexity classes.

Observation 2.12. *Let $\mathcal{C}$ be a sub-class of $\mathsf{VP}^{\mathrm{poly}}$. Then, every $\mathsf{C} \in \mathcal{C}$ of size s can be encoded using $\mathrm{poly}(s)$-many bits and evaluated on a given assignment in time $\mathrm{poly}(s)$.*

Some important families of polynomials in VP [10] are the families of *determinant*, *elementary symmetric polynomials*, *power symmetric polynomials* and *iterated matrix multiplication*. Let VP_{nb} be defined as the set of all polynomial families $\{f_n\}$ such that there exists a polynomial function $p : \mathbb{N} \to \mathbb{N}$ where for every $n \in \mathbb{N}$, f_n has at most $p(n)$ variables and f_n is computed by an arithmetic circuit of size $p(n)$. Note that the degree of f_n can be at most $2^{p(n)}$. Thus, $\mathsf{VP} \subsetneq \mathsf{VP}_{nb}$. The class VP_{nb} was defined by Malod (see [Mal03, Mal07]).

Definition 2.13 (VNP)**.** *A family of polynomials $\{g_n\}$ over a field $\mathbb{F}$ is said to be in VNP if there exist polynomial functions $p, q : \mathbb{N} \to \mathbb{N}$ and a polynomial family $\{f_n\}$ in VP such that for every n,*

$$g_n(x_1, \ldots, x_{p(n)}) = \sum_{b_1,\ldots,b_{q(n)} \in \{0,1\}} f_{q(n)}(x_1, \ldots, x_{p(n)}, b_1, \ldots, b_{q(n)}).$$

VP and VNP are algebraic analogues of P and NP, respectively. Clearly, $\mathsf{VP} \subseteq \mathsf{VNP}$ and one of the main objectives of algebraic complexity theory is to determine the exact relationship between the two classes. Like NP-complete problems, we also have the notion of VNP-complete polynomial families which 'capture' the complexity of VNP. To define a VNP-complete polynomial family, we need the following.

Definition 2.14 (p-projections)**.** *Let $\mathbf{f} := \{f_n\}$ and $\mathbf{g} := \{g_n\}$ be two families of polynomials over a field $\mathbb{F}$, where for every n, f_n and g_n are n-variate polynomials. We say that the polynomial family $\mathbf{g}$ is a p-projection of $\mathbf{f}$ if there exists a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for every n : $g_n(x_1, \ldots, x_n) = f_{p(n)}(a_1, \ldots, a_{p(n)})$, where $a_1, \ldots, a_{p(n)} \in \{x_1, \ldots, x_n\} \cup \mathbb{F}$. In other words, g_n is obtained by substituting the $p(n)$ variables of $f_{p(n)}$ with $a_1, \ldots, a_{p(n)}$.*

Definition 2.15 (VNP-completeness)**.** *A polynomial family $\mathbf{f} = \{f_n\}$ is said to be VNP-complete if $\mathbf{f} \in \mathsf{VNP}$ and for every polynomial family $\mathbf{g} \in \mathsf{VNP}$, $\mathbf{g}$ is a p-projection of $\mathbf{f}$.*

VNP-complete polynomial families play a central role in the VP versus VNP question. In particular, if some VNP-complete polynomial family is in VP then we get $\mathsf{VP} = \mathsf{VNP}$. In [Val79], Valiant showed that the family of *Permanent polynomials* defined below is VNP-complete over any field not having characteristic equal to 2 . Let $n \in \mathbb{N}$ be non-zero and $X = (x_{i,j})_{i,j \in [n]}$ be the matrix of formal variables. Let $S(n)$ be the set of all permutations on $[n]$. Then, the *permanent* polynomial is defined as follows:

$$\mathrm{Perm}_n(X) := \sum_{\sigma \in S(n)} x_{1,\sigma(1)} \cdots x_{n,\sigma(n)}. \tag{1}$$

Let $\mathcal{PERM} := \{\mathrm{Perm}_n\}$. It is conjectured that $\mathcal{PERM} \notin \mathsf{VP}$, popularly known as Valiant's hypothesis. We will need $\mathcal{PERM}$ for defining our main polynomial in Section 5. In addition, some graph-based VNP-complete polynomial families are also known (see [Bür00]). Apart from this, circuit-independent VP-complete polynomial families have also been discovered. The first set of such families was given in [DMM+16]. For more details on algebraic complexity classes, we direct interested readers to [Bür00, SY10, Sap15].

Now, we present the algebraic variant of PSPACE. We start with the following definition taken from [KP09].

[10] In fact, these polynomials are in $\mathsf{VP}^{\mathrm{poly}}$ and even in VP^0.

Definition 2.16 (Coefficient function)**.** *Let* $\mathbf{f} := \{f_n\}$ *be a family of multivariate polynomials over* $\mathbb{Z}$*. Then, the coefficient function of* $\mathbf{f}$*, denoted* $c_{\mathbf{f}}$*, maps the tuple* $(n, \mathbf{e}, i)$ *to the* i*-th bit of the coefficient of the monomial* $\mathbf{x}^{\mathbf{e}}$ *in* f_n [11]*. Further,* $c_{\mathbf{f}}(n, \mathbf{e}, 0)$ *denotes the sign of the coefficient of* $\mathbf{x}^{\mathbf{e}}$ *in* f_n*.*

Let $\mathbf{f} = \{f_n\}$ be a family of polynomials over $\mathbb{Z}$ and $c_{\mathbf{f}}$ be the coefficient function of $\mathbf{f}$. Then observe that we can express the polynomial $f_n(\mathbf{x})$ as follows for every $n \in \mathbb{N}$:

$$f_n(\mathbf{x}) = \sum_{\mathbf{e}} \left(c_{\mathbf{f}}(n, \mathbf{e}, 0) \sum_{i \in \mathbb{N}, i \geq 1} 2^{i-1} \cdot c_{\mathbf{f}}(n, \mathbf{e}, i) \cdot \mathbf{x}^{\mathbf{e}} \right).$$

We can identify a monomial $\mathbf{x}^{\mathbf{e}}$ with its coefficient vector $\mathbf{e}$ in binary and hence the coefficient function $c_{\mathbf{f}}$ for a family of polynomials $\mathbf{f} = \{f_n\}$ can be viewed as a function from $\{0,1\}^*$ to $\{0,1\}$.

Definition 2.17 (Uniform $\mathsf{VPSPACE}^0$, $\mathsf{VPSPACE}^0$, and $\mathsf{VPSPACE}^0_b$ [KP09])**.** *Let* $\mathbf{f} := \{f_n\}$ *be a family of multivariate polynomials over* $\mathbb{Z}$ *and* $c_{\mathbf{f}}$ *be its coefficient function. Then,* $\mathbf{f} \in$ (Uniform) $\mathsf{VPSPACE}^0$ *if there exists a polynomial function* $p : \mathbb{N} \to \mathbb{N}$ *such that the following properties are satisfied for every* $n \in \mathbb{N}$*.*

- *The number of variables in* f_n *is at most* $p(n)$*.*
- *The degree of* f_n *is at most* $2^{p(n)}$*.*
- *The bit size of every coefficient of* f_n *is at most* $2^{p(n)}$*.*
- *The coefficient function* $c_{\mathbf{f}}$ *can be computed in* (PSPACE) PSPACE/poly*.*

Moreover, a polynomial family $\{f_n\} \in \mathsf{VPSPACE}^0$ *is in the sub-class* $\mathsf{VPSPACE}^0_b$ *if there exists a polynomial function* $p : \mathbb{N} \to \mathbb{N}$ *such that for every* $n \in \mathbb{N}$*, the degree and coefficient size of* f_n *is at most* $p(n)$*.* [12]

As $\mathsf{PSPACE} \subseteq \mathsf{PSPACE/poly}$, clearly $\mathsf{Uniform\,VPSPACE}^0 \subseteq \mathsf{VPSPACE}^0$.

Definition 2.18 (Uniform VPAR^0)**.** *A polynomial family* $\mathbf{f} = \{f_n\}$ *over* $\mathbb{Z}$ *is in* $\mathsf{Uniform\,VPAR}^0$ *if and only if there exists a family of* PSPACE*-uniform constant-free arithmetic circuits* $\{\mathsf{C}_n\}$*, such that for every* n*, the circuit* C_n *computes* f_n*, has depth* $\mathrm{poly}(n)$ *and may have size exponential in* n*.* [13]

The following proposition from [KP09] relate the two classes $\mathsf{Uniform\,VPSPACE}^0$ and $\mathsf{Uniform\,VPAR}^0$.

Proposition 2.19 (Proposition 3.3 of [KP09])**.** $\mathsf{Uniform\,VPSPACE}^0 = \mathsf{Uniform\,VPAR}^0$.

Definition 2.20 (VPSPACE [KP09])**.** *Let* $\mathbb{F}$ *be a field containing the set of integers. Then,* VPSPACE *(or* $\mathsf{VPSPACE}_b$*) is the class of all polynomial families* $\{g_n\}$ *over* $\mathbb{F}$ *such that for every* $n \in \mathbb{N}, g_n$ *is a* $\mathrm{poly}(n)$*-variate polynomial and there exists a family* $\{f_n(\mathbf{x}, \mathbf{y})\} \in \mathsf{VPSPACE}^0$ *(respectively, a polynomial familty* $\{f_n(\mathbf{x}, \mathbf{y})\} \in \mathsf{VPSPACE}^0_b$*), a family of tuples* $\{\mathbf{a}_n\}$ *over* $\mathbb{F}$*, and a polynomial function* $p : \mathbb{N} \to \mathbb{N}$ *such that for every* $n \in \mathbb{N}, g_n(\mathbf{x}) = f_{p(n)}(\mathbf{x}, \mathbf{a}_n)$*.*

We direct interested readers to [Poi08, KP09, Mal11, MR13, CT23] for more details on VPSPACE. [Poi08, Mal11, MR13] also gave other equivalent definitions of VPSPACE.

2.3.1 Polynomial Identity Testing

Polynomial identity testing (PIT) is the problem of checking whether the given polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is identically zero or not. An important variant of PIT is the black-box PIT, where we only have oracle access to f, which returns the value $f(\mathbf{a})$ in a unit time for any $\mathbf{a} \in \mathbb{F}^n$. A randomized polynomial-time black-box PIT algorithm is known due to the DeMillo-Lipton-Schwartz-Zippel lemma [DL78, Zip79, Sch80]. Given an n-variate, degree d polynomial f, this algorithm picks an assignment $\mathbf{a} \in S^n$ uniformly at random, where $S \subseteq \mathbb{F}, |S| > d$. The running time of this algorithm is $\mathrm{poly}(n, d)$ and the success probability is at least $1 - \frac{d}{|S|}$. An important notion for a black-box PIT algorithm is that of a *hitting set*.

[11] Suppose $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{x} = \{x_1, \ldots, x_n\}$. Then, $\mathbf{x}^{\mathbf{e}} := x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$.

[12] The subscript b of $\mathsf{Uniform\,VPSPACE}^0_b$ means bounded.

[13] A circuit family $\{\mathsf{C}_n\}$ is said to be PSPACE-uniform if there exists a Turing machine M which on input 1^n outputs the description of C_n using $\mathrm{poly}(n)$ space. Further, $\{\mathsf{C}_n\}$ is constant-free if for every n, the only constant that appears in C_n is 1.

Definition 2.21 (Hitting set)**.** *Let $\mathcal{C}$ be a class of n-variate arithmetic circuits over a field $\mathbb{F}$. Then $\mathcal{H} \subseteq \mathbb{F}^n$ is a* hitting set *for $\mathcal{C}$ if for every non-zero polynomial $f \in \mathcal{C}$, there exists a point $\mathbf{a} \in \mathcal{H}$ such that $f(\mathbf{a}) \neq 0$.*

Let $\mathcal{C}$ be a class of n-variate, degree-d arithmetic circuits. It immediately follows from the above definition that if we can construct a hitting set $\mathcal{H}$ of size $\text{poly}(n, d)$ for $\mathcal{C}$ in $\text{poly}(n, d)$ time then we get a polynomial-time black-box PIT for $\mathcal{C}$. Black-box PIT has been extensively studied in the time-complexity domain for several important classes of arithmetic circuits. In this paper, we consider PIT in the space-complexity framework, which is discussed in details in [BGNV26].

Definition 2.22 (Space-explicit hitting set)**.** *Let $\mathcal{C}$ be a class of arithmetic circuits that compute n-variate, degree-d polynomials over a field $\mathbb{F}$. Let $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a function. A hitting set $\mathcal{H}$ for $\mathcal{C}$ is said to be $t(n, d)$-space-explicit if there exists a deterministic Turing machine that computes every $\mathbf{a} \in \mathcal{H}$ using $O(t(n, d))$ amount of work space.*

3 The Polynomial $\widetilde{\text{TQBF}}_n$

We first define three operators on the ring of polynomials. These are required to describe the main polynomial, which was first introduced in the Ph.D. thesis of Carmosino [Car19] and was inspired by the work of Trevisan and Vadhan [TV07]. Then we note some important properties of this polynomial. For clarity of presentation, we have shifted all the proofs from here to Section B of the appendix.

3.1 Some Useful Operators on $\mathbb{F}[\mathbf{x}]$

We now define three operators on the ring of multivariate polynomials over a field. The first two operators 'arithmetize' the universal and existential quantifiers and the third one is used to 'linearize' a polynomial. These operators will be crucial to define the main polynomial in the next section. They also played an important role in the proof of IP = PSPACE [Sha92]. See also [LFKN90] and [BF91].

Definition 3.1. *Let $\mathbb{F}$ be a field and $\mathbf{x}$ be a set of variables.*

1. *The universal operator on $\mathbb{F}[\mathbf{x}]$ with respect to a variable $z \in \mathbf{x}$, denoted $\forall_z$, is defined as*
$$\forall_z : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[\mathbf{x}];\ f \mapsto f|_{z=1} \times f|_{z=0}.$$
2. *The existential operator on $\mathbb{F}[\mathbf{x}]$ with respect to a variable $z \in \mathbf{x}$, denoted $\exists_z$, is defined as*
$$\exists_z : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[\mathbf{x}];\ f \mapsto 1 - (1 - f|_{z=1}) \times (1 - f|_{z=0}).$$
3. *The linearization operator on $\mathbb{F}[\mathbf{x}]$ with respect to $z \in \mathbf{x}$, denoted Λ_z, is defined as*
$$\Lambda_z : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[\mathbf{x}]; f \mapsto z \times f|_{z=1} + (1 - z) \times f|_{z=0}.$$
Moreover, if $X := \{x_1, \ldots, x_k\} \subseteq \mathbf{x}$, then we define $\Lambda_X f := \Lambda_{x_1} \Lambda_{x_2} \cdots \Lambda_{x_k} f$.

We now note some important properties of these operators. Their proofs are in Section B.

Observation 3.2. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate, degree d-polynomial and $z \in \mathbf{x}$ be arbitrary.*

1. *$\forall_z f, \exists_z f$ are polynomials in $\mathbb{F}[\mathbf{x} \setminus \{z\}]$ and $\deg(\forall_z f) \leq 2d$, $\deg(\exists_z f) \leq 2d$.*
2. *The polynomial $\Lambda_z f$ is linear in z, i.e., $\deg_z(\Lambda_z f) \leq 1$.*
3. *Let $X := \{x_1, \ldots, x_k\} \subseteq \mathbf{x}$ be arbitrary. Then, the operator Λ_X is $\mathbb{F}$-linear over $\mathbb{F}[\mathbf{x}]$.*

Proposition 3.3. *Let $f \in \mathbb{F}[\mathbf{x}]$ and $X := \{x_1, \ldots, x_k\} \subseteq \mathbf{x}$ be an arbitrary set. Then for every permutations σ, τ on $[k]$, we have $\Lambda_{x_{\sigma(1)}} \Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}} f = \Lambda_{x_{\tau(1)}} \Lambda_{x_{\tau(2)}} \cdots \Lambda_{x_{\tau(k)}} f$. In other words, $\Lambda_X f = \Lambda_{x_{\sigma(1)}} \Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}} f$ for every permutation σ on $[k]$.*

Now, we show that every polynomial f agrees with $\Lambda_X f$ on $\{0,1\}^n$.

Proposition 3.4. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial and $z \in \mathbf{x}$ be an arbitrary variable. Then for every $\mathbf{a} \in \{0,1\}^n$, $f|_{\mathbf{x}=\mathbf{a}} = \Lambda_z f|_{\mathbf{x}=\mathbf{a}}$.* [14]

Proposition 3.5. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial and $X \subseteq \mathbf{x}$. Then for $\mathbf{a} \in \{0,1\}^n$, $f|_{\mathbf{x}=\mathbf{a}} = \Lambda_X f|_{\mathbf{x}=\mathbf{a}}$.*

3.2 Defining $\widetilde{\text{TQBF}}$

We first define the "terms" of the main polynomial, which we call $\widetilde{\text{TQBF}}$. The definition of these terms follow a nice recursive pattern and these are obtained by 'arithmetizing' the problem TQ-3-CNF (Definition 2.3). For simplicity, we abuse the notation and call the corresponding polynomial as $\widetilde{\text{TQBF}}$ and not $\widetilde{\text{TQ-3-CNF}}$. We start with the following.

Definition 3.6 (Universal 3-CNF)**.** *Let $\mathbf{x}, \mathbf{y}$ be sets of Boolean variables of sizes n and m. A Boolean formula $\Phi_n(\mathbf{x}, \mathbf{y})$ is said to be a universal 3-CNF if for every n-variate 3-CNF $\varphi(\mathbf{x})$, there exists an assignment $\mathbf{b} \in \{0,1\}^m$ such that $\Phi_n(\mathbf{x}, \mathbf{b}) = \varphi(\mathbf{x})$.*

Now, we describe a 'canonical' universal 3-CNF, which would be used to define $\widetilde{\text{TQBF}}$. Let $n \in \mathbb{N}$, $\mathbf{x} := \{x_1, \ldots, x_n\}$ and $\mathbf{y} := \{y_{\mathbf{i},\mathbf{e}} : \mathbf{i} \in [n]^3, \mathbf{e} \in \{0,1\}^3\}$ be sets of Boolean variables. Note that $|\mathbf{y}| = 8n^3$. For $x \in \mathbf{x}$, let $x^{[0]}$ and $x^{[1]}$ denote x and $\neg x$. Consider the following Boolean formula

$$\Phi_n(\mathbf{x}, \mathbf{y}) := \bigwedge_{\mathbf{i}:=(i_1,i_2,i_3)\in[n]^3} \bigwedge_{\mathbf{e}:=(e_1,e_2,e_3)\in\{0,1\}^3} \left(\left(x_{i_1}^{[e_1]} \vee x_{i_2}^{[e_2]} \vee x_{i_3}^{[e_3]} \right) \vee \neg y_{\mathbf{i},\mathbf{e}} \right), \tag{2}$$

Observation 3.7. *The Boolean formula $\Phi_n(\mathbf{x}, \mathbf{y})$ defined in Equation (2) is a universal 3-CNF. Furthermore, given an n-variate 3-CNF $\varphi(\mathbf{x})$, an assignment $\mathbf{b}_\varphi \in \{0,1\}^{8n^3}$ satisfying $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x}, \mathbf{b}_\varphi)$ can be computed in time $O(n^3)$.*

Arithmetization of $\Phi_n(\mathbf{x}, \mathbf{y})$. Now we arithmetize the universal 3-CNF $\Phi_n(\mathbf{x}, \mathbf{y})$. Fix an arbitrary field $\mathbb{F}$ and the arithmetized polynomial will be over $\mathbb{F}$. Arithmetization of an n-variate Boolean formula $\varphi(\mathbf{x})$ is the process of obtaining a multivariate polynomial $\widehat{\varphi}(\mathbf{x})$ [15] such that φ and $\widehat{\varphi}$ agree on the Boolean hypercube, i.e., for every $\mathbf{a} \in \{0,1\}^n, \varphi(\mathbf{a}) = \widehat{\varphi}(\mathbf{a})$ [16]. We use the following notational convention here - a Greek letter denotes a Boolean formula and the same letter with a hat denotes the corresponding arithmetized polynomial. Arithmetization of Boolean formulas have been widely used in many important works in computational complexity such as [LFKN92, Sha92, TV07]. The rules of arithmetization are as follows: Let $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x})$ be two n-variate Boolean formulas. Then $\neg\varphi_1$ is arithmetized as $1 - \widehat{\varphi}_1$, and $\varphi := \varphi_1 \wedge \varphi_2$ and $\varphi' := \varphi_1 \vee \varphi_2$ are arithmetized as $\widehat{\varphi} := \widehat{\varphi}_1 \cdot \widehat{\varphi}_2$ and $\widehat{\varphi'} := 1 - (1 - \widehat{\varphi}_1)(1 - \widehat{\varphi}_2)$, respectively. Then for every $\mathbf{a} \in \{0,1\}^n, \varphi(\mathbf{a}) = \widehat{\varphi}(\mathbf{a})$ and $\varphi'(\mathbf{a}) = \widehat{\varphi'}(\mathbf{a})$. On recursively applying the rules, it is not difficult to show that we get that the arithmetization of the clause $\left(x_{i_1}^{[e_1]} \vee x_{i_2}^{[e_2]} \vee x_{i_3}^{[e_3]} \right)$ is $\left[1 - (1 - (e_1 - a_1 x_{i_1}))(1 - (e_2 - a_2 x_{i_2}))(1 - (e_3 - a_3 x_{i_3}))\right]$, such that if $e_j = 0, a_j := -1$, and if $e_j = 1, a_j := 1$ for every $j \in [3]$. Recall that $x_i^{[0]}, x_i^{[1]}$ denote x_i and $\neg x_i$, respectively. It follows from the above discussion that the arithmetization of the universal 3-CNF $\Phi_n(\mathbf{x}, \mathbf{y})$ defined in Equation (2) is as follows.

$$\widehat{\Phi}_n(\mathbf{x}, \mathbf{y}) := \prod_{\mathbf{i}:=(i_1,i_2,i_3)\in[n]^3, \mathbf{e}:=(e_1,e_2,e_3)\in\{0,1\}^3} \left[1 - (1 - (e_1 - a_1 x_{i_1}))(1 - (e_2 - a_2 x_{i_2}))(1 - (e_3 - a_3 x_{i_3})) \cdot y_{\mathbf{i},\mathbf{e}}\right]. \tag{3}$$

The following observation is implied by arithmetization.

[14] The notation $\Lambda_z f|_{\mathbf{x}=\mathbf{a}}$ means that we first apply the operator Λ_z to f and then substitute $\mathbf{x}$ with $\mathbf{a}$.

[15] $\mathbf{x}$ in $\varphi(\mathbf{x})$ and $\widehat{\varphi}(\mathbf{x})$ represent the sets of Boolean variables and formal variables that take values from the underlying field. These two sets have the same cardinalities. The difference between the two should be clear from the context.

[16] Whenever we talk about evaluating a polynomial on a Boolean vector, it means that we evaluate it on a vector whose entries are the multiplicative identity 1 and the additive identity 0 of $\mathbb{F}$.

Observation 3.8. *Let $\Phi_n(\mathbf{x},\mathbf{y})$ and $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ be the universal 3-CNF and the polynomial defined in Equations (2) and (3), respectively. Then for every $\mathbf{a}\in\{0,1\}^n$ and $\mathbf{b}\in\{0,1\}^m, \Phi_n(\mathbf{a},\mathbf{b}) = \widehat{\Phi}_n(\mathbf{a},\mathbf{b})$, where $n=|\mathbf{x}|$ and $m=|\mathbf{y}|$.*

Proposition 3.9. *The polynomial family $\{\widehat{\Phi}_n(\mathbf{x},\mathbf{y})\}$ is in $\mathsf{VP}^{\mathrm{poly}}$, has $\{0,1,-1\}$ constants [17] and hence in Uniform $\mathsf{VPSPACE}^0$ (Definition 2.17), where $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ is the polynomial defined in Equation (3).*

Defining the terms of $\widetilde{\mathrm{TQBF}}_n$. Let $\Psi_n(\mathbf{x},\mathbf{y}) := Q_1x_1\cdots Q_nx_n\,\Phi_n(\mathbf{x},\mathbf{y})$, where $Q_1,\ldots,Q_n$ are alternating quantifiers, $Q_1=\exists$, and $\Phi_n(\mathbf{x},\mathbf{y})$ is the universal 3-CNF defined in Equation (2) [18]. Recall that $|\mathbf{x}|=n$ and $|\mathbf{y}|=m$. Since $\Phi_n(\mathbf{x},\mathbf{y})$ is a universal 3-CNF, it follows that $\Psi_n(\mathbf{x},\mathbf{y})$ is a universal TQ-3-CNF (Definition 2.3), i.e., for every n-variate TQ-3-CNF $\phi_n(\mathbf{x}) = Q_1x_1\cdots Q_nx_n\,\varphi_n(\mathbf{x})$, where $Q_1,\cdots,Q_n$ are the quantifiers used in the definition of $\Psi_n(\mathbf{x},\mathbf{y})$ and $\varphi_n(\mathbf{x})$ is an arbitrary 3-CNF, there exists an assignment $\mathbf{b}_\phi\in\{0,1\}^m$ such that $\Psi_n(\mathbf{x},\mathbf{b}_\phi)=\phi_n(\mathbf{x})$. As mentioned before, for simplicity of exposition, we will call the polynomial as $\widetilde{\mathrm{TQBF}}_n$ and not $\widetilde{\mathrm{TQ\text{-}3\text{-}CNF}}_n$. The terms of $\widetilde{\mathrm{TQBF}}_n$ will be defined by uncovering the quantifiers one by one from right to left in $\Psi_n(\mathbf{x},\mathbf{y})$, arithmetizing them using universal and existential operators in Definition 3.1 and then applying the linearization operators (Λ) at every step to control the degree of the resulting polynomial. Let us discuss the arithmetization of quantifiers $Q_1,\ldots,Q_n$. Suppose $Q_i=\forall$ (or $\exists$). Since Q_i acts on the variable x_i in $\Psi_n(\mathbf{x},\mathbf{y})$, the arithmetization of Q_i, denoted $\widehat{Q}_i$, is given by the operator $\forall_{x_i}$ (respectively, $\exists_{x_i}$) given in Definition 3.1.

Proposition 3.10. *Let $\varphi_n(\mathbf{x})$ be an n-variate 3-CNF and $f(\mathbf{x})$ be a polynomial satisfying that for every $\mathbf{a}\in\{0,1\}^n, \varphi(\mathbf{a})=f(\mathbf{a})$. Let $x\in\mathbf{x}$ be an arbitrary variable and $\mathbf{x}':=\mathbf{x}\setminus\{x\}$. Then the following properties hold for every $\mathbf{a}'\in\{0,1\}^{n-1}$.*
1. $\forall x\,\varphi|_{\mathbf{x}'=\mathbf{a}'} = \forall_x f|_{\mathbf{x}'=\mathbf{a}'}$. *2.* $\exists x\,\varphi|_{\mathbf{x}'=\mathbf{a}'} = \exists_x f|_{\mathbf{x}'=\mathbf{a}'}$.

Corollary 3.11. *Let $\varphi(\mathbf{x})$ be an n-variate 3-CNF and $f(\mathbf{x})$ be a polynomial satisfying that for every $\mathbf{a}\in\{0,1\}^n, \varphi(\mathbf{a})=f(\mathbf{a})$. Let $x\in\mathbf{x}$ be an arbitrary variable and $\mathbf{x}':=\mathbf{x}\setminus\{x\}$. Then the following hold for every $\mathbf{a}'\in\{0,1\}^{n-1}$ and for every $X\subseteq\mathbf{x}'$.*
1. $\forall x\,\varphi|_{\mathbf{x}'=\mathbf{a}'} = \Lambda_X\forall_x\widehat{\varphi}|_{\mathbf{x}'=\mathbf{a}'}$. *2.* $\exists x\,\varphi|_{\mathbf{x}'=\mathbf{a}'} = \Lambda_X\exists_x\widehat{\varphi}|_{\mathbf{x}'=\mathbf{a}'}$.

Its proof follows from Propositions 3.5 and 3.10. Now, we start the description of the polynomial. For simplicity of notation, we rename the variables in $\mathbf{y}$ as $\mathbf{y}:=\{y_1,\ldots,y_m\}$, where $m=8n^3$. We define all the terms of $\widetilde{\mathrm{TQBF}}_n$ in n 'layers'. The i-th layer corresponds to the quantifier Q_i in $\Psi_n(\mathbf{x},\mathbf{y})$. We start by arithmetizing the quantifier Q_n in $\Psi_n(\mathbf{x},\mathbf{y})$. So, our first polynomial is $f_{n,0}:=\widehat{Q}_n\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$, where $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ is defined in Equation (3) and $\widehat{Q}_n$ is either $\forall_{x_n}$ or $\exists_{x_n}$ depending on whether Q_n is $\forall$ or $\exists$. The polynomial $f_{n,0}$ is the starting point of Layer n [19]. As noted in Observation 3.2, the degree of $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ might have doubled after applying $\widehat{Q}_n$. As a result of this, if we arithmetize every Q_i sequentially, the degree of the resulting polynomial can become exponential in n. But we need the degree of the resulting polynomial to be $\mathrm{poly}(n)$, particularly for Theorem 4.7. To achieve this, we apply the operator Λ_z for every variable appearing in $f_{n,0}$, i.e., for every $z\in\{x_1,\ldots,x_{n-1},y_1,\ldots,y_m\}$ [20], one by one starting with x_1 till we reach y_m. In particular, for every $j\in[n-1]$, $f_{n,j}:=\Lambda_{x_j}f_{n,j-1}$, $f_{n,n}:=\Lambda_{y_1}f_{n,n-1}$ and for $j'\in[2,m]$, $f_{n,n+j'-1}:=\Lambda_{y_{j'}}f_{n,n+j'-2}$. Hence,

Layer n: $f_{n,0}, f_{n,1},\ldots,f_{n,n+m-1}$.

In $f_{n,j}$, n and j correspond to the layer number and the variable with respect to which the linearization operator (Λ) is applied. Now, we are ready to define Layer $n-1$. To obtain the first polynomial of this layer, we apply $\widehat{Q}_{n-1}$ to the last polynomial of Layer n, which was $f_{n,n+m-1}$. Thus, $f_{n-1,0}:=\widehat{Q}_{n-1}f_{n,n+m-1}$. Now, as before, we apply the linearization operator on $f_{n-1,0}$ with respect to every variable appearing in $f_{n-1,0}$, i.e., with respect to every $z\in\{x_1,\ldots,x_{n-2},y_1,\ldots,y_m\}$, one by one starting with x_1 till

[17] Infact, such a sub-class of $\mathsf{VP}^{\mathrm{poly}}$ is called VP^0 or constant-free VP, which was defined by Malod [Mal03].
[18] Note that these quantifiers are different from the operators $\forall_z$ and $\exists_z$ given in Definition 3.1. The difference would be clear from the context.
[19] For notational convenience, we number the layers in the descending order.
[20] It follows from Definition 3.1 that $f_{n,0}\in\mathbb{F}[x_1,\ldots,x_{n-1},y_1,\ldots,y_m]$

we reach y_m. Thus, for every $j \in [n-2]$, $f_{n-1,j} := \Lambda_{x_j} f_{n-1,j-1}$, $f_{n-1,n-1} := \Lambda_{y_1} f_{n-1,n-2}$ and for $j' \in [2,m]$, $f_{n-1,n+j'-2} := \Lambda_{y_{j'}} f_{n-1,n+j'-3}$. Hence,

Layer $n-1$: $f_{n-1,0}, f_{n-1,1}, \ldots, f_{n-1,n+m-2}$.

Now, we define $f_{n-2,0}$ by applying $\widehat{Q}_3$ to $f_{n-1,n+m-2}$ and continue the process until we reach $f_{1,m}$. Thus, we immediately get the following recursive definition of $f_{i,j}$'s. For $i \in \{n-1, n-2, \ldots, 1\}$, the polynomials of Layer i are defined as follows: The first polynomial of this layer, namely $f_{i,0}$, is obtained by applying $\widehat{Q}_i$ to $f_{i+1,m+i}$, which is the last polynomial of Layer $i+1$. Thus,

$$f_{i,0} := \widehat{Q}_i \, f_{i+1,m+i}. \tag{4}$$

After this, we obtain the other polynomials of Layer i by applying Λ with respect to the variables in $f_{i,0}$, which are $\{x_1, \ldots, x_{i-1}, y_1, \ldots, y_m\}$. The operators Λ are applied with respect to variables in order $x_1, \cdots, x_{i-1}, y_1, \ldots, y_m$, i.e., for $j \in [i-1]$, and for $j' \in [2,m]$,

$$f_{i,j} := \Lambda_{x_j} f_{i,j-1}, \quad f_{i,i+j'-1} := \Lambda_{y_{j'}} f_{i,i+j'-2}, \quad \text{and, } f_{i,i} := \Lambda_{y_1} f_{i,i-1}.$$

For reader's convenience, we present all these layers in the tabular form below. Whenever some (i,j)-th entry of the table is $-$ it means that the polynomial $f_{i,j}$ is not present in Layer i.

Layer n:	$f_{n,0}$	$\cdots$	$f_{n,m}$	$f_{n,m+1}$	$\cdots$	$f_{n,m+i-1}$	$f_{n,m+i}$	$\cdots$	$f_{n,m+n-1}$	$-$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
Layer i:	$f_{i,0}$	$\cdots$	$f_{i,m}$	$f_{i,m+1}$	$\cdots$	$f_{i,m+i-1}$	$-$	$\cdots$	$-$	$-$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
Layer 1:	$f_{1,0}$	$\cdots$	$f_{1,m}$	$-$	$\cdots$	$-$	$-$	$\cdots$	$-$	$-$

Definition of the polynomial. Let $n \in \mathbb{N}$ and $\mathbf{z} := \{z_{i,j} : i \in [n], j \in \{0,1,\ldots,m+i-1\}\}$ be a set of fresh variables. Recall $f_{i,j}, i \in [n], j \in \{0,1,\ldots,n+m-i\}$ defined above. Then

$$\widetilde{\mathrm{TQBF}}_n(\mathbf{x},\mathbf{y},\mathbf{z}) := \sum_{i\in[n]} \left(\sum_{j\in[0,m+i-1]} z_{i,j} \cdot f_{i,j} \right). \tag{5}$$

For many parts, the most important term of $\widetilde{\mathrm{TQBF}}_n(\mathbf{x},\mathbf{y},\mathbf{z})$ is $f_{1,m}$. We recall the definition of $f_{1,m}$ here. For $j \in [n]$, let $X_j := \{x_1, \ldots, x_j\} \uplus \mathbf{y}$ and let $X_0 := \mathbf{y}$. Then

$$f_{1,m} = \Lambda_{X_0} \widehat{Q}_1 \Lambda_{X_1} \widehat{Q}_2 \cdots \Lambda_{X_{n-2}} \widehat{Q}_{n-1} \, \Lambda_{X_{n-1}} \widehat{Q}_n \widehat{\Phi}(\mathbf{x},\mathbf{y}), \tag{6}$$

where $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ is defined in Equation (3). Observation 3.3 implies that it does not matter in which variable order we apply the Λ operator on X_j for $j \in [0, n-1]$.

Observation 3.12. *The number of variables in* $\widetilde{\mathrm{TQBF}}_n$ *is* $n + 8n^3 + 8n^4 + \binom{n-1}{2}$.

Note that $f_{n,0}$ is non-multilinear and neitheris $\widetilde{\mathrm{TQBF}}$. It is easy to show the following.

Observation 3.13. $\deg(\widetilde{\mathrm{TQBF}}_n) = O(n^4)$.

Finally, we define the relevant polynomial families.

Definition 3.14. *We define* $\widetilde{\mathcal{TQBF}}$ *to be the family of the* $\widetilde{\mathrm{TQBF}}_n$ *polynomial, i.e.,* $\widetilde{\mathcal{TQBF}} := \left\{ \widetilde{\mathrm{TQBF}}_n : n \in \mathbb{N} \right\}$.
For notational convince we also define the family $\widetilde{\mathcal{TQBF}'}$ *as a projection of* $\widetilde{\mathcal{TQBF}}$.
Formally: $\widetilde{\mathrm{TQBF}'}_n(\mathbf{y}) \triangleq \widetilde{\mathrm{TQBF}}_n(\mathbf{0},\mathbf{y},\mathbf{z} \setminus \{z_{1,m}\} = \mathbf{0}, z_{1,m} = 1)$.

Indeed, observe that the number of variables in $\widetilde{\mathrm{TQBF}'}_n(\mathbf{y})$ is $8n^3$ and the polynomials does not depend on the $\mathbf{x}$ variables.

4 Properties of $\widetilde{\mathcal{TQBF}}$

4.1 The circuit complexity of $\widetilde{\mathcal{TQBF}}$

Recall Definition 2.17 for the following lemma.

Lemma 4.1. $\widetilde{\mathcal{TQBF}} \in \mathsf{Uniform\,VPSPACE}^0$.

Proof. It follows from Proposition 2.19 that to prove this lemma, it is sufficient to prove that $\widetilde{\mathcal{TQBF}} \in \mathsf{Uniform\,VPAR}^0$ (Definition 2.18). Let $n \in \mathbb{N}$ be arbitrarily fixed. Recall from Equation (5) that $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is the sum of polynomially many summands of the kind $z_{i,j} f_{i,j}$, where the recursive definitions of $f_{i,j}$'s are given in Section 3.2. To prove $\widetilde{\mathcal{TQBF}} \in \mathsf{Uniform\,VPAR}^0$, note that it is sufficient to show that every $f_{i,j}$ in every $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is in $\mathsf{Uniform\,VPAR}^0$. Let us pick an arbitrary $f_{i,j}$. It follows from the construction of the terms of $\widetilde{\mathrm{TQBF}}_n$ given in the previous section that $f_{i,j}$ looks as follows, where $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ is the polynomial defined in Equation (3).

$$f_{i,j} = O_1 \cdots O_u \, \widehat{\Phi}_n(\mathbf{x}, \mathbf{y}),$$

where $u \in [10n^4]$ is the number of operators applied on $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ and every O_k is either universal or existential or linearization operator described in Definition 3.1. We prove the lemma by induction on the number of operators u applied on $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ in $f_{i,j}$. Suppose $u = 0$. Then we know from Claim 3.9 that $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y}) \in \mathsf{Uniform\,VPAR}^0$. Hence the base case holds. Now suppose that $g := O_2 \cdots O_u \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ is in $\mathsf{Uniform\,VPAR}^0$. Then we know from Proposition 2.19 and Definition 2.18 that there exists an arithmetic circuit C' that computes g such that C' has polynomial depth and the description of C' can be computed using $h(n)$ space, where $h : \mathbb{N} \to \mathbb{N}$ is a polynomial function. Since $f_{i,j} = O_1 \, g$, we prove the lemma by considering all possibilities for O_1.

1. $O_1 = \forall_x$ for some $x \in \mathbf{x}$: Definition 3.1 implies that

   $$f_{i,j} = g|_{x=0} \times g|_{x=1}.$$

   Let $\mathsf{C}'|_{x=0}, \mathsf{C}'|_{x=1}$ be restrictions of C' that compute $g|_{x=0}$ and $g|_{x=1}$, respectively. Let

   $$\mathsf{C} := \mathsf{C}'|_{x=0} \times \mathsf{C}'|_{x=1}.$$

   Clearly, C computes $f_{i,j}$. It follows from the induction hypothesis that C has polynomial depth. C can possibly have size $2^{p(n)}$ for some polynomial function $p : \mathbb{N} \to \mathbb{N}$. We know from the induction hypothesis that we can compute the circuit C' using $h(n)$ space. Now we compute the circuit C as follows: We first compute $\mathsf{C}'|_{x=0}$ using $h(n)$ space and write it on the output tape, then we reuse the space and compute $\mathsf{C}'|_{x=1}$ and write on the output tape. Now we create a new node with label $\times$ and connect the root nodes of $\mathsf{C}'|_{x=0}$ and $\mathsf{C}'|_{x=1}$ with it. As the size of C is at most $2^{p(n)}$, observe that the label of this new node can be created using $O(p(n))$ bits. Hence, the creation of the new node and linking it appropriately can be done using $O(p(n))$ space. This implies that we can compute C using $O(h(n) + p(n))$ space.

2. $O_1 = \exists_x$ for some $x \in \mathbf{x}$: It follows from Definition 3.1 that

   $$f_{i,j} = g|_{x=0} + g|_{x=1} - g|_{x=0} \times g|_{x=1}.$$

   Let $\mathsf{C}'|_{x=0}, \mathsf{C}'|_{x=1}$ be the circuits defined in the first case and

   $$\mathsf{C} := \mathsf{C}'|_{x=0} + \mathsf{C}'|_{x=1} - \mathsf{C}'|_{x=0} \times \mathsf{C}'|_{x=1}.$$

   In this case also the size of C is at most $2^{p(n)}$ for some polynomial function $p : \mathbb{N} \to \mathbb{N}$ and C computes $f_{i,j}$. Now, using an argument similar to the one used in the first case, we can show that C can be computed using $q(n)$ space for some polynomial function $q : \mathbb{N} \to \mathbb{N}$.

3. $\underline{O_1 = \Lambda_x \text{ for some } x \in \mathbf{x} \cup \mathbf{y}}$: We know from Definition 3.1 that

$$f_{i,j} = x \cdot g|_{x=1} + (1-x) \cdot g|_{x=0}.$$

Let $\mathsf{C}'|_{x=0}, \mathsf{C}'|_{x=1}$ be the circuits defined in the first case. Let

$$\mathsf{C} := x \times \mathsf{C}'|_{x=1} + (1-x) \times \mathsf{C}'|_{x=0}.$$

The circuit C computes $f_{i,j}$ and has size at most $2^{p(n)}$ for some polynomial function $p : \mathbb{N} \to \mathbb{N}$. Now, using an argument similar to the one used in the first case, we can show that C can be computed using $q(n)$ space for some polynomial function $q : \mathbb{N} \to \mathbb{N}$.

Thus we get that in all the three cases, the resulting circuit C computing $f_{i,j}$ has polynomial depth and its description can be computed using polynomial amount of space. Also, by construction, the only non-zero coefficient appearing in C is 1. Thus, the family of C is in $\mathsf{Uniform\,VPAR}^0$. Now it follows from Proposition 2.19 that the family of $f_{i,j}$ is in $\mathsf{Uniform\,VPSPACE}^0$. □

As degree and coefficients-size of $\widetilde{\mathrm{TQBF}}_n$ are polynomially bounded, we get the following.

Corollary 4.2. $\widetilde{\mathcal{TQBF}} \in \mathsf{Uniform\,VPSPACE}^0_b$.

4.2 PSPACE-completeness of $L_{\mathrm{TQBF}'}$ - Proof of Theorem 1.

In this section we establish the computational power of $\widetilde{\mathcal{TQBF}}$ w.r.t to Boolean complexity by showing that the graph of $\widetilde{\mathcal{TQBF}'}$, the projection of $\widetilde{\mathcal{TQBF}}$, and hence of $\widetilde{\mathcal{TQBF}}$ itself, is PSPACE-complete, thus proving Theorem 1.

Lemma 4.3. *Let $\mathbf{x} = \{x_1, \ldots, x_n\}$, $\varphi(\mathbf{x})$ be a 3-CNF in $\mathbf{x}$ and $f_{i,j}$'s be the terms of the polynomial $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z})$ defined in Equation (5). For $j \in [n]$, we define $\mathbf{x}_j := \{x_1, \ldots, x_j\}$. Let $\mathbf{b} \in \{0,1\}^{|\mathbf{y}|}$ be the assignment from Observation 3.7. Then the following holds for every $i \in [n]$ and for every $\mathbf{a}_{i-1} \in \{0,1\}^{i-1}$.*

$$Q_i x_i \cdots Q_n x_n \, \varphi \,|_{\mathbf{x}_{i-1}=\mathbf{a}_{i-1}} = f_{i,m+i-1}|_{\mathbf{x}_{i-1}=\mathbf{a}_{i-1},\mathbf{y}=\mathbf{b}}, \tag{7}$$

where every $Q_i \in \{\forall, \exists\}$, $Q_1 = \exists$, and all quantifiers are alternating. [21]

Proof. Recall the universal 3-CNF $\Phi_n(\mathbf{x}, \mathbf{y})$ defined in Equation (2). By Observation 3.7: $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x}, \mathbf{b})$. Fix an $i \in [n]$ arbitrarily. Let ℓ be the number of quantifiers applied on $\varphi(\mathbf{x})$. We prove the lemma by induction on ℓ.

Base case: Let $\ell = 1$. We want to show that for every $\mathbf{a}_{n-1} \in \{0,1\}^{n-1}$,

$$Q_n x_n \, \varphi \,|_{\mathbf{x}_{n-1}=\mathbf{a}_{n-1}} = f_{n,m+n-1}|_{\mathbf{x}_{n-1}=\mathbf{a}_{n-1},\mathbf{y}=\mathbf{b}}.$$

Let $X_{n-1} := (\mathbf{x} \setminus \{x_n\}) \cup \mathbf{y}$. Recall from Section 3.2 that $f_{n,m+n-1} = \Lambda_{X_{n-1}} \widehat{Q}_n \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$, where $\Lambda_{X_{n-1}}$ denotes the iterated application of Λ_z for every $z \in X_{n-1}$ (see Definition 3.1). Since $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x}, \mathbf{b})$, Observation 3.8 implies that for every $\mathbf{b} \in \{0,1\}^n, \varphi(\mathbf{a}) = \widehat{\Phi}_n(\mathbf{a}, \mathbf{b})$. Now, it follows from Corollary 3.11 that

$$Q_n x_n \, \varphi \,|_{\mathbf{x}_{n-1}=\mathbf{a}_{n-1}} = \Lambda_{X_{n-1}} \widehat{Q}_n \widehat{\Phi}_n|_{\mathbf{x}_{n-1}=\mathbf{a}_{n-1},\mathbf{y}=\mathbf{b}}.$$

We know from the definition of the terms $f_{i,j}$'s of $\widetilde{\mathrm{TQBF}}_n$ that $f_{n,n+m-1} = \Lambda_{X_{n-1}} \widehat{Q}_n \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$. This proves the base case.

Induction hypothesis: Suppose the lemma holds for $\ell = n - i$. Then for every $\mathbf{a}_{i-1} \in \{0,1\}^{i-1}$,

$$Q_{i+1} x_{i+1} \cdots Q_n x_n \, \varphi \,|_{\mathbf{x}_i=\mathbf{a}_i} = f_{i+1,m+i}|_{\mathbf{x}_i=\mathbf{a}_i,\mathbf{y}=\mathbf{b}}.$$

[21] In Equation (7), $Q_i x_i \cdots Q_n x_n \, \varphi|_{\mathbf{x}_{i-1}=\mathbf{b}_{i-1}}$ means that we are first applying the quantifiers $Q_i, \ldots, Q_n$ to $\varphi(\mathbf{x})$ and then substituting $\mathbf{x}_{i-1} = \mathbf{b}_{i-1}$. A similar thing holds for $f_{i,m+i-1}|_{\mathbf{x}_{i-1}=\mathbf{a}_{i-1},\mathbf{y}=\mathbf{a}}$ also.

Induction step: Now we want to prove the lemma for $Q_i x_i \Psi_{i+1}$, where $\Psi_{i+1} := Q_{i+1}x_{i+1}\cdots Q_n x_n \varphi$. We know from the induction hypothesis that $\Psi_{i+1}|_{\mathbf{x}_i=\mathbf{a}_i} = f_{i+1,m+i}|_{\mathbf{x}_i=\mathbf{a}_i,\mathbf{y}=\mathbf{b}}$. Recall from Equation (4) that $f_{i,0} = \widehat{Q}_i f_{i+1,m+i}$. Now, it follows from Proposition 3.10 that for every $\mathbf{a}_i \in \{0,1\}^i, Q_i x_i \Psi_{i+1}|_{\mathbf{x}_i=\mathbf{a}_i} = f_{i,0}|_{\mathbf{x}_i=\mathbf{a}_i}$. Let $\mathbf{x}' := \mathbf{x} \setminus \mathbf{x}_i$. Since $f_{i,m+i-1} = \Lambda_{\mathbf{x}'\cup\mathbf{y}} f_{i,0}$, on applying Corollary 3.11, we get that for every $\mathbf{a}_i \in \{0,1\}^i, Q_i x_i \Psi_{i-1}|_{\mathbf{x}_i=\mathbf{a}_i} = f_{i,m+i-1}|_{\mathbf{x}_i=\mathbf{a}_i,\mathbf{y}=\mathbf{b}}$. Hence proved. □

This lemma implies the following important result, which is the second property of $\widetilde{\mathrm{TQBF}}_n(\mathbf{x},\mathbf{y},\mathbf{z})$. We recall the definition of $\widetilde{\mathrm{TQBF}}'_n$ for reader's convenience.

Theorem 4.4 (Theorem 1 restated and extended)**.** *The language*

$$L_{\mathrm{TQBF}'} \triangleq \left\{ y \in \{0,1\}^* \;\middle|\; |y| = 8n^3 \wedge \widetilde{\mathrm{TQBF}}'_n(y) = 1 \right\}$$

is PSPACE*-complete. Furthermore, the reduction can be implemented by Boolean formulas. Finally, for any* $n \in \mathbb{N}$ *and all* $y \in \{0,1\}^{8n^3}$*:* $\widetilde{\mathrm{TQBF}}'_n(y) \in \{0,1\}$*.*

Proof. The containment $L_{\mathrm{TQBF}'} \in$ PSPACE follows from Lemma 4.1. For the hardness, let $\varphi(\mathbf{x})$ be an n-variate 3-CNF and $\Psi := Q_1 x_1 \cdots Q_n x_n\, \varphi(\mathbf{x})$, where $Q_i \in \{\forall, \exists\}$ for every $i \in [n]$, $Q_1 = \exists$ and these quantifiers are alternating. Let $\mathbf{b}_\varphi \in \{0,1\}^{8n^3}$ be the corresponding assignment from Observation 3.7, which is computable in polynomial time. We argue that $\Psi \in$ TQ-3-CNF if and only if $\widetilde{\mathrm{TQBF}}'_n(\mathbf{b}_\varphi) = 1$.

By Lemma 4.3 (set $i = 1$), $\Psi \in$ TQ-3-CNF if and only if $f_{1,m}|_{\mathbf{y}=\mathbf{b}_\varphi} = 1$. At the same time, by definition, $\widetilde{\mathrm{TQBF}}'_n(\mathbf{y}) = f_{1,m}(\mathbf{y})$. Therefore, $\Psi \in$ TQ-3-CNF if and only if $\widetilde{\mathrm{TQBF}}'_n(\mathbf{b}_\varphi) = 1$, as required.

Since the reduction to TQ-3-CNF can be implemented by Boolean formulas (see e.g. [AB09]) and $\mathbf{b}_\varphi$ can be "read-off' the input formula, the entire procedure can be implemented by Boolean formulas. Finally, applying Lemma 4.3 with $i = 1$, again, $\widetilde{\mathrm{TQBF}}'_n(\mathbf{b}) = f_{1,m}(\mathbf{b}) \in \{0,1\}$. □

4.3 Consequences in Boolean Complexity, If $\widetilde{\mathrm{TQBF}}_n$ has Small Circuits

We observe that if $\widetilde{\mathcal{TQBF}}$ admits "small" arithmetic circuits with bounded coefficients, then these circuits can be used as non-uniform advice. In particular, we prove the following:

Lemma 4.5. *Suppose* $\widetilde{\mathcal{TQBF}} \in \mathsf{VP}^{\mathrm{poly}}$*. Then* PSPACE $\subseteq$ P/poly*.*

Proof. By definition, for every $n \in \mathbb{N}$, the polynomial $\widetilde{\mathrm{TQBF}}_n(\mathbf{x},\mathbf{y},\mathbf{z})$ is computed by an arithmetic circuit $\mathsf{C}_n(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathsf{VP}^{\mathrm{poly}}$ having size and syntactic degree $\mathrm{poly}(n)$. Observation 2.12 implies that C_n can be described using $\mathrm{poly}(n)$ bits. Furthermore, the same holds true for $\mathsf{C}_n(0,\mathbf{y},\mathbf{z} \setminus \{z_{1,m}\} = \mathbf{0}, z_{1,m} = 1)$ that computes $\widetilde{\mathcal{TQBF}}'_n$ (see Definition 3.14). To complete the proof, we invoke Theorem 1 with encoding of the circuit C_n as advice. □

Remark 4.6. *The assumption that the circuits have bounded coefficient (i.e.* $\mathsf{VP}^{\mathrm{poly}}$*) appears to be crucial. The only known way to extend the result to the unbounded case (i.e.* VP*) requires the GRH (see [Bür00, KP09]).*

4.4 Self-Testability - Proof of Theorem 7

Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a fixed polynomial and C be an arithmetic circuit over $\mathbb{F}$. It is natural to ask if we can test whether C computes f, where access to f is not given explicitly. We call this the *self-testability* of f. If f is a member of a polynomial family in VP then the DeMillo-Lipton-Schwartz-Zippel lemma gives a randomized polynomial-time algorithm to test this. But if f belongs to the polynomial family not known to be in VP, then it is unclear whether an efficient randomized self-testability algorithm always exists. In the following theorem, we show a 'weaker' self-testability $\widetilde{\mathcal{TQBF}}$. Efficient self-testability algorithms are

also known for two important polynomial families not known to be in VP - the permanent polynomial [Lip89, Mul10], and the Nisan-Wigderson design polynomial [GS19]. These polynomial families play crucial roles in algebraic complexity.

Let $\mathcal{C}$ be a class of arithmetic circuits. Then, $\Sigma^{[k]}\Pi^{[s]}\mathcal{C}$ is defined as the set of the circuits of the kind $A_1 + \cdots + A_k$, where every $A_i = B_{i,1} \cdots B_{i,s}$ and every $B_{i,j} \in \mathcal{C}$. In the following theorem, we prove a weaker version of self-testability for $\widetilde{\mathcal{TQBF}}$, i.e., we only test if the coefficient of every $z_{i,j}$ in C indeed computes the polynomial $f_{i,j}$, as we do not need the full power of self-testability. We remark that by making a somewhat stronger assumption that we are given oracle access to a PIT algorithm for $\Sigma^{[\mathrm{poly}]}\prod^{[2]}\mathcal{C}$ (instead of merely $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$) in the following theorem, we can achieve full self-testability for $\widetilde{\mathcal{TQBF}}$.

Theorem 4.7 (Self-testability of $\widetilde{\mathcal{TQBF}}$)**.** *Let $n \in \mathbb{N}, \mathbb{F}$ be a field, and $\mathcal{C}$ be an arbitrary class over $\mathbb{F}$ such that $\mathcal{C}$ is closed under projections, multiplication with a linear univariate polynomial, and the polynomial $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ defined in Equation (3) is in $\mathcal{C}$. Then, there exists a deterministic polynomial-time algorithm A that given a circuit $\mathsf{C} \in \mathcal{C}$ of syntactic degree d as input decides whether for every $i \in [n], j \in [0, m+i-1]$ the restriction of $\mathsf{C}(\mathbf{x}, \mathbf{y}, \mathbf{z} \setminus \{z_{i,j}\} = \mathbf{0}, z_{i,j} = 1)$ computes the polynomial $f_{i,j}$, defined in Section 3.2, given oracle access to a PIT algorithm for $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$.*

Proof. We know that $\widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}) := \sum_{i \in [n]} \left(\sum_{j \in [0, m+i-1]} z_{i,j} \cdot f_{i,j} \right)$, where the definitions of $f_{i,j}$'s are given in Section 3.2. For $i \in [n], j \in [0, m+i-1]$, let $\mathsf{C}_{i,j}$ be obtained from C by substituting every $\mathbf{z}$ variable except $z_{i,j}$ equal to 0 and $z_{i,j} = 1$. We want to test whether $\mathsf{C}_{i,j}$ computes $f_{i,j}$. Let $f_{i,j}$ be chosen arbitrarily. Then, as seen in the proof of Lemma 4.1, $f_{i,j}$ looks as

$$f_{i,j} = O_1 \cdots O_u \, \widehat{\Phi}_n(\mathbf{x}, \mathbf{y}),$$

where $u \in [10n^4]$ is the number of operators applied on $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ and every O_k is either universal or existential or linearization operator described in Definition 3.1. Let $g := O_2 \cdots O_u \widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$ and C' be an arithmetic circuit which computes g. It follows from the recursive construction of $f_{i,j}$'s that if $j \neq 0$ then $g = f_{i,j-1}$, otherwise $g = f_{i+1,m+i}$. As argued in the proof of Lemma 4.1, there are three possibilities.

1. If $O_1 = \forall_x$ for some $x \in \mathbf{x}$, then $f_{i,j} = g|_{x=0} \times g|_{x=1}$. Thus, if $\mathsf{C}_{i,j}$ computes $f_{i,j}$ then it should satisfy that $\mathsf{C}_{i,j} = \mathsf{C}'|_{x=0} \times \mathsf{C}'|_{x=1}$.
2. If $O_1 = \exists_x$ for some $x \in \mathbf{x}$ then $f_{i,j} = g|_{x=0} + g|_{x=1} - g|_{x=0} \times g|_{x=1}$. Thus, if $\mathsf{C}_{i,j}$ computes $f_{i,j}$ then it should satisfy that $\mathsf{C}_{i,j} := \mathsf{C}'|_{x=0} + \mathsf{C}'|_{x=1} - \mathsf{C}'|_{x=0} \times \mathsf{C}'|_{x=1}$.
3. If $O_1 = \Lambda_x$ for some $x \in \mathbf{x} \cup \mathbf{y}$ then $f_{i,j} = x \cdot g|_{x=1} + (1-x) \cdot g|_{x=0}$. Thus, if $\mathsf{C}_{i,j}$ computes $f_{i,j}$ then it should satisfy that $\mathsf{C}_{i,j} := x \times \mathsf{C}'|_{x=1} + (1-x) \times \mathsf{C}'|_{x=0}$.

We know from the discussion in Section 3.2 that the first 'slice' of $\widetilde{\mathrm{TQBF}}_n$ is $f_{n,0}$, which is obtained by applying the quantifier $\widehat{Q}_n$ to the polynomial $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})$, where $\widehat{Q}_n$ is either universal or existential operator. The algorithm A first tests if $\mathsf{C}_{n,0}$ computes the polynomial $f_{n,0}$ as follows: As $\mathcal{C}$ is closed under projection and as $\widehat{\Phi}_n(\mathbf{x}, \mathbf{y}) \in \mathcal{C}$, we get that $\mathsf{C}_{n,0} \in \mathcal{C}$. Also, as $\deg(\widehat{\Phi}_n(\mathbf{x}, \mathbf{y})) = \mathrm{poly}(n)$, the above discussion implies that by invoking the PIT on $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$, A tests in $\mathrm{poly}(n)$ time whether $\mathsf{C}_{n,0}$ computes $f_{n,0}$ or not. If not, A aborts. Otherwise, it tests whether $\mathsf{C}_{n,1}$ computes $f_{n,1}$. Recall that the polynomial $f_{n,1}$ is constructed from $f_{n,0}$, and as $\mathsf{C}_{n,0}$ computes $f_{n,0}$, A again uses the PIT for $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$ to test whether $\mathsf{C}_{n,1}$ computes $f_{n,1}$. More generally, for any $i \in [n], j \in [0, m-i+1]$, the algorithm A comes to the stage of testing whether $\mathsf{C}_{i,j}$ computes $f_{i,j}$ only if for every $0 \leq j' < j, \mathsf{C}_{i,j'}$ computes $f_{i,j'}$ and for every $i' \in \{n, n-1, \ldots, i+1\}, j' \in [0, m+i'-1], \mathsf{C}_{i',j'}$ computes $f_{i',j'}$. Now, as $\deg(f_{i,j}) = \mathrm{poly}(n)$ (Observation 3.13), A uses the PIT for $\Sigma^{[4]}\Pi^{[2]}\mathcal{C}$ to test whether $\mathsf{C}_{i,j}$ computes $f_{i,j}$. Clearly, A is deterministic (Observation 3.13) and as $\deg(\widetilde{\mathrm{TQBF}}_n) = \mathrm{poly}(n)$ and the number of slices $f_{i,j}$'s are $O(n^4)$, A runs in polynomial time.

□

4.5 Consequences of Self-Testability of $\widetilde{\mathcal{TQBF}}$ - Proof of Corollary 8

Let $\mathcal{C} \subseteq \mathsf{VP}^{\mathrm{poly}}$ be a class of arithmetic circuits that satisfy that $\widehat{\Phi}_n \in \mathcal{C}, \Sigma^{[4]}\Pi^{[2]}\mathcal{C} \subseteq \mathcal{C}$ and $\mathcal{C}$ be closed under multiplication with univariate linear polynomials. In this section, we prove that if $\widetilde{\mathrm{TQBF}}_n$ is computed by a circuit from a class $\mathcal{C}$ then PSPACE collapses to $\mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$, where $\mathsf{NP}^{\mathsf{PIT}_{\mathcal{C}}}$ is the set of languages decided by non-deterministic polynomial-time Turing machine with oracle access to a PIT algorithm for $\mathcal{C}$.

1. Suppose $\widetilde{\mathcal{TQBF}} \in \mathcal{C}$. Then there exists an arithmetic circuit $\mathsf{C} \in \mathcal{C}$ of size and syntactic degree $\mathrm{poly}(n)$ that computes $\widetilde{\mathrm{TQBF}}_n$. Furthermore, by Observation 2.12, such C can be encoded using $\mathrm{poly}(n)$ many bits. Our goal is to guess such a circuit and verify that it, indeed, computes $\widetilde{\mathrm{TQBF}}_n$. To this end, we first guess a circuit $\mathsf{C} \in \mathcal{C}$. As the size of C is polynomial, this guessing can be carried out in NP. As a next step, we test whether C represents a valid circuit in $\mathcal{C}$. Subsequently, using Observation 2.10 we compute the syntactic degree of C and verify that it is indeed $\mathrm{poly}(n)$. Now, once we get hold of a right circuit $\mathsf{C} \in \mathcal{C}$ of a "low" syntactic degree, we use the oracle to $\mathsf{PIT}_{\mathcal{C}}$ to invoke the self-testability procedure from Theorem 4.7. If the procedure fails, we reject. If the procedure accepts, we compute $C'(\mathbf{y}) \triangleq \mathsf{C}(0, \mathbf{y}, \mathbf{z} \setminus \{z_{1,m}\} = \mathbf{0}, z_{1,m} = 1)$ and use it to decide any language in PSPACE. To finish the proof, we observe that:
   - By assumption, there exists a non-deterministic guess that will pass all the tests, including self-testability (Theorem 4.7).
   - If a circuit C passes all the tests, $C'(y)$ must compute the polynomial $f_{1,m} = \widetilde{\mathcal{TQBF}'}_n$. The result then follows from Theorem 1.
2. Suppose $\mathcal{C}$ admits a deterministic (or randomized) polynomial-time reconstruction algorithm. The idea is to use dynamic programming to learn the polynomial "slice-by-slice". First learn $f_{n,0}$ and the subsequent $f_{i,j}$ until we learn $f_{1,m}$. This will allow us to decide PSPACE in P (respectively BPP).
3. Any reconstruction algorithm gives rise to a black-box PIT algorithm by querying it on the "zero" polynomial. Therefore, we obtain a sub-polynomial black-box PIT. The result then follows from Corollary 3.

5 The Final Polynomial $\mathcal{TP}$ and Its Applications

In this section, we define the ultimate polynomial family, which is a mix of $\widetilde{\mathcal{TQBF}}$ and $\mathcal{PERM}$. This is used to obtain our main results. For $n \in \mathbb{N}$, let $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}$ be sets of distinct variables where $|\mathbf{x}| = n, |\mathbf{y}| = 8n^3, |\mathbf{z}| = 8n^4 + \binom{n-1}{2}$ and $|\mathbf{u}| = n^2$. We define the polynomial TP_n as

$$\mathrm{TP}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}) := \widetilde{\mathrm{TQBF}}_n(\mathbf{x}, \mathbf{y}, \mathbf{z}) + \mathrm{Perm}_n(\mathbf{u}). \tag{8}$$

We define the corresponding polynomial family as $\mathcal{TP} := \{\mathrm{TP}_n : n \in \mathbb{N}\}$. Note that as $\widetilde{\mathrm{TQBF}}$ and Perm are defined in different variables, no monomial of $\widetilde{\mathrm{TQBF}}$ can be cancelled by a monomial of Perm and vice versa. In particular, $\mathrm{TP}_n|_{\mathbf{z}=0} = \mathrm{Perm}_n(\mathbf{u})$ and $\mathrm{TP}_n|_{\mathbf{u}=0} = \widetilde{\mathrm{TQBF}}_n$. Since $\mathcal{PERM}$ is VNP-complete, and in fact $\mathcal{PERM} \in \mathsf{VNP}^0$, the following property follows from Lemma 5.2 and Corollary 4.2.

Claim 5.1. *The polynomial family $\mathcal{TP}$ is in $\mathsf{VPSPACE}^0_b$.*

Lemma 5.2 ([KP09, Bür24])**.** *$\mathsf{VNP} \subseteq \mathsf{VPSPACE}_b$ and $\mathsf{VNP}^0 \subseteq \mathsf{VPSPACE}^0_b$.*

5.1 Proof of Theorem 5

In this section we prove Theorem 5. First, we recall the relevant result from [KP09].

Proposition 5.3 (Proposition 4.1 of [KP09])**.** *If $\mathsf{PSPACE/poly} = \mathsf{P/poly}$ and $\mathsf{VP} = \mathsf{VNP}$ then $\mathsf{VPSPACE}_b = \mathsf{VP}$. The converse holds, assuming the generalized Riemann hypothesis (GRH).*

We now prove Theorem 5 by showing that the converse of the result of [KP09], mentioned in Proposition 5.3, holds without the GRH in the constant-free setting. In fact, we prove a somewhat more general result. We remark that it is currently unclear how to extend the original argument of [KP09] to the constant-free setting without assuming the GRH.

Lemma 5.4. *Let $\mathcal{C}$ be a class of arithmetic circuits. Suppose $\mathsf{VPSPACE}_b^0 \subseteq \mathcal{C} \subseteq \mathsf{VP}^{\mathrm{poly}}$. Then $\mathsf{VNP}^0 \subseteq \mathcal{C}$ and $\mathsf{PSPACE/poly} = \mathsf{P/poly}$.*

Proof. First, by Lemma 5.2, $\mathsf{VNP}^0 \subseteq \mathsf{VPSPACE}_b^0 \subseteq \mathcal{C}$. We will show the second result using the polynomial family $\widetilde{\mathcal{TQBF}}$. By Corollary 4.2, $\widetilde{\mathcal{TQBF}} \in \mathsf{VPSPACE}_b^0 \subseteq \mathcal{C} \subseteq \mathsf{VP}^{\mathrm{poly}}$. By Lemma 4.5, this implies $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. Finally, from Lemma 2.7 it follows that $\mathsf{PSPACE/poly} \subseteq \mathsf{P/poly}$. Hence proved. □

To complete the picture, we provide an alternative proof for the forward (unconditional) direction of Proposition 5.3. As before, we prove a somewhat stronger statement. To that end, we require the following result which is implicit in the original proof of [KP09].

Lemma 5.5 (Implicit in Proposition 4.1 of [KP09])**.** *Suppose $\mathsf{PSPACE/poly} = \mathsf{P/poly}$. Then $\mathsf{VPSPACE}_b = \mathsf{VNP}$.*

We now prove the relevant claim. In fact, we a prove a somewhat stronger statement.

Lemma 5.6. *Let $\mathcal{C}$ be a class of arithmetic circuits. Suppose that $\mathsf{VNP} \subseteq \mathcal{C}$ and $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. Then $\mathsf{VPSPACE}_b \subseteq \mathcal{C}$.*

Proof. As $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$, by Lemma 2.7 we obtain that $\mathsf{PSPACE/poly} = \mathsf{P/poly}$. This, in turn, implies (by the above lemma) that $\mathsf{VPSPACE}_b = \mathsf{VNP}$. Subsequently, as $\mathsf{VNP} \subseteq \mathcal{C}$, the claim follows. □

5.2 Traits of $\mathsf{VPSPACE}_b$-completeness in $\mathcal{TP}$ - Proof of Theorem 2

In the following result, we note that if $\mathcal{TP}$ is contained in any subclass $\mathcal{C} \subseteq \mathsf{VNP}$ then so is the entire class $\mathsf{VPSPACE}_b$. For completeness, we restate Theorem 2.

Theorem 5.7 (Theorem 2 restated)**.** *For any circuit class $\mathcal{C} \subseteq \mathsf{VNP}$: $\mathcal{TP} \in \mathcal{C}$ if and only if $\mathsf{VPSPACE}_b \subseteq \mathcal{C}$.*

As a first step we show that $\widetilde{\mathcal{TQBF}}$ has the flavor of a complete family w.r.t to the containment of $\mathsf{VPSPACE}_b$ in VNP.

Theorem 5.8. *$\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$ if and only if $\mathsf{VPSPACE}_b = \mathsf{VNP}$.*

Remark: For simplicity of exposition we show the proof for uniform $\mathsf{VPSPACE}_b$ (Definition 2.17). The non-uniform case then follows from Lemma A.1.

Proof. As $\widetilde{\mathcal{TQBF}} \in \mathsf{VPSPACE}_b$, one direction is trivial. We now prove that if $\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$ then $\mathsf{VPSPACE}_b = \mathsf{VNP}$. As $\mathsf{VNP} \subseteq \mathsf{VPSPACE}_b$ (Lemma 5.2), it suffices to show that $\mathsf{VPSPACE}_b \subseteq \mathsf{VNP}$. Let $\mathbf{f} = \{f_n\} \subseteq \mathsf{VPSPACE}_b^0$. Then there exists a polynomial function $p(n)$ such that we can express the polynomial $f_n(\mathbf{x})$ as follows for every $n \in \mathbb{N}$:

$$f_n(\mathbf{x}) = \sum_{\mathbf{e} \in \{0,\ldots,p(n)\}^{p(n)}} \left(c_{\mathbf{f}}(n, \mathbf{e}, 0) \sum_{i=1}^{p(n)} 2^{i-1} \cdot c_{\mathbf{f}}(n, \mathbf{e}, i) \cdot \mathbf{x}^{\mathbf{e}} \right),$$

In addition, the coefficient function $c_{\mathbf{f}}(n, \mathbf{e}, i)$ is computable in $\mathsf{DSPACE}[p(n)]$. By Theorem 4.4, as the language $L_{\mathrm{TQBF}'}$ is PSPACE-complete, there exist $m = \mathrm{poly}(n)$ and a mapping φ_f computable by polynomial-sized Boolean formulas such that

$$c_{\mathbf{f}}(n, \mathbf{e}, i) = 1 \iff \widetilde{\mathrm{TQBF}'}_m(\varphi_f(n, \mathbf{e}, i)) = 1.$$

Furthermore, by the same theorem, for all $y \in \{0,1\}^{8m^3}$: $\widetilde{\mathrm{TQBF}'}_m(y) \in \{0,1\}$. Let $\widehat{\varphi}_f$ be the polynomial map with $8m^3$ outputs resulting from the arithmetization of the map φ_f. As φ_f is computable by a polynomial-size Boolean formula, it follows that $\widehat{\varphi}_f \in \mathsf{VP}$. We obtain:

$$f_n(\mathbf{x}) = \sum_{\mathbf{e}\in\{0,\ldots,p(n)\}^{p(n)}} \left(\widetilde{\mathrm{TQBF}'}_m(\widehat{\varphi}_f(n,\mathbf{e},0)) \sum_{i=1}^{p(n)} 2^{i-1} \cdot \widetilde{\mathrm{TQBF}'}_m(\widehat{\varphi}_f(n,\mathbf{e},i)) \cdot \mathbf{x}^{\mathbf{e}} \right),$$

Since $\widetilde{\mathcal{TQBF}} \in \mathsf{VNP}$, we have that $\widetilde{\mathcal{TQBF}'} \in \mathsf{VNP}$ as VNP is closed under projections. In addition, as $\widehat{\varphi}_f$ is a VP-mapping (i.e. each output coordinate of $\widehat{\varphi}_f$ is a polynomial in VP) the composition $\widetilde{\mathrm{TQBF}'}_m(\widehat{\varphi}_f(n,\mathbf{e},i)) \in \mathsf{VNP}$ and the same is true for the entire expression in the brackets. Finally, recall that VNP is closed under exponential summation. As a result, $\mathbf{f} \in \mathsf{VNP}$. The containment extends to $\mathsf{VPSPACE}_b$ since $\mathsf{VPSPACE}_b$ is itself a projection of $\mathsf{VPSPACE}_b^0$ (see Definition 2.20). This establishes that $\mathsf{VPSPACE}_b \subseteq \mathsf{VNP}$. □

We now move to the proof of Theorem 2.

Proof. As $\mathcal{TP} \in \mathsf{VPSPACE}_b$, one direction is trivial. Now, we prove the other direction. Let $\mathcal{C} \subseteq \mathsf{VNP}$ be arbitrary circuit class and suppose $\mathcal{TP} \in \mathcal{C}$. Since $\mathcal{TP} \in \mathsf{VPSPACE}_b$ (Claim 5.1) and $\mathcal{TP} = \widetilde{\mathcal{TQBF}} + \mathcal{PERM}$, we get that $\widetilde{\mathcal{TQBF}}, \mathcal{PERM} \in \mathcal{C}$ by restricting the variables of $\mathcal{TP}$ to $\mathbf{u} = 0$ and $\mathbf{z} = 0$, respectively. Consequently, by Theorem 5.8: $\mathsf{VPSPACE}_b = \mathsf{VNP}$ and since $\mathcal{PERM}$ is complete for VNP we obtain that

$$\mathsf{VPSPACE}_b = \mathsf{VNP} \subseteq \mathcal{C}$$

as required. □

5.3 Inconsistent Triad - Proof of Corollary 6

First, we restate Corollary 6 for completeness.

Corollary 5.9 (Corollary 6 restated)**.** *Let $\mathcal{C}$ be an arithmetic circuit class over a field $\mathbb{F}$. Then the following three assumptions cannot be simultaneously true.*

1. $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$
2. *The Permanent is computable by arithmetic circuits from $\mathcal{C}$ over $\mathbb{F}$.*
3. *There is a sub-polynomial space black-box* PIT *algorithm for $\mathcal{C}$.*

Proof. Suppose that all the three assumptions were true. As $\mathcal{PERM}$ is VNP-complete, we get that $\mathsf{VNP} \subseteq \mathcal{C}$. By Lemma 5.6, we have that $\mathsf{VPSPACE}_b \subseteq \mathcal{C}$. Yet, by Lemma 1.1, $\mathsf{VPSPACE}_b$ does not have polynomial size circuits from $\mathcal{C}$. □

6 Discussion & Open Questions

Our main focus in this work is $\mathsf{VPSPACE}_b$ which is the algebraic analogue of PSPACE in the bounded-degree regime. This class is relatively less explored in comparison to VP and VNP, the algebraic analogues of P and NP, respectively. We study several interesting properties of $\mathsf{VPSPACE}_b$ by analysing the structural richness of $\widetilde{\mathcal{TQBF}}$, a polynomial family in $\mathsf{VPSPACE}_b$, which was first introduced in [Car19]. We present a cleaner and simpler construction of this polynomial family. We also introduced another interesting polynomial family, called $\mathcal{TP}$, by merging $\mathcal{PERM}$ with $\widetilde{\mathcal{TQBF}}$. Although, $\mathcal{TP}$ 'plays a role' of a $\mathsf{VPSPACE}_b$-complete polynomial (see Theorem 2), we do not know whether it is $\mathsf{VPSPACE}_b$-complete under p-projections.

Open Question 1. Is $\mathcal{TP}$ $\mathsf{VPSPACE}_b$-complete under p-projections?

We have studied many properties of $\widetilde{\mathcal{TQBF}}$, but we do not know whether it 'captures' VNP. An affirmative answer to the following will help us replace $\mathcal{TP}$ with $\widetilde{\mathcal{TQBF}}$ everywhere.

Open Question 2. Is the polynomial family $\mathcal{PERM}$ a p-projection of $\widetilde{\mathcal{TQBF}}$?

It would be interesting to see if we can make all our results that hold for $\mathsf{VP}^{\mathrm{poly}}$ work for VP. We think that this could be done by applying the techniques of Koiran and Perifel used in the proof of Proposition 3 of [KP09], and assuming the GRH. This assumption is required in [KP09] to perform an interplay between Boolean and arithmetic circuits. Some other works pursuing connections between these two circuits are [Hru16, AGM17, OSTY25].

Open Question 3. Can we extend all our results for $\mathsf{VP}^{\mathrm{poly}}$ to VP without the GRH?

As a final remark, we observe that all existing Karp–Lipton style collapse results proceed in an "oblivious" manner: the assumed small circuit depends only on the input length, and not on the specific input itself! Consequently, these collapse results can in fact be strengthened to yield collapses into the corresponding oblivious versions of the complexity classes: that is, into ONP and OMA. Note that both of these classes are trivially contained in $\mathsf{P/poly}$ by hard-coding the witnesses. For more information on oblivious complexity classes see [CR06, FSW09, GM15, GLV24, HV26]. Hence, we can obtain the following extensions:

- Corollary 8: If $\widetilde{\mathcal{TQBF}} \in \mathcal{C}$ then $\mathsf{PSPACE} \subseteq \mathsf{ONP}^{\mathsf{PIT}_{\mathcal{C}}} \subseteq \mathsf{P/poly}$.
- [KI04]: If $\mathcal{PERM} \in \mathsf{VP}^{\mathrm{poly}}$ then $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{ONP}^{\mathsf{PIT}} \subseteq \mathsf{P/poly}$.
- [BFL91]: If $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ then $\mathsf{PSPACE} \subseteq \mathsf{OMA} \subseteq \mathsf{P/poly}$.

7 Acknowledgments

The authors would like to express their gratitude to Guillaume Malod and Nitin Saxena for many useful discussions. In particular, the authors would like to thank Guillaume Malod for discussing the work of [Poi08]. The author would like to thank the anonymous referees for their useful comments on the previous version of the paper. Part of this project by N. Gupta was done when he was a postdoctoral researcher at Boston College. He would like to sincerely thank Boston College for the financial and institutional support provided during this time.

References

[AB03] M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *JACM*, 50(4):429–443, 2003. 1

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 8, 18

[AGKS15] M. Agrawal, R. Gurjar, A. Korwar, and N. Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. 1

[AGM17] E. Allender, A. Gál, and I. Mertz. Dual VP classes. *Comput. Complex.*, 26(3):583–625, 2017. 23

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *LNCS*, pages 92–105, 2005. 1, 3

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004. 1

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988. 2

[ASSS16] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting sets, lower bounds for depth-d occur-k formulas and depth-3 transcendence degree-k circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016. 1, 3

[AvMV15] M. Anderson, D. van Melkebeek, and I. Volkovich. Derandomizing polynomial identity testing for multilinear constant-read formulae. *Computational Complexity*, 24(4):695–776, 2015. 1, 3

[BF91] L. Babai and L. Fortnow. Arithmetization: A new method in structural complexity theory. *Comput. Complex.*, 1:41–66, 1991. 12

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. 5, 7, 23

[BGKS22] V. Bhargava, A. Garg, N. Kayal, and C. Saha. Learning generalized depth three arithmetic circuits in the non-degenerate case. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 21:1–21:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 2

[BGNV26] P. Bisht, N. Gupta, P. Nimbhorkar, and I. Volkovich. Launching identity testing into (bounded) space. In *17th Latin American Theoretical Informatics Symposium - LATIN*, 2026. 3, 12

[BGV23] P. Bisht, N. Gupta, and I. Volkovich. Towards identity testing for sums of products of read-once and multilinear bounded-read formulae. In *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2023, December 18-20, 2023, IIIT Hyderabad, Telangana, India*, volume 284 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. Full version available at https://eccc.weizmann.ac.il/report/2023/109/. 1

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988. 2

[BSV21] V. Bhargava, S. Saraf, and I. Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 809–822. ACM, 2021. Full version at https://eccc.weizmann.ac.il/report/2021/045/. 2

[BSV25] V. Bhargava, S. Saraf, and I. Volkovich. Reconstruction of depth-4 multilinear circuits. *ACM Trans. Comput. Theory*, 17(3), 2025. 2

[BSV26] V. Bhargava, S. Saraf, and I. Volkovich. Linear independence, alternants and applications. *SIAM J. Comput.*, 55(1):28–64, 2026. 1

[Bür00] P. Bürgisser. Completeness and reduction in algebraic complexity theory. 7, 01 2000. 1, 4, 10, 18

[Bür24] P. Bürgisser. Completeness classes in algebraic complexity theory. *arXiv*, 2024. 9, 20

[Car19] M. L. Carmosino. *Connections Between Complexity Lower Bounds and Meta-Computational Upper Bounds*. PhD thesis, University Of California San Diego, 2019. 2, 7, 12, 22

[CGT24] P. Chatterjee, K. Gajjar, and A. Tengse. Monotone classes beyond vnp. *Theoretical Computer Science*, 1009:114689, 2024. 1

[CIKK15] M. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova. Tighter connections between derandomization and circuit lower bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, Princeton, NJ, USA, August 24-26, 2015*, LIPIcs, pages 645–658. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. 7

[CR06] V. T. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *STACS*, pages 230–241, 2006. 23

[CT23] P. Chatterjee and A. Tengse. On annihilators of explicit polynomial maps, 2023. Link - https://arxiv.org/abs/2309.07612. 4, 11

[DDS21] P. Dutta, P. Dwivedi, and N. Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 11:1–11:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1

[DG24] P. Dutta and S. Ghosh. Advances in polynomial identity testing, sigact news complexity theory column 121. *SIGACT News*, 55(2):53ï¿½88, 2024. 1

[DL78] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. 1, 11

[DMM+16] A. Durand, M. Mahajan, G. Malod, N. de Rugy-Altherre, and N. Saurabh. Homomorphism Polynomials Complete for VP. *Chicago Journal of Theoretical Computer Science*, 2016. 10

[FK09] L. Fortnow and A. R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009. 2

[FS12] M. A. Forbes and A. Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:115, 2012. 1

[FSS14] M. A. Forbes, R. Saptharishi, and A. Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Symposium on Theory of Computing, STOC*, pages 867–875, 2014. 1

[FSW09] L. Fortnow, R. Santhanam, and R. Williams. Fixed-polynomial size circuit bounds. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 19–26. IEEE Computer Society, 2009. 23

[GKL12] A. Gupta, N. Kayal, and S. V. Lokam. Reconstruction of depth-4 multilinear circuits with top fanin 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 625–642, 2012. Full version at https://eccc.weizmann.ac.il/report/2011/153. 2

[GKQ14] A. Gupta, N. Kayal, and Y. Qiao. Random arithmetic formulas can be reconstructed efficiently. *Computational Complexity*, 23(2):207–303, 2014. 2

[GKS17] R. Gurjar, A. Korwar, and N. Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017. 1, 3

[GKS20] A. Garg, N. Kayal, and C. Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. *arXiv preprint arXiv:2004.06898*, 2020. 2

[GLV24] K. Gajulapalli, Z. Li, and I. Volkovich. Oblivious complexity classes revisited: Lower bounds and hierarchies. In *44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2024*, volume 323 of *LIPIcs*, pages 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 23

[GM15] O. Goldreich and O. Meir. Input-oblivious proof systems and a uniform complexity perspective on P/poly. *ACM Transactions on Computation Theory (TOCT)*, 7(4):1–13, 2015. 23

[GS19] N. Gupta and C. Saha. On the symmetries of and equivalence test for design polynomials. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019*, volume 138 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 5, 19

[GST23] N. Gupta, C. Saha, and B. Thankey. Equivalence test for read-once arithmetic formulas. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4205–4272. SIAM, 2023. 2

[Hru16] P. Hrubeš. On hardness of multilinearization and vnp-completeness in characteristic 2. *ACM Trans. Comput. Theory*, 9(1):1:1–1:14, 2016. 23

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pages 262–272, 1980. 1, 3

[HV26] E. Hirsch and I. Volkovich. Upper and Lower Bounds for the Linear Ordering Principle. In Meena Mahajan, Florin Manea, Annabelle McIver, and Nguy˜ên Kim Th?ng, editors, *43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026)*, volume 364 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:19, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 23

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 1, 4, 7, 8, 23

[KNST17] N. Kayal, V. Nair, C. Saha, and S. Tavenas. Reconstruction of full rank algebraic branching programs. In *32nd Computational Complexity Conference, CCC 2017.*, pages 21:1–21:61, 2017. 2

[KP09] P. Koiran and S. Perifel. VPSPACE and a transfer theorem over the reals. *Comput. Complex.*, 18(4):551–575, 2009. 1, 3, 4, 6, 10, 11, 18, 20, 21, 23

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001. 1, 2

[KS06] A. Klivans and A. Shpilka. Learning restricted models of arithmetic circuits. *Theory of computing*, 2(10):185–206, 2006. 2

[KS19] N. Kayal and C. Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019.*, pages 413–424, 2019. 2

[KUW86] R. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random nc. *Combinatorica*, 6:35–48, 1 1986. 1

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proofs. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 1–10, 1990. 7, 12

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *JACM*, 39(4):859–868, 1992. 13

[Lip89] R. J. Lipton. New directions in testing. In *Distributed Computing And Cryptography*, 1989. 5, 19

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979. 1

[LST24] N. Limaye, S. Srinivasan, and S. Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. *Commun. ACM*, 67(2):101–108, 2024. 1

[Mal03] G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claud Bernard Lyon 1, 2003. 1, 4, 9, 10, 14, 31

[Mal07] G. Malod. The complexity of polynomials and their coefficient functions. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 193–204, 2007. 1, 10

[Mal11] G. Malod. Succinct algebraic branching programs characterizing non-uniform complexity classes. FCT'11, page 205 216. Springer-Verlag, 2011. 3, 4, 7, 11

[MR13] M. Mahajan and B. V. R. Rao. Small space analogues of valiant s classes and the limitations of skew formulas. *computational complexity*, 22:1–38, 03 2013. 4, 11

[MRS16] M. Mahajan, B.V.R. Rao, and K. Sreenivasaiah. Building above read-once polynomials: Identity testing and hardness of representation. *Algorithmica*, 76:890–909, 2016. 1

[MST16] M. Mahajan, N. Saurabh, and S. Tavenas. VNP=VP in the multilinear world. *Information Processing Letters*, 116(2):179–182, 2016. 1

[Mul10] K. Mulmuley. Explicit proofs and the flip. *ArXiv*, abs/1009.0246, 2010. 5, 19

[MV18] D. Minahan and I. Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *ACM Transactions on Computation Theory (TOCT)*, 10(3):10:1–10:11, 2018. 1, 3

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. 1

[OSTY25] I. Orzel, S. Srinivasan, S Tavenas, and A. Yehudayoff. The algebraic cost of a boolean sum. *CoRR*, abs/2502.02442, 2025. 23

[Poi08] B. Poizat. A la recherche de la definition de la complexite d'espace pour le calcul des polynomes a la maniere de valiant. *J. Symb. Log.*, 73(4):1179–1201, 2008. 4, 7, 11, 23

[PS21] S. Peleg and A. Shpilka. Polynomial time deterministic identity testing algorithm for $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits via edelstein-kelly type theorem for quadratic polynomials. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 259–271. ACM, 2021. 1

[Sap15] R. Saptharishi. A survey of lower bounds in arithmetic circuit complexity. 2015. 10

[Sax09] N. Saxena. Progress on polynomial identity testing. *Bulletin of EATCS*, 99:49–79, 2009. 1

[Sax14] N. Saxena. Progress on polynomial identity testing - II. *CoRR*, abs/1401.0976, 2014. 1

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. 1, 11

[Sha92] A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. 7, 12, 13

[Shp09] A. Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. on Computing*, 38(6):2130–2161, 2009. 2

[Sin16] G. Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 31:1–31:53, 2016. 2

[Sin22] G. Sinha. Efficient reconstruction of depth three arithmetic circuits with top fan-in two. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 118:1–118:33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 2

[Sri20] S. Srinivasan. Strongly exponential separation between monotone VP and monotone VNP. *ACM Trans. Comput. Theory*, 12(4), sep 2020. 1

[SS12] N. Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012. 1

[SS13] N. Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33, 2013. 1

[SS25] S. Saraf and D. Shringi. Reconstruction of depth 3 arithmetic circuits with top fan-in 3. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference, CCC 2025, Toronto, Canada, August 5-8, 2025*, volume 339 of *LIPIcs*, pages 21:1–21:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. 2

[SS26] N. Varadarajan S. Saraf, D. Shringi. Reconstruction of depth-3 arithmetic circuits with constant top fan-in . *Electronic Colloquium on Computational Complexity (ECCC)*, (222), 2026. 2

[ST17] O. Svensson and J. Tarnawski. The matching problem in general graphs is in quasi-NC. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 696–707, 2017. 1

[SV15] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015. 1

[SV18] S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. *Combinatorica*, 38(5):1205–1238, 2018. 1

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. 1, 2, 3, 10

[TV07] L. Trevisan and S. P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. 2, 7, 12, 13

[Val79] L. G. Valiant. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 249–261, 1979. 1, 9, 10

[Val82] L. G. Valiant. Reducibility by algebraic projections. *L'Enseignement mathematique*, 28:253 –268, 1982. 10

[Vol16] I. Volkovich. A guide to learning arithmetic circuits. In *Proceedings of the 29th Conference on Learning Theory, (COLT)*, pages 1540–1561, 2016. 2

[Vol17] I. Volkovich. On some computations on sparse polynomials. In *APPROX-RANDOM*, pages 48:1–4:21, 2017. 2

[vzG93] J. von zur Gathen. Parallel linear algebra. *Synthesis of parallel algorithms*, pages 574–617, 1993. 3

[Yeh19] A. Yehudayoff. Separating monotone VP and VNP. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 425ï¿½429, New York, NY, USA, 2019. Association for Computing Machinery. 1

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979. 1, 11

A Missing proofs from Section 2

A.1 Proof of Lemma 2.7

First, we remark that the definition of the non-uniform version of DSPACE (Definition 2.5) does not specify the behaviour of M when it is executed on a "wrong" (or arbitrary) advice. The next lemma, shows that we can assume, with out loss of generality, that M will halt using the same asymptotic amount of space given **any** advice.

Lemma A.1. *Let $s : \mathbb{N} \to \mathbb{N}$ be a constructable function. Let $L \in \mathsf{DSPACE}[s(n)]/s(n)$ and $\alpha_n \in \{0,1\}^{s(n)}$ be the associated advice. Then there exist a language $L' \in \mathsf{DSPACE}[s(n)]$ such that $L = \left\{ w \;\middle|\; (w, \alpha_{|w|}) \in L' \right\}$.*

Proof. Let $M(w, y)$ be the corresponding Turing machine for L. Let us define a new Turing machine $M'(w, y)$:

- Simulate M on (w, y).
- If M attempts to make more than $2^{O(s(|w|))}$ steps or use more than $O(s(|w|))$ space, halt and reject.
- Otherwise, output $M(w, y)$

Let L' be the language of M'. First of all, observe that M halts for any input. Next, we claim that $L' \in \mathsf{DSPACE}[s(n)]$. By definition, the simulation of M takes at most $O(s(|w|))$ space. Additional $O(s(|w|))$ space is required to keep track of the number of steps and memory cells used by M. Now, consider an arbitrary input w. By the definition, when M is executed on $(w, \alpha_{|w|})$ it will halt and use at most $O(s(|w|))$ space. Therefore, it will halt within $2^{O(s(|w|))}$ steps. Given that, $M'(w, \alpha_{|w|}) = M(w, \alpha_{|w|})$. Consequently:

$$w \in L \iff M(w, \alpha_{|w|}) = 1 \iff M'(w, \alpha_{|w|}) = 1 \iff (w, \alpha_{|w|}) \in L'.$$

□

We will now use the above Lemma to prove Lemma 2.7.

Lemma A.2 (Lemma 2.7 restated)**.** $\mathsf{PSPACE/poly} \subseteq \mathsf{P/poly}$ *if and only if* $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$.

Proof. As $\mathsf{PSPACE} \subseteq \mathsf{PSPACE/poly}$, one direction of the lemma is trivial. Suppose $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. We want to show that $\mathsf{PSPACE/poly} \subseteq \mathsf{P/poly}$. Let $L \in \mathsf{PSPACE/poly}$. By definition, there exists a polynomial function $s(n)$ such that $L \in \mathsf{DSPACE}[s(n)]/s(n)$. Let $L' \in \mathsf{DSPACE}[s(n)] \subseteq \mathsf{PSPACE}$ be the language guaranteed by Lemma A.1. From the assumption, $L' \in \mathsf{P/poly}$ and therefore there exists a polynomial-time deterministic Turing machine $A\left((w, y), z\right)$ and advice strings $\{\beta_n\}_{n \in \mathbb{N}}$ such that for any w, y:

$$(w, y) \in L' \iff A\left((w, y), \beta_{|(w,y)|}\right) = 1.$$

Let us now define a new Turing machine $B\left(w, (y, z)\right) = A\left((w, y), z\right)$ and $\gamma_n \triangleq (\alpha_n, \beta_{|(1^n, \alpha_n)|})$. By definition, B is a polynomial-time Turing machine. We claim that it decides the language L, given the advice γ_n:

$$w \in L \iff (w, \alpha_{|w|}) \in L' \iff A\left((w, \alpha_{|w|}), \beta_{|(w,\alpha_{|w|})|}\right) = 1 \iff B\left(w, (\alpha_{|w|}, \beta_{|(w,\alpha_{|w|})|})\right) = 1 \iff$$
$$B\left(w, (\alpha_{|w|}, \beta_{|(1^{|w|}, \alpha_{|w|})|})\right) = 1 \iff B(w, \gamma_{|w|}) = 1$$

□

B Missing proofs from Section 3

B.1 Proof of Observation 3.2

Observation 3.2. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate, degree d-polynomial and $z \in \mathbf{x}$ be arbitrary.*

1. *$\forall_z f, \exists_z f$ are polynomials in $\mathbb{F}[\mathbf{x} \setminus \{z\}]$ and $\deg(\forall_z f) \leq 2d$, $\deg(\exists_z f) \leq 2d$.*
2. *The polynomial $\Lambda_z f$ is linear in z, i.e., $\deg_z(\Lambda_z f) \leq 1$.*
3. *Let $X := \{x_1, \ldots, x_k\} \subseteq \mathbf{x}$ be arbitrary. Then, the operator Λ_X is $\mathbb{F}$-linear over $\mathbb{F}[\mathbf{x}]$.*

Proof. The first two points immediately follow from Definition 3.1. We prove that Λ_X is $\mathbb{F}$-linear by induction on the size of X, i.e., k. Let $k = 0$. Then there is nothing to prove and the base case holds. Suppose the proposition holds for $k-1$. Let $X' := \{x_2, \ldots, x_k\}$. Then by induction hypothesis, for every $f, g \in \mathbb{F}[\mathbf{x}]$ and $\alpha, \beta \in \mathbb{F}, \Lambda_{X'}(\alpha f + \beta g) = \alpha\Lambda_{X'} f + \beta\Lambda_{X'} g$. We want to prove $\Lambda_X(\alpha f + \beta g) = \alpha\Lambda_X f + \beta\Lambda_X g$. We know from Definition 3.1 that $\Lambda_X = \Lambda_{x_1}\Lambda_{X'}$. By the induction hypothesis, $\Lambda_X(\alpha f + \beta g) = \Lambda_{x_1}(\alpha\Lambda_{X'} f + \beta\Lambda_{X'} g)$. On applying the definition of Λ_{x_1} (Definition 3.1), we get

$$\Lambda_X(\alpha f + \beta g) = x_1 \times (\alpha\Lambda_{X'} f + \beta\Lambda_{X'} g)\,|_{x_1=1} + (1 - x_1) \times (\alpha\Lambda_{X'} f + \beta\Lambda_{X'} g)\,|_{x_1=0}.$$

For any polynomials $p, q \in \mathbb{F}[\mathbf{x}], (p+q)|_{x_i=\gamma} = p|_{x_i=\gamma} + q|_{x_i=\gamma}$, where $\gamma \in \mathbb{F}$. This immediately implies that the above equation can be rearranged as follows.

$$\Lambda_X(\alpha f + \beta g) = \alpha\left(x_1 \times \Lambda_{X'} f|_{x_1=1} + (1 - x_1) \times \Lambda_{X'} f|_{x_1=0}\right) + \beta\left(x_1 \times \Lambda_{X'} g|_{x_1=1} + (1 - x_1) \times \Lambda_{X'} g|_{x_1=0}\right).$$

Hence, we get $\Lambda_X(\alpha f + \beta g) = \alpha\Lambda_X f + \beta\Lambda_X g$. This proves that Λ_X is $\mathbb{F}$-linear over the vector space $\mathbb{F}[\mathbf{x}]$. □

B.2 Proof of Proposition 3.3

Proposition 3.3. *Let $f \in \mathbb{F}[\mathbf{x}]$ and $X := \{x_1, \ldots, x_k\} \subseteq \mathbf{x}$ be an arbitrary set. Then for every permutations σ, τ on $[k]$, we have $\Lambda_{x_{\sigma(1)}}\Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}} f = \Lambda_{x_{\tau(1)}}\Lambda_{x_{\tau(2)}} \cdots \Lambda_{x_{\tau(k)}} f$. In other words, $\Lambda_X f = \Lambda_{x_{\sigma(1)}}\Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}} f$ for every permutation σ on $[k]$.*

Proof. Let σ be an arbitrary permutation on $[k]$. We have noted in Observation 3.2 that the composition operator $\Lambda_{x_{\sigma(1)}}\Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}}$ is $\mathbb{F}$-linear. Thus, to prove the observation, it is sufficient to assume that f is a monomial. For simplicity, let $\mathbf{x} = \{x_1, \ldots, x_k, x_{k+1}, \ldots, x_n\}$. Then, f looks as $f = x_1^{e_1} \cdots x_n^{e_n}$, where $e_1, \ldots, e_n \in \mathbb{N}$. For every $i \in [k]$, it follows from Definition 3.1 that if $e_i = 0$ then $\Lambda_{x_i} f = f$, otherwise $\Lambda_{x_i} f = x_1^{e_1} \cdots x_{i-1}^{e_{i-1}} \cdot x_i \cdot x_{i+1}^{e_{i+1}} \cdots x_n^{e_n}$. Let A be the set of all variables in $\{x_1, \ldots, x_k\}$ appearing in f. Then, it follows from the above discussion that for every permutation σ on $[k]$,

$$\Lambda_{x_{\sigma(1)}}\Lambda_{x_{\sigma(2)}} \cdots \Lambda_{x_{\sigma(k)}} f = \prod_{x \in A} x \cdot x_{k+1}^{e_{k+1}} \cdots x_n^{e_n}.$$

This proves the desired result. □

B.3 Proof of Proposition 3.4

Proposition 3.4. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial and $z \in \mathbf{x}$ be an arbitrary variable. Then for every $\mathbf{a} \in \{0,1\}^n$, $f|_{\mathbf{x}=\mathbf{a}} = \Lambda_z f|_{\mathbf{x}=\mathbf{a}}$.* [22]

Proof. Let $\mathbf{a} \in \{0,1\}^n$ be arbitrary. We want to show that $f(\mathbf{a}) = \Lambda_z f(\mathbf{a})$. Since Λ_z is $\mathbb{F}$-linear on $\mathbb{F}[\mathbf{x}]$ (Observation 3.2), it suffices to assume without loss of generality that f is a monomial. For the sake of discussion, let $\mathbf{x} = \{x_1, \ldots, x_{n-1}, z\}$. Then $f = z^{e_z} \prod_{i \in [n-1]} x_i^{e_i}$, where $e_1, \ldots, e_{n-1}, e_z \in \mathbb{N}$. Suppose $e_z = 0$. Then it follows from Definition 3.1 that $\Lambda_z f = f$ and there is nothing to prove. Now, assume that $e_z \neq 0$. Then Definition 3.1 implies that $\Lambda_z f = z \prod_{i \in [n-1]} x_i^{e_i}$. Let $\mathbf{a} = (a_1, \ldots, a_{n-1}, a_z) \in \{0,1\}^n$. Then $f|_{\mathbf{x}=\mathbf{a}} = a_z^{e_z} \prod_{i \in [n-1]} a_i^{e_i}$ and $\Lambda_z f|_{\mathbf{x}=\mathbf{a}} = a_z \prod_{i \in [n-1]} a_i^{e_i}$. Since $a_z \in \{0,1\}$ and $e_z \neq 0$, we get $a_z = a_z^{e_z}$, which implies $f|_{\mathbf{x}=\mathbf{a}} = \Lambda_z f|_{\mathbf{x}=\mathbf{a}}$. □

B.4 Proof of Proposition 3.5

Proposition 3.5. *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial and $X \subseteq \mathbf{x}$. Then for $\mathbf{a} \in \{0,1\}^n$, $f|_{\mathbf{x}=\mathbf{a}} = \Lambda_X f|_{\mathbf{x}=\mathbf{a}}$.*

Proof. We prove this by induction on $k := |X|$. Let $k = 0$. Then there is nothing to prove and the base case holds. Suppose the proposition holds for $k-1$. Without loss of generality, let $X = \{x_1, \ldots, x_k\}$. Then we know that $\Lambda_X f = \Lambda_{x_1}(\Lambda_{x_2} \cdots \Lambda_{x_k} f)$. Let $\mathbf{a} \in \{0,1\}^n$ be arbitrary. Then from the induction hypothesis, $\Lambda_{x_2} \cdots \Lambda_{x_k} f|_{\mathbf{x}=\mathbf{a}} = f|_{\mathbf{x}=\mathbf{a}}$. Now, it follows from proposition 3.4 that $\Lambda_X f|_{\mathbf{x}=\mathbf{a}} = f|_{\mathbf{x}=\mathbf{a}}$. □

[22] The notation $\Lambda_z f|_{\mathbf{x}=\mathbf{a}}$ means that we first apply the operator Λ_z to f and then substitute $\mathbf{x}$ with $\mathbf{a}$.

B.5 Proof of Observation 3.7

Observation 3.7. *The Boolean formula $\Phi_n(\mathbf{x},\mathbf{y})$ defined in Equation (2) is a universal 3-CNF. Furthermore, given an n-variate 3-CNF $\varphi(\mathbf{x})$, an assignment $\mathbf{b}_\varphi \in \{0,1\}^{8n^3}$ satisfying $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x},\mathbf{b}_\varphi)$ can be computed in time $O(n^3)$.*

Proof. Let $\mathbf{x} = \{x_1,\ldots,x_n\}$. For every $\mathbf{i} := (i_1,i_2,i_3) \in [n]^3$ and $\mathbf{e} := (e_1,e_2,e_3) \in \{0,1\}^3$, check if the clause $(x_{i_1}^{[e_1]} \vee x_{i_2}^{[e_2]} \vee x_{i_3}^{[e_3]})$ is present in φ or not. If yes, set the coordinate $b_{\mathbf{i},\mathbf{e}} = 1$, otherwise set it as 0. Let $\mathbf{b}_\varphi := (b_{\mathbf{i},\mathbf{e}})_{\{\mathbf{i}\in[n]^3,\mathbf{e}\in\{0,1\}^3\}}$. Clearly, $\varphi(\mathbf{x}) = \Phi_n(\mathbf{x},\mathbf{b}_\varphi)$ and $\mathbf{b}_\varphi$ can be computed in $O(n^3)$ time. □

B.6 Proof of Proposition 3.9

Proposition 3.9. *The polynomial family $\{\widehat{\Phi}_n(\mathbf{x},\mathbf{y})\}$ is in $\mathsf{VP}^{\mathrm{poly}}$, has $\{0,1,-1\}$ constants [23] and hence in $\mathsf{Uniform\,VPSPACE}^0$ (Definition 2.17), where $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ is the polynomial defined in Equation (3).*

Proof. Observe that it follows from Equation (3) that $\widehat{\Phi}_n(\mathbf{x},\mathbf{y})$ is computed by a constant-depth arithmetic circuit C_n of polynomial size and the only coefficients appearing in C_n are 0 and 1. We claim that the description of C_n can be computed in $\mathrm{poly}(n)$ time [24]. Let us pick $\mathbf{i} \in [n]^3$ and $\mathbf{e} \in \{0,1\}^3$ arbitrarily. Note that the circuit for the term $T_{\mathbf{i},\mathbf{e}} := \left[1 - (x_{i_1} - e_1)(x_{i_2} - e_2)(x_{i_3} - e_3) \cdot y_{\mathbf{i},\mathbf{e}}\right]$ mentioned in Equation (3) can be constructed in $\mathrm{poly}(n)$ time. Now we go over all $\mathbf{i} \in [n]^3, \mathbf{e} \in \{0,1\}^3$ and compute the circuits of all $T_{\mathbf{i},\mathbf{e}}$ in $\mathrm{poly}(n)$ time. After that we attach all these circuits to a product gate, which can also be done in polynomial time. Clearly, this product gate computes $\widehat{\Phi}_n(\mathbf{x},\mathbf{e})$. This implies that the family $\{\widehat{\Phi}_n(\mathbf{x},\mathbf{y})\}$ is in $\mathsf{Uniform\,VPAR}^0$ (Definition 2.18). It follows from Proposition 2.19 that $\{\widehat{\Phi}_n(\mathbf{x},\mathbf{y})\} \in \mathsf{Uniform\,VPSPACE}^0$. □

B.7 Proof of proposition 3.10

Proposition 3.10. *Let $\varphi_n(\mathbf{x})$ be an n-variate 3-CNF and $f(\mathbf{x})$ be a polynomial satisfying that for every $\mathbf{a} \in \{0,1\}^n, \varphi(\mathbf{a}) = f(\mathbf{a})$. Let $x \in \mathbf{x}$ be an arbitrary variable and $\mathbf{x}' := \mathbf{x} \setminus \{x\}$. Then the following properties hold for every $\mathbf{a}' \in \{0,1\}^{n-1}$.*
1. $\forall x\, \varphi|_{\mathbf{x}'=\mathbf{a}'} = \forall_x f|_{\mathbf{x}'=\mathbf{a}'}$. *2.* $\exists x\, \varphi|_{\mathbf{x}'=\mathbf{a}'} = \exists_x f|_{\mathbf{x}'=\mathbf{a}'}$.

Proof. Let $\mathbf{a}' \in \{0,1\}^{n-1}$ be an arbitrary assignment. By definition, $\forall x\, \varphi|_{\mathbf{x}'=\mathbf{a}'} = 1$ if and only if $\varphi|_{x=0,\mathbf{x}'=\mathbf{a}'} \wedge \varphi|_{x=1,\mathbf{x}'=\mathbf{a}'} = 1$. We know for every $a \in \{0,1\}$, $\varphi|_{x=a,\mathbf{x}'=\mathbf{a}'} = f|_{x=a,\mathbf{x}'=\mathbf{a}'}$. Thus, we get $\forall x\, \varphi|_{\mathbf{x}'=\mathbf{a}'} = 1$ if and only if $(f|_{x=0} \times f|_{x=1})|_{\mathbf{x}'=\mathbf{a}} = 1$. By Definition 3.1, we get that $\forall x\, \varphi|_{\mathbf{x}'=\mathbf{a}'} = \forall_x f|_{\mathbf{x}'=\mathbf{a}'}$. Similarly, using DeMorgan's law and rules of arithmetization, we can prove the second part. □

B.8 Proof of Observation 3.12

Observation 3.12. *The number of variables in $\widetilde{\mathrm{TQBF}}_n$ is $n + 8n^3 + 8n^4 + \binom{n-1}{2}$.*

Proof. There are three pairwise disjoint sets of variables in the definition of $\widetilde{\mathrm{TQBF}}_n$, namely $\mathbf{x},\mathbf{y}$ and $\mathbf{z}$. We know $|\mathbf{x}| = n$ and $|\mathbf{y}| = m = 8n^3$. The number of variables in $\mathbf{z}$ is equal to the number of $f_{i,j}$ polynomials. It is easy to verify from the table given above that $|\mathbf{z}| = \binom{n-1}{2} + nm = \binom{n-1}{2} + 8n^4$. □

[23] Infact, such a sub-class of $\mathsf{VP}^{\mathrm{poly}}$ is called VP^0 or constant-free VP, which was defined by Malod [Mal03].

[24] Recall that an arithmetic circuit is a directed acyclic graph. By saying that we can compute the description of a circuit C we mean that we are computing the underlying graph of C.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il