Electronic Colloquium on Computational Complexity, Report No. 11 (2010)

Read-Once Polynomial Identity Testing [*]

Amir Shpilka[†] Ilya Volkovich[†]

Abstract

An *arithmetic read-once formula* (ROF for short) is a formula (a circuit whose underlying graph is a tree) in which the operations are $\{+, \times\}$ and such that every input variable labels at most one leaf. A *preprocessed ROF* (PROF for short) is a ROF in which we are allowed to replace each variable x_i with a univariate polynomial $T_i(x_i)$. In this paper we study the problems of giving deterministic identity testing for models related to preprocessed ROFs. Our main result gives PIT algorithms for the sum of k preprocessed ROFs, of individual degrees at most d (i.e. each $T_i(x_i)$ is of degree at most d), that run in time $(nd)^{\mathcal{O}(k)}$ in the non black-box model and in time $(nd)^{\mathcal{O}(k+\log n)}$ in the black-box model. We also obtain better algorithms where the formulas have a small depth that lead to an improvement on the best PIT algorithm for multilinear $\Sigma\Pi\Sigma(k)$ circuits.

Our main technique is to prove a *hardness of representation* result. Namely, a theorem showing a relatively mild lower bound on the sum of k PROFs. We then use this lower bound in order to design our PIT algorithm.

[*]Extended abstract appeared in the Proceedings of the 40th Annual STOC 2008, pages 507-516 and in the Proceedings *APPROX-RANDOM*, 2009, pages 700-713

[†]Faculty of Computer Science, Technion, Haifa 32000, Israel. Email: `{shpilka,ilyav}@cs.technion.ac.il`. Research supported by the Israel Science Foundation (grant number 439/06).

ISSN 1433-8092

1 Introduction

In this paper we study the polynomial identity testing problem for several models based on read-once formulas. In the polynomial identity testing problem (PIT for short) we are given (either explicitly or via black-box access) an arithmetic circuit (or formula) and we have to decide whether the circuit computes the zero polynomial. This problem has a well known randomized algorithm due to Schwartz, Zippel and DeMillo and Lipton [Zip79, Sch80, DL78]. In this work however, we are interested in the question of giving a deterministic algorithm to the problem.

In general, the PIT problem is believed to be very difficult and several results connecting deterministic algorithms for PIT and lower bounds for arithmetic circuits are known [HS80, KI04, Agr05, DSY09, AV08]. However, for several special cases in which the underlying circuit comes from a restricted class of arithmetic circuits, efficient deterministic PIT algorithms were found. For example, efficient deterministic identity testing algorithms are known for depth-2 arithmetic circuits [BOT88, KS01, LV03] (and references within), for depth-3 arithmetic circuits with bounded top fan-in (also known as $\Sigma\Pi\Sigma(k)$ circuits) [DS06, KS07, AM07, KS08, KS09b, SS09] and for non-commutative arithmetic formulas [RS05]. Interestingly, [AV08] showed that polynomial time deterministic black-box PIT algorithms for depth-4 arithmetic circuits imply exponential lower bounds on the size of general arithmetic circuits and a quasi-polynomial time algorithm for the general PIT problem. Indeed, efficient deterministic PIT algorithms are known only for restricted classes of depth-4 circuits [AM07, Sax08, KMSV10].

In view of the difficulty in providing efficient deterministic PIT algorithms and the tight connection to lower bounds it is natural to study the PIT problem for models for which lower bounds are known. In particular, the recent results of [Raz04, Raz05, RSY08, RY08], giving lower bounds for multilinear circuits and formulas, suggest that efficient deterministic PIT algorithms for multilinear formulas may be at reach. Unfortunately, except for the models of multilinear depth-2 circuits and multilinear $\Sigma\Pi\Sigma(k)$ and $\Sigma\Pi\Sigma\Pi(k)$ circuits no such algorithm is known. This difficulty motivates the study of restricted models of multilinear formulas in hope of gaining insight on the general case.

In this work we consider a restricted model of multilinear formulas - sums of read-once formulas (and the more general case of sums of preprocessed read-once formulas). An *arithmetic read-once formula* (ROF for short) is a formula (a circuit in which the fan-out of every gate is at most 1) in which the operations are $\{+, \times\}$ and such that every input variable labels at most one leaf. Read-once formulas can be thought of as the simplest form of multilinear formulas. Although ROFs form a very restricted model of computation they received a lot of attention both in the boolean world [KLN$^+$93, AHK93, BHH95b] and in the algebraic world [HH91, BHH95a, BB98, BC98]. However, no deterministic sub-exponential time black-box PIT algorithm for arithmetic ROF was known prior to this work. We give the first sub-exponential (in fact, quasi-polynomial) time deterministic PIT algorithms for (sums of) read-once arithmetic formulas in the black-box and non black-box settings. Besides being a relaxation of the general model of multilinear formulas, another motivation for our work is to better understand recent results on depth-3 circuits. It is not difficult to see that a multilinear depth-3 $\Sigma\Pi\Sigma(k)$ circuit is a sum of k read-once formulas of a very restricted form (i.e. each multiplication gate is a ROF). Thus, our work can be seen as a (significant) generalization and extension of previous results for multilinear $\Sigma\Pi\Sigma(k)$ circuit [DS06, KS07, KS08, SS09].

1.1 Our results and techniques

Our black-box PIT algorithms use the notion of *generators*. A generator for a circuit class $\mathcal{C}$ is a mapping $\mathcal{G} : \mathbb{F}^t \to \mathbb{F}^n$, such that for any nonzero polynomial P computed by a circuit from $\mathcal{C}$ it holds that $P \circ \mathcal{G} \not\equiv 0$. By considering the image of $\mathcal{G}$ on W^t, where $W \subseteq \mathbb{F}$ is of polynomial size, we obtain a hitting set for $\mathcal{C}$ (more details given in Section 3). We now state our results.

Theorem 1. *Given black-box access to a preprocessed read-once formula Φ in n variables, with individual degrees at most d, there is a deterministic algorithm that checks whether $\Phi \equiv 0$. The running time of the algorithm is $(nd)^{\mathcal{O}(\log n)}$.*

The intuition for the proof is that if a PROF is not zero then it can be written as combination of two smaller PROFs, such that one of them depends on at most $n/2$ variables. Using this observation we design a generator $G : \mathbb{F}^{\mathcal{O}(\log n)} \to \mathbb{F}^n$ such that $\Phi \circ G \not\equiv 0$. Our next result is an efficient non black-box PIT algorithm for the sum of a small number of PROFs.

Theorem 2. *Given k preprocessed read-once formulas in n variables, with individual degrees at most d, there is a deterministic algorithm that checks whether they sum to zero or not. The running time of the algorithm is $(nd)^{\mathcal{O}(k)}$.*

Combining Theorems 1,2 we obtain our main result: a black-box algorithm for the sum of k PROFs. The extension from a PIT algorithm for a single (preprocessed) ROF to an algorithm for the sum of k (preprocessed) ROFs is via *hardness of representation* approach. This approach enables us to transform a mild lower bound for a very structured polynomial into a PIT for sum of (preprocessed) ROFs.

Theorem 3 (Main). *Given black-box access to $\Phi = \Phi_1 + \cdots + \Phi_k$, where the Φ_i-s are preprocessed-read-once formulas in n variables, with individual degrees at most d, there is a deterministic algorithm that checks whether $\Phi \equiv 0$. The running time of the algorithm is $(nd)^{\mathcal{O}(\log n + k)}$.*

In fact, we design a generator $G : \mathbb{F}^{\mathcal{O}(\log n + k)} \to \mathbb{F}^n$, for sum of k PROFs. Using the same techniques we design a different generator for the sum of PROFs of a small depth (see Definition 6.4), which yields a better running time PIT algorithm.

Theorem 4. *There is an $(nd)^{\mathcal{O}(D+k)}$ time deterministic algorithm for checking whether a black-box holding the sum of k preprocessed depth-D read-once formulas on n-variables, with individual degrees bounded by d, computes the zero polynomial.*

As a corollary we obtain an $n^{\mathcal{O}(k)}$ time PIT algorithm for multilinear $\Sigma\Pi\Sigma(k)$ circuits (a multilinear $\Sigma\Pi\Sigma(k)$ circuit can be considered as a sum of k ROFs of depth-2), which gives the best known running-time algorithm for this circuit class. Moreover, our algorithm also holds for the more general case of preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuits (see Section 8 for the definition).

Theorem 5. *Let C be a preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuit with individual degrees bounded by d. Then there is a deterministic black-box PIT algorithm for C that runs in time $(nd)^{\mathcal{O}(k)}$.*

We note that, unlike previous works on $\Sigma\Pi\Sigma(k)$ circuits [DS06, KS08, SS09, KS09b], this result *does not* rely on bounds on the rank of zero $\Sigma\Pi\Sigma(k)$ circuits (see Section 8). In addition to the multilinear case, we obtain a new PIT algorithm (by constructing a appropriate generator) for general $\Sigma\Pi\Sigma(k)$ circuits that has (roughly) the same running time as the algorithm obtained from the results of [KS08, SS09].

Theorem 6. *Let C be a $\Sigma\Pi\Sigma(k, d)$ circuit over $\mathbb{F}$. There is a deterministic black-box PIT algorithm for C that runs in time $(nd)^{\mathcal{O}(k^3 \log d)}$ when $\mathbb{F}$ is finite and in time $(nd)^{k^{\mathcal{O}(k)}}$ when $\mathbb{F} = \mathbb{R}$ or $\mathbb{Q}$.*

Finally, we show that it is possible to generalize the previous theorems to the case where we have a sum of ROFs that is read-k. That is, every variable appears in at most k of the formulas (see Definition 7.8).

Theorem 7. *Let $\Phi = \sum_{m=1}^{r} \Phi_m$ be a read-k sum of PROFs. Then there is an $(nd)^{\mathcal{O}(\log n)}$ time deterministic black-box PIT algorithm for Φ. If in addition we are guaranteed that the Φ_m's are depth-D PROFs then there is an $(nd)^{\mathcal{O}(D+k)}$ time deterministic black-box PIT algorithm for Φ. In the non black-box setting there is an $(nd)^{\mathcal{O}(k)}$ deterministic PIT algorithm for Φ.*

1.2 Comparison to Previous Works

Read-once arithmetic formulas received a lot of attention in the context of learning theory and exact learning algorithms were given to them. We shall now discuss the different models in which it was studied and highlight the differences from our work.

In [HH91] learning algorithms for read-once arithmetic formulas that use *membership and equivalence* queries were given. A membership query to a ROF $\Phi(\bar{x})$ is simply a query that asks for the value of $\Phi(\bar{x})$ on a specific input. An equivalence query on the other hand, gives the oracle a certain hypothesis, $\Psi(\bar{x})$, and the oracle answers "equal" if $\Phi \equiv \Psi$ or returns an input $\bar{\alpha}$ such that $\Phi(\bar{\alpha}) \neq \Psi(\bar{\alpha})$. Following [HH91] other works gave learning algorithms for various extensions of read-once formulas [BHH95a, BB98, BC98]. All those works (including the original work [HH91]) give randomized learning algorithms. While our work only considers PIT algorithms it also led to new deterministic learning algorithms for read-once formulas (see [SV08, SV09]).

Our results are also related to the model of depth-3 $\Sigma\Pi\Sigma(k)$ circuits. This model was extensively studied in recent years [DS06, KS07, AM07, KS08, Shp09, SS09, KS09a, KS09b] as it stands between the simpler depth-2 case and the depth-4 case that, when studying lower bounds and polynomial identity testing, is (almost) as hard as the general case [AV08]. Prior to this work the best known black-box PIT algorithm for degree d $\Sigma\Pi\Sigma(k)$ circuits had running time $n^{\mathcal{O}(k^3 \log d)}$ for the general case [KS08, SS09] and $n^{2^{\mathcal{O}(k^2)}}$ for the multilinear case [KS08]. Both results were obtained via the rank-bound (see Section 8). We improve the algorithm for the multilinear case and obtain an $n^{\mathcal{O}(k)}$ algorithm that also works in the preprocessed case. Our PIT algorithm uses a different technique than previous approaches. In addition, applying a recent result of [SS09] we obtain a new PIT algorithm for the general case with (roughly) the same running time $(nd)^{\mathcal{O}(k^3 \log d)}$.

Note that Theorem 5 actually gives a PIT for a restricted class of depth-4 circuits. At the time of our work this was the first PIT algorithms for a class of depth-4 circuits. Other known PIT algorithms for depth-4 circuits were non black-box and cover only other very restricted cases. Arvind and Mukhopadhyay [AM07] gave a polynomial time PIT algorithm for the case that $k = \mathcal{O}(1)$ and the additional requirement that each linear function depends on a constant number of variables. Saxena [Sax08] gave a polynomial time PIT algorithm for the case where each linear product consists of a constant number of linear functions (but the top fan-in k may be unbounded). Very recently, [KMSV10] gave the first quasi-polynomial time black-box PIT algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits. This work uses several ideas that appeared in the conference versions of this paper [SV08, SV09].

1.3 Organization

The paper is organized as follows. In Section 2 we give the basic definitions and notations. Then, in Sections 3 and 4 we introduce the tools that we use in the proofs. Specifically, in Section 3 we define the notion of a *generator* for a class of arithmetic circuits and in Section 4 we show how to acquire a justifying assignment given a PIT algorithm. After this, we give the formal definition of our model (Section 5) and in the following sections we prove our theorems: The new black-box PIT algorithm for PROFs (Theorem 1) is given in Section 6. This section also contains a PIT algorithm for bounded depth PROFs (Theorem 4 for the case $k = 1$). In Section 7 we consider sums of PROFs and prove Theorems 2, 3, 4 and 7. Finally, in Section 8 we give PIT algorithm for depth-3 circuits and special cases of depth-4 circuits, proving Theorems 5 and 6.

2 Preliminaries

For a positive integer n we denote $[n] = \{1, \ldots, n\}$. Let $\mathbb{F}$ be a field and $\overline{\mathbb{F}}$ its algebraic closure. For a polynomial $P(x_1, \ldots, x_n)$, a variable x_i and a field element α we denote with $P|_{x_i=\alpha}$ the polynomial resulting after substituting α to x_i. The following definitions are for a polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$ and an assignment $\bar{a} \in \mathbb{F}^n$. We say that P *depends* on x_i if there exist $\bar{a} \in \overline{\mathbb{F}}^n$ and $b \in \overline{\mathbb{F}}$ such that $P(a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq P(a_1, a_2, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n)$. We denote $\mathrm{var}(P) \triangleq \{x_i \mid P \text{ depends on } x_i\}$. Sometimes, we will abuse notations and consider the set $\mathrm{var}(P)$ as a set of indices (i.e. $\{i \mid P \text{ depends on } x_i\}$) rather than variables.

Intuitively, P depends on x_i if x_i "appears" when P is listed as a sum of monomials. Given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$ we define $P|_{x_I=\bar{a}_I}$ to be the polynomial resulting from substituting a_i to the variable x_i for every $i \in I$. In particular $\mathrm{var}(P|_{x_I=\bar{a}_I}) \subseteq \{x_i \mid i \in [n] \setminus I\}$.

Example 2.1. *Let* $P(x_1, x_2, x_3) = 2x_2x_3 + 1$, $I = \{2\}$ *and* $\bar{a} = (0, 0, 0)$. *Then* $P|_{x_I=\bar{a}_I}(\bar{x}) = P(x_1, 0, x_3) = 1$. *Note that* $\mathrm{var}(P) = \{x_2, x_3\}$ *but* $\mathrm{var}(P|_{x_I=\bar{a}_I}(\bar{x})) = \emptyset$.

We can conclude that by substituting a value to a variable of P we, obviously, eliminate the dependence of P on this variable, however we may also eliminate the dependence of P on other variables and thus lose more information than intended. For the purposes of identity testing we cannot allow losing any information as it would affect our final answer. We now define a lossless type of an assignment. Similar definitions were given in [HH91] and [BHH95a], but we repeat the definitions here to ease the reading of the paper (we also slightly change some of the definitions).

Definition 2.2 (Justifying assignment). *We say that* $\bar{a} \in \mathbb{F}^n$ *is a* justifying assignment *of* P *if for each subset (of indices)* $I \subseteq \mathrm{var}(P)$ *we have that* $\mathrm{var}(P|_{x_I=\bar{a}_I}) = \mathrm{var}(P) \setminus I$. *We say that* $\bar{a}$ *is a* weakly-justifying assignment *of* P *if* $\mathrm{var}(P|_{x_I=\bar{a}_I}) = \mathrm{var}(P) \setminus I$ *when* $|I| = 1$. *We shall also think of justification as a property of the polynomial. We say that* P *is* $\bar{a}$-justified *if* $\bar{a}$ *is a justifying assignment of* P. *Similarly we define the term* weakly-$\bar{a}$-justified.

Clearly, justification implies weak-justification, but not vice versa. The following proposition provides a simple condition that ensures that an assignment is justifying for P.

Proposition 2.3. *An assignment* $\bar{a} \in \mathbb{F}^n$ *is a justifying assignment of* P *if and only if* $\mathrm{var}(P|_{x_I=\bar{a}_I}) = \mathrm{var}(P) \setminus I$ *for every subset* I *of size* $|\mathrm{var}(P)| - 1$.

Note that by shifting we can make any polynomial $\bar{0}$-justified .

Proposition 2.4. *Let* $\bar{a} \in \mathbb{F}^n$ *and* $P(\bar{x})$ *be a (weakly)* $\bar{a}$*-justified polynomial. Then* $P_{\bar{a}}(\bar{x}) \triangleq P(x_1 + a_1, \ldots, x_n + a_n)$ *is a (weakly)* $\bar{0}$*-justified polynomial. Moreover,* $P_{\bar{a}} \equiv 0$ *if and only if* $P \equiv 0$.

2.1 Partial Derivatives

The concept of a *partial derivative* of a multivariate polynomials and its properties (for example: P depends on x_i if and only if $\frac{\partial P}{\partial x_i} \not\equiv 0$) are well-known and well-studied for continuous domains (such as: $\mathbb{R}$,$\mathbb{C}$ etc.). Here we extend of the concept for polynomials over arbitrary fields, by defining the *Discrete* partial derivatives. Discrete partial derivatives will play a major role in the analysis of our algorithms.

Definition 2.5. *Let P be an n variate polynomial over a field $\mathbb{F}$. We define the* discrete partial derivative *of P with respect to x_i as $\frac{\partial P}{\partial x_i} = P|_{x_i=1} - P|_{x_i=0}$.*

Notice that if P is a multilinear polynomial then this definition coincides with the "analytical" one when $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$. The following lemma is easy to verify and we will use it implicitly from now on.

Lemma 2.6. *The following properties hold for every multilinear polynomial P; P depends on x_i if and only if $\frac{\partial P}{\partial x_i} \not\equiv 0$; $\frac{\partial P}{\partial x_i}$ does* not *depend on x_i (in particular $\frac{\partial^2 P}{\partial x_i^2} \equiv 0$); $\frac{\partial^2 P}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}(\frac{\partial P}{\partial x_j}) = \frac{\partial^2 P}{\partial x_j \partial x_i}$; $\forall i \neq j$ $\frac{\partial P}{\partial x_i}|_{x_j=a} = \frac{\partial}{\partial x_i}(P|_{x_j=a})$; $\bar{a} \in \mathbb{F}^n$ is a justifying assignment of P if and only if $\forall i \in \mathrm{var}(P)$ it holds that $\frac{\partial P}{\partial x_i}(\bar{a}) \not\equiv 0$.*

The following lemma shows that when dealing with multilinear polynomials the usual rules of partial derivatives hold for discrete partial derivatives.

Lemma 2.7. *Let P, G, Q be multilinear polynomials. Then the following derivation rules hold (with the appropriate implicit restrictions)*

1. ***Sum Rule**. If $Q = P + G$ then trivially $\frac{\partial Q}{\partial x_i} = \frac{\partial P}{\partial x_i} + \frac{\partial G}{\partial x_i}$.*
2. ***Product Rule**. If $Q = P \cdot G$ then either $\frac{\partial P}{\partial x_i} \equiv 0$ or $\frac{\partial G}{\partial x_i} \equiv 0$ holds. Hence, $\frac{\partial Q}{\partial x_i} = \frac{\partial P}{\partial x_i} \cdot G + P \cdot \frac{\partial G}{\partial x_i}$.*
3. ***Chain Rule**. Let $Q(y, \bar{z})$ be a polynomial such that $P(\bar{x}, \bar{z}) \equiv Q(G(\bar{x}), \bar{z})$. Then $\frac{\partial P}{\partial x_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial G}{\partial x_i}$. Notice that since Q is a multilinear polynomial, $\frac{\partial Q}{\partial y}$ does not depend on y.*

Note that these properties do not hold for general polynomials. For example, when $P(x) = x^2 - x$ we get that $\frac{\partial P}{\partial x} \equiv 0$. Thus, in order to handle general polynomial we need the following extension.

Definition 2.8. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. The* directed *partial derivative of P w.r.t. x_i and direction $\alpha \in \mathbb{F}$ is defined as $\frac{\partial P}{\partial_\alpha x_i} \triangleq P|_{x_i=\alpha} - P|_{x_i=0}$.*

We can now define the notion of a *witness*.

Definition 2.9. *We say that $0 \neq \alpha \in \mathbb{F}$ is a* witness *for x_i in P if $\frac{\partial P}{\partial_\alpha x_i} \not\equiv 0$ or $x_i \notin \mathrm{var}(P)$. The vector $\bar{\alpha} \in \mathbb{F}^n$ is a witness of P if each α_i is a witness for x_i in P.*

The following proposition is immediate.

Proposition 2.10. *Let $\bar{\alpha} \in \mathbb{F}^n$ be a witness for P. Then (for each i) P depends on x_i if and only if $\frac{\partial P}{\partial_{\alpha_i} x_i} \not\equiv 0$. Furthermore, $\bar{a} \in \mathbb{F}^n$ is a justifying assignment for P iff for every $x_i \in \mathrm{var}(P)$ it holds that $\frac{\partial P}{\partial_{\alpha_i} x_i}(\bar{a}) \neq 0$. I.e., $\bar{a}$ is a nonzero of $\frac{\partial P}{\partial_{\alpha_i} x_i}$.*

The next lemma shows that a sufficiently large field contains many witnesses.

Lemma 2.11. *Let $P(\bar{x})$ be a polynomial with individual degrees bounded by d and let $W \subseteq \mathbb{F}$ be a subset of size $d+1$ (we assume that $|\mathbb{F}| > d$). Then W^n contains a witness for P.*

Proof. Note that for each variable we can find the witness separately as they are uncorrelated. Consider $i \in [n]$ and define $\varphi_i(\bar{x}, w) \triangleq \frac{\partial P}{\partial_w x_i}$ (recall Definition 2.8). In this way we obtain a set of polynomials in the variables $\bar{x}$ and w with individual degrees bounded by d. Thus, α is a witness for x_i in P if and only if $\varphi_i(\bar{x}, \alpha) \not\equiv 0$ or $x_i \notin \mathrm{var}(P)$. If $x_i \notin \mathrm{var}(P)$ then any $\alpha \in W$ is a witness. Otherwise, $\varphi_i(\bar{x}, w)$ is a nonzero polynomial of degree d in w and therefore W contains a nonzero assignment for it (see e.g. Lemma 2.13). I.e. a witness for x_i. We can repeat the same reasoning for every $i \in [n]$. □

Definition 2.12. *For a non-empty subset $I \subseteq [n]$, $I = \left\{i_1, \ldots, i_{|I|}\right\}$, we define the iterated partial derivative with respect to I as $\partial_I P \triangleq \frac{\partial^{|I|} P}{\partial x_{i_1} \partial x_{i_2} \partial x_{i_3} \cdots \partial x_{i_{|I|}}}$.*

Let $\mathcal{C}, \mathcal{C}'$ be circuit class. We say that $\mathcal{C}$ *contains* the polynomial P, and denote it by $P \in \mathcal{C}$, if P can be computed by some circuit from $\mathcal{C}$. Similarly, we denote $\mathcal{C} \subseteq \mathcal{C}'$ whenever each polynomial in $\mathcal{C}$ also belongs to $\mathcal{C}'$. In such a case we say that $\mathcal{C}'$ *contains* $\mathcal{C}$. The class of (discrete) *Directed Partial Derivatives* of $\mathcal{C}$ is denoted by $\partial \mathcal{C} \triangleq \left\{ \frac{\partial P}{\partial_\alpha x_i} \mid P \in \mathcal{C}, i \in [n], \alpha \in \mathbb{F} \right\}$. We say that $\mathcal{C}$ is *closed under partial derivatives* if $\partial \mathcal{C} \subseteq \mathcal{C}$.

2.2 Some useful facts about polynomials

We conclude this section with two well-known facts concerning polynomials.

Lemma 2.13. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Suppose that for every $i \in [n]$ the individual degree of x_i is bounded by d_i, and let $S_i \subseteq \mathbb{F}$ be such that $|S_i| > d_i$. We denote $S = S_1 \times S_2 \times \cdots \times S_n$ then $P \equiv 0$ iff $P|_S \equiv 0$.*

A proof can be found in [Alo99].

Lemma 2.14 (Gauss). *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n, y]$ be a nonzero polynomial and $g \in \mathbb{F}[x_1, \ldots, x_n]$ such that $P|_{y=g(\bar{x})} \equiv 0$ then $y - g(\bar{x})$ is an irreducible factor of P in the ring $\mathbb{F}[x_1, x_2, \ldots, x_n, y]$.*

3 Generators and Hitting sets for arithmetic circuits

In this section, we formally define the notion of generators for polynomials, describe a few of its useful properties and give the connection to hitting sets. Intuitively, a generator $\mathcal{G}$ for a circuit class $\mathcal{C}$, is a function that stretches t independent variables into $n >> t$ dependent variables that can be *plugged* into any polynomial $P \in \mathcal{C}$ without vanishing it. Recall that a hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ for a circuit class $\mathcal{C}$ is a set such that for every nonzero polynomial $P \in \mathcal{C}$, there exists $\bar{a} \in \mathcal{H}$, such that $P(\bar{a}) \neq 0$. In identity testing, generators and hitting sets play the same role. Given a generator one can easily construct a hitting set by evaluation the generator on a large enough set of points. Conversely, given a hitting set $\mathcal{H}$ it is easy to construct a generator by taking a low degree curve through $\mathcal{H}$.

Definition 3.1. *A mapping $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^t \to \mathbb{F}^n$ is a* generator *for the circuit class $\mathcal{C}$ if for every nonzero n-variate polynomial P computed by $\mathcal{C}$ it holds that $P(\mathcal{G}) \not\equiv 0$.*

A generator can also be viewed as a mapping containing a hitting set for $\mathcal{C}$ in its image. That is, for every nonzero $P \in \mathcal{C}$ there exists $\bar{a} \in \mathrm{Im}(\mathcal{G})$ such that $P(\bar{a}) \neq 0$ (where $\mathrm{Im}(\mathcal{G}) \triangleq \mathcal{G}(\overline{\mathbb{F}}^t)$). All our black-box PIT algorithms are, in fact, generators for some (relatively) small t. The following is an immediate and important property of a generator.

Observation 3.2. *Let $P = P_1 \cdot P_2, \cdot \ldots \cdot P_k$ be a product of nonzero polynomials $P_i \in \mathcal{C}$ and let $\mathcal{G}$ be a generator for $\mathcal{C}$. Then $P(\mathcal{G}) \not\equiv 0$.*

We now describes an efficient way for constructing a generator to a circuit class $\mathcal{C}$ from a hitting set $\mathcal{H}$ for $\mathcal{C}$. The construction is performed by passing a low degree curve through $\mathcal{H}$ using polynomial interpolation: Choose an arbitrary subset $V \subseteq \mathbb{F}$ of size n and set $t \triangleq \lceil \log_n |\mathcal{H}| \rceil$. Clearly, $|\mathcal{H}| \leq n^t < n |\mathcal{H}|$. Denote $\mathcal{H} = \left\{\bar{a}^1, \bar{a}^2, \ldots, \bar{a}^{|\mathcal{H}|}\right\}$ where $\bar{a}^j = (a_1^j, a_2^j, \ldots, a_n^j)$. Let φ :

$V^t \to \{1,2,\ldots,|\mathcal{H}|\} \subseteq \mathbb{N}$ be some surjection. We define the functions $h_i(\bar{y}) : \mathbb{F}^t \to \mathbb{F}$ to be the interpolation polynomial of the i-th coordinates of the vectors in $\mathcal{H}$. That is, $h_i(\bar{y})$ is a t-variate polynomial, of degree at most $n-1$ in each variable, such that for every $\bar{b} \in V^t$ we have that $h_i(\bar{b}) = a_i^{\varphi(\bar{b})}$. Finally, let $h(\bar{y}) : \mathbb{F}^t \to \mathbb{F}^n$ be defined as $h(\bar{y}) \stackrel{\Delta}{=} (h_1(\bar{y}), h_2(\bar{y}), \ldots, h_n(\bar{y}))$. From the construction it is clear that $\mathcal{H} \subseteq \mathrm{Im}\,(h)$. We thus get that h is a generator for $\mathcal{C}$.

Lemma 3.3. *The procedure described above constructs a map $h(\bar{y}) : \mathbb{F}^t \to \mathbb{F}^n$ with individual degrees bounded by $n-1$, in time $\mathrm{poly}(n, |\mathcal{H}|)$, that is a generator for the circuit class $\mathcal{C}$.*

Proof. Let $P \in \mathcal{C}$ be a nonzero polynomial. From the definition of $\mathcal{H}$ there exists $\bar{a} \in \mathcal{H}$ such that $P(\bar{a}) \neq 0$. As $\mathcal{H} \subseteq \mathrm{Im}\,(h)$ it follows that $\bar{a} \in \mathrm{Im}\,(h)$ and consequently $P(h(\bar{y})) \not\equiv 0$. The claim regarding the degree follows form the construction of h_i-s. In addition, note that the h_i-s can be computed in time polynomial in $|\mathcal{H}|$ using simple interpolation. □

Next we describe the obvious way of obtaining a hitting set from a generator.

Lemma 3.4. *Let $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^t \to \mathbb{F}^n$ be a generator for a circuit class $\mathcal{C}$ such that the individual degrees of the $\mathcal{G}^i$-s are bounded by Δ. Let $W \subseteq \mathbb{F}$ be of size $nd\Delta$. Then, $\mathcal{H} \stackrel{\Delta}{=} \mathcal{G}(W^t)$ is a hitting set, of size $|\mathcal{H}| \leq (nd\Delta)^t$, for polynomials $P \in \mathcal{C}$ of individual degrees at most d.*

Proof. Let $P \in \mathcal{C}$ be a nonzero polynomial with individual degrees at most d. By definition, $P(\mathcal{G})$ is a nonzero t-variate polynomial with individual degrees bounded by $nd\Delta$. Lemma 2.13 implies that $P(\mathcal{G})|_{W^t} \not\equiv 0$. Equivalently, $P|_{\mathcal{H}} \not\equiv 0$. Finally, note that $|\mathcal{H}| \leq |W|^t \leq (nd\Delta)^t$. □

3.1 The Mapping G_k

In this section we define a mapping that will be one of the main ingredients in our PIT algorithms. We start with some notations. The *Hamming weight* of a vector $\bar{a} \in \mathbb{F}^n$ is defined as: $\mathrm{w_H}(\bar{a}) \stackrel{\Delta}{=} |\{i \mid a_i \neq 0\}|$. That is, the number of nonzero coordinates. For a set $0 \in W \subseteq \mathbb{F}$ and $k \leq t$ we define $\mathcal{A}_k^t(W)$ to be the set of all vectors in W^t with Hamming weight at most k, i.e. the set of vectors that have **at most** k nonzero coordinates. Formally: $\mathcal{A}_k^t(W) \stackrel{\Delta}{=} \left\{\bar{a} \in W^t \mid \mathrm{w_H}(\bar{a}) \leq k\right\}$. It is easy to see that $\left|\mathcal{A}_k^t(W)\right| = \sum_{j=0}^{k} \binom{t}{j} \cdot (|W|-1)^j = (t \cdot (|W|-1))^{\mathcal{O}(k)}$. From now on we assume that $|\mathbb{F}| > n$ as we are allowed to use elements from an appropriate extension field. Throughout the entire paper we fix a set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}$ of n distinct elements.

Definition 3.5. *For every $i \in [n]$ let $u_i(w) : \mathbb{F} \to \mathbb{F}$ be the i-th Lagrange Interpolation polynomial for the set A. Namely, $u_i(w)$ is a degree $n-1$ polynomial satisfying: $u_i(\alpha_j) = 1$ when $j = i$ and $u_i(\alpha_j) = 0$ when $j \neq i$. For every $i \in [n]$ and $k \geq 1$ we define $G_k^i(y_1, \ldots, y_k, z_1, \ldots, z_k) : \mathbb{F}^{2k} \to \mathbb{F}$ as $G_k^i(y_1, \ldots, y_k, z_1, \ldots, z_k) \stackrel{\Delta}{=} \sum_{j=1}^{k} u_i(y_j) \cdot z_j$. Finally, let $G_k(y_1, \ldots, y_k, z_1, \ldots, z_k) : \mathbb{F}^{2k} \to \mathbb{F}^n$ be defined as*

$$G_k(y_1, \ldots, y_k, z_1, \ldots, z_k) \stackrel{\Delta}{=} \left(G_k^1, G_k^2, \ldots, G_k^n\right) = \left(\sum_{j=1}^{k} u_1(y_j) \cdot z_j, \sum_{j=1}^{k} u_2(y_j) \cdot z_j, \ldots, \sum_{j=1}^{k} u_n(y_j) \cdot z_j\right).$$

The following simple observation plays an important role in our algorithms.

Observation 3.6. *Denote with $\bar{e}_i \in \{0,1\}^n$ the vector that has 1 in the i-th coordinate and 0 elsewhere. Then, $G_{k+1} = G_k + \sum_{i=1}^{n} u_i(y_{k+1}) \cdot z_{k+1} \cdot \bar{e}_i$. Hence, for every $k \geq 1$ and $\alpha_m \in A$ we have that $G_{k+1}|_{y_{k+1} = \alpha_m} = G_k + z_{k+1} \cdot \bar{e}_m$. Hence, for every $W \subseteq \mathbb{F}$ it holds that $\mathcal{A}_k^n(W) \subseteq \mathrm{Im}\,(G_k)$.*

4 From PIT to Justifying Assignments

We now show how to obtain a justifying assignment from a PIT algorithm. We shall consider a circuit class $\mathcal{C}$ (e.g. depth-3 circuit, sum of ROFs, etc.) for which there exists another circuit class $\mathcal{C}'$ such that $\partial\mathcal{C} \subseteq \mathcal{C}'$ and $\mathcal{C}'$ has an efficient PIT algorithm.[1] Algorithm 1 returns a justifying assignment for P (if it fails to do so, it returns "ERROR"). The algorithm will invoke (as a subroutine) the PIT algorithm for $\mathcal{C}'$. Before giving the algorithm we explain the intuition behind it. Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial with individual degrees bounded by d. What we are after is a vector $(a_1, \ldots, a_n) \in \mathbb{F}^n$ such that if P depends on x_i then the polynomial $P|_{x_j=a_j}$ (for any $j \neq i$) also depends on x_i. Our approach will be to consider a witness for P, $\bar{\alpha} \in \mathbb{F}^n$, and look for $\bar{a} \in \mathbb{F}^n$ such that if $\frac{\partial P}{\partial_{\alpha_i} x_i} \not\equiv 0$ then also $\frac{\partial P}{\partial_{\alpha_i} x_i}|_{x_j=a_j} \not\equiv 0$ (see Proposition 2.10). Therefore, we consider the polynomials $\left\{ g_i = \frac{\partial P}{\partial_{\alpha_i} x_i} \right\}_{i\in[n]}$ and look for a vector $\bar{a}$, such that for any $j \neq i$, $g_i|_{x_j=a_j} \not\equiv 0$. As the degree of x_i in each g_i is bounded by d, there are at most d 'bad' possible values for a_j. Namely, for most values of a_j, we have that $g_i|_{x_j=a_j} \not\equiv 0$. Hence, if we check enough values then we should find some a_j that is good for all g_i. To verify that $g_i|_{x_j=a_j} \not\equiv 0$ we use the PIT algorithm for $\mathcal{C}'$. In fact, what we just described only gives a weakly justifying assignment. To find a justifying assignment we have to verify that after we assign a_j to x_j for all $i \neq j$, g_i remains nonzero. We manage to do it by first finding a_1, then we assign a_1 to x_1 to get a new set of polynomials g_i's etc. Using this approach we can actually find a justifying assignment for more than one polynomial. That is, we find a *Common Justifying Assignment* for a set of polynomials $\{P_m\}_{m\in[k]}$. A common justifying assignment is an assignment $\bar{a}$ that is simultaneously a justifying assignment for each P_m. We now give the algorithm and its analysis.

Algorithm 1 Acquire Common Justifying Assignment

Input: Circuits $C_1, \ldots, C_k$ from $\mathcal{C}$ computing $P_1, \ldots, P_k$ with individual degrees bounded by d, Access to a PIT algorithm for $\mathcal{C}'$ such that $\partial\mathcal{C} \subseteq \mathcal{C}'$.

Output: A Common Justifying Assignment $\bar{a}$ for $P_1, P_2, \ldots, P_k$

```
1: Find $\{\bar{\alpha}^m\}_{m\in[k]}$ such that $\bar{\alpha}^m$ is a witness for $P_m$ {see Lemma 4.1 }
2: For $i \in [n]$, $m \in [k]$ set $g_i^m = \frac{\partial P_m}{\partial_{\alpha_i^m} x_i}$
   We describe an iteration $j \in [n]$ for finding the value of $a_j$ in $\bar{a}$:
3: for $j = 1 \ldots n$ do
4:    Find $c_j \in \mathbb{F}$ such that for every $m \in [k]$ and $i \neq j \in [n]$: if $g_i^m \not\equiv 0$ then $g_i^m|_{x_j=c_j} \not\equiv 0$.
5:    For every $m \in [k]$ and $i \neq j \in [n]$, set $g_i^m \leftarrow g_i^m|_{x_j=c_j}$.
6:    Set $a_j \leftarrow c_j$
```

Lemma 4.1. *Let $\mathbb{F}$ be a field of size $|\mathbb{F}| > d$. Let P be a polynomial, with individual degrees bounded by d, that is computed by a circuit class $\mathcal{C}$ over $\mathbb{F}$. Let $\mathcal{C}'$ be a circuit class such that $\partial\mathcal{C} \subseteq \mathcal{C}'$. Then there is an algorithm that when given access to P (either explicitly or via black-box access, depending on the PIT algorithm for $\mathcal{C}'$) computes* $\mathrm{var}(P)$ *and outputs a witness $\bar{\alpha}$ for P in time $\mathcal{O}(nd \cdot T_{\mathcal{C}'})$, where $T_{\mathcal{C}'}$ is the running time of the PIT algorithm for $\mathcal{C}'$.*

Proof. The proof of Lemma 2.11 gives a simple algorithm for finding witnesses. Indeed, consider $\varphi_i(\bar{x}, w) \triangleq \frac{\partial P}{\partial_w x_i}$. Clearly $\varphi_i \in \mathcal{C}'$. Note that if $x_i \in \mathrm{var}(P)$ then $\varphi_i(\bar{x}, w) \not\equiv 0$. Pick $d + 1$ different

[1]Note, that in most cases an identity testing algorithm for $\mathcal{C}$ can be slightly modified to yield an identity testing algorithm for $\partial\mathcal{C}$.

elements $v_0, \dots, v_d \in \mathbb{F}$ and for each of them check, using the PIT algorithm for $\mathcal{C}'$, whether $\varphi_i(\bar{x}, v_i) \neq 0$. Let α_i be the first v_j for which $\varphi_i(\bar{x}, v_j) \neq 0$ (if no such j exists then take $\alpha_i = 0$). Set $\bar{\alpha} = (\alpha_1, \dots, \alpha_n)$. Lemma 2.11 implies that $\bar{\alpha}$ is the required witness. The claim regarding the running time is clear.

Note that the same approach also determines, for each variable x_i, whether the polynomial depends on x_i or not. Therefore, the algorithm can be used to compute $\mathrm{var}(P_m)$ as well.

□

We now give the analysis of the second step of the algorithm.

Lemma 4.2. *Let $\mathbb{F}$ be a field with $|\mathbb{F}| > knd$ and $V \subseteq \mathbb{F}$ be of size $|V| \geq knd$. Let $\{P_m\}_{m\in[k]}$ be a set of polynomials with individual degrees bounded by d that are computed by circuits from $\mathcal{C}$. Let $\mathcal{C}'$ be a circuit class such that $\partial\mathcal{C} \subseteq \mathcal{C}'$. Then, Algorithm 1 returns a common justifying assignment $\bar{a}$ for $\{P_m\}_{m\in[k]}$ in time $\mathcal{O}(n^3k^2d \cdot T_{\mathcal{C}'})$, where $T_{\mathcal{C}'}$ is the running time of the PIT algorithm for $\mathcal{C}'$.*

Proof. We show that each iteration $j \in [n]$ succeeds, and that the algorithm outputs a justifying assignment. In order to succeed in j-th phase the algorithm must find $c_j \in V$ that is good for every $g_i^m \not\equiv 0$. Namely, for every m and $i \neq j$ if $g_i^m \not\equiv 0$ then $g_i^m|_{x_j=c_j} \not\equiv 0$. Note, that g_i^m's are polynomials with individual degrees bounded by d and hence, by Lemma 2.14, each g_i^m has *at most* d roots of the form of $x_j = c_j$. Therefore, there are at most $kd(n-1)$ 'bad' values of c_j (i.e. values for which there exist m and $i \neq j$ with $g_i^m \not\equiv 0$ and $g_i^m|_{x_j=c_j} \equiv 0$). Consequently, V contains at least one 'good' c_j. From the definition $g_i^m \in \mathcal{C}'$, therefore we can use the supplied PIT algorithm for $\mathcal{C}'$ to find such c_j. In addition, notice before that first iteration $g_i^m \not\equiv 0$ iff $x_i \in \mathrm{var}(P_m)$ and for each $j \in [n]$, if g_i^m is nonzero before the j-iteration then it remains nonzero after that iteration. We conclude that after the n-th iteration is (successfully) completed we have that for every $m \in [k]$ and $x_i \in \mathrm{var}(P_m)$ it holds that $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\bar{a}) = g_i^m \neq 0$. This follows from the definition of the g_i^m's and the fact that at each iteration we substitute a_j to x_j in every g_i^m. This ensures that $\bar{a}$ is indeed a common justifying assignment.

Next we analyze the running time. By Lemma 4.1 finding $\{\bar{\alpha}^m\}_{m\in[k]}$ requires $\mathcal{O}(knd)$ PIT checks. The computation of $\{g_i^m\}_{i\in[n],m\in[k]}$ can be done in $\mathcal{O}(nk)$ time. The execution of each iteration j requires for each $c \in V$ to perform $k(n-1)$ PIT checks, thus in every iteration we preform at most $k(n-1) \cdot |V| < n^2k^2d$ PIT checks. Therefore, we do at most $\mathcal{O}(n^3k^2d + knd)$ PIT checks during the execution. Hence the total running time of the algorithm is $\mathcal{O}(n^3k^2d \cdot T_{\mathcal{C}'})$, where $T_{\mathcal{C}'}$ is the cost of every PIT check for a circuit in $\mathcal{C}'$. □

4.1 From a Generator to a Justifying Set

Algorithm 1 shows how to find a common justifying assignment for a set of polynomials in an adaptive manner, even if the PIT for $\mathcal{C}'$ is in the black-box setting. In this section we give a non adaptive version of the algorithm. More precisely, given a generator (a black-box PIT algorithm) $\mathcal{G}$ for $\mathcal{C}'$ (satisfying $\partial\mathcal{C} \subseteq \mathcal{C}'$) we construct a (k,d)-*justifying set* for $\mathcal{C}$ (assuming that $|\mathbb{F}|$ is large enough). That is, a set of elements $\mathcal{J}_{\mathcal{G}}^{k,d} \subseteq \mathbb{F}^n$ that contains a common justifying assignment for any set of k polynomials, with individual degrees bounded by d, that are computed by $\mathcal{C}$. The construction is performed by evaluating the generator on many points. In particular, we show that $\mathrm{Im}\,(\mathcal{G})$ contains a common justifying assignment for any set of polynomials computed by $\mathcal{C}$. Note that in this case we will not find the witnesses explicitly, but rather rely on their existence.

Lemma 4.3. *Let $\{P_m(\bar{x})\}_{m\in[k]}$ be a set of k polynomials over $\mathbb{F}$, with individual degrees bounded by d, that are computed by circuits from $\mathcal{C}$. Let $\mathcal{C}'$ be (another) circuit class such that $\partial\mathcal{C} \subseteq \mathcal{C}'$. Let*

$\mathcal{G} = (\mathcal{G}^1, \dots, \mathcal{G}^n) : \mathbb{F}^t \to \mathbb{F}^n$ be a generator for $\mathcal{C}'$ such that the individual degrees in each $\mathcal{G}^i$ are bounded by Δ. Let $V \subseteq \mathbb{F}$ be of size $|V| = kn^2 d\Delta + 1$. Then $\mathcal{J}_{\mathcal{G}}^{k,d} \triangleq \mathcal{G}(V^t)$ contains a common justifying assignment for $P_1, \dots, P_k$ (that is, $\mathcal{J}_{\mathcal{G}}^{k,d}$ is a (k,d)-justifying set for $\mathcal{C}$).

Proof. By Lemma 2.11 $\mathbb{F}$ contains witnesses $\{\bar{\alpha}^m\}_{m\in[k]}$ for $\{P_m(\bar{x})\}_{m\in[k]}$. For $i \in [n]$ and $m \in [k]$ define $g_i^m \triangleq \frac{\partial P_m}{\partial_{\alpha_i^m} x_i} \circ \mathcal{G}$. From the definitions of the generator and $\mathcal{C}'$ we get that if $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i} \not\equiv 0$ then $g_i^m \not\equiv 0$. Consider the polynomial $g \triangleq \prod_{i,m|\ g_i^m \not\equiv 0} g_i^m$. It follows that g is a nonzero t-variate polynomial of degree at most $nk \cdot nd\Delta$ in each variable. Lemma 2.13 implies that $g|_{V^t} \not\equiv 0$. Equivalently, there exists $\bar{\gamma} \in V^t$ such that for each pair $i \in [n]$ and $m \in [k]$ if $g_i^m \not\equiv 0$ then $g_i^m(\bar{\gamma}) \neq 0$. Now, let $i \in [n]$ and $m \in [k]$ be such that $x_i \in \mathrm{var}(P_m)$. Then $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i} \not\equiv 0$ and thus $g_i^m = \frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\mathcal{G}) \not\equiv 0$. From the choice of $\bar{\gamma}$ we obtain that $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\mathcal{G}(\bar{\gamma})) = g_i^m(\bar{\gamma}) \neq 0$ and hence $\bar{a} = \mathcal{G}(\bar{\gamma})$ is a justifying assignment for every P_m (recall Proposition 2.10). Finally, note that $\left|\mathcal{J}_{\mathcal{G}}^{k,d}\right| \leq (kn^2 d\Delta + 1)^t$. □

The following is an immediate corollary of the proof.

Corollary 4.4. *Let $\mathcal{C}, \mathcal{C}'$ and $\mathcal{G}$ be as in Lemma 4.3 and let $k \geq 1$. Then $\mathrm{Im}(\mathcal{G})$ contains a common justifying assignment for any set of k polynomials computed by $\mathcal{C}$.*

5 Read-Once formulas

In this section we discuss our computational model. We first consider the basic model of read-once formulas and cover some of its main properties. Then, we introduce the model of preprocessed-read-once formulas and give its corresponding properties.

5.1 Read-Once Formulas and Read-Once Polynomials

Most of the definitions that we give in this section are from [HH91], or their small variants. We start by formally defining the notions of a read-once formula, a skeleton of a read-once formula and a read-once polynomial.

Definition 5.1. *An* arithmetic read-once formula *(ROF for short) Φ over a field $\mathbb{F}$ in the variables $\bar{x} = (x_1, \dots, x_n)$ is a binary tree whose leafs are labelled with the input variables and whose internal nodes are labelled with the arithmetic operations $\{+, \times\}$ and with a pair of field elements[2] $(\alpha, \beta) \in \mathbb{F}^2$. Each input variable can label at most one leaf. The computation is performed in the following way. A leaf labelled with the variable x_i and with (α, β) computes the polynomial $\alpha \cdot x_i + \beta$. If a node v is labelled with the operation* op *and with (α, β), and its children compute the polynomials Φ_{v_1} and Φ_{v_2} then the polynomial computed at v is $\Phi_v = \alpha \cdot (\Phi_{v_1}\ \mathrm{op}\ \Phi_{v_2}) + \beta$. We say that a ROF Φ is* non-degenerate *if it depends on all the variables appearing in it.*

A polynomial $P(\bar{x})$ is a *read-once polynomial* (ROP for short) if it can be computed by a read-once formula. Clearly, ROPs are a subclass of multilinear polynomials.

A ROF is called a *multiplicative ROF* if it has no addition gates. A polynomial computed by a multiplicative ROF is called a *multiplicative ROP*. Note that because we allow gates to apply

[2]This is a slightly more general model than the usual definition of read-once formulas.

linear functions on the results of their operations, the output of a multiplicative ROF can be more than just a monomial.

Example 5.2. *The polynomial* $(5x_1 \cdot x_2 + 1) \cdot ((-x_3 + 2) \cdot (2x_4 - 1) + 5)$ *has a multiplicative ROF.*

The following lemma follows easily from the definition.

Lemma 5.3 (ROP Structural Lemma)**.** *Every ROP* $P(\bar{x})$ *such that* $|\mathrm{var}(P)| \geq 2$ *can be presented in* exactly *one of the following forms:*

1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$
2. $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$

where P_1 *and* P_2 *are non-constant variable disjoint ROPs and* c *is a constant.*

Proof. Let Φ be a ROF computing P. Let v be the top gate of Φ and Φ_1, Φ_2 be the inputs of v. Clearly, Φ_1,Φ_2 are variable-disjoint ROFs. Now, If v is an addition gate then $\Phi = a \cdot (\Phi_1 + \Phi_2) + b = (a \cdot \Phi_1) + (a \cdot \Phi_2 + b)$. By setting $P_1 \overset{\Delta}{=} a \cdot \Phi_1$, $P_2 \overset{\Delta}{=} a \cdot \Phi_2 + b$ we are done. Otherwise, v is multiplication gate. Therefore $\Phi = a \cdot (\Phi_1 \cdot \Phi_2) + b$ and we proceed similarly. Note, that P_1 and P_2 are not necessarily unique, however it can be seen that P cannot be represented in both forms 1 and 2. Indeed, assume for contradiction that $P_1 + P_2 = P = P_1' \cdot P_2' + c$. If there exist x_i, x_j such that $x_i \in \mathrm{var}(P_1) \cap \mathrm{var}(P_1')$ and $x_j \in \mathrm{var}(P_2) \cap \mathrm{var}(P_2')$ (or vice versa) then we immediately get a contradiction as $x_i \cdot x_j$ will appear in some monomial in the RHS but not in the LHS. If no such x_i and x_j exist then either P_1' or P_2' contain no variable, in contradiction. $\square$

The following is another simple claim regarding representations of ROFs.

Lemma 5.4. *Let* $P(\bar{x})$ *be a ROP and* v *a node in a ROF* Φ *computing* P*. Denote by* $p_v(\bar{x})$ *the polynomial that is computed by* v*. Then there exists a polynomial* $Q(y, \bar{x})$ *such that* $Q(p_v(\bar{x}), \bar{x}) \equiv P(\bar{x})$ *and, in addition,* p_v *and* Q *can be computed by variable-disjoint ROFs.*

Proof. Consider Φ's graph of computation. Denote with Ψ the sub-formula whose top gate is v. Let φ be the rest of the graph. The output of Ψ is wired as one of the inputs of φ. We denote this input by y. We define Q to be the polynomial computed by φ. Consequently, $Q(p_v(\bar{x}) \ , \bar{x}) \equiv P(\bar{x})$ and p_v, Q are variable-disjoint ROPs as they are computed by different parts of the same ROF. $\square$

Definition 5.5. *Let* $V \subseteq \mathrm{var}(\Phi)$, $|V| \geq 2$ *be a subset of the input variables of a ROF* Φ*. We define the* first common gate *of* V*,* $\mathrm{fcg}(V)$*, to be the first gate in the graph of computation of* Φ *common to all the paths from the inputs of* V *to the root of the formula.*

We note, that $\mathrm{fcg}(V)$ is in fact the least common ancestor of the nodes in V when looking at the formula as a tree.

5.2 Partial Derivatives of ROPs

In this section we list some important properties the partial derivatives of ROPs. Clearly, a partial derivative of a multilinear polynomial is a multilinear polynomial. In particular, a partial derivative of a ROP is a multilinear polynomial as well. The following lemma gives a stronger statement.

Lemma 5.6. *A partial derivative of a ROP is a ROP.*

Proof. Let P be a ROP and $i \in [n]$. We prove the claim by induction on $k = |\mathrm{var}(P)|$. For $k = 0, 1$ the claim is trivial. For $k \geq 2$ we get by Lemma 5.3 that P can be in a one of two forms.

Case 1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$. Since P_1 and P_2 are variable disjoint we can assume w.l.o.g. that $\frac{\partial P}{\partial x_i} = \frac{\partial P_1}{\partial x_i}$. In addition, $|\mathrm{var}(P_1)| < |\mathrm{var}(P)|$ and so by the induction hypothesis we get that $\frac{\partial P}{\partial x_i} = \frac{\partial P_1}{\partial x_i}$ is a ROP.

Case 2. $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$. Again we assume w.l.o.g. that $\frac{\partial P}{\partial x_i} = \frac{\partial P_1}{\partial x_i} \cdot P_2$. As before, $\frac{\partial P_1}{\partial x_i}$ is a ROP. Since P_1 and P_2 are variable disjoint and P_2 is a ROP, we have that $\frac{\partial P}{\partial x_i} = \frac{\partial P_1}{\partial x_i} \cdot P_2$ is a ROP as well. $\square$

We now give two useful properties of the derivatives of ROPs.

Observation 5.7. *Let P be a ROP and let $x_i, x_j \in \mathrm{var}(P)$. Let Φ be a ROF computing P and $v = \mathrm{fcg}\{x_j, x_i\}$. Then v is a multiplication gate iff $\frac{\partial^2 P}{\partial x_i \partial x_j} \not\equiv 0$.*

Lemma 5.8 (2-nd Derivative Lemma). *Let $P(x_1, x_2, \ldots, x_n)$ be a ROP such that $\frac{\partial^2 P}{\partial x_i \partial x_j} \not\equiv 0$ and $\frac{\partial^2 P}{\partial x_i \partial x_j}|_{x_k=\alpha} \equiv 0$ for three different indices i, j, k and $\alpha \in \mathbb{F}$. Then, $\frac{\partial P}{\partial x_i}|_{x_k=\alpha} \equiv 0$ or $\frac{\partial P}{\partial x_j}|_{x_k=\alpha} \equiv 0$.*

Proof. First, notice that $x_j, x_i \in \mathrm{var}(P)$. Let Φ be a ROF computing P and $v = \mathrm{fcg}\{x_j, x_i\}$. Let $G(\bar{x})$ be the polynomial computed by v in Φ. From Lemma 5.4 there exists a ROP $Q(y, \bar{x})$ such that $Q(G(\bar{x}), \bar{x}) \equiv P(\bar{x})$. Clearly, $x_j, x_i \in \mathrm{var}(G) \setminus \mathrm{var}(Q)$. As $\frac{\partial^2 P}{\partial x_i \partial x_j} \not\equiv 0$, it follows that v is a multiplication gate. By the definition of fcg and Lemma 5.3 we obtain that $G(\bar{x})$ can be presented as $P_1 \cdot P_2 + c$ where $x_i \in \mathrm{var}(P_1), x_j \in \mathrm{var}(P_2)$ and $c \in \mathbb{F}$. By the chain rule (Lemma 2.7):

$$\frac{\partial P}{\partial x_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial G}{\partial x_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial P_1}{\partial x_i} \cdot P_2 \quad \text{and} \quad \frac{\partial P}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot \frac{\partial G}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot P_1 \cdot \frac{\partial P_2}{\partial x_j} .$$

Consequently, $\frac{\partial^2 P}{\partial x_i \partial x_j} = \frac{\partial Q}{\partial y} \cdot \frac{\partial P_1}{\partial x_i} \cdot \frac{\partial P_2}{\partial x_j}$. This implies that:

$$\frac{\partial P}{\partial x_i}|_{x_k=\alpha} \cdot \frac{\partial P}{\partial x_j}|_{x_k=\alpha} = \left(\frac{\partial Q}{\partial y} \cdot \frac{\partial P_1}{\partial x_i} \cdot P_2\right)|_{x_k=\alpha} \cdot \left(\frac{\partial Q}{\partial y} \cdot P_1 \cdot \frac{\partial P_2}{\partial x_j}\right)|_{x_k=\alpha} =$$

$$\left(\frac{\partial Q}{\partial y} \cdot \frac{\partial P_1}{\partial x_i} \cdot \frac{\partial P_2}{\partial x_j}\right)|_{x_k=\alpha} \cdot \left(\frac{\partial Q}{\partial y} \cdot P_1 \cdot P_2\right)|_{x_k=\alpha} = \frac{\partial^2 P}{\partial x_i \partial x_j}|_{x_k=\alpha} \cdot \left(\frac{\partial Q}{\partial y} \cdot P_1 \cdot P_2\right)|_{x_k=\alpha} \equiv 0 .$$

In particular, either $\frac{\partial P}{\partial x_i}|_{x_k=\alpha} \equiv 0$ or $\frac{\partial P}{\partial x_j}|_{x_k=\alpha} \equiv 0$. $\square$

As a corollary we obtain an example of multilinear polynomial which is not a ROP.

Example 5.9. *The polynomial $P(x_1, x_2, x_3) = x_1 x_2 x_3 + x_1 + x_2$ is not a ROP. To see this apply Lemma 5.8 on P with the parameters $i = 1, j = 2, k = 3, \alpha = 0$.*

5.3 Multiplicative and $\bar{0}$-Justified ROPs

Recall that multiplicative ROFs are ROFs with no addition gates. Observation 5.7 provides an algebraic characterization of ROPs computed by such ROFs (i.e. multiplicative ROPs).

Lemma 5.10. *A ROP P is a multiplicative ROP iff for any two variables $x_i \neq x_j \in \mathrm{var}(P)$ we have that $\frac{\partial^2 P}{\partial x_i \partial x_j} \not\equiv 0$.*

From now on, whenever we discuss multiplicative ROP we shall use the property described in the claim as an alternative definition. The following lemma is a generalization of Lemma 5.6 to weakly-$\bar{0}$-justified ROPs.

Lemma 5.11. *A partial derivative of a weakly-$\bar{0}$-justified ROP is a weakly-$\bar{0}$-justified ROP.*

Proof. Let P to be a weakly-$\bar{0}$-justified ROP. From Lemma 5.6 it is enough to show that the partial derivatives of P are weakly-$\bar{0}$-justified . Assume for a contradiction that for some $i \in [n]$, we have that $\frac{\partial P}{\partial x_i}$ is not weakly-$\bar{0}$-justified . That is, there exist some $j, k \in [n]$ such that $\frac{\partial P}{\partial x_i}$ depends on x_j however $\frac{\partial P}{\partial x_i}|_{x_k=0}$ does not. In other words, we have that: $\frac{\partial^2 P}{\partial x_i \partial x_j} \not\equiv 0$ and $\frac{\partial^2 P}{\partial x_i \partial x_j}|_{x_k=0} \equiv 0$. Note that $\{x_i, x_j\} \subseteq \mathrm{var}(P)$. Lemma 5.8 implies that either $\frac{\partial P}{\partial x_i}|_{x_k=0} \equiv 0$ or $\frac{\partial P}{\partial x_j}|_{x_k=0} \equiv 0$ holds. On the other hand, $\{x_i, x_j\} \subseteq \mathrm{var}(P|_{x_k=0})$ since P is a weakly-$\bar{0}$-justified ROP and hence $\frac{\partial P}{\partial x_i}|_{x_k=0} \not\equiv 0$ and $\frac{\partial P}{\partial x_j}|_{x_k=0} \not\equiv 0$, in contradiction. □

It is also possible to extend this proof for $\bar{0}$-justified ROPs. In a similar (and simpler) way, we observe the following.

Lemma 5.12. *Every factor of a weakly-$\bar{0}$-justified ROP is a weakly-$\bar{0}$-justified ROP.*

Proof. Let $P = h_1 \cdot h_2$ be a ROP, where h_1 and h_2 are two polynomials. As P is multilinear, h_1 and h_2 must be variable-disjoint. Therefore, we can write $P(\bar{x}, \bar{y}) = h_1(\bar{x}) \cdot h_2(\bar{y})$. As P is weakly-$\bar{0}$-justified so are h_1 and h_2. Furthermore, it holds that $h_1(\bar{0}), h_2(\bar{0}) \neq 0$. Consequently, $h_1(\bar{x}) = P(\bar{x}, \bar{0})/h_2(\bar{0})$ is a weakly-$\bar{0}$-justified ROP and so is $h_2(\bar{y})$. □

We note that almost the same proof shows that a factor of any ROP is also a ROP. We conclude this section with a very useful property of multiplicative ROPs that encapsulates the previously discussed properties.

Lemma 5.13. *Let P be a multiplicative ROP with $|\mathrm{var}(P)| \geq 2$. Then, for every $x_i \in \mathrm{var}(P)$ there exists $x_j \in \mathrm{var}(P)$ such that $\frac{\partial P}{\partial x_j} = (x_i - \alpha)h_j(\bar{x})$ for some $\alpha \in \mathbb{F}$ and ROP $h_j(\bar{x})$, when $\mathrm{var}(h_j) = \mathrm{var}(P) \setminus \{x_i, x_j\}$ (in particular, $\frac{\partial P}{\partial x_j}|_{x_i=\alpha} \equiv 0$). If, in addition, P is weakly-$\bar{0}$-justified then so is $h_j(\bar{x})$. Moreover, $\alpha \neq 0$ and there exists **at most** one element $\beta \neq \alpha \in \mathbb{F}$ such that $P|_{x_i=\beta}$ is not weakly-$\bar{0}$-justified .*

Proof. Let Φ be a multiplicative ROF computing P. As $|\mathrm{var}(\Phi)| = |\mathrm{var}(P)| \geq 2$, Φ has at least one gate. Let v be the unique entering gate[3] of x_i. We denote by $p_v(\bar{x})$ the ROP that is computed by v. Assume w.l.o.g that $\mathrm{var}(p_v) = \{x_1, x_2, \ldots x_{i-1}, x_i\}$. By Lemma 5.4 there exists some ROP $Q(y, x_{i+1}, \ldots, x_n)$ such that $Q(p_v(x_1, x_2, \ldots, x_i), x_{i+1}, \ldots, x_n) \equiv P(x_1, x_2, \ldots, x_n)$. Since v is a multiplication gate (recall that Φ is a multiplicative ROF) and the entering gate of x_i we get, in a similar manner to Lemma 5.3, that p_v can be written as $p_v(\bar{x}) = (x_i - \alpha)H(\bar{x}) + c$ for some ROP $H(\bar{x})$ such that $\mathrm{var}(H) \neq \emptyset$ and $x_i \notin \mathrm{var}(H)$. By the chain rule, for every $x_j \in \mathrm{var}(H)$ it holds that:

$$\frac{\partial P}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot \frac{\partial p_v}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot (x_i - \alpha) \cdot \frac{\partial H}{\partial x_j} = (x_i - \alpha) \cdot h_j(\bar{x}), \tag{1}$$

where $h_j(\bar{x}) \triangleq \frac{\partial Q}{\partial y} \cdot \frac{\partial H}{\partial x_j}$. As $\frac{\partial P}{\partial x_j}$ is a ROP then so is h_j. Now, if P is weakly-$\bar{0}$-justified then by Lemma 5.11 we get that $\frac{\partial P}{\partial x_j}$ is a weakly-$\bar{0}$-justified ROP as well. By Lemma 5.12 $h_j(\bar{x})$ is a weakly-$\bar{0}$-justified ROP and $\alpha \neq 0$. Assume that for some $\beta \neq \alpha \in \mathbb{F}$ the polynomial $P|_{x_i=\beta}$ is not

[3]The entering gate of x_i is the neighbor of the leaf labelled by x_i.

weakly-$\bar{0}$-justified . We will show that the value of β is uniquely defined. Applying a reasoning similar to the one in Lemma 5.8 we obtain that there exist $x_\ell, x_k \neq x_i \in \mathrm{var}(P)$ such that $\frac{\partial P}{\partial x_\ell}|_{x_i=\beta} \not\equiv 0$ however $\frac{\partial P}{\partial x_\ell}|_{x_i=\beta, x_k=0} \equiv 0$. In particular, $\frac{\partial P}{\partial x_\ell} \not\equiv 0$ which implies $\frac{\partial P}{\partial x_\ell}|_{x_k=0} \not\equiv 0$ since P is weakly-$\bar{0}$-justified . In other words, the substitution $x_i = \beta$ affects the dependence of $P|_{x_k=0}$ on x_ℓ. We consider two cases.

Case $x_\ell \in \mathrm{var}(H)$ (i.e. $1 \leq \ell \leq i-1$): By Equation (1) it holds that $\frac{\partial P}{\partial x_\ell}|_{x_k=0, x_i=\beta} = (\beta - \alpha) \cdot h_\ell|_{x_k=0}$ while $\frac{\partial P}{\partial x_\ell}|_{x_k=0} = (x_i - \alpha) \cdot h_\ell|_{x_k=0}$. As $\beta - \alpha \neq 0$ we conclude that this case is impossible.

Case $x_\ell \in \mathrm{var}(Q)$ (i.e. $i+1 \leq \ell \leq n$): Here we have that $\frac{\partial P}{\partial x_\ell} = \frac{\partial Q}{\partial x_\ell}|_{y=p_v(\bar{x})}$. Define $p_{v,0} \stackrel{\Delta}{=} p_v|_{x_k=0}$ and $p_{v,0,\beta} \stackrel{\Delta}{=} p_{v,0}|_{x_i=\beta} = p_v|_{x_k=0, x_i=\beta}$. We get that

$$\frac{\partial Q}{\partial x_\ell}|_{x_k=0, y=p_{v,0}} = \frac{\partial P}{\partial x_\ell}|_{x_k=0} \not\equiv 0.$$

In particular, it implies that $\frac{\partial Q}{\partial x_\ell}|_{x_k=0} \not\equiv 0$. On the other hand,

$$\left(\frac{\partial Q}{\partial x_\ell}|_{x_k=0}\right)|_{y=p_{v,0,\beta}} = \frac{\partial Q}{\partial x_\ell}|_{x_k=0, y=p_{v,0,\beta}} = \frac{\partial P}{\partial x_\ell}|_{x_k=0, x_i=\beta} \equiv 0.$$

Therefore, from Lemma 2.14 we conclude that $y - p_{v,0,\beta}$ is a factor of $\frac{\partial Q}{\partial x_\ell}|_{x_k=0}$. Since $\mathrm{var}(p_{v,0,\beta}) \cap \mathrm{var}(Q) = \emptyset$ and Q is a multilinear polynomial it follows that there exists (exactly one) $\gamma \in \mathbb{F}$ such that $p_{v,0,\beta} \equiv \gamma$ (otherwise $p_{v,0,\beta}$ introduces variables that do not appear in Q) and $y - \gamma$ is a factor of $\frac{\partial Q}{\partial x_\ell}|_{x_k=0}$. Recall that $p_v(\bar{x}) = (x_i - \alpha)H(\bar{x}) + c$. Thus, $H|_{x_k=0} = \frac{\gamma - c}{\beta - \alpha}$ (recall that $\beta - \alpha \neq 0$). As H is a non-constant polynomial, it must the case that $x_k \in \mathrm{var}(H)$. Finally, notice that since P is a weakly-$\bar{0}$-justified polynomial then it must be the case that $\mathrm{var}(H) = \{x_k\}$ (otherwise, if there is $x_m \neq x_k \in \mathrm{var}(H)$ then $\frac{\partial P}{\partial x_m}|_{x_k=0} \equiv 0$ in contradiction). We conclude that H is a univariate polynomial in x_k and that the value of β is uniquely defined by α, γ, c and H, which, in turn, are uniquely defined by P. $\square$

5.4 Preprocessed Read-Once Polynomials

In this subsection we extend the model of ROFs by allowing a *preprocessing* step of the input variables. While the basic model is read-once in its variables, the extended model can be considered as read-once in univariate polynomials.

Definition 5.14. *A* preprocessing *is a transformation $T(\bar{x}) : \mathbb{F}^n \to \mathbb{F}^n$ of the form $T(\bar{x}) \stackrel{\Delta}{=} (T_1(x_1), T_2(x_2), \ldots, T_n(x_n))$ such that each T_i is a* non-constant univariate polynomial. *We say that a preprocessing is* standard *if in addition to the above each T_i is monic and satisfies $T_i(0) = 0$.*

Notice that preprocessings do not affect the PIT problem in the non-black setting as for every n-variate polynomial $P(\bar{y})$ it holds that $P(\bar{y}) \equiv 0$ if and only if $P(T(\bar{x})) \equiv 0$. We now give a formal definition and list some immediate properties.

Definition 5.15. *A* preprocessed arithmetic read-once formula *(PROF for short) over a field $\mathbb{F}$ in the variables $\bar{x} = (x_1, \ldots, x_n)$ is a binary tree whose leafs are labelled with non-constant univariate polynomials $T_1(x_1), T_2(x_2), \ldots, T_n(x_n)$ (all together forming a preprocessing) and whose internal nodes are labelled with the arithmetic operations $\{+, \times\}$ and with a pair of field elements $(\alpha, \beta) \in \mathbb{F}^2$. Each T_i can label at most one leaf. The computation is performed in the following way. A leaf labelled with the polynomial $T_i(x_i)$ and with (α, β) computes the polynomial $\alpha \cdot T_i(x_i) + \beta$. If a node v is labelled with the operation* op *and with (α, β), and its children compute the polynomials Φ_{v_1} and Φ_{v_2} then the polynomial computed at v is $\Phi_v = \alpha \cdot (\Phi_{v_1}$ op $\Phi_{v_2}) + \beta$.*

A polynomial $P(\bar{x})$ is a *Preprocessed Read-Once Polynomial* (PROP for short) if it can be computed by a preprocessed read-once formula. A *Decomposition* of a polynomial P is a couple $Q(\bar{z}), T(\bar{x})$ such that $P(\bar{x}) = Q(T(\bar{x}))$ when Q is a ROP and T is a preprocessing. A *Standard Decomposition* is as above with the additional requirement that T is a standard preprocessing. An immediate consequence from the definition is that each PROP admits a decomposition. To provide additional intuition we start with a simple, yet important lemma.

Lemma 5.16. *Every PROP P admits a standard decomposition.*

Proof. Let (Q, T) be a decomposition of P and $c_i \neq 0$ denote the leading coefficient of x_i in the polynomial $T_i(x_i)$ for $i \in [n]$ (c_i is well-defined since T_i is non-constant). Consider the shifted polynomials:

$$Q'(\bar{z}) \triangleq Q\left(c_1 \cdot z_1 + T_1(0), c_2 \cdot z_2 + T_2(0), \ldots, c_n \cdot z_n + T_n(0)\right)$$
$$T_i'(x_i) \triangleq \frac{T_i(x_i) - T_i(0)}{c_i}\ ,\ T(\bar{x}) \triangleq \left(T_1'(x_1), T_2'(x_2), \ldots, T_n'(x_n)\right).$$

It is easy to verify that (Q', T') is a standard decomposition of P. $\square$

Lemma 5.17. *Let P be a PROP, and let $(Q(\bar{z}), T(\bar{x}))$ be a standard decomposition for P. Then P is (weakly) $\bar{0}$-justified if and only if Q is (weakly) $\bar{0}$-justified . More generally: P is $\bar{a}$-justified iff Q is $T(\bar{a})$-justified.*

Since the above properties trivially hold, we will use them implicitly. The following two lemmas are the PROPs analogs of Lemmas 5.3 and 5.6.

Lemma 5.18 (PROP Structural Lemma)**.** *Every PROP $P(\bar{x})$ such that $|\mathrm{var}(P)| \geq 2$ can be presented in* exactly *one of the following forms: $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$, or $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$, where P_1 and P_2 are non-constant, variable-disjoint PROPs and c is a constant.*

Lemma 5.19. *A partial derivative of a PROP is a PROP.*

The following lemma exhibits yet another important property of PROFs. Note that for zero-characteristic fields the claim holds for every polynomial, however, this in not the case for fields of finite characteristic.

Lemma 5.20. *Let P be a PROP and $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^t \to \mathbb{F}^n$ be such that $P(\mathcal{G})$ is a non-constant polynomial. Then there exists $x_m \in \mathrm{var}(P)$ such that $P\left(\mathcal{G}^1, \ldots, \mathcal{G}^{m-1},\ x_m\ , \mathcal{G}^{m+1}, \ldots, \mathcal{G}^n\right)$ (the polynomial resulting from substituting $\mathcal{G}^i$ for x_i for every $i \neq m$) depends on x_m.*

Proof. We prove the claim by induction on $k = |\mathrm{var}(P)|$. Clearly, $k \geq 1$. We also note that for $k = 1$ the claim is trivial. For $k \geq 2$ we get by Lemma 5.18 that P can be in a one of two forms.

Case 1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$. Since $(P_1 + P_2)(\mathcal{G})$ is a non-constant polynomial we get that w.l.o.g. $P_1(\mathcal{G})$ is a non-constant polynomial. In addition, $|\mathrm{var}(P_1)| < |\mathrm{var}(P)|$ and so by the induction hypothesis we get that there exists $x_m \in \mathrm{var}(P_1)$ such that $P_1\left(\mathcal{G}^1, \ldots, \mathcal{G}^{m-1},\ x_m\ , \mathcal{G}^{m+1}, \ldots, \mathcal{G}^n\right)$ depends on x_m. As P_1 and P_2 are variable disjoint we obtain that $P\left(\mathcal{G}^1, \ldots, \mathcal{G}^{m-1},\ x_m\ , \mathcal{G}^{m+1}, \ldots, \mathcal{G}^n\right)$ depends on x_m as well.

Case 2. $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$. Again we assume w.l.o.g. that $P_1(\mathcal{G})$ is a non-constant polynomial and $P_2(\mathcal{G}) \not\equiv 0$. As before, there exists $x_m \in \mathrm{var}(P_1)$ such that $P_1\left(\mathcal{G}^1, \ldots, \mathcal{G}^{m-1},\ x_m\ , \mathcal{G}^{m+1}, \ldots, \mathcal{G}^n\right)$ depends on x_m, and from variable disjointedness and the fact that $P_2(\mathcal{G}) \not\equiv 0$ we obtain that $P\left(\mathcal{G}^1, \ldots, \mathcal{G}^{m-1},\ x_m\ , \mathcal{G}^{m+1}, \ldots, \mathcal{G}^n\right)$ depends on x_m as well. $\square$

The following example demonstrates that this is not the case for general polynomials over finite characteristic fields.

Example 5.21. *Let $\mathbb{F}$ be a field of characteristic p. Consider $Q(x_1,\dots,x_{p+1}) = \sum_{i=1}^{p+1}\prod_{j\neq i} x_j$. Note that $Q(y,y,\dots,y) = (p+1)\cdot y^p = y^p$ is a non-constant polynomial, while for every m we get that $Q(y,\dots,y,x_m,y,\dots,y) = p\cdot x_m\cdot y^{p-1} + y^p = y^p$ which does not depend on x_m.*

6 Black-Box PIT for Preprocessed Read-Once Polynomials

In this section we give black-box PIT algorithm for PROPs, thus proving Theorem 1. The main idea is to convert a PROP P, that has many variables, each with a low degree, into a polynomial P' with a smaller number of variables while maintaining a reasonable degree, such that $P' \equiv 0$ if and only if $P \equiv 0$. In fact, we construct a low-degree generator for PROPs.

Lemma 6.1. *Let $P \in \mathbb{F}[x_1,\dots,x_n]$ be a nonzero PROP with $|\mathrm{var}(P)| \le 2^t$, for some $t \ge 0$. Then $P(G_{t+1}) \not\equiv 0$. Moreover, if P is non-constant then so is $P(G_{t+1})$ (recall Definition 3.5).*

Proof. We prove the claim by induction on $|\mathrm{var}(P)|$. For $|\mathrm{var}(P)| = 0,1$ the claim is trivial. Assume that $|\mathrm{var}(P)| \ge 2$ (i.e. $t \ge 1$). By Lemma 5.18 we get that P can be in a one of two forms.

Case 1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$. Since P_1 and P_2 are variable disjoint we can assume w.l.o.g. that $|\mathrm{var}(P_1)| \le |\mathrm{var}(P)|/2$ (in particular $|\mathrm{var}(P_1)| < |\mathrm{var}(P)|$). By the induction hypothesis we see that $P_1(G_t) \not\equiv 0$ is a non-constant polynomial. Lemma 5.20 implies that there exists a variable $x_m \in \mathrm{var}(P_1)$ such that even after substituting all the other G_t^i-s, P_1 still depends on x_m. As $x_m \notin \mathrm{var}(P_2)$ we obtain that $P\left(G_t^1,\dots,G_t^{m-1},\ x_m\ ,G_t^{m+1},\dots,G_t^n\right)$ depends on x_m as well. By Observation 3.6, $P(G_{t+1})|_{y_{t+1}=\,\alpha_m} = P\left(G_t^1,\dots,G_t^{m-1},\ G_t^m + z_{t+1}\ ,G_t^{m+1},\dots,G_t^n\right)$. Since z_{t+1} only appears in the m-th coordinate it follows that $P(G_{t+1})|_{y_{t+1}=\,\alpha_m}$ depends on z_{t+1}. Hence, $P(G_{t+1})$ is a non-constant polynomial and in particular $P(G_{t+1}) \not\equiv 0$.

Case 2. $P(\bar{x}) = P_1(\bar{x})\cdot P_2(\bar{x}) + c$. As P_1, P_2 are non-constant and variable disjoint we have that $1 \le |\mathrm{var}(P_1)|, |\mathrm{var}(P_2)| < |\mathrm{var}(P)| \le 2^t$. Hence, we can apply the induction hypothesis on both P_1 and P_2. As $P(G_{t+1}) = P_1(G_{t+1})\cdot P_2(G_{t+1}) + c$, we see that $P(G_{t+1})$ is a non-constant polynomial (since $P_1(G_{t+1}), P_2(G_{t+1})$ are non-constant as well). ☐

Theorem 6.2. *Let $P \in \mathbb{F}[x_1,\dots,x_n]$ be a nonzero PROP with individual degrees bounded by d that depends on at most t variables*[4]*. Then, there exists an explicit set $\mathcal{H}$ of size $|\mathcal{H}| = (nd)^{\mathcal{O}(\log t)}$ such that $P|_{\mathcal{H}} \not\equiv 0$.*

Proof. Denote $\ell = \lceil \log_2 t\rceil + 1$. By Lemma 6.1 we get that $P(G_\ell) \not\equiv 0$. The proof follows from Lemma 3.4. Note that $|\mathcal{H}| \le (n^2 d)^{2\ell} = (nd)^{\mathcal{O}(\log t)}$. ☐

In particular, since every PROP depends on at most n variables, we obtain a quasi-polynomial $(nd)^{\mathcal{O}(\log n)}$ black-box PIT algorithm for PROPs, thus proving Theorem 1.

Remark 6.3. *Lemma 6.1 shows that G_ℓ (for the appropriate value of ℓ) is a generator for ROPs regardless of the degree of P. It can also be proved that G_ℓ is a generator for the more general model of arithmetic read-once formulas with the operations $\{+,\times,/\}$ (addition, multiplication, division), however we will not do it here.*

[4]Clearly $t \le n$ but we choose this more general statement.

6.1 Small Depth Preprocessed Alternating Read-Once Formulas

In this section we use a similar idea to construct a generator for PROPs computed by formulas of a small depth. When considering small depth (preprocessed) read-once formulas we allow the tree to have unbounded fan-in (and not just fan-in 2 as in the usual definition). Moreover, we allow small depth PROFs to use generalized multiplication gates. A generalized multiplication gate on the inputs $(x_1, \ldots, x_k)$ is allowed to compute *any* multiplicative ROP in its input variables.

Definition 6.4. *An* alternating read-once formula *(AROF) over a field* $\mathbb{F}$ *in the variables* $\bar{x} = (x_1, \ldots, x_n)$ *is a tree, of unbounded fan-in, whose leafs are labelled with the input variables and whose internal nodes are labelled with either + or MULT. Each input variable can label at most one leaf. Every leaf and every + gate are labelled with two field elements* $(\alpha, \beta) \in \mathbb{F}^2$*. In addition, any children of a MULT (+) gate is either a leaf or a + (MULT) gate The computation is performed in the following way. A leaf labelled with the variable* x_i *and with* (α, β) *computes the polynomial* $\alpha x_i + \beta$*. If a node* v*, of fan-in* k*, is labelled with + and* (α, β) *and its children compute the polynomials* $\Phi_{v_1}, \ldots, \Phi_{v_k}$ *then the polynomial computed at* v *is* $\Phi_v = \alpha \cdot \left(\sum_{i=1}^{k} \Phi_{v_1}\right) + \beta$*. If* v *is labelled with MULT then it computes a multiplicative ROP in its input variables. That is, if* v *is labelled with the multiplicative ROP* Ψ*, and its children compute the polynomials* $\Phi_{v_1}, \ldots, \Phi_{v_k}$*, then the output of* v *will be the polynomial* $\Phi_v = \Psi(\Phi_{v_1}, \ldots, \Phi_{v_k})$*. The* depth *of an AROF is defined as the depth of its tree. In other words, the length of the longest path from a leaf to the root.*

A preprocessed alternating read-once formula *(P-AROF for short) is an AROF* Φ *whose leafs are labelled with non-constant univariate polynomials* $T_1(x_1), T_2(x_2), \ldots, T_n(x_n)$ *(namely, a preprocessing) and the computation is performed as before (in a similar manner to Definition 5.15).*

Example 6.5. *The polynomial computed in Example 5.2 has an AROF of depth-*1 *that contains a single MULT gate.*

Definition 6.6. *For a PROP* $P \in \mathbb{F}[x_1, \ldots, x_n]$ *we define* $\text{depth}(P)$ *to be the depth of the shallowest P-AROF computing it.*

In fact, it can be shown that all the non-degenerate P-AROFs computing the same PROP have the same depth. We now give the analog of Lemmas 5.18, 5.19 and for the case of P-AROFs.

Lemma 6.7. *Every PROP* $P(x)$ *with* $|\text{var}(P)| \geq 2$ *of depth* D *can be presented in* exactly *one of the following forms:* $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \ldots + P_k(\bar{x})$ *or* $P(\bar{x}) = f\left(P_1(\bar{x}), P_2(\bar{x}), \ldots, P_k(\bar{x})\right)$*, where the polynomials* $\{P_j(\bar{x})\}_{j \in [k]}$ *are non-constant variable-disjoint PROPs of depth at most* $D-1$*, and* f *is a multiplicative ROP.*

The proof is similar to the proof of Lemma 5.3 so we omit it.

Lemma 6.8. *A partial derivative of a PROP* $P(\bar{x})$ *of depth* D *is a PROP of depth at most* D*.*

Proof. Let P be a PROP of depth D, $x_i \in \text{var}(P)$ and $\alpha \in \mathbb{F}$. We prove the lemma by induction on $m = |\text{var}(P)|$. For $m = 0, 1$ the claim is trivial. For $m \geq 2$ we get by Lemma 6.7 that P can be in a one of two forms.

Case 1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \ldots + P_k(\bar{x})$. In this case we get that since the P_j's are variable-disjoint PROPs we can assume w.l.o.g that $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial P_1}{\partial_\alpha x_i}$. In addition, $|\text{var}(P_1)| < |\text{var}(P)|$. By the induction hypothesis we get that $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial P_1}{\partial_\alpha x_i}$ is a PROP of depth at most $D-1$.

Case 2. $P(\bar{x}) = f\left(P_1(\bar{x}), P_2(\bar{x}), \ldots, P_k(\bar{x})\right)$, where f is a multiplicative ROP in $\{y_1, y_2, \ldots y_k\}$. Assume w.l.o.g that $x_i \in \text{var}(P_1)$. By the chain rule we get that $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial f}{\partial y_1}(P_1, \ldots, P_k) \cdot \frac{\partial P_1}{\partial_\alpha x_i}$. As f is a multiplicative ROP, we get that $\frac{\partial f}{\partial y_1}$ is a multiplicative ROP in the variables $y_2, \ldots, y_k$. In addition, our induction hypothesis implies that $\frac{\partial P_1}{\partial_\alpha x_i}$ is a PROP of depth at most $D-1$ (as the depth of P_1 is at most $D-1$). As the P_j's are variable disjoint it follows that $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial f}{\partial y_1}(P_1, \ldots, P_k) \cdot \frac{\partial P_1}{\partial_\alpha x_i}$ is a PROP of depth at most D. □

We now give a generator for small depth P-AROFs. The idea is to 'reduce' the depth of the formula level by level. In a P-AROF each pair of adjacent levels consists of + and MULT gates. To reduce a + gate, we use Lemma 5.20. To reduce a MULT gate, we use the following lemma. Note that in the proof of Lemma 6.1 we made an explicit usage of Lemma 6.9 for the case $k = 2$.

Lemma 6.9. *Let $Q(x_1, \ldots, x_k) : \mathbb{F}^k \to \mathbb{F}$ be a non-constant multiplicative ROP and $h_1(\bar{y}), \ldots, h_k(\bar{y})$ be non-constant polynomials. Then $Q(h_1, \ldots, h_k)$ is a non-constant polynomial.*

Proof. The proof follows immediately by a simple induction on the structure of the multiplicative ROF for Q. We just notice that the top gate is $\times$ and by induction the children are non-constant and so their product is non-constant. The base case of the induction is trivial. □

Finally, we can state the depth-version of Lemma 6.1.

Lemma 6.10. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-constant PROP of depth $\leq D$. Then $P(G_{D+1})$ is a non-constant polynomial (in particular $P(G_{D+1}) \not\equiv 0$).*

Proof. We prove the claim by induction on depth(P). For depth$(P) = 0$ we get that $|\text{var}(P)| \leq 1$ and the proof is trivial. Now assume that depth$(P) \geq 1$. This implies $|\text{var}(P)| \geq 2$. By Lemma 6.7, P can be written in exactly one of the following two forms.

Case 1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \ldots + P_k(\bar{x})$, where the polynomials $P_j(\bar{x})$ are non-constant variable-disjoint PROPs of depth at most $D-1$: By the induction hypothesis we see that $P_1(G_D)$ is a non-constant polynomial. By Lemma 5.20 there is a variable $x_m \in \text{var}(P_1)$ such that even after substituting all the other G_D^i-s, P_1 still depends on x_m. As $x_m \notin \text{var}(P_j)$ for $2 \leq j \leq k$ it follows that $P\left(G_D^1, \ldots, G_D^{m-1}, \; x_m \;, G_D^{m+1}, \ldots, G_D^n\right)$ depends on x_m as well. By Observation 3.6 we get that $P(G_{D+1})|_{y_{D+1} = \alpha_m} = P\left(G_D^1, \ldots, G_D^{m-1}, \; G_D^m + z_{D+1} \;, G_D^{m+1}, \ldots, G_D^n\right)$. As z_{D+1} only appears in the m-th coordinate it follows that $P(G_D)|_{y_{D+1} = \alpha_m}$ depends on z_{D+1}. Therefore, $P(G_{D+1})$ is a non-constant polynomial and in particular $P(G_{D+1}) \not\equiv 0$.

Case 2. $P(\bar{x}) = f\left(P_1(\bar{x}), P_2(\bar{x}), \ldots, P_k(\bar{x})\right)$, where the polynomials $P_j(\bar{x})$ are non-constant variable-disjoint PROPs of depth at most $D-1$, and f is a multiplicative ROP: By applying the induction hypothesis on each P_j we get that $P_j(G_{D+1})$ is a non-constant polynomial, for every $j \in [k]$. As $P(G_{D+1}) = f\left(P_1(G_{D+1}), P_2(G_{D+1}), \ldots, P_k(G_{D+1})\right)$ it follows from Lemma 6.9 that $P(G_{D+1})$ is a non-constant polynomial. □

We now give an analog of Theorem 6.2 that clearly implies Theorem 4, for the case $k = 1$. The proof is exactly the same and so we omit it.

Theorem 6.11. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a nonzero PROP with individual degrees bounded by d and depth at most D. Then, there exists an explicit set $\mathcal{H}$ of size $|\mathcal{H}| = (nd)^{\mathcal{O}(D)}$ such that $P|_{\mathcal{H}} \not\equiv 0$.*

7 PIT for Sum of Preprocessed Read-Once Formulas

In this section we prove Theorems 2, 3, 4 and 7. Specifically, we are given k PROPs $\{F_m\}_{m\in[k]}$ and we have to find whether they sum to zero or not. In other words, let $F = F_1 + \ldots + F_k$. The problem is to decide whether $F \equiv 0$. Our algorithm for the problem has two steps. First we find a common justifying assignment to $F_1, \ldots, F_k$ using Algorithm 1. Once we have a common justifying assignment we can assume w.l.o.g. that all the input formulas are $\bar{0}$-justified (see Proposition 2.4). In the second step we simply verify that F vanishes on a relatively small set of vectors, each of weight at most $3k$. Theorem 7.4 then guarantees that $F \equiv 0$. In the black-box version of the algorithm we construct a generator that simulates this process.

7.1 Hardness of Representation

The main tool in our proof is Theorem 7.1 that shows that we cannot represent $\mathcal{P}_n \triangleq \prod_{i=1}^n x_i$ as a sum of less than $\frac{1}{3}n$ $\bar{0}$-justified ROPs. We call this approach a *hardness of representation* approach as the proof is based on the fact that a simple polynomial cannot be represented by a sum of a 'small' number of $\bar{0}$-justified ROPs. Then, using this preliminary result, we prove a stronger hardness of representation theorem (Theorem 7.2) for PROPs. Namely, we show that every nonzero polynomial that has $\mathcal{P}_n$ as a factor, cannot be written as a sum of at most $\frac{n}{3}$ $\bar{0}$-justified PROPs. For completeness we give a simple representation of $\mathcal{P}_n$ as a sum of n $\bar{0}$-justified ROPs, showing the near optimality of our bound.

Theorem 7.1. *$\mathcal{P}_n(\bar{x})$ cannot be represented as sum of $k \leq \frac{n}{3}$ weakly-$\bar{0}$-justified ROPs.*

Proof. Let $\{F_m(\bar{x})\}_{m\in[k]}$ be k weakly-$\bar{0}$-justified ROPs over $\mathbb{F}[x_1, \ldots, x_n]$. We prove the claim by induction on k. For $k = 0, 1$ the claim follows from the definition of weak-$\bar{0}$-justification. We now assume that $k \geq 2$ and that $n \geq 3k$. We shall assume for a contradiction that $\sum\limits_{m=1}^{k} F_m = \mathcal{P}_n$. The idea of the proof is to eliminate a 'large' number of ROPs at the cost of a 'small' number of variables. Specifically, we find a small set of (indices of) input variables $J \subseteq [n-1]$ and a constant $\alpha \neq 0 \in \mathbb{F}$ such that after we take a partial derivative with respect to all of the variables in J and substitute $x_n = \alpha$ (that is we consider the ROPs $\{\partial_J F_m|_{x_n=\alpha}\}_{m\in[k]}$) we eliminate 'many' F_m-s in a way that the rest of the ROPs remain weakly-$\bar{0}$-justified . We thus get a representation of $\partial_J \mathcal{P}_n|_{x_n=\alpha} = \alpha \cdot \mathcal{P}_{\hat{n}}$ (for a relatively large $\hat{n}$) as a sum of a 'small' number of weakly-$\bar{0}$-justified ROPs. Then we use the induction hypothesis to reach a contradiction. We now proceed with the proof. There are two cases to consider.

Case 1: There exist $i \neq j \in [n]$ and $m \in [k]$ such that $\frac{\partial^2 F_m}{\partial x_i \partial x_j} \equiv 0$ (namely, F_m does not contain $x_i \cdot x_j$ in any of its monomials). Assume w.l.o.g. that $i = n-1, j = n$ and $m = k$. By considering the partial derivatives with respect to $\{x_n, x_{n-1}\}$ we see that $\sum\limits_{m=1}^{k-1} \frac{\partial^2 F_m}{\partial x_n \partial x_{n-1}} = \mathcal{P}_{n-2}$. It may be the case that more than one F_m vanishes when we take a partial derivative w.r.t. $\{x_n, x_{n-1}\}$, however they cannot all vanish simultaneously (as $\mathcal{P}_n$ contains $x_n \cdot x_{n-1}$). By Lemma 5.11 we get that the polynomials $\left\{\frac{\partial^2 F_m}{\partial x_n \partial x_{n-1}}\right\}$ are weakly-$\bar{0}$-justified ROPs. Hence, we obtain a representation of $\mathcal{P}_{n-2}$ as a sum of $0 < \hat{k} \leq k-1$ weakly-$\bar{0}$-justified ROPs such that $0 < 3\hat{k} \leq 3(k-1) = 3k-3 < n-2$ which contradicts the induction hypothesis.

Case 2: For every $i \neq j \in [n]$ and $m \in [k]$ we have that $\frac{\partial^2 F_m}{\partial x_i \partial x_j} \not\equiv 0$. Thus, by Lemma 5.10 we get that the polynomials $\{F_m\}_{m\in[k]}$ are multiplicative ROPs. In addition, for every $m \in [k]$ we have that $\mathrm{var}(F_m) = [n]$. In particular, $|\mathrm{var}(F_m)| \geq 6$. Lemma 5.13 implies that $\forall m \in [k]$ there exist $j_m \in [n]$, $\alpha_m \neq 0 \in \mathbb{F}$ and a ROP $h_m(\bar{x})$ such that $\frac{\partial F_m}{\partial x_{j_m}} = (x_n - \alpha_m) h_m(\bar{x})$. Let $A = \{\alpha_m \mid m \in [k]\}$. Clearly $0 \notin A$. For every $\alpha \in A$ denote

$$E_\alpha \triangleq \{m \in [k] \mid \alpha_m = \alpha\}$$

and

$$B_\alpha \triangleq \{m \in [k] \mid \alpha_m \neq \alpha \text{ and } F_m|_{x_n=\alpha} \text{ is not weakly-}\bar{0}\text{-justified }\} .$$

Intuitively, E_α is set of the ROPs that can be eliminated by substituting $x_n = \alpha$ and B_α is set of ('bad') ROPs that will become non weakly-$\bar{0}$-justified upon the substitution and thus require a special treatment. From the definition of A we have that $|E_\alpha| \geq 1$ and $\sum\limits_{\alpha \in A} |E_\alpha| = k$. More specifically, the E_α's form a partition of $[k]$. Similarly, Lemma 5.13 implies that for each $\alpha \neq \alpha' \in A$ the sets B_α and $B_{\alpha'}$ are disjoint (since for every ROP there exists at most one bad value β of x_n) and therefore $\sum\limits_{\alpha \in A} |B_\alpha| \leq k$. Hence, there exists $\alpha_0 \in A$ such that $|B_{\alpha_0}| \leq |E_{\alpha_0}|$. Let $I = E_{\alpha_0} \cup B_{\alpha_0}$ and $J = \{j_m \mid m \in I\}$. From the definition, $I \subseteq [k]$ and $J \subseteq [n]$. In addition, $1 \leq |J| \leq |I| \leq |E_{\alpha_0}| + |B_{\alpha_0}| \leq 2\,|E_{\alpha_0}|$ and $n \notin J$. Consider the following ROPs for every $m \in [k]$: $F'_m \triangleq \partial_J F_m$. Then the ROPs F'_m's have the following properties.

1. By Lemma 5.11 we get that every F'_m is a weakly-$\bar{0}$-justified ROP.

2. For every $m \in I$ we have that $F'_m = (x_n - \alpha_m) h'_m(\bar{x})$ for some ROP $h'_m(\bar{x})$. Indeed, as $j_m \in J$ we have that

$$F'_m = \partial_J F_m = \partial_{J\setminus\{j_m\}}\left(\frac{\partial F_m}{\partial x_{j_m}}\right) = \partial_{J\setminus\{j_m\}}\left((x_n - \alpha_m) h_m(\bar{x})\right) = (x_n - \alpha_m) \cdot \partial_{J\setminus\{j_m\}} h_m(\bar{x}).$$

3. For every $m \in I$ we have that $h'_m(\bar{x})$ is a weakly-$\bar{0}$-justified ROP (this follows from Lemma 5.12 and the previous two properties).

For $m \in [k]$ consider the following ROPs: $F''_m \triangleq \partial_J F_m|_{x_n=\alpha_0} = F'_m|_{x_n=\alpha_0}$. Based on the above we can conclude that:

- For every $m \in E_{\alpha_0}$ it holds that $F''_m = (\alpha_0 - \alpha_m) h'_m(\bar{x}) \equiv 0$ (by definition of E_{α_0} we have that $\alpha_m = \alpha_0$).

- For every $m \in B_{\alpha_0}$ we have that $F''_m = (\alpha_0 - \alpha_m) h'_m(\bar{x})$ is a nonzero weakly-$\bar{0}$-justified ROP. Notice that in contrary to F_m, the structure of F'_m guarantees that it remains weakly-$\bar{0}$-justified when substituting $x_n = \alpha_0$.

- For $m \in [k] \setminus I$ the definitions of E_{α_0} and B_{α_0} guarantee that $F_m|_{x_n=\alpha_0}$ is a weakly-$\bar{0}$-justified ROP. Lemma 5.11 implies that the same holds for $F''_m = \partial_J (F_m|_{x_n=\alpha_0})$ as well. Note that in this case it is also possible that $F''_m \equiv 0$.

Thus, $F''_m \equiv 0$ for $m \in E_{\alpha_0}$ and F''_m is a weakly-$\bar{0}$-justified ROP for $m \in [k] \setminus E_{\alpha_0}$. W.l.o.g. let us assume that $J = \{\hat{n}+1, \hat{n}+2, \ldots, n-2, n-1\}$ for some $\hat{n}$. We get that $\sum\limits_{m=1}^{k} F''_m = \partial_J \mathcal{P}_n|_{x_n=\alpha_0} = \alpha_0 \cdot \mathcal{P}_{\hat{n}}$. That is, we found a representation of $\alpha_0 \cdot \mathcal{P}_{\hat{n}}$ as a sum of weakly-$\bar{0}$-justified ROPs, where at least $|E_{\alpha_0}|$ of the ROPs are zeros. Notice that $2|E_{\alpha_0}| \geq |J| = (n-1) - \hat{n}$ and $|E_\alpha| \geq 1$. Therefore, we have found a representation of $\alpha_0 \cdot \mathcal{P}_{\hat{n}}$ as a sum of $0 \leq \hat{k} < k$ weakly-$\bar{0}$-justified ROPs such that

$$0 \leq 3\hat{k} \leq 3(k - |E_\alpha|) = 3k - 3|E_\alpha| \leq n - 3|E_\alpha| = n - 1 - 2|E_\alpha| + 1 - |E_\alpha| \leq \hat{n} + 1 - |E_\alpha| \leq \hat{n}.$$

By our induction hypothesis we get that $\alpha_0 = 0$, which is a contradiction (recall that $\alpha_0 \in A$ and $0 \notin A$). Hence, $\mathcal{P}_n$ cannot be represented as a sum of less than $\frac{n}{3}$ weakly-$\bar{0}$-justified ROPs. This completes the proof of Theorem 7.1. □

We now generalize the hardness of representation theorem to the case of PROPs.

Theorem 7.2. *For every $g(\bar{x}) \not\equiv 0$ the polynomial $g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ cannot be represented as sum of k weakly-$\bar{0}$-justified PROPs for $k \leq \frac{n}{3}$.*

Proof. Let $\{F_m(\bar{x})\}_{m \in [k]}$ be k weakly-$\bar{0}$-justified PROPs with individual degrees bounded by d over $\mathbb{F}$ and $\{(Q_m(\bar{z}), T^m(\bar{x}))\}_{m \in [k]}$ be standard decompositions for them. Recall (Lemma 5.17) that $\{Q_m(\bar{z})\}_{m \in [k]}$ are weakly-$\bar{0}$-justified ROPs. Denote $T^m_i(x_i) = \sum_{j=0}^{d} \alpha_{j,i,m} \cdot x_i^j$. Assume that $F = \sum_{m=1}^{k} F_m = g(x) \cdot \mathcal{P}_n(\bar{x})$. Let $\prod_{i=1}^{n} x_i^{e_i}$ be some monomial appearing in $g(x)$. It follows that the monomial $A = \prod_{i=1}^{n} x_i^{e_i+1}$ appears in F. Since the F_m-s are ROFs we have that $A = \sum_{m=1}^{k} \tilde{F}_m$, where $(Q_m(\bar{z}), \alpha_{e_i+1,i,m} \cdot \prod_{i=1}^{n} x_i^{e_i})$ is a standard decomposition for $\tilde{F}_m$. Indeed, the polynomial computed by this sum has no constant term (as one can see by setting all the variables to zero and comparing to F. Additionally, each of its monomials must also appear in F and as the degree of x_i in it is exactly $x_i^{e_i+1}$, it must equal A. It now follows that $\sum_{m=1}^{k} Q_m = \alpha \mathcal{P}_n(\bar{x})$ where α is the coefficient of $\prod_{i=1}^{n} x_i^{e_i}$ in g. By Theorem 7.1 it follows that $n > 3k$. □

To complete the picture we show that over a large field ($|\mathbb{F}| > n$) the polynomial $\mathcal{P}_n(\bar{x})$ can be represented as a sum of n $\bar{0}$-justified ROPs.

Lemma 7.3. *Let $\mathbb{F}$ be a field with more than n elements. Then the polynomial $\mathcal{P}_n(\bar{x})$* ***can*** *be represented as a sum of n $\bar{0}$-justified ROPs.*

Proof. Let $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F} \setminus \{0\}$ be a subset of n distinct nonzero elements. For every $i \in [n]$ let $u_i(w)$ be the i-th Lagrange Interpolation polynomial over A (see Definition 3.5).Let $\varphi(\bar{x}, t) = (x_1 + t)(x_2 + t) \cdots (x_n + t) - t^n$. Since the degree of t in $\varphi(\bar{x}, t)$ is $n - 1$ we get $\varphi(\bar{x}, t) = \sum_{m=1}^{n} u_m(t) \cdot \varphi(\bar{x}, \alpha_m)$ (i.e., interpolate $\varphi(\bar{x}, t)$ as a degree $n - 1$ polynomial in t). Consequently, $\mathcal{P}_n(\bar{x}) = \varphi(\bar{x}, 0) = \sum_{m=1}^{n} u_m(0) \cdot \varphi(\bar{x}, \alpha_m) = \sum_{m=1}^{n} F_m(\bar{x}) + c$ where $F_m(\bar{x}) \triangleq u_m(0) \cdot \varphi(\bar{x}, \alpha_m)$ are $\bar{0}$-justified ROPs and $c = -\sum_{m=1}^{n} u_m(0) \cdot \alpha_m^n$. Clearly, we can add c to F_n, and so the proof is completed. □

Next, we show that if $\sum_{m=1}^{k} F_m$, a sum of $\bar{0}$-justified PROPs, vanishes on a certain small set then the sum is zero.

Theorem 7.4. *Let $\{F_m(\bar{x})\}_{m\in[k]}$ be $\bar{0}$-justified PROPs over $\mathbb{F}$ with individual degrees bounded by d. Let $W \subseteq \mathbb{F}$ be a subset of size $d+1$ [5] such that $0 \in W$. Let $F(\bar{x}) = \sum_{m=1}^{k} F_m(\bar{x})$. Then $F \equiv 0$ if and only if $F|_{\mathcal{A}^n_{3k}(W)} \equiv 0$ (recall the definition in Section 3.1).*

Proof. If $F \equiv 0$ then the claim is clear. For the other direction we apply induction on n. Our base case is when $n \le 3k$. In this case F is a polynomial in $n \le 3k$ variables of degree at most d in each variable and therefore by Lemma 2.13 we get that $F|_{\mathcal{A}^n_{3k}(W)} \equiv 0$ implies that $F \equiv 0$. We now assume that $n > 3k \ge 3$. Let $\ell \in [n]$. Consider the restriction of the F_m's and F to the subspace $x_\ell = 0$. We now show that the required conditions hold for $F' \overset{\Delta}{=} F|_{x_\ell=0}$ and $\left\{F'_m \overset{\Delta}{=} F_m|_{x_\ell=0}\right\}_{m\in[k]}$ as well. Indeed, the $\{F'_m\}_{m\in[k]}$ are $\bar{0}$-justified PROPs with individual degrees bounded by d. Moreover, $F'|_{\mathcal{A}^{n-1}_{3k}(W)} = F'|_{\mathcal{A}^n_{3k}(W)} \equiv 0$. From the induction hypothesis we conclude that $F|_{x_\ell=0} = F' \equiv 0$ and therefore x_ℓ is a factor of F (see Lemma 2.14). As this holds for every $\ell \in [n]$ we get that $\mathcal{P}_n(\bar{x})$ divides $F(\bar{x})$ or equivalently $F(\bar{x}) = g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ for some $g(\bar{x}) \in \mathbb{F}[x_1, \ldots, x_n]$. It follows that $g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ is a sum of k $\bar{0}$-justified PROPs. As $n > 3k$ we get by Theorem 7.2 that we must have that $g(\bar{x}) \equiv 0$. Hence $F = g \cdot \mathcal{P}_n \equiv 0$. This completes the proof of the theorem. $\square$

The following is an immediate corollary from Theorem 7.4 and Observation 3.6.

Corollary 7.5. *In the settings of Theorem 7.4 let $\bar{a}$ be a common justifying assignment for the PROPs $F_1, F_2, \ldots, F_k$. Then $F \equiv 0$ iff $F_{\bar{a}}|_{\mathcal{A}^n_{3k}(W)} \equiv 0$ and hence $F \equiv 0$ iff $F_{\bar{a}}(G_{3k}) \equiv 0$.*

7.2 Non Black-Box Identity Testing Algorithm for Sum of PROPs

In this section we prove Theorem 2. For the algorithm we assume that $|\mathbb{F}| > d$, where d is the bound on the individual degrees of the PROFs.

Algorithm 2 PIT algorithm for sum of preprocessed read-once formulas

Input: PROFs $F_1, \ldots, F_k$ with individual degrees bounded by d
Output: "true" iff $F \overset{\Delta}{=} F_1 + \cdots + F_k \equiv 0$

```
1: Choose $W \subseteq \mathbb{F}$ a subset of size $d+1$, such that $0 \in W$
2: Acquire a common justifying assignment $\bar{a}$ for $F_1, F_2, \ldots, F_k$ {using Algorithm 1}.
3: Check that $F_{\bar{a}}|_{\mathcal{A}^n_{3k}(W)} \equiv 0$.
```

Lemma 7.6. *Algorithm 2 runs in time $(nd)^{\mathcal{O}(k)}$ and correctly determines whether $F \equiv 0$.*

Proof. We start by showing the correctness of the algorithm. If the algorithm did not return "true" then $F_{\bar{a}}$ evaluates to a nonzero value which implies that $F_{\bar{a}} \not\equiv 0$ and hence $F \not\equiv 0$. If, on the other hand, the algorithm outputs "true", then $F_{\bar{a}}|_{\mathcal{A}^n_{3k}(W)} \equiv 0$, where $\bar{a}$ is common justifying assignment for the PROPs $F_1, \ldots, F_k$. Corollary 7.5 now implies that $F \equiv 0$.

To analyze the running time we first note that given a PROF (explicitly) we can determine whether it computes the zero polynomial in time $\mathcal{O}(n)$ by a simple traversal over the formula. Therefore, acquiring a common justifying assignment $\bar{a}$ for the formulas requires time $\mathcal{O}(n^4k^2d)$ (set $T_{\mathcal{C}'} = \mathcal{O}(n)$ in Lemma 4.2). Verifying that $F_{\bar{a}}|_{\mathcal{A}^n_{3k}(W)} \equiv 0$ requires at most $|\mathcal{A}^n_{3k}(W)| \cdot k$ time. Hence, the running time is at most $k \cdot (nd)^{\mathcal{O}(k)} = (nd)^{\mathcal{O}(k)}$ (see Section 3.1). $\square$

Theorem 2 is an immediate corollary of Lemma 7.6.

[5] We implicitly assume that $|\mathbb{F}| > d$.

7.3 Black-Box Identity Testing Algorithm for Sum of PROPs

In this section we prove Theorems 3, 4 and 7. The idea is to give a generator that, in some sense, simulates Algorithm 2. Specifically, a generator whose image contains a common justifying assignment and the set $\mathcal{A}^n_{3k}(W)$ (for an appropriate subset W). For that purpose we use Corollary 4.4 of Lemma 4.3 that shows how to obtain a justifying set from a black-box PIT algorithm. Thus, we actually show how to construct a generator for sum of k PROFs from a generator for a single PROF.

Theorem 7.7. *Let $F_1, \ldots, F_k$ be PROPs, with individual degrees bounded by d, that are computed by a circuit class $\mathcal{C}$ such that $F = \sum_{m=1}^{k} F_m \not\equiv 0$. Let $\mathcal{C}'$ be a circuit class such that $\partial\mathcal{C} \subseteq \mathcal{C}'$ [6]. Let $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^t \to \mathbb{F}^n$ be a generator for $\mathcal{C}'$. Then $F(\mathcal{G} + G_{3k}) \not\equiv 0$. That is, the mapping $\mathcal{G} + G_{3k} : \mathbb{F}^{t+6k} \to \mathbb{F}^n$, obtained by component-wise addition, is a generator for sum of k PROPs.*

Proof. By Corollary 4.4 there exists $\bar{\gamma} \in \overline{\mathbb{F}}^t$ such that $\bar{a} = \mathcal{G}(\bar{\gamma})$ is a common justifying assignment for $F_1, \ldots, F_k$. Now, by Corollary 7.5 we get: $F\left(\mathcal{G}(\bar{\gamma}) + G_{3k}\right) = F_{\bar{a}}(G_{3k}) \not\equiv 0$. In particular, $F(\mathcal{G} + G_{3k}) \not\equiv 0$. □

Using Theorem 7.7 and Lemma 3.4 we prove Theorems 3 and 7.

Proof of Theorem 3. From Lemma 6.1 we get that for $\ell = \lceil \log_2 n \rceil + 1$ the mapping $G_\ell : \mathbb{F}^{2\ell} \to \mathbb{F}^n$ is a generator for PROFs. Lemma 5.19 implies that PROFs are closed under partial derivatives. Hence, by Theorem 7.7 we get that the mapping $G_{\ell+3k}$ is a generator for sum of k PROFs. The hitting set produced by Lemma 3.4 is of size $|\mathcal{H}| = (n^2 d)^{\mathcal{O}(6k+2\ell)} = (nd)^{\mathcal{O}(k+\log n)}$. □

The next case is when all the F_m's are bounded depth PROFs.

Proof of Theorem 4. Lemma 6.10 implies that the mapping $G_\ell : \mathbb{F}^{2\ell} \to \mathbb{F}^n$, for $\ell = D + 1$, is a generator for depth-D PROFs. By Lemma 6.8, this circuit class is closed under partial derivatives. Therefore, it follows from Theorem 7.7 that $G_{\ell+3k}$ is a generator for sum of k PROFs of depth at most D. Lemma 3.4 now gives a hitting set of size $|\mathcal{H}| = (n^2 d)^{\mathcal{O}(6k+2\ell)} = (nd)^{\mathcal{O}(D+k)}$. □

The last result in this vein is a black-box PIT algorithm for the case where the black box holds a sum of PROFs that is a read-k (i.e., every variable appears in at most k PROFs).

Definition 7.8. *Let $\{F_m\}_{m\in[r]}$ be PROFs. We say that $F = \sum_{m=1}^{r} F_m$ is a* read-k sum *if for each $i \in [n]$ there are at most k functions F_m that depend on x_i. In other words, each variable is read at most k times in F.*

We can easily extend a PIT algorithm for sum of k PROFs to a PIT algorithm for read-k sum with the following observation:

Observation 7.9. *Let F be a read-k sum. Then $\frac{\partial F}{\partial_\alpha x_i}$ is a sum of (at most) k PROFs, for every $i \in [n]$ and each $\alpha \in \mathbb{F}$.*

Proof of Theorem 7. Given a read-k sum F we can, by Lemma 4.1, find $\mathrm{var}(F)$. Observation 7.9 implies that we can use Theorems 2, 3 and 4 as the corresponding PIT algorithm. □

[6] Note that we can let $\mathcal{C}$ be the class of PROFs, however we give the more general statement in order to apply it for models for which we have a more efficient generator than the one for PROFs.

8 Depth-3 Arithmetic Circuits

In this section we give a new black-box PIT algorithm for depth-3 circuits based on the *hardness of representation* approach. We also derive a new PIT algorithm for multilinear depth-3 circuits and a special case of depth-4 circuits based on Theorem 4. We first define the relevant models and discuss the known results.

Definition 8.1. *A* linear function *over* $\mathbb{F}$ *is a polynomial of the form* $L(\bar{x}) = \sum_{i=1}^{n} b_i x_i + b_0$*, where* $\forall i\ b_i \in \mathbb{F}$*. A polynomial* $P(\bar{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *is a* linear product *if it can be represented as a product of linear functions:* $P(\bar{x}) = \prod_j L_j(\bar{x})$ *where each* $L_j(\bar{x})$ *is a linear function.*

Definition 8.2. *A depth-3* $\Sigma\Pi\Sigma(k)$ *circuit* C *computes a polynomial of the form* $C(\bar{x}) = \sum_{m=1}^{k} F_m(\bar{x}) = \sum_{m=1}^{k} \prod_{j=1}^{d_m} L_{m,j}(\bar{x})$*, where each* $L_{m,j}(\bar{x})$ *is a linear function. The* F_m*-s are the multiplication gates of the circuit. Note that the* F_m*-s are, in fact, linear products. We denote by* $\Sigma\Pi\Sigma(k,d)$ *a* $\Sigma\Pi\Sigma(k)$ *circuit such that each multiplication gate has degree at most* d*. I.e.* $d_m \leq d$ *for every* m*. A* multilinear $\Sigma\Pi\Sigma(k)$ *circuit is a* $\Sigma\Pi\Sigma(k)$ *circuit such that each* F_m *is a multilinear polynomial. Note, that in this case the degree is bounded from above by* n*. Moreover, note that in the multilinear case each* F_m *is a ROP.*

As before, we shall also consider preprocessed $\Sigma\Pi\Sigma(k)$ circuits, that form a special subclass of depth-4 circuits. The definition is similar to the way PROFs are generated from ROFs.

Definition 8.3. *A* preprocessed linear function *is a polynomial of the form* $L(\bar{x}) = \sum_{i=1}^{n} T_i(x_i)$*, where each* $T_i(x_i)$ *is a univariate polynomials. A polynomial* $F(\bar{x})$ *is a* preprocessed linear product *if it can be represented as a product of preprocessed linear functions. A* Preprocessed $\Sigma\Pi\Sigma(k)$ *(or* P$\Sigma\Pi\Sigma(k)$ *- for short) computes a polynomial of the form:* $C(\bar{x}) = \sum_{m=1}^{k} F_m(\bar{x})$*, where the* F_m*-s are preprocessed linear products.*

As a corollary of Theorem 4 we obtain a PIT algorithm for preprocessed multilinear depth-3 circuits (Theorem 5). Indeed, in a multilinear $\Sigma\Pi\Sigma(k)$ circuit each multiplication gate is a depth-2 ROP. Therefore, a preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuit is actually a sum of k depth-2 PROPs. Having this in mind we can apply the results of Section 7 (i.e. Theorems 7.2 and 7.4) [7]. Thus, Theorem 5 is in fact an immediate corollary.

Proof of Theorem 5. By the above discussion, a preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuit is a sum of k depth-2 PROPs with the same individual degrees. The result follows from Theorem 4. □

We now describe a new algorithm for PIT of general $\Sigma\Pi\Sigma(k)$ circuits. Before presenting our algorithm we give several notations (originally defined in [DS06]) and discuss related results.

Definition 8.4. *Let* $C(\bar{x}) = \sum_{m=1}^{k} F_m(\bar{x})$ *be a* $\Sigma\Pi\Sigma(k)$ *circuit. We say that* C *is* minimal *if no subset of the multiplication gates sums to zero. We define* gcd(C) *as the linear product of all the non-constant linear functions that belong to all the* F_m*'s. I.e.* $\gcd(C) = \gcd(F_1, \ldots, F_k)$*. We say*

[7] Theorem 7.2 is tight for multilinear depth-3 circuits since Lemma 7.3 gives a representation of $\mathcal{P}_n$ as a sum of n $\bar{0}$-justified linear products and a constant, and by a slightly more sophisticated argument one can get rid of the constant.

that C is simple *if* $\gcd(C) = 1$*. The simplification of C, denoted by* $\mathrm{sim}(C)$*, is defined as* $C/\gcd(C)$*. Note that if C is a $\Sigma\Pi\Sigma(k,d)$ then so is* $\mathrm{sim}(C)$*. Let* $\mathrm{rank}(C)$ *be defined as the dimension of the span of the linear functions in C, viewed as $(n+1)$-dimensional vectors over $\mathbb{F}^{n+1}$.*

In [DS06] it was proved that the rank of a $\Sigma\Pi\Sigma(k)$ circuit computing the identically zero polynomial cannot be too large. Specifically, if the circuit is simple and minimal, then the dimension of the linear space spanned by all the linear functions in the circuit is relatively small. This bound on the rank was recently improved by [SS09] for finite fields and by [KS09b] over $\mathbb{R}$ and $\mathbb{Q}$.

Theorem 8.5 ([SS09, KS09b])**.** *Let $C \equiv 0$ be a simple and minimal $\Sigma\Pi\Sigma(k,d)$ circuit over a field $\mathbb{F}$. Then* $\mathrm{rank}(C) < R(k,d)$*, where, when $\mathbb{F}$ is finite, $R(k,d) = \mathcal{O}(k^3 \log d)$ and when $\mathbb{F} = \mathbb{R}$ or $\mathbb{Q}$, $R(k,d) = k^{\mathcal{O}(k)}$.*

In [KS08] a black-box PIT algorithm for $\Sigma\Pi\Sigma(k,d)$ circuits was given. The running time is $\mathrm{poly}(n) \cdot d^{R(k,d)}$. Hence, [SS09, KS09b] get the following corollary.

Theorem 8.6 ([SS09, KS09b])**.** *There is a deterministic black-box PIT algorithm for $\Sigma\Pi\Sigma(k,d)$ circuits over $\mathbb{F}$ that runs in time* $\mathrm{poly}(n) \cdot d^{R(k,d)} = \mathrm{poly}(n) \cdot d^{\mathcal{O}(k^3 \log d)}$ *when $\mathbb{F}$ is finite and in time* $\mathrm{poly}(n) \cdot d^{k^{\mathcal{O}(k)}}$ *when $\mathbb{F} = \mathbb{R}$ or $\mathbb{Q}$, $R(k,d) = k^{\mathcal{O}(k)}$.*

On the other hand, in the *non black-box* model there is a $\mathrm{poly}(n, d^k)$ time PIT algorithm.

Theorem 8.7 ([KS07])**.** *There is a deterministic non black-box PIT algorithm for $\Sigma\Pi\Sigma(k,d)$ circuits that runs in* $\mathrm{poly}(n, d^k)$ *time.*

8.1 New Black-Box PIT algorithm for depth-3 $\Sigma\Pi\Sigma(k,d)$ circuits

In this section we give a different black-box PIT algorithm for $\Sigma\Pi\Sigma(k,d)$ circuits based on the recent result of [SS09, KS09b] using our hardness of representation approach. For this we will use a result of [SS09, KS09b] that generalizes Theorem 8.5, giving an upper bound on the rank of the *linear factors* of a polynomial that is computed by a simple, minimal and nonzero $\Sigma\Pi\Sigma(k,d)$ circuit [8].

Definition 8.8. *Let $P(\bar{x}) = h_1(\bar{x}) \cdot h_2(\bar{x}) \cdot \ldots \cdot h_t(\bar{x})$ be a nonzero polynomial and its irreducible factors, respectively. We denote by* $\mathrm{Lin}(P)$ *the set of (non-constant) linear factors of P. Formally,* $\mathrm{Lin}(P) \triangleq \{h_i \mid h_i \text{ is a linear factor of } P\}$*.*

Lemma 8.9 ([SS09, KS09b])**.** *Let $P(x_1, \ldots, x_n)$ be a polynomial computed by a simple, minimal and nonzero $\Sigma\Pi\Sigma(k,d)$ circuit then* $\mathrm{rank}(\mathrm{Lin}(P)) \le R(k,d)$*.*

This lemma allows us to establish a hardness of representation theorem for $\Sigma\Pi\Sigma(k,d)$ circuits. To ease the notations in the proof, it is more convenient to consider[9] nonzero-$\bar{0}$-justified linear functions and linear products.

Definition 8.10. *Given an assignment $\bar{a} \in \mathbb{F}^n$ and a polynomial P we say that $\bar{a}$ is a* nonzero-justifying assignment *of P if $\bar{a}$ is a justifying assignment of P and in addition $P(\bar{a}) \neq 0$. We say that P is* nonzero-$\bar{a}$-justified *if $\bar{a}$ is a nonzero-justifying assignment of P.*

[8]This result appears only in [SS09]. However, the proof of [KS09b] can be easily extended to this case as well.

[9]It is possible to get similar algorithms using the usual $\bar{0}$-justified polynomials, instead of the nonzero ones, however this requires additional technical work that does not contribute to the understanding of the model.

Observation 8.11. *Let $F(\bar{x}) = \prod_j L_j(\bar{x})$ be a (preprocessed) linear product. Then F is nonzero-$\bar{0}$-justified iff $F(\bar{0}) \neq 0$. In addition, F is nonzero-$\bar{0}$-justified iff $L_j(\bar{0}) \neq 0$ for each j.*

We now give an efficient PIT algorithm using techniques from Section 7.

Lemma 8.12. *The function $G_1(y_1, z_1)$ (recall Definitions 3.1, 3.5) is a generator for preprocessed linear products.*

Proof. From Observation 3.2 it is sufficient to show that the claim holds for a single non-constant preprocessed linear function $L(\bar{x}) = \sum_{i=1}^{n} T_i(x_i)$. From the definition, there exists j such that $T_j(x_j)$ is a non-constant polynomial. As $L(G_1(\alpha_j, z_1)) = \sum_{i \neq j} T_i(0) + T_j(z_1)$ (see Observation 3.6) we get that $L(G_1(y_1, z_1))$ is a non-constant polynomial. □

The following corollary is obtained in a similar fashion to Corollary 4.4 (by slightly changing the proof of Lemma 4.3).

Corollary 8.13. *The image of G_1, $\mathrm{Im}(G_1)$, contains a common nonzero-justifying assignment for any set of preprocessed linear products.*

Next, we prove a hardness of representation result for depth-3 circuits and give an analog of Theorem 7.4.

Theorem 8.14. *For every $g(\bar{x}) \not\equiv 0$ the polynomial $g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ cannot be represented as sum of k nonzero-$\bar{0}$-justified linear products of (a total) degree d when $n > R(k,d)$.*

Proof. Let $C = \sum_{m=1}^{k} F_m$ compute $g(x) \cdot \mathcal{P}_n$. Assume, w.l.o.g., that C is minimal. As all the linear functions in C are nonzero-$\bar{0}$-justified we have that $x_i \notin \gcd(C)$, for any $i \in [n]$. Therefore, $\gcd(C, \mathcal{P}_n) = 1$. Consequently, if we consider the simplification of C we get $C' \stackrel{\Delta}{=} \mathrm{sim}(C) = g'(x) \cdot \mathcal{P}_n(\bar{x})$, where $g' = g/\gcd(C) \not\equiv 0$. That is, $g' \cdot \mathcal{P}_n$ is computed by a simple, minimal, nonzero circuit C' (C' remains minimal after the simplification). Hence, x_i is a linear factor of C', for every $i \in [n]$. Lemma 8.9 implies that $n \leq \mathrm{rank}(\mathrm{Lin}(g' \cdot \mathcal{P}_n)) \leq R(k,d)$, as required. □

Theorem 8.15. *Let $C(\bar{x}) = \sum_{m=1}^{k} F_m(\bar{x})$, where each $F_m(\bar{x})$ is a nonzero-$\bar{0}$-justified linear product over $\mathbb{F}$, of total degree at most d. Let $W \subseteq \mathbb{F}$ be of size $d+1$, such that $0 \in W$. Then $C \equiv 0$ if and only if $C|_{\mathcal{A}^n_{R(k,d)}(W)} \equiv 0$.*

Finally, in a similar fashion to the proof of Theorem 7.7, (using Corollary 8.13 instead of Corollary 4.4) we obtain the following theorem that implies Theorem 6. (Recall Theorem 8.5).

Theorem 8.16. *Let C be a $\Sigma\Pi\Sigma(k,d)$ circuit over $\mathbb{F}$ then $C(G_{R(k,d)+1}) \not\equiv 0$. In addition, Lemma 3.4 gives a hitting set for such circuits of size $|\mathcal{H}| = (nd)^{\mathcal{O}(R(k,d))} = (nd)^{\mathcal{O}(k^3 \log d)}$ when $\mathbb{F}$ is finite and of size $|\mathcal{H}| = (nd)^{\mathcal{O}(R(k,d))} = (nd)^{k^{\mathcal{O}(k)}}$ when $\mathbb{F} = \mathbb{R}$ or $\mathbb{Q}$.*

References

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

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.

[Alo99] N. Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8:7–29, 1999.

[AM07] V. Arvind and P. Mukhopadhyay. The monomial ideal membership problem and polynomial identity testing. In *Proceedings of the 18th ISAAC*, pages 800–811, 2007.

[AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual FOCS*, pages 67–75, 2008.

[BB98] D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *J. of Computer and System Sciences*, 56(1):112–124, 1998.

[BC98] N. H. Bshouty and R. Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM J. on Computing*, 27(2):401–413, 1998.

[BHH95a] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995.

[BHH95b] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates. *JCSS*, 50:521–542, 1995.

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

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

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

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

[HH91] T. R. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the 4th Annual COLT*, pages 326–336, 1991.

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th annual STOC*, pages 262–272, 1980.

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

[KLN$^+$93] M. Karchmer, N. Linial, I. Newman, M. E. Saks, and A. Wigderson. Combinatorial characterization of read-once formulae. *Discrete Mathematics*, 114(1-3):275–282, 1993.

[KMSV10] Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. Submitted, 2010.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual STOC*, pages 216–223, 2001.

[KS07] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

[KS08] Z. S. Karnin and A. Shpilka. Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual CCC*, pages 280–291, 2008.

[KS09a] Z. S. Karnin and A. Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual CCC*, pages 274–285, 2009.

[KS09b] N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, (32), 2009.

[LV03] R. J. Lipton and N. K. Vishnoi. Deterministic identity testing for multivariate polynomials. In *Proceedings of the 14th annual SODA*, pages 756–760, 2003.

[Raz04] R. Raz. Multilinear $NC_1 \neq$ Multilinear NC_2. In *Proceedings of the 45th Annual FOCS*, pages 344–351, 2004.

[Raz05] R. Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual STOC*, pages 11–20, 2005.

[RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RSY08] R. Raz, A. Shpilka, and A. Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. on Computing*, 38(4):1624–1647, 2008.

[RY08] R. Raz and A. Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. In *IEEE Conference on Computational Complexity*, pages 128–139, 2008.

[Sax08] N. Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP (1)*, pages 60–71, 2008.

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

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

[SS09] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. In *Proceedings of the 24th annual CCC*, 2009.

[SV08] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. In *Proceedings of the 40th Annual STOC*, pages 507–516, 2008.

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation*, pages 216–226. 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de