Lower Bounds from Succinct Hitting Sets

Prerona Chatterjee [*] Anamay Tengse [†]

Abstract

We investigate the consequences of the existence of "efficiently describable" hitting sets for polynomial sized algebraic circuit (VP), in particular, *VP-succinct hitting sets*. Existence of such hitting sets is known to be equivalent to a "natural-proofs-barrier" towards algebraic circuit lower bounds, from the works that introduced this concept [FSV18, GKSS17]. We show that the existence of VP-succinct hitting sets for VP would either imply that $\mathsf{VP} \neq \mathsf{VNP}$, or yield a fairly strong lower bound against TC^0 circuits, assuming the Generalized Riemann Hypothesis (GRH).

This result is a consequence of showing that designing efficiently describable (VP-explicit) hitting set generators for a class $\mathcal{C}$, is essentially the same as proving a separation between $\mathcal{C}$ and VPSPACE: the algebraic analogue of PSPACE. More formally, we prove an upper bound on *equations* for polynomial sized algebraic circuits (VP), in terms of VPSPACE.

Using the same upper bound, we also show that even *sub-polynomially explicit hitting sets* for VP — much weaker than VP-succinct hitting sets that are almost polylog-explicit — would imply that either $\mathsf{VP} \neq \mathsf{VNP}$ or that $\mathsf{P} \neq \mathsf{PSPACE}$. This motivates us to define the concept of *cryptographic hitting sets*, which we believe is interesting on its own.

[*]Department of Mathematics, IIT Madras. Parts of this work were supported by the Azrieli International Postdoctoral Fellowship, the Israel Science Foundation (grant number 514/20), the Len Blavatnik and the Blavatnik Family foundation, and a fellowship of the DAE, India for TIFR Mumbai. Email: prerona.ch@gmail.com.

[†]School of Computer Sciences, NISER, Bhubaneswar. Parts of this work were supported by the Israel Science Foundation (grant No. 716/20 and grant No. 843/23), and a fellowship of the DAE, India for TIFR Mumbai. Email: anamay.tengse@gmail.com.

1 Introduction

Proving lower bounds against boolean circuits is extremely hard. This is not news to anyone in the area, and perhaps, coming to terms with this reality is a rite of passage for all complexity theorists. More concretely, the famous work of Razborov and Rudich [RR97] rules out most "natural" strategies for proving lower bounds against any boolean circuit class that is rich enough to do strong-enough cryptography.

Every boolean function can be seen as a multilinear polynomial, and it is therefore quite natural to try and understand the complexity of computing these polynomials *syntactically* (using sum and product gates). Computing a function syntactically can only be a harder task, since any such implementation must compute the same function. This suggests that proving algebraic circuit lower bounds should only be easier.

Take the case of the #SAT function, whose syntactic counterpart is the Permanent polynomial [Val79b]. The latter captures the algebraic counterpart of NP, called VNP [Val79a], and it is known that showing VP $=$ VNP ("algebraic P $=$ NP") would immediately give efficient boolean circuits for SAT, barring some unlikely issues due to field constants [Bür00]. It can further be shown that any algebraic circuit computing a degree d polynomial does not benefit much from having intermediate computations of degree more than d [Str73]. Algebraic circuit complexity therefore focusses on studying the complexity of syntactically computing *low-degree* polynomials using algebraic operations of sums and products as gates. The reader may refer to some comprehensive surveys [SY10, CKW11, Sap15] for an overview of the area.

Despite this supposed advantage towards proving hardness, almost all interesting algebraic circuit classes have been doing a pretty good impression of their boolean counterparts as they continue to resist some promising attacks (like shifted partials: [AV08, GKKS14, FLMS15], set-multilinearization: [BDS24, CKSS24]). As a result, the one-line summary of the best known lower bounds does not read much better than the boolean world: $\Omega(n \log n)$ for circuits [BS83, Smo97], $\Omega(n^2)$ for ABPs [CKSV22], $\Omega(n^2)$ for formulas [Kal85, CKSV22].

Algebraic Natural Proofs

This state of affairs has inspired studies in search of a formal barrier towards proving strong algebraic circuit lower bounds using the conventional methods, along the lines of the aforementioned work of Razborov and Rudich [RR97]. The analogous notion that captures almost all known algebraic circuit lower bounds, is the framework of *algebraic(ally) natural proofs*. This framework was jointly proposed in the works of Forbes, Shpilka and Volk [FSV18], and Grochow, Kumar, Saks and Saraf [GKSS17] achieves this well[1]. This framework is obtained essentially by "algebraizing" the framework of natural proofs from [RR97].

Intuitively, say we wish to prove a lower bound of s on the size of any circuit computing a polynomial f. This can be viewed as a task of proving non-membership in a set: the set $\mathcal{C}$ that contains all polynomials computable with size less than s, and we want to show $f \notin \mathcal{C}$. What would help here is a polynomial D that can certify this fact. In particular, if D were to evaluate to zero on all members of $\mathcal{C}$, and to some nonzero value on f, then D would act as a proof of $f \notin \mathcal{C}$. Further, for D to be a non-trivial proof, its complexity needs to be significantly better than just enumerating over all circuits of size less than s. It turns out that most known lower-bounds

[1] These build on the works of Aaronson and Drucker [AD08], and Grochow [Gro15].

against algebraic models yield such "distinguishers" D that are themselves computable efficiently, and hence they form the basis for *algebraic natural proofs*.

More formally, a $\mathcal{D}$-natural proof for $\mathcal{C}$ is a polynomial $P \in \mathcal{D}$ that vanishes on the coefficient vectors of all polynomial in $\mathcal{C}$ (see Section 2.1 for details). Motivated from algebraic geometry, sometimes such polynomials are also called *equations for* $\mathcal{C}$. Stated in these terms, almost every known algebraic circuit lower bound against a class $\mathcal{C}$ gives a VP-natural proof for $\mathcal{C}$. Here VP is the class of polynomials[2] whose degree and circuit size grows polynomially with their arity. Thus, asking whether natural techniques could prove a super-polynomial algebraic circuit lower bound, translates to asking: does VP have any VP-natural proofs?

In this context, a work of Kumar, Ramya, Saptharishi and Tengse [KRST22] shows that if the Permanent is exponentially hard[3], then there are no VP-natural proofs for VNP. In particular, as the Permanent is widely believed to be exponentially hard, if one believes that VP has VP-natural proofs, then they should expect VP $\neq$ VNP to be *provable using natural techniques*. A natural question is therefore, what are the consequences of VP *not admitting* any VP-natural proofs?

Succinct Hitting Sets

Suppose that the polynomials in $\mathcal{C}$ (their coefficient vectors, rather) form a *hitting set* for the class VP: low-degree polynomials with efficient circuits. Just by definition, this means that no polynomial in VP can vanish over the entire class $\mathcal{C}$. Stated simply, this would mean that the above *natural* lower bound strategy is ineffective for the class $\mathcal{C}$.

Specifically, if $\mathcal{C}$ were to be the class of n-variate, multilinear polynomials with circuits of size $\text{poly}(n)$, then the coefficient vectors have length $N = 2^n$. If $\mathcal{C}$ forms a hitting set for $\text{VP}(N)$, then it means that $\text{VP}(N)$ has a hitting set that has an extremely succinct description: as the set of coefficient vectors of circuits of size about $\text{poly}(n) = \text{poly}(\log N)$. Such a hitting set is therefore said to be a *C-succinct hitting set* for VP. As argued above, the existence of VP-succinct hitting sets for VP *is equivalent to* the non-existence of VP-natural proofs for VP. This is the core observation in the works of Forbes, Shpilka and Volk [FSV18], and Grochow, Kumar, Saks and Saraf [GKSS17] (see Section 2.1 for formal statements).

Therefore, VP does not have VP-natural proofs *if and only if* algebraic circuits of size and degree $\text{poly}(N)$ have hitting sets that can be described by circuits of size about $\text{poly}(\log N)$. It is not hard to see that non-trivial hitting sets for a class almost immediately yield polynomials that hard for that class (see, for example, [HS80]). So what sort of hardness results do we get from these $\text{polylog}(N)$-succinct hitting sets? We show that existence of such hitting sets would imply that either VP $\neq$ VNP or that NC^1 is separate from TC^0 in a very strong sense (Theorem 1.2).

Cryptographic Hitting Sets. On our way to proving this statement, we encounter an object that is weaker than ($\text{polylog}(N)$) succinct hitting sets, but which we strongly believe is equally interesting on its own. We call these objects *cryptographic hitting sets*.

Informally, we say that a hitting set H for a class C is *cryptographic*, if H can be generated or described by a circuit in a class C', where C' is seemingly weaker than C. For example, if C is the class of algebraic circuits of size s, then the class of algebraic formulas of size $\text{poly}(s)$, or even the class of circuits of size $t = s^{o(1)}$ would be valid choices for C'. We define this concept formally as

[2] Technically, the classes VP, VNP contain polynomial families. We ignore this distinction for now for simplicity.

[3] Here exponential hardness means 2^{n^ε}-hardness for some constant $\varepsilon > 0$.

Definition 1.8. The term *cryptographic* here is chosen because this setting of parameters — where a weaker class is required to fool a stronger class — is prevalent in cryptography.

We then show an exciting consequence of cryptographic hitting set generators, that have a super-polynomial "stretch". In particular, we prove that for the correct setting of parameters, if VP-cryptographic HSGs exist for VP then either VP $\neq$ VNP or $\mathsf{NC}^1 \not\subseteq \mathsf{TC}^0$ (Theorem 1.9).

Note that we do not even know of a well-founded hardness assumption which implies the existence of cryptographic hitting sets for algebraic circuits. The only example is a construction due to Kayal [Kay09], whose setting of parameters is rather unwieldy for algebraic circuit complexity. The work gives a polynomial map $\mathcal{M} : \mathbb{F}^N \to \mathbb{F}^{N+1}$ that can be described by constant depth formulas, such that any polynomial P that vanishes on it: $P \circ \mathcal{M} \equiv 0$, has degree $2^{\Omega(N)}$. The map is therefore a *hitting set generator* for N-variate polynomials of degree $\mathrm{poly}(N)$. This setting of parameters is unwieldy for two reasons: (1) the "stretch" of the generator is linear (can be made $N \to N + N^{1+\varepsilon}$ for any $\varepsilon > 0$), and (2) we would ideally like a map that is hard to annihilate, *despite* the existence of low-degree annihilators; this comes from the motivation to study "syntactic" algebraic computation, as outlined earlier.

1.1 Our contributions

The main contribution of this paper are some interesting connections between lower bounds, identity testing and algebraic natural proofs.

Lower Bounds from Succinct Hitting Set Generators

Before stating our theorem, let us recall that a family of polynomial maps $\left\{H_n : \mathbb{C}^n \to \mathbb{C}^{N(n)}\right\}$ is a hitting set generator for a class of N-variate polynomials of degree-$D(N)$, if any family $\left\{A_N : A_N \in \mathbb{F}[x_1, \ldots, x_{N(n)}]\right\}$ that annihilates $\{H_n\}$ is outside the class.

Succinct hitting set generators [FSV18, GKSS17] are then defined as follows .

Definition 1.1 ($\mathcal{C}$-Succinct HSG for $\mathcal{D}$)**.** *For functions $d, D : \mathbb{N} \to \mathbb{N}$, let*

- *$\mathcal{C}_d$ be a class of n-variate polynomial families of degree $d(n)$, and*
- *$\mathcal{D}_D$ be a class of N-variate polynomial families of degree $D(N)$ with $N = \binom{n+d(n)}{n}$.*

We say that $\left\{H_n : \mathbb{C}^n \to \mathbb{C}^N\right\}$ is a family of $\mathcal{C}_d$-succinct hitting set generators for $\mathcal{D}_D$, if the following are true.

1. *$\{H_n\} \in \mathcal{C}_d$.*
2. *For infinitely many $n \in \mathbb{N}$, H_n is a hitting set generator for $\mathcal{D}_D(N)$.* $\Diamond$

Our main result shows that the existence of VP-succinct hitting sets for VP (or equivalently, the existence of a barrier in proving VP $\neq$ VNP via *natural techniques*) imply strong lower bounds.

Theorem 1.2 (Hardness from Succinct Hitting Sets)**.** *Assuming the Generalized Riemann Hypothesis, if* VP*-succinct hitting set generators exist for* VP*, then at least one of the following must be true.*

1. VP $\neq$ VNP.

2. *For any $\ell(m) = o(1)$, there is a family of functions $\{h_m\}$ in uniform NC^1 such that any uniform constant-depth threshold circuit computing it must have size larger than $\exp(\exp(\log^{\ell(m)} m))$, where the uniformity is* DLOGTIME *(in terms of the respective sizes).*

Further, if VP *does not admit* VNP*-natural proofs then* $\mathsf{P} \neq \mathsf{SPACE}(\log^{\log^*(n)}(n))$.

Remark 1.3. *A couple of remarks about Theorem 1.2 are in order.*

1. *Comparing this result with that of Kabanets and Impagliazzo [KI04], the (second) conclusion here is seemingly much stronger than* $\mathsf{NEXP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$. *This is to be expected, given that the hypothesis here demands a much more explicit derandomization of identity testing.*

2. *Note that any function on n variables in uniform NC^1 is trivially computable by uniform constant-depth threshold circuits of size $\exp(\mathrm{poly}(n)) = \exp(\exp(O(\log n)))$; the statement in Theorem 1.2 says that only a polynomial improvement in the second exponent is possible over this trivial bound. For comparison, $\exp(\exp(\log^{\Omega(1)} n))$ grows faster than a "half-exponential" function[4].* ◊

Theorem 1.2 essentially follows from the following upper bound on equations for evaluation vectors (similar to truth tables) of $\mathsf{VP}(n)$.

Theorem 1.4 (Equations for VP)**.** *For an arbitrary $d(n) \in \mathrm{poly}(n)$, let $N(n) = \binom{n+d(n)}{n}$. Then, for $t(n) := n^{\log^* n}$, there is a family of $N(n)$-variate, multilinear polynomials $\{P_N\}$ that depends only on the first $t(n)$ variables, satisfying the following.*

- *The family $\{P_N\}$ is a family of equations for* the evaluation vectors *of $\mathsf{VP}_{d(n)}$.*
- *The coefficient functions of $\{P_N\}$ are computable in (uniform) space $t(n) = n^{\log^* n}$.*

Remark 1.5. *We make a few remarks about Theorem 1.4.*

1. *A $\mathsf{VP}(N)$[5] upper bound on the equations of $\mathsf{VP}(n)$ would rule out a barrier in proving that* VP $\neq$ VNP *via natural proof strategies. The bound we show is incomparable to $\mathsf{VP}(N)$.*

2. *The function $\log^*(n)$ can be replaced by any growing function in n.*

3. *Even though 'algebraic natural proofs' are defined in terms of coefficient vectors instead of evaluation vectors, these notions are more or less equivalent. This is formalized as follows.*

   Proposition 1.6 (Informal version of Proposition 4.6)**.** *For any $d(n)$ and $N = \binom{n+d}{n}$, equations for* coefficient vectors *of VP_d are computable in $\mathsf{VP}(N)$ (or $\mathsf{VNP}(N)$), if and only if, equations for* evaluation vectors *of VP_d are computable in $\mathsf{VP}(N)$ (or $\mathsf{VNP}(N)$).* ◊

Defining Cryptographic Hitting Set Generators

As mentioned earlier, our proof for Theorem 1.2 goes via studying objects that are seemingly much weaker than succinct hitting sets. We now define them formally.

Firstly, we define *explicit polynomial maps* as follows.

Definition 1.7. *For a family $\{H_n : \mathbb{C}^n \to \mathbb{C}^N\}$ of maps and a class $\mathcal{C}$, we say that a circuit $C(\mathbf{z}, \mathbf{y})$* encodes *$H$ if there are assignments $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ to $\mathbf{y}$, such that for each $i \in [N]$, $C(\mathbf{x}, \mathbf{a}_i) = h_i(\mathbf{x})$.*

Analogously, we say that $\{H_n\}$ is $\mathcal{C}$-explicit *if there is a family $\{C_n\} \in \mathcal{C}$ such that, for every n, C_n encodes H_n.* ◊

[4] The half-exponential function $h(n)$ is defined so that $h(h(n)) = 2^n$. For example, $2^{n^\varepsilon} \gg h(n) \gg \mathrm{quasipoly}(n)$.

[5] Here $\mathcal{C}(n)$ and $\mathcal{D}(N)$ mean that the complexities are polynomial in the parameters n and $N \approx 2^n$, respectively.

Cryptographic HSGs We are now ready to define cryptographic hitting set generators.

Recall that a polynomial map $H(x_1, \ldots, x_n) = (h_1(\mathbf{x}), \ldots, h_N(\mathbf{x}))$ is said to have degree d if $h_i(\mathbf{x})$ has degree at most d for every $i \in [N]$.

Definition 1.8 ($\mathcal{C}$-Cryptographic HSG for $\mathcal{D}$)**.** *For functions $d, D : \mathbb{N} \to \mathbb{N}$, let*

- *$\mathcal{C}_d$ be a class of n-variate polynomial families of degree $d(n)$, and*
- *$\mathcal{D}_D$ be a class of N-variate polynomial families of degree $D(N)$.*

Further, let $\left\{H_n : \mathbb{C}^n \to \mathbb{C}^{N(n)}\right\}$ be a family of polynomial maps with $N(n) > n$. We say that $\{H_n\}$ is a family of $\mathcal{C}_d$-cryptographic hitting set generators for $\mathcal{D}_D$, if the following are true.

1. *$\mathcal{C}_d \subsetneq \mathcal{D}_D$.*
2. *$\{H_n\}$ is $\mathcal{C}_d$-explicit.*
3. *For infinitely many $n \in \mathbb{N}$, H_n is a hitting set generator for $\mathcal{D}_D(N)$.* ◊

Lower Bounds from Cryptographic HSGs

We focus on VP-explicit families of polynomial maps $\{H_n\}$, where the degree and size of encoding circuits increases polynomially in the number of parameters n, and show that hitting set generators for algebraic circuits that have a super-polynomial *stretch* would have similar (but slightly weaker) consequences as the ones in Theorem 1.2.

Theorem 1.9 (Lower Bounds from Cryptographic HSGs)**.** *Let $\left\{H_n : \mathbb{C}^n \to \mathbb{C}^N\right\}$ be a family of polynomial maps of degree $d = n^8$ with $N \geq 2n$, and let $D(N) = N^{10}$.*

Assuming the Generalized Riemann Hypothesis, if the family $\{H_n\}$ is a VP_d-cryptographic hitting set generator for VP_D, at least one of the following must be true.

1. *$\mathsf{VP} \neq \mathsf{VNP}$.*
2. *Uniform $\mathsf{NC}^1 \not\subseteq$ uniform TC^0, where the uniformity is $\mathsf{DLOGTIME}$.*

Further, if the family $\{H_n\}$ is a hitting set generator for VNP_d, then $\mathsf{P} \neq \mathsf{PSPACE}$.

Remark 1.10. *In Theorem 1.9, by "$\{H_n\}$ is a VP_d-cryptographic hitting set generator for VP_D" we mean that there is a fixed constant e such that $\{H_n\}$ can be encoded using a size n^e circuit, but $\{H_n\}$ hits circuits of size N^a* for all *constants a. This is what makes $\{H_n\}$ a cryptographic HSG.* ◊

Annihilators of explicit polynomials maps

All of our results follow from an upper bound that we are able to show on annihilators of explicit polynomial maps. The formal statement requires the notion of special evaluation gates, called *projection gates* (see Definition 2.19), and is stated as Theorem 3.1.

Projection gates characterize a class called VPSPACE in the same way as (usual) algebraic circuits characterize VP. A detailed discussion on this class can be found in Section 2.2. But over any fixed finite field, in particular, it coincides with the class of polynomials whose bits of coefficients can be computed in non-uniform PSPACE. In particular, a slightly weaker version of Theorem 1.4 can be restated using this vocabulary as follows.

Theorem 1.11 (Upper bound for annihilators of VP)**.** *Let $d(m)$ be an arbitrary polynomial function of m. For any VP_d-explicit family of maps $\{\mathcal{G}_m\}$, where each $\mathcal{G}_m$ has n outputs with $n \geq 2m$, there is a family $\{A_m\}$ in $\mathsf{VPSPACE}_b$ of degree $O(m^2 d)$ such that A_m annihilates $\mathcal{G}_m$ for all large enough m.*

In fact, the upper bound in this statement holds even when the encoding circuit uses projection gates (see Theorem 3.6). That is, VPSPACE even annihilates (low-degree) VPSPACE-explicit maps[6]. Using this, the hypothesis of Theorem 1.9 can also be weakened to a family of VNP or even $\mathsf{VPSPACE}_b$-explicit HSGs (see Theorem 4.1). It is therefore believable that Theorem 1.4 can be improved; we expand on this in Section 1.3. That said, as we have already seen, it is strong enough to have some very interesting consequences (Theorem 1.2, Theorem 1.9).

1.2 Proof Overview

The main ideas behind our proofs are a combination of elementary techniques, and we will therefore try to give a nearly complete outline of our proofs.

Upper bound on the annihilators

We begin with an overview of the proof of Theorem 1.11 and then discuss how Theorem 1.4 follows from it.

Let $\mathcal{G} = (g_1(\mathbf{z}), \ldots, g_n(\mathbf{z}))$ be an arbitrary tuple of degree-d polynomials in $m < n/2$ variables, that is VP_d-explicit. The explicitness of $\mathbf{g}$ will be used later in the proof.

Observe that an annihilator $A(x_1, \ldots, x_n) \in \mathbb{C}[\mathbf{x}]$ of *individual* degree $(D-1)$ for this tuple is precisely a (non-trivial) $\mathbb{C}$-linear dependency between products of $g_1, \ldots, g_n$ of total degree at most $n(D-1)$, where each g_i is multiplied at most $(D-1)$ times. Now, any such product is itself a polynomial in m variables of individual degree less than $d' := (nD \cdot d)$, and hence belongs to a vector space of dimension less than d'^m over $\mathbb{C}$. On the other hand, there are at least D^n of these products, so if D is large enough to ensure $D^n \geq d'^m = (ndD)^m$ then we are guaranteed some non-trivial linear dependency. For $n \geq 2m$, this happens for a $D \leq nd$.

Now we turn to finding the coefficients of this dependency, which are going to be the coefficients of our annihilator. For this, consider a matrix M with rows labelled by monomials in $\mathbf{z}$ of individual degree at most $(d'-1)$, and columns labelled by monomials in $\mathbf{x}$ of individual degree at most $(D-1)$. Order both the rows and columns lexicographically. For a monomial $\mathbf{x}^{\mathbf{e}} = x_1^{e_1} \cdots x_n^{e_n}$, the $\mathbf{e}^{th}$ column of M is the coefficient vector of $\mathbf{g}^{\mathbf{e}}$. In this language, any column-dependency of M is the coefficient vector of an annihilator of $(g_1, \ldots, g_n)$.

Since we are chasing an annihilator with as efficient a description as possible, it makes sense to pick the "lexicographically first" one. That is, let $\mathbf{e}$ be the "smallest" exponent such that $\mathbf{g}^{\mathbf{e}}$ is dependent on the set of $\mathbf{g}^{\mathbf{e}'}$s where all $\mathbf{e}'$s are "smaller than" $\mathbf{e}$. Our annihilator will precisely be this unique dependency (up to scaling by $\mathbb{C}$).

Given a linear system of equations, $Ax = 0$, with a unique (up to scaling) solution, *Cramer's rule* describes the solution in terms of determinants of submatrices of A. We will fix the 'last coefficient' to be $(-\det(A'))$, where A' is the submatrix that has all but the last column. This is done to ensure that all the coefficients are integers (and minors of A). In fact, it turns out that stated this way, the annihilator itself can be written as a *single* determinant just by inserting all the

[6]This can be phrased as 'VPSPACE is closed under taking annihilators', in the same way as 'VP is closed under taking homogenous components' and 'VNP is closed under taking arbitrary coefficients'.

(symbolic) monomials $\mathbf{x}^{\mathbf{e}}$ as the (new) last row! We will call this 'special Cramer's rule' for the remainder of this overview.

At this point, note that there is a minor issue. For all we know, the "first" dependency that we have chosen gives us a submatrix (all columns up to $\mathbf{e}$) that has more rows than columns, and hence we cannot apply Cramer's rule. Further, it is not even clear if the basis of rows has a small description (smaller than just its characteristic vector). We get around this by using a rank extractor, in particular the construction used by Forbes and Shpilka [FS12] (see Lemma 2.8), because it fits snugly with the structure of the matrix M. The arguments up to this point have been formalized in Lemma 3.2.

Now that we are in the setting of 'special Cramer's rule': we have an $(r+1) \times (r+1)$ matrix $\widetilde{M}$ whose determinant is the annihilator. Since $r \leq (md)^{O(m)}$ this structure is not enough on its own to guarantee a non-trivial upper bound. But we have even more structure in $\widetilde{M}$, owing to the explicitness of $\mathbf{g}$. Specifically we have that, for all $i \leq r$, the $(i, \mathbf{e}_j)^{th}$ entry of $\widetilde{M}$ is the evaluation of $\mathbf{g}^{\mathbf{e}_j}$ at a point, $\beta^{(i-1)} \in \mathbb{Z}^m$ and this can be calculated efficiently by a constant-free algebraic circuit when i is given in its binary representation. Since computing a circuit for $\mathbf{g}^{\mathbf{e}_j}$ given the binary encoding of $\mathbf{e}_j$ is also doable efficiently, we say that the matrix $\widetilde{M}$ is efficiently encodable using algebraic circuits. More formally, there is a size $\mathrm{poly}(\log r) = \mathrm{poly}(m, d, s)$ algebraic circuit which, when given bit-vectors corresponding to $i, j \in [r+1]$, outputs $\widetilde{M}[i, j]$. Note that this output can sometimes be a monomial in $\mathbf{x}$. The formal argument can be found in the proof of Lemma 3.5.

Finally, we use the fact that the determinant of an explicit matrix can itself be computed in a "memory-efficient" way. More formally, we use the elegant construction due to Mahajan and Vinay [MV97], of an *algebraic branching program* (see Definition 2.3) for the determinant polynomial, along with repeated squaring using *projection gates* (see Definition 2.19). Here, the fact that projection gates let us evaluate a polynomial at constant cost proves to be crucial. Quantitatively, given any matrix that is encoded by a circuit of size s', we can compute the determinant of this matrix using a circuit with projection gates of size $\mathrm{poly}(s')$. In fact, Malod [Mal11] uses this to give an alternative characterization of VPSPACE. This argument has been formalized in Proposition 2.30.

Putting all the pieces together (see Figure 1), we get a circuit with projection gates, of size $\mathrm{poly}(m, d, s)$ that computes the annihilator of the map $\mathcal{G} = (g_1, \ldots, g_n)$ as required. A complete proof of Theorem 1.11 is given in Section 3.

The observation that allows us to move from Theorem 1.11 to Theorem 1.4 is that we can work with *evaluation vectors* instead of *coefficient vectors*, especially when dealing with VP-natural or VNP-natural proofs (due to Proposition 1.6). This means that a family of equations can be constructed to vanish on the evaluation vector of the *universal circuit* of a slightly super-polynomial size, say $n^{O(\log^* n)}$. Clearly, this evaluation vector is a size$(n^{O(\log^* n)})$-explicit map. Applying Theorem 1.11, we get a family of equations that can be computed by circuits with projection gates, of size $n^{O(\log^* n)}$. Finally applying the characterization of VPSPACE in terms of the computability of its coefficients (Definition 2.18) gives us the required statement. A formal proof of Theorem 1.4 can be found in Section 4.2.

Remark 1.12. *Perhaps the first approach that one would take for proving Theorem 1.4 would be to avoid using the notion of* VPSPACE*. A sketch of this alternate proof is as follows.*

- *Given an $N \times N$ matrix with integer entries of bit-complexity b, there is an algorithm that uses space at most $\mathrm{poly}(\log N, b)$, and simulates bit-access to its determinant [Ber84, Csa76].*
- *It can be shown that the definition of "bit-complexity of the entries" in the above result can be taken*

as "space complexity of an algorithm that provides bit-access to the entries".

- *This when combined with the use of explicit rank-extractors [GR05], which makes the matrix A have rank that is one less than full as well as have entries that have small "bit-complexity", essentially finishes the proof with an application of Cramer's Rule.*

We believe that this alternate proof is relatively less clean, as it requires multiple switches between non-uniform algebraic computation and uniform boolean computation when one tries to formalize it fully. For instance, in the second step, the bound on the space complexity of the entries only holds when the map $\mathcal{G}$ itself is encoded by a constant-free algebraic circuit*, and showing that requires using the definition of* VP *as a* projection *of the class*[7] VP^0.

In comparison, our proof only uses non-uniform algebraic computation with the use of projection gates *and then uses the characterization of* $\mathsf{VPSPACE}^0$ *as (families of) polynomials whose coefficients are computable in non-uniform* PSPACE *(see Definition 2.18) only once at the the end.* ◊

Lower bounds from Hitting Set Generators

Cryptographic HSGs An overview of how Theorem 1.9 follows from Theorem 1.11 is the following. Suppose, for contradiction, that VP = VNP and uniform $\mathsf{NC}^1 \subseteq$ uniform TC^0. Further, let us assume that the Generalized Reimann Hypothesis is true.

It can be shown, using a padding argument (see, for example, Complexity Zoo), that if uniform $\mathsf{NC}^1 \subseteq$ uniform TC^0 then the counting hierarchy (CH) is the same as PSPACE. This would then imply that $\mathsf{CH}_{/\mathsf{poly}} = \mathsf{PSPACE}_{/\mathsf{poly}}$. On the other hand, since VP = VNP, $\mathsf{CH}_{/\mathsf{poly}} = \mathsf{P}_{/\mathsf{poly}}$. Thus, we get that $\mathsf{P}_{/\mathsf{poly}} = \mathsf{PSPACE}_{/\mathsf{poly}}$, which would imply that $\mathsf{VNP} = \mathsf{VPSPACE}_b$ due to Valiant's Criterion and the definition of $\mathsf{VPSPACE}_b$. Reusing the assumption that VP = VNP, we get that $\mathsf{VP} = \mathsf{VPSPACE}_b$. For formal statements, see Proposition 2.26 and Proposition 2.27.

This means that any circuit with *projection gates* can be efficiently simulated by a circuit without *projection gates*. In particular, this implies that the circuits which compute the family of annihilators of the hitting set generator $\{\mathcal{H}_m\}$ can be converted to "usual" algebraic circuits of polynomially larger size. Finally, since the degree function $d(n)$ is large enough for the annihilators ensured by Theorem 1.11, we get that there is annihilator for $\{\mathcal{H}_m\}$ in VP_d. This contradicts its 'hitting property' in the hypothesis, thus completing the proof.

To prove the second statement, we assume for contradiction that P = PSPACE. Thus, $\mathsf{P}_{/\mathsf{poly}} = \mathsf{PSPACE}_{/\mathsf{poly}}$ and therefore it must that $\mathsf{VNP} = \mathsf{VPSPACE}_b$. An argument similar to the previous case then completes the proof. Theorem 1.9 is proved formally in Section 4.1.

Succinct HSGs The only difference between Succinct HSGs and Cryptographic HSGs is the complexity of the circuit that encodes the generator. Since the parameters are much better in the case of succinct HSGs — barely super-polylogarithmic vs sub-polynomial in the number of outputs — it allows us to extract much better lower bounds against constant-depth theshold circuits. A formal proof of Theorem 1.2 is given in Section 4.3.

[7] The class of polynomial families that are computable efficiently by constant-free algebraic circuits.

1.3 Open directions

- An intriguing takeaway from our results, is the case that it makes for studying 'cryptographic HSGs' as in Theorem 1.9 and the more general Theorem 4.1. Specifically, we ask whether the arguments of Heintz and Schnorr [HS80] can be derandomized within space that is sub-polynomial in the parameters n, d and s. As this would satisfy the hypothesis of Theorem 4.1, achieving this even under a hardness assumption that is weaker than (or incomparable to) $\mathsf{P} \neq \mathsf{PSPACE}$ would be interesting.

  Along the same lines, under what hardness assumptions do 'sub-exponentially strong cryptographic HSGs' corresponding to Theorem 1.2 exist? There are some caveats[8] due to which boolean cryptographic primitives do not directly translate to HSGs. In particular, and specifically in the context of natural proofs, what would be an appropriate analogue for the work of Håstad, Impagliazzo, Levin and Luby [HILL99]?

- Another interesting thread is to improve the upper bound in this paper on annihilators of explicit polynomial maps (Theorem 1.11). Since we expect $\mathsf{VPSPACE}_b$ to be exponentially stronger than VP (and VNP), any bound that is better than exponential, say e.g. $2^{o(s(n))}$ for size-$s(n)$-explicit maps, in terms of $\mathsf{VP}(2^n)$ (or $\mathsf{VNP}(2^n)$) would be interesting, even if it requires assuming some standard hypotheses.

  Annihilators from the 'minimal dependency', as in Lemma 3.2, are perhaps unlikely to imply such a bound without observing some structure of VP. This is mainly because such an approach gives coefficients that are determinants of matrices of dimension $2^{\mathrm{poly}(s)}$ and one has to argue that all these coefficients can somehow be "compressed" into a smaller circuit.

  Note that Theorem 3.6 implies that the bound from Theorem 1.4 extends to equations for VNP. So, can we prove a better bound just for VP? Certainly, deriving any property of VP that does not hold for VNP should probably assume hardness of VNP (e.g. [KRST22]).

1.4 Organisation of the paper

The rest of our paper is organized as follows.

We first formalize some relevant concepts in Section 2. This section also includes some simple propositions (see Section 2.2, Section 2.4) that will be important for the proof of our main theorems. We then prove the VPSPACE upper bound on annihilators for VP (Theorem 1.11) in Section 3 and our main theorems (Theorem 1.9, Theorem 1.4, Theorem 1.2) in Section 4. For completeness, in Appendix A, we include a somewhat detailed proof sketch of the equivalence of the two definitions of VPSPACE that we use.

[8] This discussion is beyond the scope of this paper. We encourage the reader to refer to the literature on algebraic natural proofs (e.g. [AD08],[FSV18]) for some details.

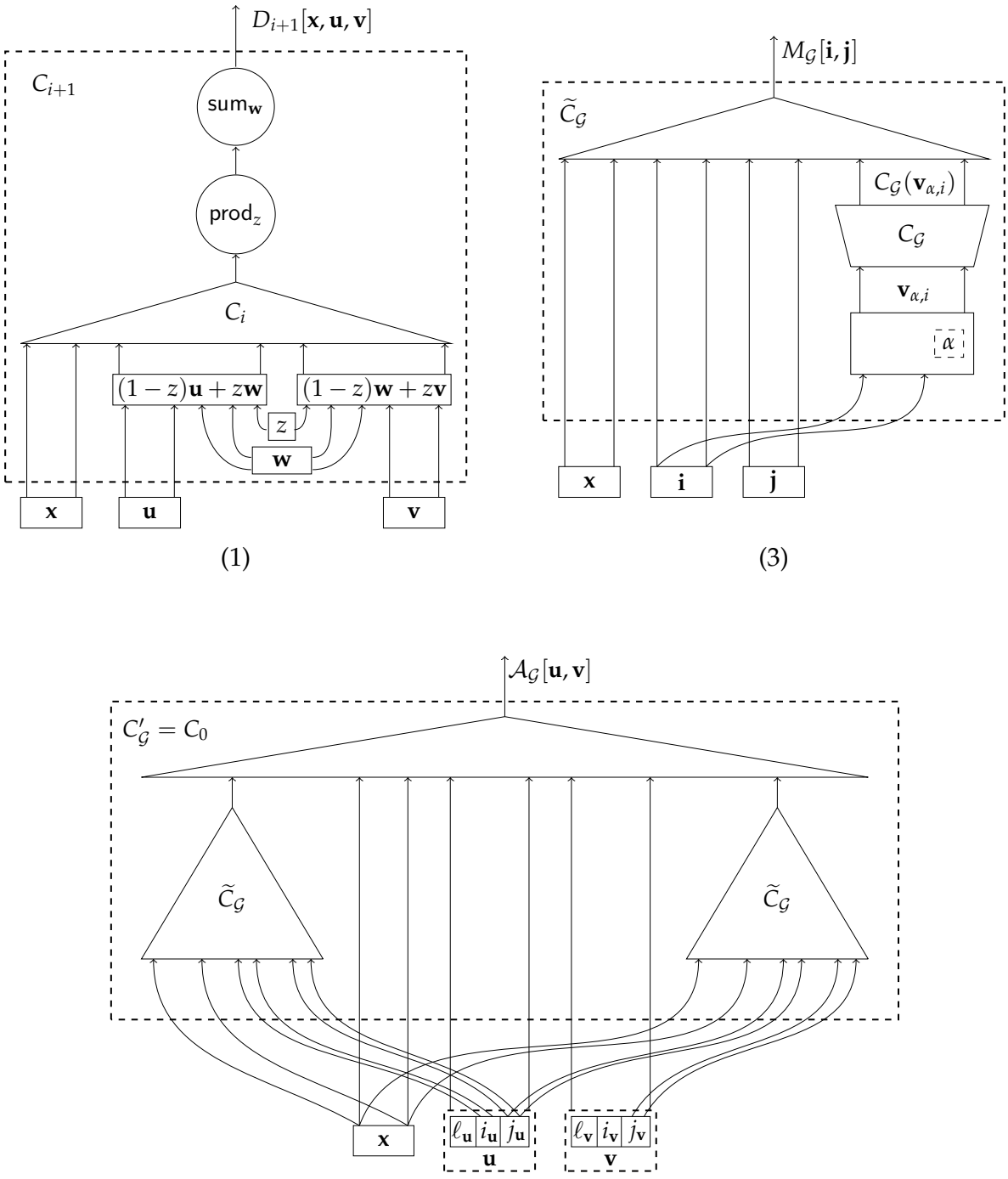

Figure 1: The annihilator is computed by using (1) repeatedly to get $C_{\log N}$, if N is the length of the ABP described by (2) (Claim 2.32); (2) describes the ABP computing $\det(\widetilde{M})$ using (3) (Claim 2.31); (3) describes the matrix $\widetilde{M}$, such that $\det(\widetilde{M}) \circ \mathcal{G} \equiv 0$, in terms of the generator $\mathcal{G}$ (Lemma 3.5).

2 Preliminaries

- We use $[n]$ to denote the set $\{1, \ldots, n\}$.
- We use boldface letters such as $\mathbf{x}, \mathbf{y}$ to denote tuples, typically of variables. When necessary, we adorn them with a subscript such as $\mathbf{y}_{[n]}$ to denote the length of the tuple. We also use $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $\prod x_i^{e_i}$.
- We use $\{f_n\}_{n \in \mathbb{N}}$ to denote families of polynomials. We drop the index set whenever it is clear from context. For a given polynomial f we denote by $\deg(f)$ its degree. For a polynomial $f(\mathbf{x}, \mathbf{y}, \ldots)$ on multiple sets of variables, we use $\deg_{\mathbf{x}}(f)$, $\deg_{\mathbf{y}}(f)$, etc., to denote the degree in the variables from the respective sets.
- For a given n-variate polynomial $f(\mathbf{x})$ of degree d, and a monomial m, we use $\text{coeff}_m(f)$ to refer to the coefficient of m in f. We further use $\overline{\text{coeff}}(f)$ to denote the vector[9] of coefficients of f.
- For a matrix M, we use $M[i, j]$ to refer to its $(i, j)^{th}$ entry. We commonly start our indices from 1, but sometimes start them from 0 when it helps the exposition. Whenever this is done, we make it clear.

Models of Algebraic Computation

We now formally define the models of algebraic computation that we need for this work.

We first define algebraic circuits and their constant-free version.

Definition 2.1 (Algebraic Circuits)**.** *An* algebraic circuit *is specified by a directed acyclic graph, with leaves (nodes with in-degree zero, called* inputs*) labelled by field constants or variables, and internal nodes labelled by $+$ or $\times$. The nodes with out-degree zero are called the* outputs *of the circuit. Computation proceeds in the natural way, where inductively each $+$ gate computes the sum of its children and each $\times$ gate computes the product of its children.*

The size *of the circuit is defined as the number of edges (or* wires*) in the underlying graph.* ◊

Definition 2.2 (Constant-free Algebraic Circuits)**.** *An algebraic circuit is said to be "constant-free", if the only field constants that appear in it are 1 and -1. In order that the circuit is able to compute rational numbers, we allow it to have division gates, where both the inputs are field constants.* ◊

We also formally define algebraic branching programs.

Definition 2.3 (Algebraic Branching Programs (ABPs))**.** *An* algebraic branching program *is specified by a layered graph where each edge is labelled by an affine linear form and the first and the last layer have one vertex each, called the "source" and the "sink" vertex respectively. The polynomial computed by an ABP is equal to the sum of the weights of all paths from the start vertex to the end vertex in the ABP, where the weight of a path is equal to the product of the labels of all the edges on it.*

The width of a layer in an ABP is the number of vertices in it and the width of an ABP is the width of the layer that has the maximum number of vertices in it.

The size of an ABP is the number of edges in it. ◊

[9] We do not explicitly mention the monomial ordering used for this vector representation, since all our statements work for any monomial ordering.

Remark 2.4. *We will use ABPs to mean ABPs that have been defined as above, but with edge labels also being allowed to be monomials of degree that is at most the size of the ABP.* ◊

Algebraic Complexity Classes

We now define the basic algebraic complexity classes.

Definition 2.5 (VP)**.** *A family $\{f_n\}_n$ of polynomials is said to be in* VP*, if there exists a constant $c \in \mathbb{N}$ such that for all large n, f_n depends on at most n^c variables, has degree at most n^c, and is computable by an algebraic circuit of size at most n^c.* ◊

Definition 2.6 (VNP)**.** *A family $\{f_n\}_n$ of polynomials is said to be in* VNP*, if there exists a constant $c \in \mathbb{N}$, and an m-variate family $\{g_m\} \in$ VP with $m, \text{size}(g_m) \leq n^c$, such that for all large enough n, f_n satisfies the following.*

$$f_n(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^{|\mathbf{y}|}} g_m(\mathbf{x}, \mathbf{y} = \mathbf{a})$$ ◊

We will also be needing the following subclass of VP, especially in the context of algebraic natural proofs.

Definition 2.7 (VP_d)**.** *For a function $d(n) \in \text{poly}(n)$, we define VP_d to be the class of all degree-d families that belong to* VP*. Equivalently, a family $\{f_n\} \in$ VP is said to be in VP_d if for all large enough n, $\deg(f_n) \leq d(n)$.*

We also define VNP_d to contain all the degree-d families from VNP*.* ◊

Rank Extractor We will also need the concept of rank extractors. A rank extractor is a family of maps which have rank preserving properties. The following lemma shows the existence of such a family in the setting that we will need.

Lemma 2.8 ([FS12])**.** *Let $1 \leq r \leq n$ and let $M \in \mathbb{C}^{n \times r}$ be of rank r. Define $A_\alpha \in \mathbb{C}^{r \times n}$ by $A_\alpha[i,j] = \alpha^{ij}$. Then there exists $\alpha \in [nr]$ such that $\text{rank}(A_\alpha \cdot M) = r$.* □

Polynomials for boolean operations Finally we note that there are computationally simple polynomials which capture certain boolean operations. Since this is easy to check, we omit the proof.

Observation 2.9. *Given vectors $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$, the following polynomials (or vectors of polynomials) have constant-free circuits of size $O(\ell^2)$.*

- $\text{EQ}(\mathbf{a}, \mathbf{b})$*: outputs* 1 *if* $\mathbf{a} = \mathbf{b}$*,* 0 *otherwise.*
- $\text{GT}(\mathbf{a}, \mathbf{b})$*: outputs* 1 *if* $\mathbf{a} > \mathbf{b}$ *when seen as binary encodings of positive integers,* 0 *otherwise.*
- $\text{LT}(\mathbf{a}, \mathbf{b})$*: outputs* 1 *if* $\mathbf{a} < \mathbf{b}$ *when seen as binary encodings of positive integers,* 0 *otherwise.*
- $\text{INC}(\mathbf{a})$*: outputs a vector* $\mathbf{b}$ *such that* $\mathbf{b} = \mathbf{a} + 1$*, when seen as positive integers.* □

2.1 Universal Circuits and Natural Proofs

A circuit is said to be universal for circuits of size s if every such circuit is a simple projection of it.

Definition 2.10 (Universal Circuit [SY10])**.** *A circuit $\mathcal{U}$ is called universal for n-input circuits of size s, that compute polynomials of degree d, if the following holds.*

For any polynomial $f(x_1, \ldots, x_n)$ of degree d that can be computed by a circuit of size s, there exists a circuit φ computing f as well, such that the computation graph of $\mathcal{U}$ is the same as the graph of φ. ◊

The following lemma due to Raz [Raz10] shows the existence of such circuits. For a proof sketch that yields the exact statement given below, please refer to [CKR+20].

Lemma 2.11 (Existence of Universal Circuits [Raz10])**.** *Let $\mathbb{F}$ be any field and $n, s \geq 1$ and $d \geq 0$. Then there exists an algebraic circuit $\mathcal{U}$ of size $\text{poly}(n, d, s)$ computing a polynomial in $\mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_r]$ with $r \leq \text{poly}(n, d, s)$ such that:*

- $\deg_{\mathbf{x}}(\mathcal{U}(\mathbf{x}, \mathbf{y})), \deg_{\mathbf{y}}(\mathcal{U}(\mathbf{x}, \mathbf{y})) \leq \text{poly}(d)$*;*
- *for any $f \in \mathbb{F}[x_1, \ldots, x_n]$ with $\deg_{\mathbf{x}}(f) \leq d$ that is computable by an algebraic circuit of size s, there exists an $\mathbf{a} \in \mathbb{F}^r$ such that $f(\mathbf{x}) = \mathcal{U}(\mathbf{x}, \mathbf{a})$.* □

Algebraic Natural Proofs

We now define the concepts around natural proofs that we will need [FSV18, GKSS17].

Definition 2.12 (Degree-d families)**.** *For any function $d : \mathbb{N} \to \mathbb{N}$, we denote by $\mathcal{P}_d$ the class of all degree-d polynomial families. That is, a family $\{f_n\}$ of n-variate polynomials belongs to $\mathcal{P}_d$ if $\deg(f_n) \leq d(n)$ for all large enough n.* ◊

Definition 2.13 ($\mathcal{D}$-Natural Proofs for $\mathcal{C}$)**.** *Let $d(n), D(N)$ be polynomially growing functions, and let $\mathcal{C} \in \mathcal{P}_d$ and $\mathcal{D} \in \mathcal{P}_D$ be any classes of polynomial families. For $N(n) = \binom{n+d(n)}{n}$, we say that an N-variate family $\{A_N\} \in \mathcal{D}$ is a* $\mathcal{D}$-natural proof for $\mathcal{C}$ *if for any family $\{f_n\} \in \mathcal{C}$ we have that the polynomial $A_{N(n)}$ vanishes on the coefficient vector of f_n, for all large enough n.* ◊

Definition 2.14 (VP, VNP, and natural proofs)**.** *For a fixed $d(n) \in \text{poly}(n)$, we say that* VP-natural proofs exist for VP_d *if for some $D(N) \in \text{poly}(N)$, there exists a family $\{A_N\}$ that is a VP_D-natural proof for VP_d. We say that* VP-natural proofs exist for VP *if for every $d(n) \in \text{poly}(n)$, there is some* VP*-natural proof for VP_d.*

Similarly, we say that there are VNP-natural proofs for VP*, if for each $d(n) \in \text{poly}(n)$, there is some* VNP*-natural proof.* ◊

Succinct Hitting Sets

For classes of polynomial families $\mathcal{C}, \mathcal{D}$, we have already defined $\mathcal{C}$-succinct hitting set generators for $\mathcal{D}$ (Definition 1.1). We now define VP-succinct hitting set generators for VP. Note that they directly contradict VP-natural proofs for VP [FSV18, GKSS17].

Definition 2.15 (Succinct hitting sets for VP)**.** *For fixed $d(n) \in \text{poly}(n)$, we say that* VP_d-succinct hitting sets exist for VP *if for every $D(N), S(N) \in \text{poly}(N)$, there is some family $\{h_n\}$ that is a VP_d-succinct hitting set generator (as in Definition 1.1) for size-$s(N)$ families in VP_D.*

We say that VP-succinct hitting sets exist for VP *if for some $d(n) \in \text{poly}(n)$, there are VP_d-succinct hitting sets for* VP*. Finally, we say that there are* VNP-succinct hitting sets for VP*, if for some $d(n) \in \text{poly}(n)$, there are VNP_d-succinct hitting sets for* VP*.* ◊

We refer the reader to the full version (on arxiv) of [CKR+20] for detailed definitions of succinct HSGs and natural proofs.

2.2 Algebraic analogues of bounded-space computation: VPSPACE

The upper bound on the annihilator that we show is in terms of space complexity.

Definition 2.16 (SPACE). *We use* $\mathsf{SPACE}(s(n))$ *to denote the class of languages that can be decided by deterministic Turing Machines in space* $s(n)$ *for all inputs of length* n. ◊

Defining the class

One way to define the algebraic analogue of PSPACE is using the space needed to compute coefficients of the polynomial, which was the route taken by Koiran and Perifel [KP09].

Definition 2.17 (Coefficient function). *Suppose* $f(x_1, \ldots, x_n)$ *is a polynomial of* individual *degree* d*, and each of its coefficient is an integer of absolute value* $< 2^\ell$.

Then, for $M = n \cdot \lceil \log(d+1) \rceil + (\ell + 1)$*, we define the coefficient function of* f *to be the* M*-bit boolean function* Φ_f *such that* $\Phi_f(e_1, \ldots, e_n, i)$ *outputs the* i^{th} *bit of the coefficient of the monomial* $x_1^{e_1} \cdots x_n^{e_n}$*, where the* 0^{th} *bit encodes the sign.*

A family $\{f_n\}$ *of integer polynomials naturally defines a family* $\{\Phi_n\}$ *of coefficient functions.* ◊

Definition 2.18 ($\mathsf{VPSPACE}^0$ from coefficient functions [KP09]). *A family* $\{P_N\}$ *of integer polynomials is said to be in* $\mathsf{VPSPACE}^0$ *if, for all large* N*, the polynomial* P_N *has* $\mathrm{poly}(N)$ *variables, degree* $2^{\mathrm{poly}(N)}$*, each coefficient of* P_N *has at most* $2^{\mathrm{poly}(N)}$ *bits, and if the family of coefficient functions of* $\{P_N\}$ *is computable in* $\mathsf{PSPACE}_{/\mathsf{poly}}$. ◊

Poizat [Poi08] on the other hand defined VPSPACE without going into boolean computation, using a new type of gate called projection gate, defined below. He showed[10] that this definition is equivalent to that of Koiran and Perifel.

Definition 2.19 (Projection gates). *A projection gate is labelled by a variable, say* w*, and a constant* $b \in \{0, 1\}$*. We denote such a gate by* $\mathrm{fix}_{w=b}$*. The gate* $\mathrm{fix}_{w=b}$ *"projects"* w *to* b *in the input polynomial. That is,* $\mathrm{fix}_{w=b} f(w, \mathbf{x}) = f(b, \mathbf{x})$. ◊

Definition 2.20 (Circuits with projections [Poi08]). *A family* $\{P_N\}$ *of integer polynomials is said to be in* VPPROJ^0 *if for all large* N*, there is a constant-free algebraic circuit, say* C*, that additionally have access to projection gates such that* C *has size* $\mathrm{poly}(N)$ *and* P_N *is the polynomial computed by* C. ◊

Proposition 2.21 ([Poi08, Mal11]). $\mathsf{VPSPACE}^0 = \mathsf{VPPROJ}^0$.

We believe that this statement is not entirely obvious, so we give a fairly detailed proof sketch of this in Appendix A. Finally, for polynomials with arbitrary constants, VPSPACE is defined using $\mathsf{VPSPACE}^0$ as follows.

Definition 2.22 (VPSPACE). *Let* $\mathbb{F}$ *be the field of rationals, reals or complexes. A family* $\{P_N\}$ *of polynomials over* $\mathbb{F}$ *is in* VPSPACE*, if there exists an* $M = \mathrm{poly}(N)$ *and a family* $\{Q_M\} \in \mathsf{VPSPACE}^0$*, such that for all* N*,* P_N *is obtained from* Q_M *by setting some variables to constants from* $\mathbb{F}$. ◊

For convenience, we also define two new types of gates: summation and production gates.

Definition 2.23 (Summations and Productions). *The summation and production operations are defined, using the operation of projection as defined in Definition 2.19, as follows.*

- $\mathtt{sum}_z f(z, \mathbf{x}) := \mathrm{fix}_{z=0} f(z, \mathbf{x}) + \mathrm{fix}_{z=1} f(z, \mathbf{x})$

[10] Poizat's paper (written in French) includes a rough sketch of this proof. Malod's paper [Mal11] (in English) quotes his result but does not provide a proof.

- $\texttt{prod}_z f(z, \mathbf{x}) := \text{fix}_{z=0} f(z, \mathbf{x}) \times \text{fix}_{z=1} f(z, \mathbf{x})$ $\Diamond$

It is easy to see that, by definition, these gates can be simulated using projection gates and the usual sum, product gates.

Before moving on, it is important to note that there are (at least) two more equivalent characterizations of VPSPACE due to Malod [Mal11], and Mahajan and Rao [MR13]. We omit the details of these definitions as we do not directly use them.

Comparison with VP

Given that VPSPACE corresponds to a class as powerful as PSPACE, it is worth checking that the known "hard polynomial families" are outside VPSPACE as well. In this context, it makes sense to define a bounded degree analogue of VPSPACE written as $\mathsf{VPSPACE}_b$. As Koiran and Perifel [KP09] showed, the degree bound is inconsequential when comparing VP and VPSPACE.

Proposition 2.24 (Restatement of [KP09, Lemma 4])**.** *Let* $\mathsf{VPSPACE}_b$ *be the polynomial families of degree* $\text{poly}(n)$ *that belong to* VPSPACE*, and similarly, let* VP_{nb} *be the class of polynomial families of unbounded degree that have algebraic circuits of size* $\text{poly}(n)$*.*

Then, $\mathsf{VP} = \mathsf{VPSPACE}_b$ *if and only if* $\mathsf{VP}_{nb} = \mathsf{VPSPACE}$*.* □

We can now summarize the relationship of $\mathsf{VPSPACE}_b$ with known hard polynomials as follows, essentially by using the arguments in the well-known book by Bürgisser [Bür00].

- Since a *random polynomial* does not have small circuits with projection gates with high probability, random polynomial families are outside $\mathsf{VPSPACE}_b$.
- Any construction that involves exponentially many "independent irrational numbers", e.g. $h_c(\mathbf{x}) = \sum_{\mathbf{e}} \sqrt{p_{\mathbf{e}}} \cdot \mathbf{x}^{\mathbf{e}}$ for distinct primes $\{p_{\mathbf{e}}\}$, also works against $\mathsf{VPSPACE}_b$. This is because the coefficients of polynomials in $\mathsf{VPSPACE}_b$ can always be written as integer polynomials that depend on at most polynomially many scalars from the underlying field.
- Finally, *Strassen's multilinear polynomial*, defined[11] as $h_s(\mathbf{x}) = \sum_{0 \le i < 2^n} 2^{2^i} \mathbf{x}^{\mathbf{i}}$ has coefficients that are triply exponential in the number of variables n. This means that we will need doubly exponentially many bits to even index into the bits of the coefficients, which puts the (corresponding family of) coefficient functions outside PSPACE.

Consequences of Separating VP and VPSPACE

Koiran and Perifel [KP09] showed that separating VP and $\mathsf{VPSPACE}_b$ would imply interesting lower bounds.

Proposition 2.25 ([KP09, Proposition 3])**.** *If* $\mathsf{VP} \neq \mathsf{VPSPACE}_b$*, then either* $\mathsf{VP} \neq \mathsf{VNP}$ *or* $\mathsf{P}_{/\text{poly}} \neq \mathsf{PSPACE}_{/\text{poly}}$*.*

Assuming the Generalized Riemann Hypothesis (GRH), the converse is also true. That is, if $\mathsf{VP} \neq \mathsf{VNP}$ *or* $\mathsf{P}_{/\text{poly}} \neq \mathsf{PSPACE}_{/\text{poly}}$ *then* $\mathsf{VP} \neq \mathsf{VPSPACE}_b$*.* □

On the other hand, it can be shown via a padding argument that if $\mathsf{TC}^0 = \mathsf{NC}^1$ then the counting hierarchy (CH) is the same as PSPACE (see, for example, [CMTV98] or Complexity Zoo) implying that $\mathsf{CH}_{/\text{poly}} = \mathsf{PSPACE}_{/\text{poly}}$ as well. Further, the proof can be modified to show the following.

[11] Here $\mathbf{i}$ is the vector corresponding to the binary representation of i.

Proposition 2.26. *There exists a constant a for which the following holds.*
For any function $m(n) = \Omega(\log n)$, suppose that every $\mathrm{DTIME}(m(n))$-uniform NC^1 circuit on $2^{a \cdot m(n)}$ inputs, can be simulated by a $\mathrm{DTIME}(n^c)$-uniform constant depth threshold circuit of size 2^{n^c} for some constant c. Then $\mathsf{SPACE}(m(n)) \subseteq \mathsf{CH}$, both defined for input length n. □

At the same time, the proof of Proposition 2.25 can be generalized to extract the following statement when we additionally assume the GRH. Here $\mathsf{CH}(t(n))$, $\mathsf{VP}(t(n))$, $\mathsf{VPSPACE}(t(n))$ refer to these classes defined with respect to the input length being $t(n)$.

Proposition 2.27. *Assuming the Generalized Riemann Hypothesis, for any function $m(n) = \Omega(\log n)$, if $\mathsf{VPSPACE}_b(m(n)) \not\subseteq \mathsf{VP}(n)$, then either $\mathsf{VP} \neq \mathsf{VNP}$ or $\mathsf{SPACE}(m(n)) \not\subseteq \mathsf{CH}(n)$.*

Sketch. The argument is easier for the contrapositive. We assume the GRH throughout.
If $\mathsf{VP} = \mathsf{VNP}$, then from a work of Bürgisser [Bür09], we know that CH collapses to $\mathsf{P}_{/\mathsf{poly}}$. Also, by Valiant's criterion [Val79a], we have that any polynomial family whose coefficient function (family) is in $\mathsf{P}_{/\mathsf{poly}}$ is in VNP.

Putting these together, we get that if $\mathsf{VP} = \mathsf{VNP}$, then any polynomial family whose coefficient function belongs to $\mathsf{CH}_{/\mathsf{poly}}$ must belong to VP. However, if additionally $\mathsf{SPACE}(m(n)) \subseteq \mathsf{CH}(n)$, then all the polynomial families whose coefficient functions can be computed in (non-uniform) space $O(m(n))$ belong to VP, which means that $\mathsf{VPSPACE}_b(m(n)) \subseteq \mathsf{VP}(n)$. □

2.3 Explicit Objects

Next, along the same lines as *explicit* polynomial maps (Definition 1.7), we define explicit matrices and explicit ABPs. As before, we use the word *explicit* to mean encodable by efficient circuits.

Definition 2.28 (Explicit matrices)**.** *A circuit $C(\mathbf{x}, \mathbf{a}, \mathbf{b})$ is said to* encode *a matrix $M \in \mathbb{F}^{r \times c}$ if $|\mathbf{a}| = \lceil \log r \rceil$, $|\mathbf{b}| = \lceil \log c \rceil$ and for $\mathbf{i}$, $\mathbf{j}$ being binary representations of i, j respectively, $C(\mathbf{x}, \mathbf{i}, \mathbf{j}) = M[i, j]$.*

Analogously, for a family of matrices $\{M_{r,c} : M_{r,c}$ has r rows and c columns$\}$ and a class $\mathcal{C}$, we say that $\{M_{r,c}\}$ is $\mathcal{C}$-explicit *if there is a family $\{C_{r,c}\} \in \mathcal{C}$ such that, for every r, c, $C_{r,c}$ encodes $M_{r,c}$.* ◊

Definition 2.29 (Explicit ABPs)**.** *A circuit $C_A(\mathbf{x}, (\mathbf{a}, \mathbf{b}), (\mathbf{a}', \mathbf{b}'))$ is said to* encode *an algebraic branching program A, of width w and d layers, if $|\mathbf{a}| = |\mathbf{a}'| = \lceil \log d \rceil$, $|\mathbf{b}| = |\mathbf{b}'| = \lceil \log w \rceil$ and*

$$\forall \mathbf{i}, \mathbf{i}' \in \{0,1\}^{\lceil \log d \rceil} \quad \forall \mathbf{j}, \mathbf{j}' \in \{0,1\}^{\lceil \log w \rceil}, \\ C_A(\mathbf{x}, (\mathbf{i}, \mathbf{j}), (\mathbf{i}', \mathbf{j}')) \text{ is the label on the edge } ((i, j), (i', j')) \text{ in } A.$$

Here $\mathbf{i}$, $\mathbf{j}$, $\mathbf{i}'$, $\mathbf{j}'$ are the binary representations of i, j, i', j' respectively and $((i, j), (i', j'))$ denotes the edge between the j-th vertex in layer i and the j'-th vertex in layer i'.

Analogously, for a family of algebraic branching programs $\{A_{w,d} : A_{w,d}$ has width w and d layers$\}$ and a class $\mathcal{C}$, we say that $\{A_{w,d}\}$ is $\mathcal{C}$-explicit *if there is a family $\{C_{w,d}\} \in \mathcal{C}$ such that, for every w, d, $C_{w,d}$ encodes $M_{w,d}$.* ◊

2.4 Computing determinants of explicit matrices in VPSPACE

We now show that given a matrix that is encoded by a small circuit, its determinant can be computed using a small circuit with projection gates. A similar result already appears in a work by Malod [Mal11] and we provide a proof here for completeness.

Proposition 2.30 (Computing determinant of explicit matrices). *Let $M \in \mathbb{C}^{N \times N}$ be a matrix that is encoded by a circuit $C(\mathbf{x}, \mathbf{a}, \mathbf{b})$ with $|\mathbf{a}| = |\mathbf{b}| = \lceil \log N \rceil$.*

Then $\det(M)$ can be computed by a circuit C' with projection gates, of size $O(\text{size}(C) + \log^2 N)$. Moreover, if C is constant-free, then so is C'.

Proof. The proof has two parts. First we show that $\det(M)$ can be computed by an explicit Algebraic Branching Program (ABP) and then show that the polynomial computed by an explicit ABP can also be efficiently computed by circuits with projection gates.

The first step is easy to deduce from the well-known elegant construction due to Mahajan and Vinay [MV97] and the second step is just a careful application of "repeated squaring" for matrices. We now prove these formally.

Claim 2.31. *There is an explicit ABP, say $A(\mathbf{x})$, encoded by a circuit of size $\text{poly}(\text{size}(C), \log N)$ that computes $\det(M)$.*

Proof. We just describe the ABP computing $\det(M)$ here and also the circuit that encodes it. For a crisp proof of why this ABP computes the determinant, we direct the reader to Saptharishi's survey [Sap15, Section 3.3.3].

The ABP has $N+1$ layers of vertices and, except for the first and the last layer, each layer has $O(N^2)$ vertices. Vertices of the ABP are labelled by $(\ell, (i,j))$ where $\ell \in [N+1]$ denotes the layer and $(i,j) \in [N] \times [N]$ indexes a particular vertex in that layer. The edges are given as below.

- For each vertex $(\ell, (i,j))$ with $\ell \in [N-1]$, there is an edge to each $(\ell+1, (i,k))$ where $k > i$. The label of the edge is $M[j,k]$.
- For each vertex $(\ell, (i,j))$ with $\ell \in [N-1]$, there is an edge to each $(\ell+1, (k,k))$ where $k > i$. The label of the edge is $-M[j,i]$.
- All vertices $(N, (i,j))$ have an edge to the sink $(N+1, (1,1))$, with the label $-M[j,i]$.

We now describe the circuit $\widetilde{C}$ that encodes the above ABP, using the circuit C. Each vertex of the ABP is a vector of $3\lceil \log N \rceil$ bits: $\lceil \log N \rceil$ each for ℓ, i and j as described in the above construction. In the circuit, the two input vertices are $\mathbf{u} \equiv (\ell_u, (i_u, j_u))$ and $\mathbf{v} \equiv (\ell_v, (i_v, j_v))$[12]. Using the polynomials defined in Observation 2.9, we get the following expression for $\widetilde{C}$.

$$\begin{aligned}
\text{Let } G(\mathbf{x}, \mathbf{u}, \mathbf{v}) &= \text{EQ}(\ell_v, \text{INC}(\ell_u)) \cdot \text{EQ}(i_u, i_v) \cdot \text{GT}(j_v, i_u) \cdot C(\mathbf{x}, j_u, j_v) \\
&\quad + \text{EQ}(\ell_v, \text{INC}(\ell_u)) \cdot \text{GT}(i_v, i_u) \cdot \text{EQ}(i_v, j_v) \cdot (-1 \cdot C(\mathbf{x}, i_u, j_u)) \\
&\quad + \text{EQ}(\ell_v, \text{INC}(\ell_u)) \cdot \text{EQ}(\ell_u, \mathbf{n}) \cdot \text{EQ}(0\ldots01, i_v) \cdot \text{EQ}(i_v, j_v) \cdot (-1 \cdot C(\mathbf{x}, i_u, j_u)), \\
\text{and valid}(\mathbf{u}) &= \text{LT}(0\ldots00, \ell_u) \cdot \text{LT}(\ell_u, \text{INC}(\text{INC}(\mathbf{n}))) \cdot \text{int}(i_u) \cdot \text{int}(j_u), \\
\text{where int}(\mathbf{a}) &= \text{LT}(0\ldots00, \mathbf{a}) \cdot \text{LT}(\mathbf{a}, \text{INC}(\mathbf{n})). \\
\text{Then } \widetilde{C}(\mathbf{x}, \mathbf{u}, \mathbf{v}) &= \text{valid}(\mathbf{u}) \cdot \text{valid}(\mathbf{v}) \cdot g(\mathbf{x}, \mathbf{u}, \mathbf{v}).
\end{aligned}$$

Here each 'product term' in the definition of G corresponds to one type of edge in the description above, and $\mathbf{n}$, $0\ldots00$ and $0\ldots01$ are the bit-vectors corresponding to the numbers N, 0 and 1 respectively. We use the polynomial "valid" to ensure that $\widetilde{C}$ outputs 0 whenever it is given a pair of labels that is of a 'non-edge'.

[12] Here ℓ,i,j for both $\mathbf{u}$ and $\mathbf{v}$ are bit-vectors. We have used plain lowercase symbols since we think of them as numbers.

The correctness of this expression is easy to check using the description of the ABP. Also, $\widetilde{C}$ has size $O(\text{size}(C) + \log^2 N)$ and is a constant-free algebraic circuit (without projection gates) if C is constant free. □

Claim 2.32. *There is a circuit C' with projection gates, of size $\text{poly}(\text{size}(\widetilde{C}), \log N)$ that computes the polynomial computed by the above ABP.*

Proof. Let A be the adjacency matrix of the graph that underlies the branching program. The polynomial computed by the ABP is just the (s,t)-th entry of the matrix A^N, where $s = (1,(1,1))$ and $t = (N+1,(1,1))$ are the tuples corresponding to the source and the sink. Since $\widetilde{C}$ encodes A, we only need to figure out how to encode A^2, which we can then use recursively to obtain access to the entries of A^{2^k} for any k. Note that this becomes much easier when N is a power of 2.

So we modify the ABP by adding layers $N+2, \ldots, N'+1$, to ensure that N' is a power of 2. Each of the new layers have a single vertex with the label $(\ell,(1,1))$ where ℓ is the layer. We also add an edge from $(\ell,1,1)$ to $(\ell+1,1,1)$, labelled with the scalar 1, for every $N < \ell \leq N'$. This new ABP is explicit since we can add the term

$$\text{EQ}(\ell_v, \text{INC}(\ell_u)) \cdot \text{GT}(\ell_u, \mathbf{n}) \cdot \text{EQ}(0\ldots01, i_u) \cdot \text{EQ}(i_u, j_u) \cdot \text{EQ}(0\ldots01, i_v) \cdot \text{EQ}(i_v, j_v) \cdot 1$$

to the polynomial G defined above, and also modify $\text{valid}(\mathbf{u})$ appropriately, to obtain a circuit that encodes the new ABP.

Now note that the (s,t)-th entry of $A^{N'}$ is the same as the (s,t)-th entry of A^N, and therefore is the polynomial we are looking for. Further, we can now easily compute the (s,t)-th entry of $A^{N'}$ using recursion, as described earlier. We now describe a circuit using projection gates that carries out this operation.

Consider the following circuits defined using projections (Definition 2.19) and summations (Definition 2.23).

$$P_1(\mathbf{x},\mathbf{u},\mathbf{w},\mathbf{v},z) := \widetilde{C}\,(\mathbf{x}, ((1-z)\mathbf{u} + z\mathbf{w}), ((1-z)\mathbf{w} + z\mathbf{v}))$$
$$D_1(\mathbf{x},\mathbf{u},\mathbf{v}) := \mathtt{sum}_{w_1}\mathtt{sum}_{w_2}\cdots\mathtt{sum}_{w_L}\,(\text{fix}_{z=0}\,P_1(\mathbf{x},\mathbf{u},\mathbf{w},\mathbf{v},z) \cdot \text{fix}_{z=1}\,P_1(\mathbf{x},\mathbf{u},\mathbf{w},\mathbf{v},z))$$

Note that D_1 encodes the adjacency matrix A^2 when $\widetilde{C}$ encodes A and $L = O(\log N)$ is the length of the vertex labels of A.

Now for every $2 \leq i \leq k = \lceil \log(N+1) \rceil$, we define P_{i+1} by using D_i in place of $\widetilde{C}$, and D_{i+1} by using P_{i+1} in place of P_1, in the definitions of P_1 and D_1 above.

As each of the P_{i+1}s and D_{i+1}s have circuits (with projection gates) of size $\text{poly}(\log N)$ with *exactly one* use of D_i and P_{i+1} respectively, all these increases are additive. So, finally, $C'(\mathbf{x}) = D_k(\mathbf{x}, s, t)$ has a circuit of size $O(\text{size}(\widetilde{C}) + \log^2 N)$ and computes the polynomial we need. Further, C' is constant-free if $\widetilde{C}$ is constant free. □

Combining both the above claims, we get that $\text{size}(C') = O(\text{size}(C) + \log^2 N)$, and also that C' is constant-free whenever C is constant-free. □

3 VPSPACE Upper Bounds for Annihilators

In this section we prove Theorem 1.11. In fact we will prove the following, more general, statement.

Theorem 3.1 (Annihilators of explicit maps). *Let m be large enough, and let $\mathcal{G} : \mathbb{F}^m \to \mathbb{F}^n$ be a polynomial map given by $(g_1(\mathbf{z}), \ldots, g_n(\mathbf{z}))$ of degree d, with $n \geq 2m$.*

There exists a constant c such that, if there is a circuit with projection gates *$C_{\mathcal{G}}(\mathbf{z}, \mathbf{y})$ of size s that encodes $\mathcal{G}$ as per Definition 1.7, then there is a circuit* with projection gates *$C'(\mathbf{x})$ of size $(m \cdot d \cdot s)^c$ computing a nonzero polynomial $A(\mathbf{x})$ of individual degree at most $3 \cdot m \cdot d$ that annihilates $\mathcal{G}$ (that is, $A \circ \mathcal{G}_m = A(g_1(\mathbf{z}), \ldots, g_n(\mathbf{z})) \equiv 0$).*

To prove this, we will first show that there is an *explicit* matrix whose determinant is the annihilator, and then show that in this case the annihilator has a small circuit with projection gates.

3.1 Annihilator as determinant of a matrix

Lemma 3.2 (Annihilator as a determinant). *Let $\mathcal{G}$ be a polynomial map as given in Theorem 3.1. Further, let $C_{\mathcal{G}}(\mathbf{z}, \mathbf{y})$ be a circuit that generates $\mathcal{G}$.*

Then for $D = \left\lceil (nd)^{m/(n-m)} \right\rceil + 1$, there exists a $K \times K$ matrix $\widetilde{M}$ with $K \leq D^n$ such that $\det(\widetilde{M})$ is an annihilator of $\mathcal{G}$ that has individual degree at most $D - 1$. Moreover, $\widetilde{M}$ can be described as follows.

There is some positive integer $\alpha < D^{2n}$ such that, for $i \in \{0, \ldots, K-1\}$ and $j \in [K]$,

$$\widetilde{M}[i, \mathbf{e}^{(j)}] = \begin{cases} (\mathcal{G}(\mathbf{v}_{\alpha,i}))^{\mathbf{e}^{(j)}} & \text{if } i \leq K-2 \\ (-1)^{K-1} \cdot \mathbf{x}^{\mathbf{e}^{(j)}} & \text{if } i = K-1, \end{cases}$$

where $\mathbf{v}_{\alpha,i} = (\alpha^i, \alpha^{i \cdot ndD}, \ldots, \alpha^{i \cdot (ndD)^{m-1}})$.

Proof. First we see how linear dependencies of specific polynomials in terms of $\mathcal{G}$ give us annihilators of the map $\mathcal{G}$. Suppose $A(\mathbf{x})$ is an annihilator of $\mathcal{G}$, and let $A(\mathbf{x}) = \sum_{\mathbf{e}} f_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$ be its sparse representation. Then $0 = A(\mathcal{G}) = \sum_{\mathbf{e}} f_{\mathbf{e}} \mathcal{G}^{\mathbf{e}}$. In other words, the coefficient vector of A is a linear dependency in the set of polynomials of the form $\mathcal{G}^{\mathbf{e}}$, as we range over all the exponent vectors $\mathbf{e}$. Clearly, the converse is also true: for any polynomial $A(\mathbf{x}) = \sum_{\mathbf{e}} f_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$, if the coefficients $\{f_{\mathbf{e}}\}$ represent a linear dependency in the polynomials $\{\mathcal{G}^{\mathbf{e}}\}$, then $F(\mathcal{G}) = 0$.

Now, for a parameter D to be fixed later, consider the following matrix M.

- Columns of M are indexed by monomials in $\mathbf{x} = \{x_1, \ldots, x_n\}$ of individual degree at most $D - 1$, ordered lexicographically.
- Rows of M are indexed by monomials in $\mathbf{z} = \{z_1, \ldots, z_m\}$ of individual degree at most $(ndD) - 1$, ordered lexicographically.
- For any valid $\mathbf{e} \in \mathbb{N}^n$, the $\mathbf{e}$th column of M is the coefficient vector of the product $\mathcal{G}^{\mathbf{e}} = g_1(\mathbf{z})^{e_1} \cdots g_n(\mathbf{z})^{e_n}$. Note that this product is an m-variate polynomial of individual degree at most $|\mathbf{e}| \leq (n \cdot (D-1)) \cdot d < (ndD) - 1$.

From the previous discussion, we can see that any nonzero dependency in the columns of M is the coefficient vector of some annihilator of $\mathcal{G}$. Now observe that M has fewer rows than columns whenever $D > (nd)^{m/(n-m)}$, since $(ndD)^m < D^n$ in this case. Therefore, there is a non-trivial dependency in its columns if we set $D = \left\lceil (nd)^{m/(n-m)} \right\rceil + 1$.

In order to ensure a *unique* dependency (up to scaling), we restrict M to its first K columns, where $K - 1$ is the largest number for which the first $K - 1$ columns are linearly independent. So now M has the following properties.

- M has $K \leq D^n$ columns and each column is labelled by a monomial in $\{x_1, \ldots, x_n\}$ of individual degree $\leq D-1$. Let $\left\{\mathbf{e}^{(j)}\right\}_{j\in[K]}$ be the labels of the columns.
- Each row is labelled by a monomial in $\{z_1, \ldots, z_m\}$ of individual degree $\leq (ndD) - 1$ and so, if the number of rows in M is R, then

$$R \leq (ndD)^m < D^n$$

- The $(\mathbf{z}^{\mathbf{e}}, \mathbf{e}^{(j)})$-th entry of M is the coefficient of $\mathbf{z}^{\mathbf{e}}$ in $\mathcal{G}^{\mathbf{e}^{(j)}}$.
- The first $K-1$ columns are linearly independent and the last column is a linear combination of the previous columns.
- $A(\mathbf{x}) = \mathbf{x}^{\mathbf{e}^{(K)}} - \sum_{j=1}^{K-1} f_{\mathbf{e}^{(j)}} \cdot \mathbf{x}^{\mathbf{e}^{(j)}}$ is an annihilator of $\mathcal{G}$,if $M[\mathbf{e}^{(K)}] = \sum_{j=1}^{K-1} f_{\mathbf{e}^{(j)}} \cdot M[\mathbf{e}^{(j)}]$.

We now construct a matrix M', with $(K-1)$ rows and K columns, such that M' has rank exactly $(K-1)$. In fact, the columns of M' will share the same dependencies as the columns of M.

Claim 3.3. *Define for $\alpha \in \mathbb{N}$, the matrix $E_\alpha \in \mathbb{C}^{(K-1)\times R}$ such that $E_\alpha[i,j] = \alpha^{ij}$ for each $0 \leq i < K-1$ and $0 \leq j < R$. For some $\alpha \leq D^{2n}$, the matrix product $M' := E_\alpha \cdot M$ has rank exactly $K-1$. Further, M' has the same dependencies in its columns as M.*

Proof. By Lemma 2.8, we immediately have that there exists $\alpha \leq R \cdot (K-1) < D^{2n}$ such that $\operatorname{rank}(E_\alpha \cdot M) = K-1$. Let $M' = E_\alpha \cdot M$. To show that M' has the same dependencies in its columns as M, let us first assume that $M\mathbf{u} = 0$ for some $\mathbf{u} \in \mathbb{C}^K$. Clearly this implies that $M'\mathbf{u} = E_\alpha \cdot M\mathbf{u} = 0$.

Conversely, let $\mathbf{v} \in \mathbb{C}^K$ be such that $M'\mathbf{v} = 0$ but $M\mathbf{v} \neq 0$. Then, clearly, $\mathbf{v} \neq \beta\mathbf{u}$ for any $\beta \in \mathbb{C}$. This shows that $\dim(\ker(M')) \geq 2$, contradicting the rank-nullity theorem since $\operatorname{rank}(M') = K-1$. Therefore, no such $\mathbf{v}$ can exist, proving that for any $\mathbf{v} \in \mathbb{C}^K$, $M'\mathbf{v} = 0 \implies M\mathbf{v} = 0$. □

For M' defined as in the claim above, we clearly have that $M'[\mathbf{e}^{(K)}] = \sum_{j=1}^{K-1} f_{\mathbf{e}^{(j)}} \cdot M'[\mathbf{e}^{(j)}]$. Also, note that the rows of M' are labelled by $\{0, \ldots, K-2\}$ and the columns are labelled by $\left\{\mathbf{e}^{(j)} : j \in [K]\right\}$. We now define $\widetilde{M}$, a $K \times K$ matrix with rows labelled by $\{0, \ldots, K-1\}$ and columns labelled by $\left\{\mathbf{e}^{(j)} : j \in [K]\right\}$, as follows.

$$\widetilde{M}[i, \mathbf{e}^{(j)}] = \begin{cases} M'[i, \mathbf{e}^{(j)}] & \text{if } i \leq K-2 \\ (-1)^i \cdot \mathbf{x}^{\mathbf{e}^{(j)}} & \text{if } i = K-1 \end{cases}$$

First, we show that the determinant of $\widetilde{M}$ is an annihilator of $\mathcal{G}$. In particular, we show that $\det(\widetilde{M})$ is a scalar multiple of $A(\mathbf{x})$. We use $M'[*, \mathbf{e}^{(j)}]$ to denote the $\mathbf{e}^{(j)}$th column of the matrix M', and $M' \setminus M'[*, \mathbf{e}^{(j)}]$ to denote the submatrix of M' obtained by deleting the $\mathbf{e}^{(j)}$th column.

Claim 3.4. $\det(\widetilde{M}) = \det(M' \setminus M'[*, \mathbf{e}^{(K)}]) \cdot A(\mathbf{x})$.

Proof. Note that, by expanding with respect to the last row,

$$\det(\widetilde{M}) = \sum_{j=1}^{K} (-1)^{j-1} \cdot \det(M' \setminus M'[*, \mathbf{e}^{(j)}]) \cdot (-1)^{K-1} \cdot \mathbf{x}^{\mathbf{e}^{(j)}}.$$

This can be re-written as

$$\frac{\det(\widetilde{M})}{\det(M' \setminus M'[*, \mathbf{e}^{(K)}])} = \mathbf{x}^{\mathbf{e}^{(K)}} + \sum_{j=1}^{K-1} (-1)^{K-j} \cdot \frac{\det(M' \setminus M'[*, \mathbf{e}^{(j)}])}{\det(M' \setminus M'[*, \mathbf{e}^{(K)}])} \cdot \mathbf{x}^{\mathbf{e}^{(j)}}.$$

Since $M'[\mathbf{e}^{(K)}] = \sum_{j=1}^{K-1} f_{\mathbf{e}^{(j)}} \cdot M'[\mathbf{e}^{(j)}]$, it is now easy to check the following using Cramer's Rule.

$$\frac{\det(\widetilde{M})}{\det(M' \setminus M'[*, \mathbf{e}^{(K)}])} = \mathbf{x}^{\mathbf{e}^{(K)}} - \sum_{j=1}^{K-1} f_{\mathbf{e}^{(j)}} \cdot \mathbf{x}^{\mathbf{e}^{(j)}} = A(\mathbf{x}). \qquad \square$$

Since $\det(M' \setminus M'[*, \mathbf{e}^{(K)}])$ is a fixed, nonzero scalar, $\det(\widetilde{M})$ is an annihilator of $\mathcal{G}$. Further, since every monomial in $\left\{\mathbf{x}^{\mathbf{e}^{(j)}} : j \in [K]\right\}$ has individual degree at most $D-1$, $\det(\widetilde{M})$ also has individual degree at most $D-1$.

We now show that $\widetilde{M}$ can be described as claimed. To show that, we note that M' has a very special structure. For any $\ell \in \{0, \ldots, R-1\}$, let $\mathbf{v}_\ell \in [ndD-1]^m$ be the unique vector such that

$$\ell = \sum_{b=1}^{m} \mathbf{v}_\ell(b) \cdot (ndD)^{b-1}.$$

Then, for $i \in \{0, \ldots, K-2\}$ and $j \in [K]$,

$$\begin{aligned} M'[i, \mathbf{e}^{(j)}] &= \sum_{\ell=0}^{R-1} \alpha^{i\ell} \cdot \mathrm{coeff}_{\mathbf{z}^{\mathbf{v}_\ell}}(\mathcal{G}^{\mathbf{e}^{(j)}}) = \sum_{\ell=0}^{R-1} \mathrm{coeff}_{\mathbf{z}^{\mathbf{v}_\ell}}(\mathcal{G}^{(\mathbf{e}^j)}) \cdot \alpha^{i \cdot \left(\sum_{b=1}^{m} \mathbf{v}_\ell(b) \cdot (ndD)^{b-1}\right)} \\ &= \sum_{\ell=0}^{R-1} \left(\mathrm{coeff}_{\mathbf{z}^{\mathbf{v}_\ell}}(\mathcal{G}^{\mathbf{e}^{(j)}}) \prod_{b=1}^{m} \alpha^{(i \cdot (ndD)^{b-1}) \mathbf{v}_\ell(b)} \right) = \sum_{\ell=0}^{R-1} \left(\mathrm{coeff}_{\mathbf{z}^{\mathbf{v}_\ell}}(\mathcal{G}^{\mathbf{e}^{(j)}}) \cdot \mathbf{v}_{\alpha,i}^{\mathbf{v}_\ell} \right) \end{aligned}$$

where $\mathbf{v}_{\alpha,i} = (\alpha^i, \alpha^{i \cdot ndD}, \ldots, \alpha^{i \cdot (ndD)^{m-1}})$. That is,

$$M'[i, \mathbf{e}^{(j)}] = (\mathcal{G}(\mathbf{v}_{\alpha,i}))^{\mathbf{e}^{(j)}}.$$

We can therefore re-write $\widetilde{M}$ as claimed. $\square$

3.2 Annihilator as determinant of an explicit matrix

We now show that the matrix described in the last section is *explicit* in the sense of Definition 2.28.

Lemma 3.5 (Annihilator as determinant of an explicit matrix)**.** *Let $\mathcal{G}$ be a polynomial map as given in Theorem 3.1. Let $C_{\mathcal{G}}(\mathbf{z}, \mathbf{y})$ be a circuit that encodes $\mathcal{G}$. Further, for $D = \left\lceil (nd)^{m/(n-m)} \right\rceil + 1$, $\alpha < D^{2n}$ and $K \leq D^n$, let $\widetilde{M}$ be defined as follows.*

For $i \in \{0, \ldots, K-1\}$ and $j \in [K]$,

$$\widetilde{M}[i, \mathbf{e}^{(j)}] = \begin{cases} (\mathcal{G}(\mathbf{v}_{\alpha,i}))^{\mathbf{e}^{(j)}} & \text{if } 0 \leq i \leq K-2 \\ (-1)^{K-1} \cdot \mathbf{x}^{\mathbf{e}^{(j)}} & \text{if } i = K-1. \end{cases}$$

where $\mathbf{v}_{\alpha,i} = (\alpha^i, \alpha^{i \cdot ndD}, \ldots, \alpha^{i \cdot (ndD)^{m-1}})$.

Then, given access to a $C_{\mathcal{G}}$ gate, there is a purely algebraic circuit[13] *$\widetilde{C}_{\mathcal{G}}$ that encodes $\widetilde{M}$. Further, $\widetilde{C}_{\mathcal{G}}$ uses only a single $C_{\mathcal{G}}$ gate and has overall size $O((nd)^3 + n \cdot \mathrm{size}(C_{\mathcal{G}}))$.*

[13] Algebraic circuit without projection gates.

Proof. Let $\mathbf{j}$ be a bit-vector of length $n \cdot \delta$, for $\delta := \lceil \log D \rceil$, which we interpret as the tuple $(\mathbf{j}^{(1)}, \ldots, \mathbf{j}^{(n)})$ where $\mathbf{j}^{(\ell)}$ is the bit-vector encoding the ℓ-th co-ordinate of $\mathbf{e}^{(j)}$. Further, let the bit-vector $\mathbf{i}$ encode the integer i. We want to construct a circuit $\widetilde{C}_{\mathcal{G}}$ such that $\widetilde{C}_{\mathcal{G}}(\mathbf{x}, \mathbf{i}, \mathbf{j}) = \widetilde{M}[i, \mathbf{e}^{(j)}]$.

Firstly, it is easy to see that there is a constant-free multi-output circuit C', of size $O(\log^2 K)$, such that $C'(\mathbf{j}) = (\mathbf{j}^{(1)}, \ldots, \mathbf{j}^{(n)})$ as described above. We will use $C'_{k,b}(\mathbf{j})$ to denote the output gate of $C'(\mathbf{j})$ which outputs the b-th bit of $\mathbf{j}^{(k)}$.

Let $\Delta = ndD$ and $\mathbf{v}_\alpha := (\alpha, \alpha^\Delta, \ldots, \alpha^{\Delta^{m-1}})$. We start by computing $L = \lceil \log K \rceil$ many distinct powers of the vector $\mathbf{v}_\alpha$, namely, $\mathbf{v}_\alpha$, $\mathbf{v}_\alpha^2$, $\mathbf{v}_\alpha^4$, …, $\mathbf{v}_\alpha^{2^{L-1}}$. Here $\mathbf{v}_\alpha^r$ is used to denote the vector $(\alpha^r, \alpha^{r\cdot\Delta}, \ldots, \alpha^{r\cdot\Delta^{m-1}})$. Since α can be computed by a constant-free circuit of size $O(n \log(nd))$, each of these L vectors can be computed by a constant-free circuit of size $O(L \cdot n \log(nd)) = O(\log K \cdot n \log(nd))$. Let these multi-output circuits be $C_0(\alpha), \ldots, C_{L-1}(\alpha)$ with $C_{\ell,k}(\alpha)$ denoting the k-th output gate in $C_\ell(\alpha)$ for $k \in \{0, \ldots, m-1\}$. That is, $C_{\ell,k}(\alpha) = \alpha^{r\cdot\Delta^k}$ for $k \in \{0, \ldots, m-1\}$.

We now describe some intermediate circuits, and then finally $\widetilde{C}_{\mathcal{G}}$. For $\mathbf{i} = (i_0, \ldots, i_{L-1})$,

$$\begin{aligned}
\mathrm{pow}(\mathbf{i}) &:= \left(\prod_{\ell=0}^{L-1} \left(i_\ell \cdot C_{\ell,0}(\alpha) + (1 - i_\ell) \cdot 1\right), \ldots, \prod_{\ell=0}^{L-1} \left(i_\ell \cdot C_{\ell,m-1}(\alpha) + (1 - i_\ell) \cdot 1\right) \right) \\
\mathrm{ROW}(\mathbf{i}) &:= \mathrm{LT}(\mathbf{i}, \mathbf{k}) \cdot C_{\mathcal{G}}(\mathrm{pow}(\mathbf{i})) + \mathrm{EQ}(\mathbf{i}, \mathbf{k}) \cdot \mathbf{x} \\
\widetilde{C}_{\mathcal{G}}(\mathbf{i}, \mathbf{j}) &:= \prod_{a \in [n]} \left(\prod_{b=0}^{\delta-1} \left(C'_{a,b}(\mathbf{j}) \cdot (\mathrm{ROW}(\mathbf{i})_a)^{2^b} + (1 - C'_{a,b}(\mathbf{j})) \cdot 1 \right) \right)
\end{aligned}$$

Here $\mathrm{pow}(\mathbf{i})$ computes the vector $\mathbf{v}_{\alpha,\mathbf{i}}$, and the polynomials $\mathrm{LT}(\cdot,\cdot)$ and $\mathrm{EQ}(\cdot,\cdot)$ are as defined in Observation 2.9. Further, $\mathbf{k}$ is the binary encoding of the integer $K-1$ and $\mathrm{ROW}(\mathbf{i})_a$ denotes the a-th output gate of $\mathrm{ROW}(\mathbf{i})$. That is, $\mathrm{ROW}(\mathbf{i})_a = \mathrm{LT}(\mathbf{i}, \mathbf{k}) \cdot g_a(\mathrm{pow}(\mathbf{i})) + \mathrm{EQ}(\mathbf{i}, \mathbf{k}) \cdot x_a$ if $\mathcal{G} = (g_1, \ldots, g_n)$.

Note that $\widetilde{C}_{\mathcal{G}}$ uses the sub-circuit $C_{\mathcal{G}}$ exactly once, in ROW, as claimed. Also, all the three expressions described above are constant-free, algebraic expressions. Finally, note that the size of $\widetilde{C}_{\mathcal{G}}$ is $O(L \cdot n \log(nd) + L + L^2 + nd + n \cdot \mathrm{size}(C_{\mathcal{G}}))$, which is $O(n^3 \cdot d^2 + n \cdot \mathrm{size}(C_{\mathcal{G}}))$. □

3.3 Completing the proof

We now have all the components necessary to complete the proof of Theorem 3.1.

Theorem 3.1 (Annihilators of explicit maps)**.** *Let m be large enough, and let $\mathcal{G} : \mathbb{F}^m \to \mathbb{F}^n$ be a polynomial map given by $(g_1(\mathbf{z}), \ldots, g_n(\mathbf{z}))$ of degree d, with $n \geq 2m$.*

There exists a constant c such that, if there is a circuit with projection gates *$C_{\mathcal{G}}(\mathbf{z}, \mathbf{y})$ of size s that encodes $\mathcal{G}$ as per Definition 1.7, then there is a circuit* with projection gates *$C'(\mathbf{x})$ of size $(m \cdot d \cdot s)^c$ computing a nonzero polynomial $A(\mathbf{x})$ of individual degree at most $3 \cdot m \cdot d$ that annihilates $\mathcal{G}$ (that is, $A \circ \mathcal{G}_m = A(g_1(\mathbf{z}), \ldots, g_n(\mathbf{z})) \equiv 0$).*

Proof. Let $\mathcal{G}' = (g_1(\mathbf{z}), \ldots, g_{2m}(\mathbf{z}))$, the first $2m$ co-ordinates. Clearly, $C_{\mathcal{G}}$ encodes $\mathcal{G}'$ as well.

Lemma 3.2 tells us that there is a $K \times K$ matrix, $\widetilde{M}$, such that $0 \not\equiv A'(\mathbf{x}) = \det(\widetilde{M})$ is an annihilator for the map $\mathcal{G}'$ (that is, $A' \circ \mathcal{G}' \equiv 0$). Here $K = (2md+1)^{2m}$ and $A'(\mathbf{x})$ has individual degree at most $\left\lceil (2md)^{m/(2m-m)} \right\rceil + 1 = 2md + 1$.

Using the fact that the entries of $\widetilde{M}$ are either evaluations of $\mathcal{G}'$ or monomials, Lemma 3.5 provides a circuit $\widetilde{C}$ that encodes the matrix $\widetilde{M}$ where $\widetilde{C}$ has size $O((2md)^3 + (2m) \cdot \mathrm{size}(C_{\mathcal{G}}))$.

Further $\widetilde{C}$ is a purely algebraic circuit[14] and also constant-free, assuming we have access to a gate computing $C_{\mathcal{G}_m}$. Finally, we use Proposition 2.30 to obtain a circuit with projections C' that computes $A'(\mathbf{x})$. The circuit C' has size $O((2md)^3 + (4m) \cdot \text{size}(C_{\mathcal{G}}) + \log^2 K) = O((2md)^3 + (4m) \cdot \text{size}(C_{\mathcal{G}})) = (m \cdot d \cdot s)^4$ for $s = \text{size}(C_{\mathcal{G}})$.

Finally, we note that for $A(x_1, \ldots, x_n) := A'(x_1, \ldots, x_{2m})$, $0 \not\equiv A(\mathbf{x})$ and $A \circ \mathcal{G}_m \equiv 0$. This completes the proof, since C' also computes A. □

Clearly Theorem 1.11 is a special cases of Theorem 3.1 and therefore follows as a corollary. We restate it here for convenience.

Theorem 1.11 (Upper bound for annihilators of VP)**.** *Let $d(m)$ be an arbitrary polynomial function of m. For any VP_d-explicit family of maps $\{\mathcal{G}_m\}$, where each $\mathcal{G}_m$ has n outputs with $n \geq 2m$, there is a family $\{A_m\}$ in $\mathsf{VPSPACE}_b$ of degree $O(m^2 d)$ such that A_m annihilates $\mathcal{G}_m$ for all large enough m.*

In fact, since the proof allows for the encoding circuit to use projection gates, we additionally have the following statement as a special case.

Theorem 3.6 (Upper bound for annihilators of VPSPACE)**.** *For any $\mathsf{VPSPACE}_b$-explicit family of maps $\{\mathcal{G}_m\}$, where each $\mathcal{G}_m$ has more than $2m$ outputs, there is a family $\{A_m\}$ in $\mathsf{VPSPACE}_b$ such that A_m annihilates $\mathcal{G}_m$ for all large enough m.*

4 Lower Bounds from Hitting Set Generators

We are now ready to formally prove our main results about the consequences of succinct and cryptographic hitting sets. As mentioned before, these consequences follow by constructing annihilators for such hitting sets by applying Theorem 3.1 with appropriate choices of parameters.

4.1 Lower Bounds from Cryptographic Hitting Set Generators

We begin by proving Theorem 1.9, which we first restate.

Theorem 1.9 (Lower Bounds from Cryptographic HSGs)**.** *Let $\{H_n : \mathbb{C}^n \to \mathbb{C}^N\}$ be a family of polynomial maps of degree $d = n^8$ with $N \geq 2n$, and let $D(N) = N^{10}$.*

Assuming the Generalized Riemann Hypothesis, if the family $\{H_n\}$ is a VP_d-cryptographic hitting set generator for VP_D, at least one of the following must be true.

1. $\mathsf{VP} \neq \mathsf{VNP}$.
2. *Uniform* $\mathsf{NC}^1 \not\subseteq$ *uniform* TC^0*, where the uniformity is* DLOGTIME.

Further, if the family $\{H_n\}$ is a hitting set generator for VNP_d, then $\mathsf{P} \neq \mathsf{PSPACE}$.

Proof. We shall prove the statement via contradiction. Suppose that $\mathsf{VP} = \mathsf{VNP}$, and that uniform $\mathsf{NC}^1 \subseteq$ uniform TC^0. Invoking the padding argument in Proposition 2.26, we get that for any $m(n) = \text{poly}(n)$, $\mathsf{SPACE}(m(n)) \subseteq \mathsf{CH}$, which means $\mathsf{PSPACE} \subseteq \mathsf{CH}$. Thus, using Proposition 2.27, we get that $\mathsf{VPSPACE}_b = \mathsf{VP}$.

[14] Algebraic circuit without projection gates.

This means that any family $\{A_N\}$ in $\mathsf{VPSPACE}_b$ of degree $d'(N) \in \mathrm{poly}(N)$ belongs to $\mathsf{VP}_{d'}$. For the specific parameters of the generator family $\{\mathcal{H}_n\}$ in the hypothesis, Theorem 3.1 implies a family $\{A_N\} \in \mathsf{VPSPACE}_b$ of annihilators of individual degree at most $(2n \cdot n^8) \leq N \cdot (N/2)^8 \leq N^9$, and thus total degree at most $N^{10} =: d(N)$. Now from our assumption, $\{A_N\} \in \mathsf{VP}_d$, which contradicts the HSG property of $\{\mathcal{H}_n\}$.

So it must be the case that either $\mathsf{VP} \neq \mathsf{VNP}$, or that uniform $\mathsf{NC}^1 \not\subseteq$ uniform TC^0. □

One can weaken the hypothesis of Theorem 1.9 to a family of VNP or even $\mathsf{VPSPACE}_b$-explicit HSGs because of Theorem 3.6, as follows.

Theorem 4.1. *Let $\{\mathcal{H}_m\}$ be a $\mathsf{VPSPACE}_b$-explicit family of m-variate polynomial maps with $n(m) > 2m$ outputs of degree $d_0(m) \in \mathrm{poly}(m)$, and let $d(n) \in \mathrm{poly}(n)$ be such that $d(n(m)) > 2m \cdot d_0(m)$, for all large m.*

If the family $\{\mathcal{H}_m\}$ is a hitting set generator for VP_d, then $\mathsf{VP} \neq \mathsf{VPSPACE}_b$. As a consequence, either $\mathsf{VP} \neq \mathsf{VNP}$ or DLOGTIME-uniform $\mathsf{NC}^1 \not\subseteq$ DLOGTIME-uniform TC^0. □

Remark 4.2 (Known "PSPACE constructions")**.** *Note that the PSPACE-construction of hitting sets for algebraic circuits by Mulmuley [Mul12] (see also Forbes and Shpilka [FS18]), does not satisfy the hypothesis of Theorem 4.1. This is essentially the same reason as why the arguments of Heintz and Schnorr [HS80] do not give a "cryptographic" generator as required in Theorem 1.9. These constructions ([Mul12, FS18]) yield a (possibly)* different *family of generators constructible in space $\mathrm{poly}(n, d, s)$ for each size s. On the other hand, Theorem 4.1 requires a single family that works for all sizes $s(n) \in \mathrm{poly}(n)$.* ◊

4.2 Upper Bound on Annihilators for Evaluation Vectors of VP

Next we prove Theorem 1.4. Before moving on to the formal statement, we state some definitions and observations.

Definition 4.3 (Interpolating set)**.** *For a set of polynomials P, a set of evaluation points I is called an* interpolating set for P*, if there is a* linear transformation *M such that for any polynomial $f \in P$, the coefficients of f can be obtained from the evaluations of f on the set I by an application of M.* ◊

Proposition 4.4 (Restatement of [BP20, Theorem 10])**.** *Let $S := \{0, 1, 2, \ldots, d\}$, then the set of evaluation points $I_{n,d} := \{(a_1, a_2, \ldots, a_n) \in S^n | a_1 + a_2 + \cdots + a_n \leq d\}$ forms an interpolating set for the set of all n-variate polynomials of total degree at most d.* □

Definition 4.5 (Evaluation vector)**.** *For any n-variate polynomial $f(\mathbf{x})$ of total degree-d, its evaluation vector is simply the vector[15] formed by evaluations of f on the set $I_{n,d}$ from Proposition 4.4.* ◊

Proposition 4.6 (Equations for evaluation vectors and natural proofs)**.** *Let $\mathcal{C}$ and $\mathcal{D}$ be either* VP *or* VNP*. For any $d(n)$, there are $\mathcal{D}$-natural proofs for $\mathcal{C}_d$ if and only if there is a family $\{A_N\} \in \mathcal{D}$ such that for each $\{f_n\} \in \mathcal{C}_d$, A_N vanishes on the evaluation vector of f_n for all large enough n.*

Proof Sketch. These statements essentially follow from the fact that for any n-variate, degree-d polynomial, we can move between its coefficient vector and evaluation vector using linear transformations. Clearly, both these transformations have (constant-free) algebraic circuits of size $\mathrm{poly}(N)$. Since both VP and VNP are rich enough to implement the above circuits, we get all the required statements. □

We now state the upper bound on equations for *evaluation vectors* of VP.

[15] The order can be picked arbitrarily.

Lemma 4.7. *There exists a constant c, such that for all large enough $n, d, s \in \mathbb{N}$, a multilinear equation for the* evaluation vectors *of the set of n-variate, degree-d polynomials computable by circuits of size s, is computable by constant-free circuits with projection gates of size at most $(nds)^c$.*

Proof. For the given parameters n, d, s, let $N = \binom{n+d}{n}$ and consider the universal circuit $\mathcal{U}(\mathbf{x}_{[n]}, \mathbf{y}_{[r]})$ as given in Lemma 2.11. Let $I \subseteq \mathbb{C}^n$ be an interpolating set of size N given by Proposition 4.4. Define a polynomial map $\mathcal{U}_{n,d,s} : \mathbb{C}^r \to \mathbb{C}^N$ such that each co-ordinate of $\mathcal{U}_{n,d,s}$ is exactly $\mathcal{U}(\mathbf{x} = \mathbf{a}, \mathbf{y})$ for a unique point $\mathbf{a} \in I$. We also know from Lemma 2.11 that $\mathcal{U}_{n,d,s}$ can be encoded by a *constant-free* circuit of size $\text{poly}(n, d, s)$, and has individual degree at most $\text{poly}(n, d, s)$.

Since any n-variate, degree-d polynomial that is computable by circuits of size s can be obtained by fixing the $\mathbf{y}$ variables to some scalars in $\mathcal{U}(\mathbf{x}, \mathbf{y})$, we have that any annihilator of $\mathcal{U}_{n,d,s}$ vanishes on the evaluation vector of *every* such polynomial.

Now, using the proof of Theorem 3.1, we can construct a *multilinear*, $\text{poly}(n, d, s)$-variate annihilator A for $\mathcal{U}_{n,d,s}$, that is computable by constant-free circuits with projections, of size at most $(nds)^c$ for some constant c. That the annihilator A can be multilinear is guaranteed by the bound on the individual degree from Lemma 3.2. □

Theorem 1.4 then follows as a simple corollary. Observe that the $\log^* n$ in the exponent can be replaced by any growing function of n.

Theorem 1.4 (Equations for VP)**.** *For an arbitrary $d(n) \in \text{poly}(n)$, let $N(n) = \binom{n+d(n)}{n}$. Then, for $t(n) := n^{\log^* n}$, there is a family of $N(n)$-variate, multilinear polynomials $\{P_N\}$ that depends only on the first $t(n)$ variables, satisfying the following.*

- *The family $\{P_N\}$ is a family of equations for* the evaluation vectors *of* $\mathsf{VP}_{d(n)}$.
- *The coefficient functions of $\{P_N\}$ are computable in (uniform) space $t(n) = n^{\log^* n}$.*

Proof. Let c be a constant as chosen in the proof of Lemma 4.7, $\beta > 2$ be a constant to be fixed later and let $s = s(n) = n^{\log^* n/(\beta \cdot c)}$. Using Lemma 4.7, define a multilinear polynomial family $\{P_N\}$ such that, for each n, $P_N(Z_1, \ldots, Z_N)$ is an annihilator for the set of n-variate, degree-d, size-s polynomials and depends only on the first $(nds)^{2c} \leq n^{\log^* n} = t(n)$ variables. As each P_N is computable using constant-free algebraic circuits with projections, of size $s^{3c} \leq t(n)$, the family $\{P_N\}$ is in $\mathsf{VPSPACE}^0$. Further, since the family of universal circuits $\{\mathcal{U}_{n,d,s}\}$ is $\mathsf{DTIME}(\text{poly}(n, d, s))$-uniform, using the characterization of $\mathsf{VPSPACE}^0$ due to Koiran and Perifel (Definition 2.18), the coefficient functions of $\{P_N\}$ can be computed by a Turing machine that uses space $s(n)^{3c \cdot \beta'}$ for some constant β'. We now fix $\beta = 3\beta'$ so that $s(n)^{3c \cdot \beta'} \leq t(n)$.

Let $\{f_n\} \in \mathsf{VP}_d$ be arbitrary. We know that for all large enough n, $\text{size}(f_n) = n^a$ for some constant a. Thus, there is a finite $n_0 \in \mathbb{N}$ such that $\text{size}(f_n) \leq s(n)$ for all $n > n_0$. Therefore, for all large enough n, the evaluation vectors of f_n are zeroes of the polynomial $P_N(Z_1, \ldots, Z_N)$ and so $\{P_N\}$ is a family of equations whose coefficients are computable in (uniform) space $t(n)$. □

4.3 Lower Bounds from Succinct Hitting Set Generators

Finally, we prove Theorem 1.2 by combining the observations in the proofs of Theorem 1.4 and Theorem 1.9. We start by proving the following consequence of a separation between $\mathsf{VPSPACE}_b$ and VP. Here, by $\mathsf{VPSPACE}_b(m)$ we mean the class of m-variate polynomial families in $\mathsf{VPSPACE}_b$;

when comparing this with $\mathsf{VP}(N)$, we treat them as N-variate families that depend only on the first m inputs.

Proposition 4.8. *Assuming the Generalized Riemann Hypothesis, suppose that for every growing function $f(n)$, and $m(n) := \log^{f(n)} n$, we have that $\mathsf{VPSPACE}_b(m) \not\subseteq \mathsf{VP}(n)$.*

Then either $\mathsf{VP} \neq \mathsf{VNP}$, or for every $g(N) = \omega(1)$ there is a boolean function family $\{h_N\}$ that depends only on its first $M = \exp((\log\log N)^{g(N)})$-inputs, such that:
(a) $\{h_N\}$ has $\mathsf{DLOGTIME}$-uniform, $O(\log M)$-depth, $\mathrm{poly}(M)$-sized, fan-in 2 boolean circuits, and
(b) $\{h_N\} \notin \mathsf{DLOGTIME}$-uniform TC^0.
Here the uniformity is in terms of the circuit size, in both cases.

Proof. Firstly, note that the point (a) above is the same as saying: $\{\tilde{h}_M\} \in \mathsf{DLOGTIME}$-uniform NC^1, for the family $\{\tilde{h}_M\}$ defined so that $\tilde{h}_M = h_N$ for all N large enough. We now prove the statement by contradiction.

So suppose that the hypothesis holds, but not the conclusion. Therefore, $\mathsf{VP} = \mathsf{VNP}$ and for some $g(N) = \omega(1)$, every $\{\tilde{h}_M\}$ in $\mathsf{DLOGTIME}$-uniform NC^1, has a $\mathsf{DLOGTIME}$-uniform constant-depth threshold circuit of size N, where N is such that $M = \exp((\log\log N)^{g(N)})$.

Now define $f(n) = g(2^n)$; note that $f(n) = \omega(1)$. Thus, from the hypothesis, we have that $\mathsf{VPSPACE}(\log^{f(n)} n) \not\subseteq \mathsf{VP}(n)$. Now using Proposition 2.27, we get that $\mathsf{SPACE}(\log^{f(n)} n) \not\subseteq \mathsf{CH}$. So, by instantiating Proposition 2.26 for $m(n) = \frac{\log^{f(n)} n}{a}$ (where a is the absolute constant), we get that there is a function family $\{p_L\}$ on $L = 2^{\log^{f(n)} n}$ inputs, which has $\mathsf{DLOGTIME}$-uniform NC^1 circuits, but requires $\mathsf{DLOGTIME}$-uniform constant depth threshold circuits of size $2^{n^{\omega(1)}}$.

However, from our previous observation, taking $N = 2^n$, we have $L = 2^{m(n)} = 2^{(\log\log N)^{g(N)}}$, and therefore $\{p_L\}$ should have $\mathsf{DLOGTIME}$-uniform constant-depth threshold circuit of size N. This is a contradiction. □

Theorem 1.2 (Hardness from Succinct Hitting Sets)**.** *Assuming the Generalized Riemann Hypothesis, if VP-succinct hitting set generators exist for VP, then at least one of the following must be true.*

1. $\mathsf{VP} \neq \mathsf{VNP}$.
2. *For any $\ell(m) = o(1)$, there is a family of functions $\{h_m\}$ in uniform NC^1 such that any uniform constant-depth threshold circuit computing it must have size larger than $\exp(\exp(\log^{\ell(m)} m))$, where the uniformity is $\mathsf{DLOGTIME}$ (in terms of the respective sizes).*

Further, if VP does not admit VNP-natural proofs then $\mathsf{P} \neq \mathsf{SPACE}(\log^{\log^*(n)}(n))$.

Proof. Suppose we have VP_d-succinct hitting sets for VP, which means that for any $D(N), S(N) \in \mathrm{poly}(N)$, there is some family $\left\{g_n^{(D,S)}\right\} \in \mathsf{VP}_d$ that is a succinct hitting set generator for size-$S(N)$ families in VP_D (see Definition 2.15). Notice that over all choices of D, S, the family $\left\{g_n^{(D,S)}\right\}$ is a degree-$d(n)$ family; the different $\left\{g_n^{(D,S)}\right\}$s could potentially have different sizes $s(n) \in \mathrm{poly}(n)$.

Now consider the universal circuit family $\left\{\mathcal{U}_{n,d(n),t(n)}(\mathbf{x},\mathbf{y})\right\}$ (Definition 2.10) with a slightly super-polynomial size parameter, say $t(n) = n^{f(n)}$ for $f(n) = \omega(1)$. Observe that for every $\left\{g_n^{(D,S)}\right\} \in \mathsf{VP}_d$ above, there exists a large enough value of n, where $g_n^{(D,S)}(\mathbf{x}) = \mathcal{U}(\mathbf{x}, \mathbf{y} = \mathbf{a})$

for some $\mathbf{a}$ in the field. As a result, the family $\left\{\mathcal{U}_{n,d(n),t(n)}(\mathbf{x},\mathbf{y})\right\}$ is a succinct hitting set generator for all of VP. That is, for $N := \binom{n+d(n)}{d(n)}$, for any family $\{P_N\} \in \mathsf{VP}(N)$, there are infinitely many values of N where P_N will *not vanish* on the coefficient vector of $\mathcal{U}$ (which is a vector of polynomials over the parameter variables $\mathbf{y}$). Using Proposition 4.6, this means that the same property holds even for the evaluation vectors of $\mathcal{U}$. Therefore, the family $\{\mathcal{U}\}$ is a family of degree-$d(n)$ polynomial maps that are encoded by circuits of size $t(n)$, such that any family of annihilators for these maps is outside $\mathsf{VP}(N)$.

We can now apply Theorem 3.1 to obtain a family of annihilators for $\{\mathcal{U}\}$, say $\{A_m\}$ in the class $\mathsf{VPSPACE}_b$, with $m = 2t(n)$, and degree at most $t(n)^5$ for all large enough n. Due to the hitting property of $\{U\}$, $\{A_m\} \notin \mathsf{VP}(N)$, and thus, $\mathsf{VPSPACE}(t(n)) \not\subseteq \mathsf{VP}(N)$. Since this works for $t(n) = n^{f(n)}$ for any $f(n) = \omega(1)$, we are in the setting of Proposition 4.8.

Therefore, either $\mathsf{VP} \neq \mathsf{VNP}$, or for any $\gamma(N) = \omega(1)$ there is a family $\{h_N\}$ depending only on its first $M = \exp((\log\log N)^{\gamma(N)})$ inputs with the respective properties. We can then view this family as an M-variate family $\{\tilde{h}_M\}$ in uniform NC^1 — as in the proof of Proposition 4.8 — that requires uniform constant-depth threshold circuits of size strictly larger than N.

Finally, since M is a growing function of N, there is some $\lambda(M) = \omega(1)$ such that $N \leq \exp(\exp(\log^{1/\lambda(M)} M))$, and we can then let $g(M) = 1/\lambda(M) = o(1)$ to derive the second conclusion. Again, as the guarantee from Proposition 4.8 holds for any growing $\gamma(N)$, the conclusion also holds for any $g(M) = o(1)$ as required.

VP-succinct hitting sets for VNP. In this case, we get that the universal circuit family $\{U\}$ is a hitting set generator for all of VNP, using the exact same argument as above. As a result, the annihilator family $\{A_m\} \notin \mathsf{VNP}$. Therefore, if it were the case that $\mathsf{SPACE}(\log^{\log^* N} N) \subseteq \mathsf{P}$, then the coefficient function of $\{A_m\}$ — taking $m = n^{\log^* n/c} = \log^{\log^* N} N$ for some constant c — would be computable in $\mathrm{DTIME}(\mathrm{poly}(N))$, thus putting it in VNP using Valiant's criterion [Val79a], leading to a contradiction. This finishes the proof. □

Acknowledgements

We thank C. Ramya and Ramprasad Saptharishi for discussions about this problem during its early stages. We are grateful to Mrinal Kumar, Ramprasad Saptharishi and Amir Shpilka for numerous informative discussions on algebraic natural proofs, and algebraic complexity in general.

We thank Robert Andrews, Vishwas Bhargava, Christian Ikenmeyer, Meena Mahajan and Nitin Saxena for their helpful comments on a presentation of these results at the Worskhop on Algebra and Computation (WAC 2023); we also thank the organizers of this workshop for this opportunity. Finally, we would like to acknowledge the workshop on Proof Complexity and Meta-mathematics at the Simons institute, which facilitated various discussions regarding natural proofs with many experts in theoretical CS. These interactions motivated this work and hence we thank the organizers of the workshop for inviting us.

We thank Robert Andrews for his comments on an earlier version of the paper that helped us improve the presentation.

References

[AD08] Scott Aaronson and Andrew Drucker. Arithmetic natural proofs theory is sought. Shtetl Optimized: Scott Aaronson's Blog, 2008. [Cited on pages 1 and 9.]

[AV08] Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. eccc:TR08-062. [Cited on page 1.]

[BDS24] C. S. Bhargav, Prateek Dwivedi, and Nitin Saxena. Lower Bounds for the Sum of Small-Size Algebraic Branching Programs. In *Theory and Applications of Models of Computation - 18th Annual Conference, TAMC 2024, Hong Kong, China, May 13-15, 2024, Proceedings*, volume 14637 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 2024. [Cited on page 1.]

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147 – 150, 1984. [Cited on page 7.]

[BP20] Markus Bläser and Anurag Pandey. Polynomial Identity Testing for Low Degree Polynomials with Optimal Randomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 8:1–8:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. [Cited on page 24.]

[BS83] Walter Baur and Volker Strassen. The Complexity of Partial Derivatives. *Theoretical Computer Science*, 22:317–330, 1983. [Cited on page 1.]

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000. [Cited on pages 1 and 15.]

[Bür09] Peter Bürgisser. On Defining Integers And Proving Arithmetic Circuit Lower Bounds. *Computational Complexity*, 18(1):81–103, 2009. [Cited on page 16.]

[CKR+20] Prerona Chatterjee, Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. On the Existence of Algebraically Natural Proofs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 870–880. IEEE, 2020. Pre-print available at arXiv:2004.14147. [Cited on page 13.]

[CKSS24] Prerona Chatterjee, Deepanshu Kush, Shubhangi Saraf, and Amir Shpilka. Lower Bounds for Set-Multilinear Branching Programs. In *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. [Cited on page 1.]

[CKSV22] Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. Quadratic Lower Bounds for Algebraic Branching Programs and Formulas. *Comput. Complex.*, 31(2):8, 2022. [Cited on page 1.]

[CKW11] Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial Derivatives in Arithmetic Complexity. *Foundations and Trends in Theoretical Computer Science*, 2011. [Cited on page 1.]

[CMTV98] Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic NC^1 Computation. *J. Comput. Syst. Sci.*, 57(2):200–212, 1998. [Cited on page 15.]

[Csa76] L. Csanky. Fast Parallel Matrix Inversion Algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976. [Cited on page 7.]

[FLMS15] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower Bounds for Depth-4 Formulas Computing Iterated Matrix Multiplication. *SIAM Journal of Computing*, 44(5):1173–1201, 2015. Preliminary version in the *46th Annual ACM Symposium on Theory of Computing (STOC 2014)*. eccc:TR13-100. [Cited on page 1.]

[FS12] Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 163–172. ACM, 2012. Pre-print available at eccc:TR11-147. [Cited on pages 7 and 12.]

[FS18] Michael A. Forbes and Amir Shpilka. A PSPACE construction of a hitting set for the closure of small algebraic circuits. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 1180–1192. ACM, 2018. arXiv:1712.09967. [Cited on page 24.]

[FSV18] Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct Hitting Sets and Barriers to Proving Lower Bounds for Algebraic Circuits. *Theory of Computing*, 14(1):1–45, 2018. Preliminary version in the *49th Annual ACM Symposium on Theory of Computing (STOC 2017)*. arXiv:1701.05328. [Cited on pages , 1, 2, 3, 9, and 13.]

[GKKS14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *Journal of the ACM*, 61(6):33:1–33:16, 2014. Preliminary version in the *28th Annual IEEE Conference on Computational Complexity (CCC 2013)*. Pre-print available at eccc:TR12-098. [Cited on page 1.]

[GKSS17] Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. Pre-print available at arXiv:1701.01717. [Cited on pages , 1, 2, 3, and 13.]

[GR05] Ariel Gabizon and Ran Raz. Deterministic Extractors for Affine Sources over Large Fields. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 407–418, 2005. [Cited on page 8.]

[Gro15] Joshua A. Grochow. Unifying Known Lower Bounds via Geometric Complexity Theory. *Computational Complexity*, 24(2):393–475, 2015. [Cited on page 1.]

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. [Cited on page 9.]

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 262–272, 1980. [Cited on pages 2, 9, and 24.]

[Kal85] Kyriakos Kalorkoti. A Lower Bound for the Formula Size of Rational Functions. *SIAM Journal of Computing*, 14(3):678–687, 1985. [Cited on page 1.]

[Kay09] Neeraj Kayal. The Complexity of the Annihilating Polynomial. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 184–193, 2009. [Cited on page 3.]

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*. [Cited on page 4.]

[KP09] Pascal Koiran and Sylvain Perifel. VPSPACE and a Transfer Theorem over the Reals. *Computational Complexity*, 18(4):551–575, 2009. [Cited on pages 14 and 15.]

[KRST22] Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. If VNP Is Hard, Then so Are Equations for It. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 44:1–44:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. arXiv:2012.07056. [Cited on pages 2 and 9.]

[Mal11] Guillaume Malod. Succinct Algebraic Branching Programs Characterizing Non-uniform Complexity Classes. In *Fundamentals of Computation Theory - 18th International Symposium, FCT 2011, Oslo, Norway, August 22-25, 2011. Proceedings*, pages 205–216, 2011. [Cited on pages 7, 14, 15, 16, 31, and 32.]

[MR13] Meena Mahajan and B. V. Raghavendra Rao. Small Space Analogues of Valiant's Classes and the Limitations of Skew Formulas. *Computational Complexity*, 22(1):1–38, 2013. [Cited on page 15.]

[Mul12] Ketan Mulmuley. Geometric Complexity Theory V: Equivalence between Blackbox Derandomization of Polynomial Identity Testing and Derandomization of Noether's Normalization Lemma. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 629–638, 2012. [Cited on page 24.]

[MV97] Meena Mahajan and V. Vinay. A Combinatorial Algorithm for the Determinant. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997)*, pages 730–738, 1997. Available on citeseer:10.1.1.31.1673. [Cited on pages 7 and 17.]

[Poi08] Bruno 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. [Cited on pages 14, 31, and 32.]

[Raz10] Ran Raz. Elusive Functions and Lower Bounds for Arithmetic Circuits. *Theory of Computing*, 6(1):135–177, 2010. [Cited on page 13.]

[RR97] Alexander A. Razborov and Steven Rudich. Natural Proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. Preliminary version in the *26th Annual ACM Symposium on Theory of Computing (STOC 1994)*. [Cited on page 1.]

[Sap15] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015. [Cited on pages 1 and 17.]

[Smo97] Roman Smolensky. Easy Lower Bound for a Strange Computational Model. *Comput. Complex.*, 6(3):213–216, 1997. [Cited on page 1.]

[Str73] Volker Strassen. Die Berechnungskomplexität Von Elementarsymmetrischen Funktionen Und Von Interpolationskoeffizienten. *Numerische Mathematik*, 20(3):238–251, 1973. [Cited on page 1.]

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010. [Cited on pages 1 and 12.]

[Val79a] Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 249–261, 1979. [Cited on pages 1, 16, and 27.]

[Val79b] Leslie G. Valiant. The Complexity of Computing the Permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. [Cited on page 1.]

A Equivalence of the definitions of VPSPACE

This section is devoted to proving Proposition 2.21, restated below.

Proposition 2.21 ([Poi08, Mal11]). $\mathsf{VPSPACE}^0 = \mathsf{VPPROJ}^0$.

Throughout this section, whenever a bit-string corresponds to an integer, the 0^{th} bit refers to the sign-bit, the first bit to the most significant bit (MSB) and the last bit to the least significant bit (LSB).

Before moving to Proposition 2.21, we sketch the proofs of a few basic facts that will be needed.

Observation A.1 (Adding in small space)**.** *Let A, B be positive integers such that $A + B$ is at most 2^{2^s}. Assuming bit-access to A and B, any bit of the sum $A + B$ can be computed in space at most $O(s)$.*

Sketch. Let $i \in [s]$ be the index of the bit of $A + B$ that we wish to compute. This will be done in i iterations, starting from the LSB to the i^{th} bit. We maintain a counter j (using s bits) to keep track of the current iteration. We also use 4 additional bits: a, b, r and c, for A, B, the result and the carry, respectively, all initialized to 0.

In iteration j, we get the j^{th} bits of A and B in a and b, respectively. We then set $r \leftarrow a \oplus b \oplus c$, and $c \leftarrow (a \wedge b) \vee (b \wedge c) \vee (a \wedge c)$, in that order. At the end of iteration i, we output r as the i^{th} bit of $A + B$.

The total space used is, clearly, at most $O(s)$ for all large s. □

Observation A.2 (Subtracting in small space)**.** *Let A, B be integers of absolute value at most 2^{2^s} such that the absolute value of $A - B$ is at most 2^{2^s}. Assuming bit-access to A and B, any bit of the difference $A - B$ can be computed in space at most $O(s)$.*

Sketch. As an initial step, we query the sign-bits (0^{th} bits) of A and B, to decide whether the absolute value of $A - B$ is the sum of the difference of their absolute values.

The rest of the algorithm is the same as that for Observation A.1, but with the correct expressions for computing r and c in each of the iterations. In order to compute the sign-bit of $A - B$, we need to check the sign-bits of A and B, and check if there is a carry from iteration 1. □

Observation A.3 (Summing a list in small space)**.** *Let $A_1, A_2, \ldots, A_L$ be positive integers such that their sum $T := A_1 + A_2 + \cdots + A_L$ is at most 2^{2^s}. Assuming bit-access to each of the A_is, any bit of the sum T can be computed in space at most $O(\lceil \log L \rceil \cdot s)$.*

Sketch. We prove this using Observation A.1, and induction on the size of the list, L. Assuming bit-access to the sums $T_1 = A_1 + \cdots + A_{L/2}$, and $T_2 = A_{L/2+1} + \cdots + A_L$, we can compute the sum T in space $2s$. In order to simulate bit-access to T_1 and T_2, we recursively run the algorithm of the two smaller lists. The key observation is that we can reuse the same $2 \cdot (\lceil \log L \rceil - 1) \cdot s$ bits of space for both these computations. The total space used is thus, at most $O(\lceil \log L \rceil \cdot s)$. □

Observation A.4 (Multiplying in small space)**.** *Let A, B be positive integers such that $A \times B$ is at most 2^{2^s}. Assuming bit-access to A and B, any bit of the product $A \times B$ can be computed in space at most $O(s^3)$.*

Sketch. Using the "school method" the product $A \times B$ can be calculated as the sum of $L = \lceil \log B \rceil$ many numbers, each of which is either 0, or a shift of A.

Therefore, we can use Observation A.3 by simulating bit-access to each of the shifts mentioned above. Here, the j^{th} bit in the k^{th} shift of A, is either 0 (if the k^{th} bit of B is 0), or the $(j + k - 1)^{th}$ bit of A. So we just need an additional space of at most $2s$ bits for this simulation.

The total space used is therefore $O(s^2) + O(s) \leq O(s^3)$, for all large enough s. □

We are now ready to prove Proposition 2.21, which we restate once more.

Proposition 2.21 ([Poi08, Mal11])**.** $\mathsf{VPSPACE}^0 = \mathsf{VPPROJ}^0$.

The statement clearly follows from the following lemmas, namely Lemma A.5 and Lemma A.6.

Lemma A.5. *Let s be large enough, and suppose $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ is computable by a constant-free, algebraic circuit with projections gates, of size s.*

Then, the coefficient function[16] Φ of f, is computable by a Turing machine that uses space at most $O(s^4)$, and takes as advice a string of length at most $O(s^2)$.

[16] Definition 2.17.

Proof. Let C be the constant-free circuit with projection gates that computes f. The advice to the Turing machine is going to be an encoding of the graph of C.

We can assume that $C(\mathbf{x}) = C_1(\mathbf{x}) - C_2(\mathbf{x})$, with C_1, C_2 being monotone (that is, they only use the positive constant 1, and addition and multiplication gates). We shall therefore assume that C itself is monotone, and show that its coefficient function can be computed in small space, and then just use Observation A.2 to finish the proof.

For each gate $g \in C$, let Φ_g be its coefficient function.

- If g is a leaf, then its coefficient function is trivial.
- If $g = u + v$, then Φ_g can be decided in space $O(s)$ using Observation A.1, by using an additional space of at most $O((s-1)^4)$ bits, to simulate Φ_u and Φ_v whenever required.
- If $g = \text{fix}_{z=0}\, u(z, \mathbf{x})$, then Φ_g is essentially Φ_u with the exponent for z fixed to zero. When $g = \text{fix}_{z=1}\, u(z, \mathbf{x})$, for any monomial $\mathbf{x}^{\mathbf{e}}$, $\text{coeff}_g(\mathbf{x}^{\mathbf{e}}) = \sum_{0 \le a \le \deg_z(u)} \text{coeff}_u(\mathbf{x}^{\mathbf{e}} \cdot z^a)$. Therefore, we can calculate Φ_g using Φ_u and Observation A.3.
- If $g = u \times v$, then the coefficient of a monomial $\mathbf{x}^{\mathbf{e}}$ in g is given by $\sum_{\mathbf{e}'} \text{coeff}_u(\mathbf{x}^{\mathbf{e}'}) \text{coeff}_v(\mathbf{x}^{\mathbf{e}-\mathbf{e}'})$. The degree of f is at most 2^s, and hence there are at most $2^{ns} \le 2^{s^2}$ terms in the above sum. Therefore, if we can simulate bit-access to each of the product terms, then we can compute any bit of the sum in additional space $O(s^3)$ using Observation A.3. Since we can inductively compute the coefficient functions of both u and $v$in space $O((s-1)^4)$, Observation A.4 gives us bit-access to each of the terms in the sum, using an additional space of at most $O(s^3)$ bits. Putting all the pieces together, Φ_g can be decided in space at most $O((s-1)^4 + s^3)$.

Thus, the total space used is at most $O((s-1)^4 + s^3) \le O(s^4)$, for all large s. □

Lemma A.6. *Let $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ be an n-variate, degree d polynomial such that its coefficient function can be computed in space s. Then there is a constant-free circuit with projection gates of size $\text{poly}(s, n, \log d)$ that can compute $f(\mathbf{x})$.*

Proof. We prove this in two steps. First we show that the coefficient function can be efficiently computed by a constant-free algebraic circuit with projection gates. We then show that if the coefficient function of a polynomial can be efficiently computed by a constant-free algebraic circuit with projection gates, then the polynomial itself can be efficiently computed by a constant-free algebraic circuit with projection gates.

Claim A.7. *Let $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ be an n-variate, degree d polynomial such that its coefficient function can be computed in space s. Then there is a constant-free algebraic circuit with projection gates of size $\text{poly}(s, n, \log d)$ that computes the coefficient function of f.*

Proof. Using the fact that totally quantified boolean expressions capture PSPACE, since the coefficient function of f can be computed using space s, we note that the coefficient function can be written as a quantified boolean formula of size $\text{poly}(s)$.

We now *arithmetize* the boolean expression in the usual way to get an algebraic circuit. Additionally, we use projection gates to simulate the quantifiers using constant sized gadgets:

- $\forall y \Psi(y) \equiv \text{fix}_{y=0}\, p(y) \times \text{fix}_{y=1}\, p(y)$,
- $\exists y \Psi(y) \equiv 1 - (1 - \text{fix}_{y=0}\, p(y)) \times (1 - \text{fix}_{y=1}\, p(y))$.

We therefore get an algebraic circuit with projection gates, of size $\text{poly}(s, n, \log d)$ that computes the coefficient function of f. □

Claim A.8. *Let $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ be an n-variate, degree d polynomial such that its coefficient function can be computed by a constant-free algebraic circuit with projection gates of size s'. Then there is a constant-free algebraic circuit with projection gates of size $\text{poly}(s', n, \log d)$ that computes f.*

Proof. It is easy to see that projection gates can efficiently simulate exponential sums. So it suffices to provide an exponential sum that computes $f(\mathbf{x})$ given access to a circuit, say $\text{CF}(\mathbf{y}_e, \mathbf{i})$, computing the coefficient function, as follows. Here $m = n \cdot \lceil \log d \rceil$ is the length of the bit-vectors for exponents, and b is the base-2-logarithm of the bit complexity of the coefficients of f.

$$f(\mathbf{x}) = \sum_{\mathbf{y}_e \in \{0,1\}^m} \left(\sum_{\mathbf{i} \in \{0,1\}^b} \text{pow}(\mathbf{i}) \cdot \text{CF}(\mathbf{y}_e, \mathbf{i}) \right) \cdot \text{check}(\mathbf{y}_e) \cdot \text{mon}(\mathbf{x}, \mathbf{y}_e)$$

The polynomials pow computes the i^{th} power of two, check outputs 1 when the exponent $\mathbf{e}$ is valid, and zero otherwise, and mon computes the $\mathbf{e}^{th}$ monomial in $\mathbf{x}$. All these polynomials have easy constructions in size $\text{poly}(m, b) = \text{poly}(n, \log d, s)$. □

The required statement clearly follows from the above two claims. □