

Improved Polynomial Identity Testing for Read-Once Formulas

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 and reconstruction algorithms for preprocessed ROFs. In particular we obtain the following results.

1. We give an $(nd)^{\mathcal{O}(\log n)}$ black-box polynomial identity-testing algorithm for PROFs in n variables of individual degrees at most d (i.e. each $T_i(x_i)$ is of degree at most d). This improves and generalizes the previous $n^{\mathcal{O}(\sqrt{n})}$ algorithm of [SV08] for ROFs.
2. Given k PROFs in n variables of individual degrees at most d we give a deterministic (non black-box) algorithm that checks whether they sum to zero or not. The running time of the algorithm is $n^{\mathcal{O}(k)}$. This result improves and extends the previous results of [SV08].
3. Combining the two results above we obtain an $(nd)^{\mathcal{O}(k+\log n)}$ time deterministic algorithm for checking whether a black box holding the sum of k n -variate PROFs computes the zero polynomial. In other words, we provide a hitting set of size $(nd)^{\mathcal{O}(k+\log n)}$ for the sum of k PROFs. This result greatly improves and extends [SV08] where an $n^{\mathcal{O}(k^2+\sqrt{n})}$ algorithm was given for identity testing the sum of k ROFs.
4. We give an $(nd)^{\mathcal{O}(D+k)}$ time deterministic black-box identity testing algorithm for the sum of k PROFs of depth D and individual degrees at most d , which improves the $n^{\mathcal{O}(D+k^2)}$ algorithm of [SV08] for depth D ROFs.

As a corollary of the results above we obtain an $n^{\mathcal{O}(k)}$ time deterministic identity testing algorithm for multilinear depth-3 $\Sigma\Pi\Sigma(k)$ circuits. This matches the *non black-box* algorithm of [KS07] for the multilinear case. In fact the same result also holds for preprocessed-multilinear- $\Sigma\Pi\Sigma(k)$ circuits which are depth-4 circuits of a restricted form.

In addition to the above results we obtain a deterministic reconstruction algorithms for preprocessed read-once formulas. The running time of the algorithm is $(nd)^{\mathcal{O}(\log n)}$ for general preprocessed read-once formulas, of individual degrees at most d , and $(nd)^{\mathcal{O}(D)}$ for depth D preprocessed read-once formulas of individual degrees at most d .

Most of our results are obtained using *hardness of representation* approach which was first used in [SV08]. This technique can be thought of as a very explicit way of transforming (mild) hardness of a very structured polynomial to an identity testing algorithm.

*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).

AMS Subject Classifications

68Q25 [Theory of Computing]: Analysis of Algorithms and Problem Complexity

Keywords

Read-Once Formulas, Identity Testing, Reconstruction, Arithmetic Circuits, Bounded Depth Circuits

Contents

1	Introduction	3
1.1	Our results and techniques	3
1.2	Comparison to previous works	5
1.3	Organization	5
2	Preliminaries	6
2.1	Partial Derivatives	8
2.2	Some useful facts about polynomials	9
2.3	Polynomials and Circuit Classes	10
3	Preprocessed-Read-Once Formulas	10
3.1	Read-Once Formulas and Read-Once Polynomials	10
3.2	Preprocessed Read-Once Polynomials	13
4	From PIT to Justifying Assignments	15
4.1	From a Generator to a Justifying Set	17
4.2	Constructing a Generator from a Hitting set	17
5	Black-Box PIT for Preprocessed Read-Once Polynomials	18
5.1	Small Depth PROPs	20
6	PIT for Sum of Preprocessed Read-Once Formulas	23
6.1	Hardness of Representation theorem	23
6.2	Vanishing Theorem	27
6.3	The Identity Testing Algorithm - Proof of Theorem 2	27
6.4	Black-Box PIT for Sum of PROPs	28
7	Depth-3 Arithmetic Circuits	29
7.1	New Black-Box PIT algorithm for depth-3 $\Sigma\Pi\Sigma(k, d)$ circuits	31
8	Reconstruction of a PROF	33
8.1	Reconstruction of a ROF	33
8.2	Identifying the Preprocessing	33
8.3	Extending the algorithm for Reconstruction of a PROF	34
9	Preprocessed Read-Once Testing	34
9.1	Generic Scheme	34
9.2	Read-Once Verification	35
9.3	Proof of Theorem 8	38

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. Schwartz and Zippel [Zip79, Sch80] gave a black-box randomized algorithm for the problem, that was later improved in several cases [CK97, LV98, AB03]. 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 [KI04, Agr05, DSY08, 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 (that computes sparse polynomials) [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, SS08] 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 a very restricted class of depth-4 circuits [AM07, Sax08] (and even those algorithms are non black-box).

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] on lower bounds for multilinear circuits and formulas suggest that giving efficient deterministic PIT algorithms for multilinear formulas may be possible. Unfortunately, except for the models of multilinear depth-2 and multilinear $\Sigma\Pi\Sigma(k)$ circuits no such algorithm is known. As a consequence the problem of PIT for read-once formulas, which can be thought of as the simplest form of multilinear formulas,¹ was considered [SV08]. There, sub-exponential time deterministic PIT algorithms for (sums of) read-once arithmetic formulas in the black-box and non black-box models were given.

1.1 Our results and techniques

In this work we improve and extend the results from [SV08] (obtained by the same authors as this paper). There are two aspects for this improvement. First, we give quasi-polynomial time identity testing algorithms, which greatly improve upon the previous sub-exponential time algorithms. Secondly, we consider the more general model of *preprocessed read-once formulas*. These are read-once formulas in which we substitute a univariate polynomial $T_i(x_i)$ for any variable x_i (see Definition 3.12). Using our new results we obtain new identity-testing algorithms for multilinear depth-3 circuits and preprocessed multilinear depth-3 circuits, which are a restricted class of depth-4 circuits. We now describe our results and outline some of the ideas behind the proofs: Our first result is a black-box identity testing algorithm for read-once formulas. In fact this improved algorithm also holds for the more general preprocessed model.

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

¹Indeed, a read-once formula is a restricted form of multilinear formulas in which a variable can label at most one leaf of the formula (see Definition 3.1)

This result improves the $n^{\mathcal{O}(\sqrt{n})}$ time algorithm of [SV08] that worked for ROFs. The main new idea is induction on n . It is not hard to show that if a PROF is not zero then there is a variable x_i such that $\partial F/\partial x_i$ is a non zero PROF depending on at most $n/2$ variables. Using this observation we design a generator $G : W^{\mathcal{O}(\log n)} \rightarrow \mathbb{F}^n$, where $W \subseteq \mathbb{F}$ is of size n^2d , such that $F \circ G \neq 0$.

Our next result is a non black-box identity testing algorithm for the sum of k PROFs. The running time is $(nd)^{\mathcal{O}(k)}$. This result is a generalization to the preprocessed case (as well as an improvement) of the corresponding result from [SV08], where an $n^{\mathcal{O}(k^2)}$ algorithm was obtained.

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. This result improves and generalizes the previous $n^{\mathcal{O}(k^2+\sqrt{n})}$ time algorithm of [SV08] for the sum of k ROFs. The extension from a PIT algorithm for a single ROF to an algorithm for the sum of k 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. This idea previously appeared in [SV08] and here we extend it to the preprocessed case as well.

Theorem 3. *Given black-box access to $F = F_1 + \dots + F_k$, where the F_i -s are preprocessed-read-once formulas in n variables, with individual degrees at most d , there is a deterministic algorithm that checks whether $F \equiv 0$. The running time of the algorithm is $(nd)^{\mathcal{O}(k+\log n)}$.*

Using our techniques together with the results of [SV08] we obtain improved PIT algorithms for the sum of small depth PROFs (see Definition 5.5).

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 of the above result 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). This improves the $n^{\mathcal{O}(k^2)}$ algorithm that is implied by the results of [SV08]. Moreover, our algorithm also holds for the more general case of preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuits (see Section 7 for the definition).

Theorem 5. *Let F 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 F that runs in time $(nd)^{\mathcal{O}(k)}$.*

We note that this result *does not* rely on bounds on the rank of zero $\Sigma\Pi\Sigma(k)$ circuits (see Section 7). In addition to the multilinear case, we obtain a new PIT algorithm for general $\Sigma\Pi\Sigma(k)$ circuits that has (roughly) the same running time as the algorithm obtained from the results of [KS08, SS08] (although our algorithm needs a smaller field size).

Theorem 6. *Let C be $\Sigma\Pi\Sigma(k, d)$ circuit. There is a deterministic black-box PIT algorithm for C that runs in time $(nd)^{\mathcal{O}(k^3 \log d)}$.*

We also consider the problem of reconstruction of preprocessed read-once formulas. Namely, algorithms that when given black-box access to a PROF reconstruct another PROF computing the same polynomial. The approach is the same as in [SV08], but using our improved Theorems 1 we obtain better results.

Theorem 7. *There is an $(nd)^{\mathcal{O}(\log n)}$ time deterministic reconstruction algorithm for n -variate PROFs of individual degrees at most d . Similarly, there is an $(nd)^{\mathcal{O}(D)}$ time deterministic reconstruction algorithm for n -variate depth- D PROFs of individual degrees at most d .*

Finally, we observe an extension of the read-once testing result of [SV08].²

Theorem 8 (Informal). *Let \mathcal{M} be a “nice” class of arithmetic circuits. Assume that there is a deterministic PIT algorithm for \mathcal{M} that runs in time $t(s)$, when given as input an n -variate circuit C of size s . Then, there is a deterministic algorithm that runs in time $\text{poly}(n, s, d, t(s))$ that when given access to an n -variate circuit from \mathcal{M} of size s and individual degrees at most d , solves the preprocessed read-once reconstruction problem for it. Namely, the algorithm decides whether the circuit computes a PROP, and if it does then it outputs a PROF for it.*

1.2 Comparison to previous works

Read-once arithmetic formulas were mostly studied in the context of computational learning theory. Various works considered the problem of reconstructing the unknown read-once formula using membership queries. A membership query to a ROF $f(\bar{x})$ is simply a query that asks for the value of $f(\bar{x})$ on a specific input. In [HH91] a deterministic learning algorithm for read-once arithmetic formulas that uses membership and *equivalence* queries was given. An equivalence query gives the oracle holding the unknown formula a certain hypothesis, $h(\bar{x})$, and the oracle answers “equal” if $f \equiv h$ or returns an input $\bar{\alpha}$ such that $f(\bar{\alpha}) \neq h(\bar{\alpha})$. In [BHH95] a different approach was taken. They considered *randomized* learning algorithms that only use membership queries. It is not difficult to see that using randomness one does not need to use equivalence queries any more. The learning algorithm of [BHH95] can reconstruct, with high probability, arithmetic read-once formulas that also use division gates (and not just $+$, \times gates). This result was later generalized by [BB98] who gave a randomized reconstruction algorithm for read-once formulas that use additions, multiplications, divisions and exponentiations.

In [SV08] a sub-exponential time (i.e. $n^{\mathcal{O}(\sqrt{n})}$ time) PIT algorithm for black-box read-once formulas was given. Using this algorithm together with the learning methods of [HH91, BHH95] a deterministic sub-exponential time reconstruction algorithm for arithmetic ROFs was obtained. In addition, PIT algorithms for sum of ROFs both in the black-box and in the non black-box models were obtained in that work.

In this work we improve the PIT algorithms for sum of ROFs. In the black-box case we give a deterministic algorithm that runs in $n^{\mathcal{O}(k+\log n)}$ time. In the non black-box case our algorithm runs in time $n^{\mathcal{O}(k)}$. Moreover, both algorithms also work for the general model of preprocessed ROFs.

We are also able to apply our methods to depth-3 circuits, also known as $\Sigma\Pi\Sigma(k)$ circuits. This model was extensively studied in recent years [DS06, KS07, AM07, SV08, KS08, SS08, Shp09] as it stands between the simpler depth-2 case and the depth-4 case that is as hard as the general case, for lower bounds and polynomial identity testing [AV08]. Prior to this work the best known black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d)$ circuits had running time $n^{\mathcal{O}(k^3 \log d)}$ for the general case [KS08, SS08] and $n^{\mathcal{O}(k^2)}$ in the multilinear case [SV08]. We improve the algorithm for the multilinear case and obtain an $n^{\mathcal{O}(k)}$ algorithm that also works in the preprocessed case.

1.3 Organization

The paper is organized as follows. In Sections 2 and 3 we give the basic definitions and notations. Then in Section 4 we show how to acquire a justifying assignment given a PIT algorithm. New

²We give an informal version here. A more formal version is given in Theorem 9.6.

black-box PIT algorithm for PROF is given in Section 5 - proving Theorem 1. We also give a PIT for bounded depth and PROF. In Section 6 we consider sums of PROPs. We prove Theorems 2, 3 and 4. Following (Section 7) we give PIT algorithm for Depth-3 circuits and special cases of Depth-4 circuits, proving Theorems 5 and 6. Finally, in Sections 8.1 and 9 we present a reconstruction algorithm for PROFs and show how to extend PIT algorithms to PROT algorithms (Theorems 7 and the formal version of Theorem 8).

2 Preliminaries

For a positive integer n we denote $[n] = \{1, \dots, n\}$. For a graph $G = (V, E)$ we denote with G^c the complement graph. That is $G^c = (V, E')$ such that for $i \neq j \in V$ we have that $(i, j) \in E$ if and only if $(i, j) \notin E'$. For a polynomial $P(x_1, \dots, x_n)$ a variable x_i and a field element α we denote with $P|_{x_i=\alpha}$ the polynomial resulting after substituting α to the variable x_i . The following definitions, taken from [SV08], definitions are for a polynomial $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ and an assignment $\bar{a} \in \mathbb{F}^n$. We say that P depends on x_i if there exist $\bar{a} \in \mathbb{F}^n$ and $b \in \mathbb{F}$ such that:

$$P(a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n) \neq P(a_1, a_2, \dots, a_{i-1}, b, a_{i+1}, \dots, a_n).$$

We denote $\text{var}(P) \triangleq \{x_i \mid P \text{ depends on } x_i\}$. Given a subset $I \subseteq [n]$ we say that P is defined on I if $\text{var}(P) \subseteq I$. Intuitively, P depends on x_i if x_i “appears” when P is listed as a sum of monomials. On the other hand, a constant function $P \equiv c$ is defined on every subset of $[n]$.

Definition 2.1. 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 $P|_{x_I=\bar{a}_I}$ is defined on $[n] \setminus I$.

Example 2.2. Let $P(x_1, x_2, x_3) = x_1^2 + 2x_2x_3 + 4x_1x_3^2$, $I = \{1, 3\}$ and $\bar{a} = (0, 4, 1)$. Then $\text{var}(P) = \{1, 2, 3\}$, $P|_{x_I=\bar{a}_I}(\bar{x}) = P(0, x_2, 1) = 2x_2$ and $\text{var}(P|_{x_I=\bar{a}_I}(\bar{x})) = \{2\}$.

Example 2.3. Let $P(x_1, x_2, x_3) = 2x_2x_3 + 1$, $I = \{2\}$. For $\bar{a} = (0, 0, 0)$ we get that $P|_{x_I=\bar{a}_I}(\bar{x}) = P(x_1, 0, x_3) = 1$. Note that $\text{var}(P) = \{2, 3\}$ but $\text{var}(P|_{x_I=\bar{a}_I}(\bar{x})) = \emptyset$.

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. To conclude we give the following trivial observation.

Observation 2.4. Let $J \subseteq I \subseteq \text{var}(P)$ be subsets of $\text{var}(P)$. Then for every assignment $\bar{a} \in \mathbb{F}^n$ it must be the case that $\text{var}(P|_{x_I=\bar{a}_I}) \subseteq \text{var}(P|_{x_J=\bar{a}_J}) \subseteq \text{var}(P) \setminus J$.

For the purposes of reconstruction and 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 [BHH95], but we repeat the definitions here to ease the reading of the paper (we also slightly change some of the definitions).

Definition 2.5 (Justifying assignment). Given an assignment $\bar{a} \in \mathbb{F}^n$ we say that \bar{a} is a justifying assignment of P if for each subset $I \subseteq \text{var}(P)$ we have that

$$\text{var}(P|_{x_I=\bar{a}_I}) = \text{var}(P) \setminus I \tag{1}$$

Though this definition is very intuitive, ensuring that a given assignment \bar{a} satisfies condition 1 for every $I \subseteq \text{var}(P)$ seems to be a computationally hard problem. The following proposition provides an alternative and more useful definition:

Proposition 2.6. *An assignment $\bar{a} \in \mathbb{F}^n$ is a justifying assignment of P if and only if condition 1 holds for every subset I of size $|\text{var}(P)| - 1$.*

Proof. Let $J \subseteq \text{var}(P)$. If $J = \text{var}(P)$ then, obviously, $\text{var}(P|_{x_J=\bar{a}_J}) = \emptyset = \text{var}(P) \setminus J$. Otherwise, let $x \in \text{var}(P) \setminus J$. Consider the set $I = \text{var}(P) \setminus \{x\}$. From the definition: $|I| = |\text{var}(P)| - 1$ and $J \subseteq I$, and thus condition 1 holds for I . Combining with Observation 2.4 we obtain $\forall x \in \text{var}(P) \setminus J$:

$$\{x\} = \text{var}(P) \setminus I = \text{var}(P|_{x_I=\bar{a}_I}) \subseteq \text{var}(P|_{x_J=\bar{a}_J}) \subseteq \text{var}(P) \setminus J$$

and consequently $\text{var}(P|_{x_J=\bar{a}_J}) = \text{var}(P) \setminus J$. The second direction of the claim is trivial. \square

The justification notion can be both expanded and weakened:

Definition 2.7. *Given an assignment $\bar{a} \in \mathbb{F}^n$ we say that \bar{a} is a non-zero-justifying assignment of P if \bar{a} is a justifying assignment of P and in addition $P(\bar{a}) \neq 0$. Given an assignment $\bar{a} \in \mathbb{F}^n$ we say that \bar{a} is a weakly-justifying assignment of P if condition 1 holds for $|I| = 1$.*

Clearly, non-zero-justification implies (regular) justification which in turn implies weak-justification, but not vice versa. Later on we will encounter several cases in which justification and weak-justification occur simultaneously. We can, as well, define justification as a property of polynomials.

Definition 2.8. *We say that a polynomial P is \bar{a} -justified if \bar{a} is justifying assignment of P . We define the terms non-zero- \bar{a} -justified and weakly- \bar{a} -justified in a similar manner.*

Due some technical reasons in later sections, we define the zero polynomial to be non-zero- $\bar{0}$ -justified. Note that we can convert any polynomial to a $\bar{0}$ -justified form by shifting.

Lemma 2.9. *Let $\bar{a} \in \mathbb{F}^n$ and let $f(\bar{x})$ be a (non-zero, weakly) \bar{a} -justified polynomial. Then*

$$f_{\bar{a}}(\bar{x}) \triangleq f(x_1 + a_1, x_2 + a_2, \dots, x_n + a_n)$$

is a (non-zero, weakly) $\bar{0}$ -justified polynomial, in addition, $f_{\bar{a}} \equiv 0$ if and only if $f \equiv 0$.

We shall make use of the following notations in some of our proofs and algorithms.

For $n \geq 1$ let $\mathcal{P}_n(\bar{x}) = \prod_{i=1}^n x_i$. When $n = 0$ we define $\mathcal{P}_0(\bar{x}) = 1$. The *Hamming weight* of a vector $\bar{a} \in \mathbb{F}^n$ is defined as: $w_H(\bar{a}) \triangleq |\{i \mid a_i \neq 0\}|$. That is, the number of non-zero coordinates.

Definition 2.10. *For a set $0 \in W \subseteq \mathbb{F}$ and $k \leq n$ we define $\mathcal{A}_k^n(W)$ to be the set of all vectors in W^n with Hamming weight at most k , that is vectors that have **at most** k non-zero coordinates.*

Formally: $\mathcal{A}_k^n(W) \triangleq \{\bar{a} \in W^n \mid w_H(\bar{a}) \leq k\}$.

An immediate conclusion from the definition is that

$$|\mathcal{A}_k^n(W)| = \sum_{j=0}^k \binom{n}{j} \cdot (|W| - 1)^j = (n \cdot (|W| - 1))^{\mathcal{O}(k)}.$$

2.1 Partial Derivatives

The concept of a *partial derivative* of a multivariate function and its properties (for example: P depends on x_i if and only if $\frac{\partial P}{\partial x_i} \neq 0$) 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 2.11. *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 $\mathbb{F} = \mathbb{C}$. The following lemma is easy to verify.

Lemma 2.12. *The following properties hold for a multilinear polynomial P*

- P depends on x_i if and only if $\frac{\partial P}{\partial x_i} \neq 0$
- $\frac{\partial P}{\partial x_i}$ does not depend on x_i (in particular $\frac{\partial^2 P}{\partial x_i^2} \equiv 0$).
- $\frac{\partial^2 P}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial P}{\partial x_j} \right) = \frac{\partial^2 P}{\partial x_j \partial x_i}$.
- $\forall i \neq j \quad \frac{\partial P}{\partial x_i} |_{x_j=a} = \frac{\partial}{\partial x_i} (P|_{x_j=a})$.
- $\bar{a} \in \mathbb{F}^n$ is a justifying assignment of P if and only if $\forall i \in \text{var}(P)$ it holds $\frac{\partial P}{\partial x_i}(\bar{a}) \neq 0$.

Since the above properties trivially hold, we will use them implicitly. The following lemma contains the equivalent derivation rules when we restrict ourselves to the multilinear polynomials domain. That is, in all the cases we must ensure that the results of the specified arithmetic operations are indeed, multilinear polynomials. As this is not the general case, it imposes various (implicit) variable-disjointness restrictions. For example - a quotient of two arbitrary multilinear polynomial is not necessarily a multilinear polynomial. Moreover, in some cases it might not be a polynomial at all.

Lemma 2.13. *Let P, G, Q be multilinear polynomials. Then the following derivation rules hold (with the appropriate implicit restrictions)*

1. **Sum Rule.** *Let $Q = P + G$ then trivially*

$$\frac{\partial Q}{\partial x_i} = \frac{\partial P}{\partial x_i} + \frac{\partial G}{\partial x_i}$$

2. **Product Rule.** *Let $Q = P \cdot G$. In this case 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. **Quotient Rule.** *Let $Q = \frac{P}{G}$. The equality*

$$G^2 \cdot \frac{\partial Q}{\partial x_i} = G \cdot \frac{\partial P}{\partial x_i} - P \cdot \frac{\partial G}{\partial x_i}$$

can be obtained by applying the Product rule on $P = Q \cdot G$.

4. **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 .

However, these basic 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, to handle general polynomial we need the following extension.

Definition 2.14. Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ 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 say that $0 \neq \alpha \in \mathbb{F}$ is a witness for x_i in P if $\frac{\partial P}{\partial_\alpha x_i} \neq 0$ or $x_i \notin \text{var}(P)$. 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 are immediate corollaries.

Corollary 2.15. 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 $\frac{\partial P}{\partial_{\alpha_i} x_i} \neq 0$

Corollary 2.16. Let $Q(y, \bar{z})$ be a multilinear polynomial such that $P(\bar{x}, \bar{z}) \equiv Q(G(\bar{x}), \bar{z})$. Then $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial G}{\partial_\alpha x_i}$ and in addition, $\frac{\partial Q}{\partial y}$ does not depend on y .

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

Lemma 2.17. 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 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 $\varphi_i(\bar{x}, w) \triangleq \frac{\partial P}{\partial_w x_i}$ (see Definition 2.14). 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 $\varphi_i(\bar{x}, \alpha) \neq 0$ or $x_i \notin \text{var}(P)$. If $x_i \notin \text{var}(P)$ then any $\alpha \in W$ is a witness. Otherwise, $\varphi_i(\bar{x}, w)$ is a non-zero polynomial and by Corollary 2.20 we conclude that W contains a witness for x_i . We can repeat the same reasoning for every $i \in [n]$. \square

Definition 2.18. For a non-empty subset $I \subseteq [n]$, $I = \{i_1, \dots, i_{|I|}\}$, we define the iterated partial derivative with respect to I in the following way:

$$\partial_I P \triangleq \frac{\partial^{|I|} P}{\partial x_{i_1} \partial x_{i_2} \partial x_{i_3} \cdots \partial x_{i_{|I|}}}$$

2.2 Some useful facts about polynomials

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

Lemma 2.19. Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ 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]. The following is an easy Corollary:

Corollary 2.20. *Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a non-zero polynomial with individual degrees bounded by d and let $|\mathbb{F}| > d$. Then there exists an assignment $\bar{a} \in \mathbb{F}^n$ such that $P|_{x_I=\bar{a}_I} \neq 0$, for every $I \subseteq [n]$.*

Lemma 2.21 (Gauss). *Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n, y]$ be a non-zero polynomial and $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$ such that $P|_{y=g(\bar{x})} \equiv 0$ then $y - g(\bar{x})$ is an irreducible factor of P in the ring $\mathbb{F}[x_1, x_2, \dots, x_n, y]$.*

2.3 Polynomials and Circuit Classes

Let \mathcal{M} be a circuit class. We shall associate with it the polynomials that can be computed by its circuits. We make the following notations that will be later used.

Definition 2.22. *We shall use the following notations.*

1. *We say that the circuit class \mathcal{M} contains the polynomial P if P can be computed by some circuit C from \mathcal{M} . We denote it by $P \in \mathcal{M}$.*
2. *We say that the circuit class \mathcal{M}_1 contains the circuit class \mathcal{M}_2 if it contains all its polynomials (e.g. $P \in \mathcal{M}_1 \implies P \in \mathcal{M}_2$). We denote it by $\mathcal{M}_1 \subseteq \mathcal{M}_2$.*
3. *For a circuit class \mathcal{M} we define the (discrete) Directed Partial Derivatives of \mathcal{M} as:*

$$\partial\mathcal{M} \triangleq \left\{ \frac{\partial P}{\partial_\alpha x_i} \mid P \in \mathcal{M}, i \in [n], \alpha \in \mathbb{F} \right\}.$$
4. *A circuit class \mathcal{M} is closed under partial derivatives if $\partial\mathcal{M} \subseteq \mathcal{M}$.*
5. *For a circuit class \mathcal{M} we define the linear closure of \mathcal{M} as:*

$$\mathcal{L}(\mathcal{M}) \triangleq \{ \alpha \cdot P + \beta \mid P \in \mathcal{M}, \alpha, \beta \in \mathbb{F} \}.$$
 We say that \mathcal{M} is linearly closed if $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{M}$.

3 Preprocessed-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 the generalized basic properties for it.

3.1 Read-Once Formulas and Read-Once Polynomials

We give some basic definitions related to read-once formulas and list some basic facts and properties of the model. Most of the definitions are from [HH91]. The listed properties and their proofs can be found in Section 3 of [SV08]. 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 3.1. *A read-once arithmetic formula (ROF for short) over a field \mathbb{F} in the variables $\bar{x} = (x_1, \dots, x_n)$ is a binary tree whose leaves are labeled with the input variables and whose internal nodes are labeled with the arithmetic operations $\{+, \times\}$ and with a pair of field elements³ $(\alpha, \beta) \in \mathbb{F}^2$. Each input variable can label at most one leaf. The computation is performed in the following way. A leaf labeled with the variable x_i and with (α, β) computes the polynomial $\alpha \cdot x_i + \beta$. If a*

³This is a slightly more general model than the usual definition of read-once formulas.

node v is labelled with the operation op and with (α, β) , and its children compute the polynomials f_{v_1} and f_{v_2} then the polynomial computed at v is

$$f_v = \alpha \cdot (f_{v_1} \text{ op } f_{v_2}) + \beta.$$

The skeleton of a ROF f is the tree obtained from f after removing all the labels (α, β) from the nodes (but keeping the operations and the input variables).

We say that a ROF (instance) f is non-degenerate if it depends on all the variables appearing in it.

As our ROF model does not allow constants as inputs (constants can only label the internal nodes of the formula), we represent the constant polynomials with a special gate **CONST** that has only one label, a . We denote by Cv_a the non-degenerate ROF computing the constant polynomial $P \equiv a$.

Definition 3.2. A polynomial $P(\bar{x})$ is a read-once polynomial (ROP for short) if it can be computed by a read-once formula. For a ROP P we define $\mathcal{ROF}(P) \triangleq \{f \mid f \text{ is a ROF computing } P\}$.

A special class of ROFs that will play an important role in our proofs is the class of multiplicative ROFs.

Definition 3.3. A ROF is called multiplicative ROF if it has no addition gates. A polynomial computed by a multiplicative ROF is called a multiplicative ROP.

We shall later see (Lemma 3.9) that this notion is well defined. Note, because we allow gates to apply linear functions on the results of their operations, the output of a multiplicative ROF can be more complicated than a monomial.

Example 3.4. The polynomial $(5x_1 \cdot x_2 + 1) \cdot ((-x_3 + 2) \cdot (2x_4 - 1) + 5)$ has a multiplicative ROF.

Listed below are some basic properties of ROPs.

Lemma 3.5 (ROP Representation Lemma). *Every ROP $P(\bar{x})$ such that $|\text{var}(P)| \geq 2$ can be presented in exactly one of the following forms:*

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

When P_1, P_2 are non-constant variable disjoint ROPs and c is a constant.

Lemma 3.6 (Lemma 3.4 of [SV08]). *Let $P(\bar{x})$ be a ROP and v a node in a ROF f computing P . We 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 are variable-disjoint ROPs.*

Lemma 3.7 (Lemmas 3.5 and 3.6 of [SV08]). *A partial derivative and a factor of a ROP is a ROP.*

Lemma 3.8 (Lemma 3.10 of [SV08]). *A partial derivative and a factor of a weakly- $\bar{0}$ -justified ROP is a weakly- $\bar{0}$ -justified ROP.*

Lemma 3.9 (Proposition 3.8 of [SV08]). *A ROP P is a multiplicative ROP iff $\forall x_i \neq x_j \in \text{var}(P)$ we have that $\frac{\partial^2 P}{\partial x_i \partial x_j} \neq 0$.*

The following is an extension of Lemma 3.9 of [SV08] that will play an important role later.

Lemma 3.10. *Let $P(x_1, x_2, \dots, x_n)$ be a multiplicative ROP with $|\text{var}(P)| \geq 2$. Then for every variable $x_i \in \text{var}(P)$ there exists another variable $x_j \in \text{var}(P)$ such that $\frac{\partial P}{\partial x_j} = (x_i - \alpha)h(\bar{x})$ for some $\alpha \in \mathbb{F}$ and ROP $h(\bar{x})$, when $\text{var}(h) = \text{var}(P) \setminus \{x_i, x_j\}$ (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 $h(\bar{x})$. 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 $f \in \mathcal{ROF}(P)$ be a multiplicative ROF. As $|\text{var}(f)| = |\text{var}(P)| \geq 2$, f has at least one gate. Let v be the unique entering gate⁴ of x_i . We denote by $p_v(\bar{x})$ the ROP that is computed by v . Assume w.l.o.g that $\text{var}(p_v) = \{x_1, x_2, \dots, x_{i-1}, x_i\}$. By Lemma 3.6 there exists some polynomial $Q(y, x_{i+1}, \dots, x_n)$ such that $Q(p_v(x_1, x_2, \dots, x_i), x_{i+1}, \dots, x_n) \equiv P(x_1, x_2, \dots, x_n)$. Since v is a multiplication gate (recall that f is a multiplicative ROF) and the entering gate of x_i we get, in a similar manner to Lemma 3.5, that p_v can be written as $p_v(\bar{x}) = (x_i - \alpha)H(\bar{x}) + c$ for some polynomial $H(\bar{x})$ such that $\text{var}(H) \neq \emptyset$ and $x_i \notin \text{var}(H)$. By the chain rule, for every $x_j \in \text{var}(H)$ it holds that :

$$\frac{\partial P}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot \frac{\partial p_v}{\partial x_j} = \frac{\partial Q}{\partial y} \cdot (x_i - \alpha) \cdot \frac{\partial H}{\partial x_j} = (x_i - \alpha) \cdot h(\bar{x}), \quad (2)$$

where $h(\bar{x}) = \frac{\partial Q}{\partial y} \cdot \frac{\partial H}{\partial x_j}$. Now, if P is weakly- $\bar{0}$ -justified then by Lemma 3.8 we get that $\frac{\partial P}{\partial x_j}$ is a weakly- $\bar{0}$ -justified ROP as well. By Lemma 3.8 $h(\bar{x})$ is a weakly- $\bar{0}$ -justified ROP and $\alpha \neq 0$.

Now, assume that for some $\beta \neq \alpha \in \mathbb{F}$ the polynomial $P|_{x_i=\beta}$ is not weakly- $\bar{0}$ -justified. We will show that the value of β is uniquely defined. Let $x_\ell, x_k \neq x_i \in \text{var}(P)$ be such that $\frac{\partial P}{\partial x_\ell}|_{x_k=0} \neq 0$ but $\frac{\partial P}{\partial x_\ell}|_{x_k=0, x_i=\beta} \equiv 0$. In other words, the substitution $x_i = \beta$ affects the dependence of P on x_ℓ . We consider two cases.

Case $x_\ell \in \text{var}(H)$ (i.e. $1 \leq \ell \leq i-1$): By Equation 2 it holds that $\frac{\partial P}{\partial x_\ell}|_{x_k=0, x_i=\beta} = (\beta - \alpha) \cdot h|_{x_k=0}$ while $\frac{\partial P}{\partial x_\ell}|_{x_k=0} = (x_i - \alpha) \cdot h|_{x_k=0}$. As $\beta - \alpha \neq 0$ we conclude that the this case is impossible.

Case $x_\ell \in \text{var}(Q)$ (i.e. $i+1 \leq \ell \leq n$): In this case $\frac{\partial P}{\partial x_\ell} = \frac{\partial Q}{\partial x_\ell}|_{y=p_v(\bar{x})}$. Define $p'_v \triangleq p_v|_{x_k=0}$ and $p''_v \triangleq p'_v|_{x_i=\beta} = p_v|_{x_k=0, x_i=\beta}$. We have that

$$\frac{\partial Q}{\partial x_\ell}|_{x_k=0, y=p'_v} = \frac{\partial P}{\partial x_\ell}|_{x_k=0} \neq 0.$$

In particular, it implies that $\frac{\partial Q}{\partial x_\ell}|_{x_k=0} \neq 0$. On the other hand,

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

Therefore, from Lemma 2.21 we conclude that $y - p''_v$ is a factor of $\frac{\partial Q}{\partial x_\ell}|_{x_k=0}$. Since $\text{var}(p''_v) \cap \text{var}(Q) = \emptyset$ and Q is a multilinear polynomial it follows that there exists exactly one constant $\gamma \in \mathbb{F}$ such that $y - \gamma$ is a factor of $\frac{\partial Q}{\partial x_\ell}|_{x_k=0}$ and $p''_v \equiv \gamma$ (otherwise p''_v introduces variables that do not appear in Q). Recall that $p_v(\bar{x}) = (x_i - \alpha)H(\bar{x}) + c$. Thus, $H|_{x_k=0} = \frac{\gamma - c}{\beta - \alpha}$ ⁵ (recall that $\beta - \alpha \neq 0$). As H is a non-constant polynomial, it must be the case that $x_k \in \text{var}(H)$. Finally, notice that since P is a weakly- $\bar{0}$ -justified polynomial then it must be the case that $\text{var}(H) = \{x_k\}$ (otherwise, if there

⁴The entering gate of x_i is the neighbor of the leaf labeled by x_i .

⁵I.e. $\beta = \frac{\gamma - c}{H|_{x_k=0}} + \alpha$.

is $x_m \neq x_k \in \text{var}(H)$ 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 and H , which, in turn, are uniquely defined by P . \square

3.2 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 3.11. A preprocessing is a transformation $T(\bar{x}) : \mathbb{F}^n \rightarrow \mathbb{F}^n$ of the form $T(\bar{x}) \triangleq (T_1(x_1), T_2(x_2), \dots, T_n(x_n))$, 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 preprocessing 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 3.12. A preprocessed read-once arithmetic formula (*PROF* for short) over a field \mathbb{F} in the variables $\bar{x} = (x_1, \dots, x_n)$ is a binary tree whose leafs are labeled with non-constant univariate polynomials $T_1(x_1), T_2(x_2), \dots, T_n(x_n)$ (all together forming a preprocessing) and whose internal nodes are labeled 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 labeled 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 f_{v_1} and f_{v_2} then the polynomial computed at v is

$$f_v = \alpha \cdot (f_{v_1} \text{ op } f_{v_2}) + \beta.$$

Similar to Definition 3.1, the skeleton of a PROF f is the tree obtained from f after removing all the labels (α, β) from the nodes (but keeping the operations and the T_i 's).

Definition 3.13. 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 preprocessing. A Standard Decomposition is as above with the additional requirement that T is a standard processing.

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 3.14. Every PROP P admits a standard decomposition. In addition, if $(Q(\bar{z}), T(\bar{x}))$ and $(Q'(\bar{z}), T'(\bar{x}))$ are two decompositions of a PROP $P(\bar{x})$, then exist two vectors $\bar{\alpha}, \bar{\beta} \in \mathbb{F}^n$ such that

$$Q(\bar{z}) = Q'(\alpha_1 \cdot z_1 + \beta_1, \dots, \alpha_n \cdot z_n + \beta_n)$$

and for every $x_i \in \text{var}(P)$

$$T_i(\bar{x}) = \alpha \cdot T'_i(\bar{x}) + \beta$$

Moreover, if (Q, T) is a decomposition of a PROP P , then any pair (Q', T') that satisfies the above conditions is also a decomposition of P .

Proof. Let (Q, T) be some decomposition of P and let $c_i \neq 0$ 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:

$$Q'(\bar{z}) \triangleq Q(c_1 \cdot z_1 + T_1(0), c_2 \cdot z_2 + T_2(0), \dots, c_n \cdot z_n + T_n(0))$$

$$T'_i(x_i) \triangleq \frac{T_i(x_i) - T_i(0)}{c_i}, \quad T(\bar{x}) \triangleq (T'_1(x_1), T'_2(x_2), \dots, T'_n(x_n)).$$

It is easy to verify that (Q', T') is a standard decomposition of P . The proof of the second part is also easy and so we omit it. \square

It follows every PROP P admits a unique (up to the indices in $[n] \setminus \text{var}(P)$) standard decomposition. The following simple Corollary provides us a simple property of PROPs.

Corollary 3.15. *Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a PROP and let $\bar{a}, \bar{b} \in \mathbb{F}^n$ be two justifying assignments of P . Then for every $x_i \in \text{var}(P)$ there exist $\alpha \neq 0, \beta \in \mathbb{F}$ such that*

$$P(a_1, a_2, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n) = \alpha \cdot P(b_1, b_2, \dots, b_{i-1}, x_i, b_{i+1}, \dots, b_n) + \beta$$

Lemma 3.16. *Let P be a PROP, and let $(Q(\bar{z}), T(\bar{x}))$ be a standard decomposition for P . Then the following properties hold:*

- $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial Q}{\partial z_i} \Big|_{\bar{z}=T(\bar{x})} \cdot \frac{\partial T_i}{\partial_\alpha x_i} = \frac{\partial Q}{\partial z_i} \Big|_{\bar{z}=T(\bar{x})} \cdot T_i(\alpha)$.
In particular $\left(T_i(\alpha) \cdot \frac{\partial Q}{\partial z_i}(\bar{z}), T(\bar{x}) \right)$ is a standard decomposition of $\frac{\partial P}{\partial_\alpha x_i}$.
- P is (non-zero, weakly) $\bar{0}$ -justified if and only if Q is (non-zero, weakly) $\bar{0}$ -justified.
More generally: P is \bar{a} -justified $\iff Q$ is $T(\bar{a})$ -justified.
- α is a witness for x_i in P if and only if $T_i(\alpha) \neq 0$.

The following two lemmas are PROP versions of Lemmas 3.5, 3.7 and 3.8.

Lemma 3.17 (PROP Representation Lemma). *Every PROP $P(\bar{x})$ such that $|\text{var}(P)| \geq 2$ can be presented in exactly one of the following forms:*

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

When P_1, P_2 are non-constant, variable-disjoint PROPs and c is a constant.

Lemma 3.18. *A partial derivative of a PROP is a PROP.*

Since the above properties trivially hold, we will use them implicitly.

Corollary 3.19. *Let P be a weakly- $\bar{0}$ -justified PROP. For any $\alpha \in \mathbb{F}$ the directed partial derivative $\frac{\partial P}{\partial_\alpha x_i}$ is a weakly- $\bar{0}$ -justified PROP (as well).*

Proof. Let $Q(\bar{z}), T(\bar{x})$ be a standard decomposition of P . From the basic properties Q is a weakly- $\bar{0}$ -justified ROP and $T_i(\alpha) \cdot \frac{\partial Q}{\partial z_i}, T$ is a standard decomposition of $\frac{\partial P}{\partial_\alpha x_i}$. Lemma 3.8 implies that $T_i(\alpha) \cdot \frac{\partial Q}{\partial z_i}$ is a weakly- $\bar{0}$ -justified ROP and the corollary follows applying the same basic properties. \square

Remark 3.20. *In the contrary to the ROPs a PROP might have factors which are not PROPs. For example consider*

$$x^3 + x - y^3 - y = (x - y)(x^2 + xy + y^2 + 1)$$

It is easy to see that $x^2 + xy + y^2 + 1$ is an irreducible polynomial PROP. To see that it is not a PROP, apply Corollary 3.15 for $\bar{a} = (0, 0)$ and $\bar{b} = (1, 1)$, to get a contradiction.

4 From PIT to Justifying Assignments

We now show how to get a justifying assignment from a PIT algorithm. We shall consider a circuit class \mathcal{M} (e.g. depth-3 circuit, sum of ROFs, etc.) for which there exists another circuit class \mathcal{M}' such that $\partial\mathcal{M} \subseteq \mathcal{M}'$ and \mathcal{M}' has an efficient PIT algorithm⁶. Algorithm 1 returns a justifying assignment for P (if it fails to do so, it returns “ERROR”). The algorithm will invoke (as a routine) the supplied PIT algorithm for \mathcal{M}' .

Before giving the algorithm we explain the intuition behind it. Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a polynomial with individual degrees bounded by d . What we are after is a vector $(a_1, \dots, a_n) \in \mathbb{F}^n$ 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 . In other words, let \bar{a} be a witness for P we want that if $\frac{\partial P}{\partial \alpha_i x_i} \neq 0$ then also $\frac{\partial P}{\partial \alpha_i x_i}|_{x_j=a_j} \neq 0$ (see Corollary 2.15). 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 $j \neq i$, $g_i|_{x_j=a_j} \neq 0$. As the degree of x_i in each g_i is bounded by d there are at most d “bad values” that we can assign to a_j . Namely, for most values of a_j , we have that $g_i|_{x_j=a_j} \neq 0$. Hence, if we check enough values we should find a good a_j for every g_i . To verify that $g_i|_{x_j=a_j} \neq 0$ we use the PIT algorithm for \mathcal{M}' . 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 $j \neq i$ then g_i is still not zero. 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. Actually, we shall acquire 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 . Acquiring a common justifying assignment is a simple generalization of the above ideas. In fact, it can be regarded as a simultaneous acquisition of a (single) justifying assignment. The following corollary (from Definition 2.14) summarizes the described idea:

Corollary 4.1. *Let $\{P_m(\bar{x})\}_{m \in [k]}$ be n -variate polynomials over \mathbb{F} . For every $m \in [k]$ let $\bar{\alpha}^m$ be a witness for P_m . Then $\bar{a} \in \mathbb{F}^n$ is a common justifying assignment for $\{P_m\}_{m \in [k]}$ if for every m and $x_i \in \text{var}(P_m)$ it holds that $\frac{\partial P_m}{\partial \alpha_i^m x_i}(\bar{a}) \neq 0$. I.e., \bar{a} is a common non-zero of the $\frac{\partial P_m}{\partial \alpha_i^m x_i}$ ’s.*

The first step in the algorithm is finding witnesses for the k polynomials (separately for each polynomial). Then, using these witnesses we acquire a common justifying assignment, variable-by-variable. The proof of Lemma 2.17 provides us with a simple algorithm for finding witnesses. Note that during the execution of the (suggested) algorithm, the algorithm determines for each variable x_i whether the polynomial depends on x_i or not. Therefore, the algorithm can be used to compute var of the polynomial as well.

⁶Note, that in most cases an identity testing algorithm for \mathcal{M} can be slightly modified to yield an identity testing algorithm for $\partial\mathcal{M}$.

Lemma 4.2. 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{M} over \mathbb{F} . Let \mathcal{M}' be (another) circuit class such that $\partial\mathcal{M} \subseteq \mathcal{M}'$. 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{M}') computes $\text{var}(P)$ and outputs a witness $\bar{\alpha}$ for P in time $\mathcal{O}(nd \cdot t)$, where t is the running time of the PIT algorithm for \mathcal{M}' .

Algorithm 1 Acquire Common Justifying Assignment

Input:

- Circuits C_1, \dots, C_k from \mathcal{M} computing P_1, \dots, P_k with individual degrees bounded by d ,
- Subset $V \subseteq \mathbb{F}$ of size $|V| = knd$,
- Access to a PIT algorithm for \mathcal{M}' such that $\partial\mathcal{M} \subseteq \mathcal{M}'$.

Output:

Common Justifying assignment \bar{a} for P_1, P_2, \dots, P_k

- 1: Find $\{\bar{\alpha}^m\}_{m \in [k]}$ - witnesses for $\{P_m\}_{m \in [k]}$ {Separately for each P_m using Lemma 4.2 }
- 2: For $i \in [n], m \in [k]$ compute $g_i^m \triangleq \frac{\partial P_m}{\partial \alpha_i^m x_i}$

We describe an iteration $j \in [n]$ for finding the value of a_j in \bar{a} :

- 1: **for** $j = 1 \dots n$ **do**
 - 2: Find a value c such that for every $m \in [k]$ and $i \neq j \in [n]$: if $g_i^m \neq 0$ then $g_i^m|_{x_j=c} \neq 0$.
 - 3: For every $m \in [k]$ and $i \neq j \in [n]$ substitute c to the variable x_j of g_i^m .
 - 4: Set $a_j \leftarrow c$
 - 5: Continue to the next j .
-

Lemma 4.3. Let \mathbb{F} be a field with $|\mathbb{F}| > knd$. Let $\{P_m\}_{m \in [k]}$ be a set of polynomials with individual degrees bounded by d that are computed by circuits from \mathcal{M} . Let \mathcal{M}' be (another) circuit class such that $\partial\mathcal{M} \subseteq \mathcal{M}'$. Let $V \subseteq \mathbb{F}$ be of size $|V| \geq knd$. Then Algorithm 1 returns a common justifying assignment \bar{a} for $\{P_m\}_{m \in [k]}$ in time $\mathcal{O}(n^3 k^2 d \cdot t)$, where t is the running time of the PIT algorithm for \mathcal{M}' .

Proof. Lemma 4.2 implies that each $\bar{\alpha}^m$ is indeed a witness for P_m . We now 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 $c \in V$ that is good for every $g_i^m \neq 0$. That is, a c such that for every m and $i \neq j$ we have that if $g_i^m \neq 0$ then $g_i^m|_{x_j=c} \neq 0$. Note, that g_i^m 's are polynomials with individual degrees bounded by d hence from Lemma 2.21 each g_i^m has at most d roots of the form of $x_j = c$. Therefore, there are at most $kd(n-1)$ “bad” values of c (e.g. values for which there exist m and $i \neq j$ with $g_i^m \neq 0$ and $g_i^m|_{x_j=c} \equiv 0$). Consequently, V contains at least one “good” value of c . In addition, notice before that first iteration $g_i^m \neq 0$ iff $x_i \in \text{var}(P_m)$ and for each $j \in [n]$, if g_i^m is non-zero *before* the j -iteration it will remain non-zero *after* the j -th iteration. We conclude that after the n -th iteration is (successfully) completed we have that for every m and $x_i \in \text{var}(P_m)$ 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. 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 perform at most $k(n-1) \cdot |V| < n^2 k^2 d$ PIT checks. Therefore, we do at most $n^3 k^2 d + knd$ PIT checks during the execution. Hence the total running time of the algorithm is $\mathcal{O}(n^3 k^2 d \cdot t)$, where t is the cost of every PIT check for a circuit in \mathcal{M}' . \square

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{M}' is in the black-box setting. In this section we give a “Black-Box” version of the algorithm. More precisely, given a hitting set (a black-box PIT algorithm) \mathcal{H} for \mathcal{M}' (when $\partial\mathcal{M} \subseteq \mathcal{M}'$) we construct a (k, d) -justifying set for \mathcal{M} (assuming that $|\mathbb{F}|$ is large enough). That is a set of elements in \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{M} . The construction is performed by evaluating the generator on “many” points. Note that in this case we will not find the witnesses explicitly, but rather rely on their existence. We start by defining the notion of a generator, which is a certain form of a hitting set.

Definition 4.4. *A map $\mathcal{G} = (\mathcal{G}^1, \dots, \mathcal{G}^n) : \mathbb{F}^q \rightarrow \mathbb{F}^n$ is a generator for the circuit class \mathcal{M} if for every non-zero n -variate polynomial P computed by \mathcal{M} it holds that $P(\mathcal{G}) \neq 0$.*

Note that, actually, it is sufficient to show that $P(\mathcal{G}) \neq 0$ only for non-constant polynomials computed by \mathcal{M} . Furthermore, for any “reasonable” \mathcal{M} this is equivalent to showing that $P(\mathcal{G})$ is a non-constant polynomial. All our PIT algorithms are in fact generators for some (relatively) small q . In the next section we show how to construct a generator from a hitting set.

Lemma 4.5. *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{M} . Let \mathcal{M}' be (another) circuit class such that $\partial\mathcal{M} \subseteq \mathcal{M}'$. Let $\mathcal{G} : \mathbb{F}^q \rightarrow \mathbb{F}^n$ be a generator for \mathcal{M}' 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 $\mathcal{J}_{\mathcal{G}}^{k,d} \triangleq \mathcal{G}(V^q)$ 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{M}).*

Proof. Let $\{\bar{\alpha}^m\}_{m \in [k]}$ be the witnesses of $\{P_m(\bar{x})\}_{m \in [k]}$, respectively. For $i \in [n], m \in [k]$ we define $g_i^m \triangleq \frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\mathcal{G})$. From the definitions of the generator and \mathcal{M}' we get that if $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i} \neq 0$ then $g_i^m \neq 0$. Consider the polynomial $g \triangleq \prod_{i,m | g_i^m \neq 0} g_i^m$. It follows that g is a non-zero q -variate polynomial of degree at most $nk \cdot nd\delta$ in each variable. Lemma 2.19 implies that $g|_{V^q} \neq 0$. Equivalently, there exists $\bar{\alpha} \in V^q$ such that for each pair $i \in [n], m \in [k]$ if $g_i^m \neq 0$ then $g_i^m(\bar{\alpha}) \neq 0$. Now, let $i \in [n], m \in [k]$ be such that $x_i \in \text{var}(P_m)$. Then $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i} \neq 0$ and thus $g_i^m = \frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\mathcal{G}) \neq 0$. From the choice of $\bar{\alpha}$ we obtain that $\frac{\partial P_m}{\partial_{\alpha_i^m} x_i}(\mathcal{G}(\bar{\alpha})) = g_i^m(\bar{\alpha}) \neq 0$ and hence $\mathcal{G}(\bar{\alpha})$ is a justifying assignment for every P_m (Corollary 4.1). As for the size note that $|\mathcal{J}_{\mathcal{G}}^{k,d}| \leq (kn^2 d \delta + 1)^q$. \square

4.2 Constructing a Generator from a Hitting set

In this section we describe an efficient way to construct a generator from a hitting set \mathcal{H} . The construction is performed by interpolating the vectors in \mathcal{H} with multivariate polynomials (i.e. by passing a low degree curve through \mathcal{H}). We extend the ideas from Section 5.2.1 of [SV08].

Given a hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ for a circuit class \mathcal{M} we chose a subset of $V \subseteq \mathbb{F}$ of size n and set $q \triangleq \lceil \log_n |\mathcal{H}| \rceil$. It follows that

$$|\mathcal{H}| \leq n^q < n |\mathcal{H}|.$$

Then, we arbitrarily order the vectors in \mathcal{H} . Let $\mathcal{H} = \{\bar{a}^1, \bar{a}^2, \dots, \bar{a}^{|\mathcal{H}|}\}$ where each $\bar{a}^j = (a_1^j, a_2^j, \dots, a_n^j)$. Let $\varphi : V^q \rightarrow \{1, 2, \dots, |\mathcal{H}|\} \subseteq \mathbb{N}$ be a surjection.

Definition 4.6. For every $i \in [n]$ we define $L_i(\bar{y}) : \mathbb{F}^q \rightarrow \mathbb{F}$ to be the interpolation polynomial of the i -th coordinates of the vectors in \mathcal{H} . That is, $L_i(\bar{y})$ is a q -variate polynomial of degree at most $n - 1$ in every variable such that for every $\bar{b} \in V^q$ we have that

$$L_i(\bar{b}) = a_i^{\varphi(\bar{b})}.$$

Let $L(\bar{y}) : \mathbb{F}^q \rightarrow \mathbb{F}^n$ be defined as

$$L(\bar{y}) \triangleq (L_1(\bar{y}), L_2(\bar{y}), \dots, L_n(\bar{y})).$$

Note that L_i -s can be computed in time polynomial in $|\mathcal{H}|$ using simple interpolation. We now prove that $L(\bar{y}) : \mathbb{F}^q \rightarrow \mathbb{F}^n$, as defined above, is a generator for \mathcal{M} . We start by a trivial observation which follows immediately from the definition of L .

Observation 4.7. For each $\bar{a} \in \mathcal{H}$ there exists $\bar{b} \in V^q$ such that $L(\bar{b}) = \bar{a}$.

Lemma 4.8. $L(\bar{y}) : \mathbb{F}^q \rightarrow \mathbb{F}^n$ is a generator for \mathcal{M} with individual degrees bounded by $n - 1$.

Proof. Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a non-zero polynomial computed by a circuit in \mathcal{M} . From the definition of \mathcal{H} there exists $\bar{a} \in \mathcal{H}$ such that $P(\bar{a}) \neq 0$. From the observation we obtain that there exists $\bar{b} \in V^q$ such that $L(\bar{b}) = \bar{a}$. In particular, $P(L(\bar{b})) = P(\bar{a}) \neq 0$ and hence $P(L(\bar{y})) \not\equiv 0$. The claim regarding the degree follows from the definition of L_i -s. \square

5 Black-Box PIT for Preprocessed Read-Once Polynomials

In this section we give black-box PIT algorithm for PROPs (Theorem 3 for the case $k = 1$). The main idea is to convert a PROP P , that has many variables, each with a “small” degree, to 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 will construct a generator for PROPs (see Definition 4.4). We shall assume that $|\mathbb{F}| > n$ as we are allowed to use elements from an appropriate extension field. Throughout the entire section we fix a set $A = \{\alpha_1, \alpha_2, \dots, \alpha_n\} \subseteq \mathbb{F}$ of n distinct elements.

Definition 5.1. For every $i \in [n]$ let $u_i(w) : \mathbb{F} \rightarrow \mathbb{F}$ be the i -th Lagrange Interpolation polynomial for the set A . That is, each $u_i(w)$ is polynomial of degree $n - 1$ that satisfies:

$$u_i(\alpha_j) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

For every $i \in [n]$ and $k \geq 1$ we define $G_k^i(y_1, \dots, y_k, z_1, \dots, z_k) : \mathbb{F}^{2k} \rightarrow \mathbb{F}$ as

$$G_k^i(y_1, \dots, y_k, z_1, \dots, z_k) \triangleq \sum_{j=1}^k u_i(y_j) \cdot z_j$$

Finally, let $G_k(y_1, \dots, y_k, z_1, \dots, z_k) : \mathbb{F}^{2k} \rightarrow \mathbb{F}^n$ be defined as

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

The following observation is crucial:

Observation 5.2. Denote with $\bar{e}_i \in \{0, 1\}^n$ the vector that has 1 in the i -th coordinate and 0 elsewhere. We observe

$$G_{k+1} = G_k + \sum_{i=1}^n u_i(y_{k+1}) \cdot z_{k+1} \cdot \bar{e}_i.$$

Hence, for every $k \geq 1$ and $\alpha_m \in A$

$$G_{k+1}|_{y_{k+1}=\alpha_m} = G_k + z_{k+1} \cdot \bar{e}_m.$$

Now we can establish a low-degree generator.

Lemma 5.3. Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a non-zero PROP with $|\text{var}(P)| \leq 2^t$, for some $t \geq 0$ then $P(G_{t+1}) \neq 0$. Moreover, if P is a non-constant polynomial then so is $P(G_{t+1})$.

Proof. Let d be a bound on the individual degrees of the variables. For simplicity we assume that w.l.o.g \mathbb{F} is “large”, that is $|\mathbb{F}| > d$ - as we can always regard the given polynomials as polynomials over some larger extension field of \mathbb{F} .

We prove the claim by induction on $|\text{var}(P)|$. For $|\text{var}(P)| = 0, 1$ the claim is trivial. Now assume that $|\text{var}(P)| \geq 2$ (which implies $t \geq 1$). By Lemma 3.17 we get that P can be in a one of the two forms:

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 $|\text{var}(P_1)| \leq \frac{|\text{var}(P)|}{2}$ (in particular $|\text{var}(P_1)| < |\text{var}(P)|$). Let $x_m \in \text{var}(P_1)$. We define

$$P' \triangleq P(G_t^1, \dots, G_t^{m-1}, x_m, G_t^{m+1}, \dots, G_t^m)$$

that is, a polynomial resulting from substituting G_t^i into the variable x_i for every $i \neq m$.

Since $|\mathbb{F}| > d$ there exists a witness $\beta \in \mathbb{F}$ (Lemma 2.17) such that

$$P'' \triangleq \frac{\partial P}{\partial_\beta x_m} = \frac{\partial P_1}{\partial_\beta x_m} \neq 0.$$

Thus, we obtain that P'' is a non-zero PROP with $|\text{var}(P'')| < |\text{var}(P_1)| \leq \frac{|\text{var}(P)|}{2} \leq 2^{t-1}$. By the induction hypothesis

$$\frac{\partial P'}{\partial_\beta x_m} = P''(G_t) \neq 0$$

which implies that P' depends on x_m . By Observation 5.2

$$P(G_{t+1})|_{y_{t+1}=\alpha_m} = P'|_{x_m=G_t^m+z_{t+1}}.$$

As 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} . As a conclusion we get that $P(G_{t+1})$ is a non-constant polynomial and particularly, $P(G_{t+1}) \neq 0$.

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 $1 \leq |\text{var}(P_1)|, |\text{var}(P_2)| < |\text{var}(P)| \leq 2^t$. 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 obtain 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).

This complete proof of the Lemma. □

Note that the $P(G_k) \neq 0$ for the appropriate value of k regardless of the degree of P . We also note that the requirement that $|\mathbb{F}| > n$ is needed for the definition of G_k .

Theorem 5.4. *Let P be an n -variate PROP with individual degrees bounded by d that depends on at most t variables⁷. Denote $\ell = \lceil \log_2 t \rceil + 1$. Let $W \subseteq \mathbb{F}$ be a set of size $n^2 d$. Let $\mathcal{H} = G_\ell(W^{2^\ell}) \subseteq \mathbb{F}^n$ (that is, we take the image of W^{2^ℓ} under G_ℓ). Then $P \equiv 0$ if and only if $P|_{\mathcal{H}} \equiv 0$.*

Proof. If $P \equiv 0$ then the claim is trivial. Assume that $P \neq 0$. By Lemma 5.3 we get that $P(G_\ell) \neq 0$. From the definition, G_ℓ^i depends on 2ℓ variables $\{y_j, z_j\}_{j \in [\ell]}$. The degrees of each y_j and each z_j in G_ℓ are $n-1$ and 1 , respectively. Hence, the degrees of each y_j and each z_j in $P(G_\ell)$ are bounded by $d(n-1)n$ and dn , respectively. Lemma 2.19 implies that $P \neq 0$ if and only if $P|_{\mathcal{H}} \neq 0$. Finally, we note that $|\mathcal{H}| \leq (n^2 d)^\ell = (nd)^{\mathcal{O}(\log t)}$. □

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. When the underlying PROF is of small depth we obtain a faster algorithm.

5.1 Small Depth PROPs

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

Definition 5.5. *An alternating read-once formula (AROF) over a field \mathbb{F} in the variables $\bar{x} = (x_1, \dots, x_n)$ is a tree, of unbounded fan-in, whose leaves are labeled with the input variables and whose internal nodes are labeled with either $+$ or MUL . Each input variable can label at most one leaf. Every leaf and every $+$ gate are labeled with two field elements $(\alpha, \beta) \in \mathbb{F}^2$. In addition, any children of a MUL ($+$) gate is either a leaf or a $+$ (MUL) gate (this is the reason for the name alternating ROF). The computation is performed in the following way. A leaf labeled 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 labeled with $+$ and (α, β) and its children compute the polynomials f_{v_1}, \dots, f_{v_k} then the polynomial computed at v is*

$$f_v = \alpha \cdot \left(\sum_{i=1}^k f_{v_i} \right) + \beta.$$

⁷Clearly $t \leq n$ but we choose this more general statement.

If v is labeled with *MUL* then it computes a multiplicative ROP in its input variables. That is, if v is labeled with the multiplicative ROP g , and its children compute the polynomials f_{v_1}, \dots, f_{v_k} , then the output of v will be the polynomial

$$f_v = g(f_{v_1}, \dots, f_{v_k}).$$

The skeleton of an AROF f is defined, as before, as the tree of f without the labels (α, β) on the nodes (but with the operations and the input variables).

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 a pair F, T where F is a AROF and T is a preprocessing such that a leaf of F labeled with x_i computes the polynomial $T_i(x_i)$ and the computation at the other gates is performed as before. (in a similar manner to Definition 3.12).

Example 5.6. The polynomial computed in Example 3.4 has an AROF of depth 1 that contains a single *MUL* gate.

First, we define the depth of a PROP. The definition is well-defined. For more information see Section 3 of [SV08].

Definition 5.7. For a PROP $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ we define: $\text{depth}(P)$ to be the depth of the P-AROF computing it.

We now give the analog of Lemmas 3.17, 3.18 and for the case of P-AROFs.

Lemma 5.8. Every PROP $P(x)$ with $|\text{var}(P)| \geq 2$ of depth D can be presented in exactly one of the following forms:

1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \dots + P_k(\bar{x})$.
2. $P(\bar{x}) = f(P_1(\bar{x}), P_2(\bar{x}), \dots, P_k(\bar{x}))$.

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.

Proof. The proof is similar to the proof of Lemma 3.17. □

Lemma 5.9. 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 at most 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 5.8 that P can be in a one of the two forms:

1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \dots + P_k(\bar{x})$. 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 $D - 1$.
2. $P(\bar{x}) = f(P_1(\bar{x}), P_2(\bar{x}), \dots, P_k(\bar{x}))$, where f is a multiplicative ROP in the variables $\{y_1, y_2, \dots, y_k\}$. As previously, we assume w.l.o.g that $x_i \in \text{var}(P_1)$. By the chain rule we get that $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial f}{\partial y_1}(P_1, \dots, P_k) \cdot \frac{\partial P_1}{\partial_\alpha x_i}$. As f is a multiplicative ROP, we get that $\frac{\partial f}{\partial y_1}$ is a multiplicative ROP in the variables y_2, \dots, 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 $\frac{\partial P}{\partial_\alpha x_i} = \frac{\partial f}{\partial y_1}(P_1, \dots, P_k) \cdot \frac{\partial P_1}{\partial_\alpha x_i}$ is a PROP of depth at most D .

This concludes the proof of the lemma. \square

Having the above we can describe a generator for PROPs computed by small depth P-AROFs. The idea is to “reduce” the depth of the formula. This is done one level at a time. In P-AROF each pair of adjacent levels consists of + and MULT gates. To reduce a + gate, we consider a partial derivative w.r.t a well chosen variable in $\text{var}(P)$. To reduce a MULT gate, we use the following lemma. Note that in the proof of Lemma 5.3 we made an explicit usage of Lemma 5.10 for the case $k = 2$.

Lemma 5.10. *Let $Q(x_1, \dots, x_k) : \mathbb{F}^k \rightarrow \mathbb{F}$ be a non-constant multiplicative ROP and let $h_1(\bar{y}), h_2(\bar{y}), \dots, h_k(\bar{y})$ be non-constant polynomials. Then $Q(h_1, \dots, h_k)$ is a non-constant polynomial.*

Proof. The proof follows immediately by a simple induction on the structure of the multiplicative ROP 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. \square

Finally, we can state the depth-version of Lemma 5.3.

Lemma 5.11. *Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a non-constant PROP of depth $\leq D$ then $P(G_{D+1})$ is a non-constant polynomial (in particular $P(G_{D+1}) \neq 0$).*

Proof. Let d be a bound on the individual degrees of the variables in P . For simplicity we assume that (w.l.o.g) $|\mathbb{F}| > d$. We prove the claim by induction on $\text{depth}(P)$. For $\text{depth}(P) = 0$ we get that $|\text{var}(P)| \leq 1$ and the proof is trivial. Now assume that $\text{depth}(P) \geq 1$. This implies $|\text{var}(P)| \geq 2$. By Lemma 5.8 we get that P can be written in exactly one of the two forms:

Case $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x}) + \dots + P_k(\bar{x})$: where the polynomials $P_j(\bar{x})$ are non-constant variable-disjoint PROPs of depth at most $D - 1$. Let $x_m \in \text{var}(P_1)$. Let

$$P' \triangleq P(G_D^1, \dots, G_D^{m-1}, x_m, G_D^{m+1}, \dots, G_D^n)$$

be the polynomial resulting from substituting G_D^i for the variable x_i for every $i \neq m$. As $|\mathbb{F}| > d$ we get by Lemma 2.17 that there exists a witness $\beta \in \mathbb{F}$ such that $P'' \triangleq \frac{\partial P}{\partial_\beta x_m} = \frac{\partial P_1}{\partial_\beta x_m} \neq 0$. Hence, by Lemma 5.9 P'' is a non-zero PROP of $\text{depth}(P'') \leq D - 1$. By the induction hypothesis we see that $P''(G_D) \neq 0$. Therefore we get that $\frac{\partial P'}{\partial_\beta x_m} = P''(G_D) \neq 0$ which implies that P' depends on x_m . By Observation 5.2 we get that $P(G_{D+1})|_{y_{D+1} = \alpha_m} = P'|_{x_m = G_t^m + z_{D+1}}$. As z_{D+1} only appears in the m -th coordinate it follows that $P(G_{D+1})|_{y_{D+1} = \alpha_m}$ depends on z_{D+1} . As a conclusion we get that $P(G_{D+1})$ is a non-constant polynomial and in particular $P(G_{D+1}) \neq 0$.

Case $P(\bar{x}) = f(P_1(\bar{x}), P_2(\bar{x}), \dots, P_k(\bar{x}))$: 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_i we obtain that for every $i \in [k]$ $P_i(G_{D+1})$ is a non-constant polynomial. As $P(G_{D+1}) = f(P_1(G_{D+1}), P_2(G_{D+1}), \dots, P_k(G_{D+1}))$ it follows from Lemma 5.10 that $P(G_{D+1})$ is a non-constant polynomial. \square

The following Theorem is an analog of Theorem 5.4.

Theorem 5.12. *Let P be a n -variate PROP with individual degrees bounded by d and depth at most D . Denote $\ell = D + 1$. Let $W \subseteq \mathbb{F}$ be a set of size $n^2 d$. Let $\mathcal{H} = G_\ell(W^{2\ell}) \subseteq \mathbb{F}^n$ (the image of $W^{2\ell}$ under G_ℓ). Clearly, $|\mathcal{H}| = (nd)^{\mathcal{O}(D)}$. Then $P \equiv 0$ if and only if $P|_{\mathcal{H}} \equiv 0$.*

6 PIT for Sum of Preprocessed Read-Once Formulas

In this section we prove Theorems 2, 3 and 4. We are given k PROPs $\{F_m\}_{m \in [k]}$ and we have to find whether they sum to zero. In other words, let $F = \sum_{m=1}^k F_m$, then we have to check whether $F \equiv 0$. Our algorithm for the problem has two steps. First we find a common justifying assignment to F_1, \dots, 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 Lemma 2.9). In the second step we simply verify that F vanishes on a relatively small set of vectors, each of weight at most $3k$. Theorem 6.4 then guarantees that $F \equiv 0$. The main tool in the proof is Theorem 6.2 that shows that we cannot represent $\mathcal{P}_n = \prod_{i=1}^n x_i$ as a sum of less than $\frac{1}{3}n$ $\bar{0}$ -justified PROPs. We call this approach a *hardness of representation* approach as the proof is based on the fact that a simple polynomial cannot be represented (computed) by a sum of a “small” number of $\bar{0}$ -justified PROPs.

6.1 Hardness of Representation theorem

In this section we give a *hardness of representation* theorem, that shows that the polynomial \mathcal{P}_n cannot be represented as a sum of $k \leq \frac{n}{3}$ $\bar{0}$ -justified ROPs. Then, using this preliminary result, we prove a stronger hardness of representation theorem for PROPs. I.e., we show that every non-zero 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 that the near optimality of our bound.

Theorem 6.1. $\mathcal{P}_n(\bar{x})$ 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 $\mathbb{F}[x_1, x_2, \dots, x_n]$. We prove the claim by induction on k . For $k = 0, 1$ the claim follows from the definition of $\bar{0}$ -weak-justification. We now assume that $k \geq 2$ and that $n \geq 3k$. We shall assume for a contradiction that $\sum_{m=1}^k F_m = \mathcal{P}_n$. The idea of the proof is to eliminate a “large” number of ROPs at a cost of a “small” number of variables. More 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” polynomials such that the rest of the ROPs remain weakly- $\bar{0}$ -justified. This way we get that a representation of polynomial $\partial_J \mathcal{P}_n|_{x_n=\alpha} = \alpha \cdot \mathcal{P}_{\hat{n}}$ (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 get that $\sum_{m=1}^{k-1} \frac{\partial^2 F_m}{\partial x_n \partial x_{n-1}} = \mathcal{P}_{n-2}$. 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 3.8 we have 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 $\frac{\partial^2 F_m}{\partial x_i \partial x_j} \neq 0$. Thus, by Lemma 3.9 we get that the polynomials $\{F_m\}_{m \in [k]}$ are multiplicative ROPs. In addition, for every $m \in [k]$ we have that $\text{var}(F_m) = [n]$. In particular, $|\text{var}(F_m)| \geq 6$. Lemma 3.10 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 $\alpha \in A$ we define:

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

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

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 $\sum_{\alpha \in A} |E_\alpha| = k$. More specifically, the E_α ’s form a partition of $[k]$. Similarly, Lemma 3.10 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 $\sum_{\alpha \in A} |B_\alpha| \leq k$. Hence, there exists $\alpha_0 \in A$ such that $|B_{\alpha_0}| \leq |E_{\alpha_0}|$.

Now, 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 3.8 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 3.8 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 non-zero 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 3.8 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 $J = \{\hat{n} + 1, \hat{n} + 2, \dots, n - 2, n - 1\}$ for some \hat{n} . We get that $\sum_{m=1}^k F''_m = \partial_J \mathcal{P}_n|_{x_n=\alpha_0} = \alpha_0 \cdot \mathcal{P}_{\hat{n}}$. 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 6.1. \square

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

Theorem 6.2. *For every $g(\bar{x}) \neq 0$ 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 let $\{(Q_m(\bar{z}), T^m(\bar{x}))\}_{m \in [k]}$ be standard decompositions for them. We assume w.l.o.g that $|\mathbb{F}| > d$, as we can always regard the given polynomials as polynomials over some extension field of \mathbb{F} . We prove the claim by induction on k . For $k = 0, 1$ the claim, as before, follows from the definition of $\bar{0}$ -weak-justification. We now consider $k \geq 2$ and that $n \geq 3k$. Assume for a contradiction that $\sum_{m=1}^k F_m = g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ for some $g(\bar{x}) \neq 0$. We show that either all the preprocessing are equal, and then we can reduce the problem to the un-preprocessed case. Or, if this is not the case then we can decrease the number of the polynomials in the sum and then use the induction hypothesis to reach a contradiction. We now proceed with the proof. Let y, w be two new indeterminates. There are two cases to consider.

Case 1: For every $i \in [n]$ and $m \in [k]$ we have that $y \cdot g|_{x_i=y} \cdot T_i^m(w) \equiv w \cdot g|_{x_i=w} \cdot T_i^m(y)$. We will argue that in this case all the preprocessing are equal (i.e., $T^m(\bar{x}) \equiv T^\ell(\bar{x})$ for every $m, \ell \in [k]$). Indeed, as for every $\ell \in [k]$ we have that $y \cdot g|_{x_i=y} \cdot T_i^\ell(w) \equiv w \cdot g|_{x_i=w} \cdot T_i^\ell(y)$ and $g \neq 0$ we get that

$$\frac{T_i^m(y)}{T_i^\ell(y)} \equiv \frac{T_i^m(w)}{T_i^\ell(w)} = c_{\ell, m}$$

for some constant $c_{\ell, m} \in \mathbb{F}$. Indeed, the LHS is a polynomial in y and the RHS is a polynomial in w . As $\{T^m(\bar{x})\}_{m \in [k]}$ are standard preprocessing it must be the case that $T_i^m(x_i) = T_i^\ell(x_i)$. Consequently, there exists a set of univariate polynomials $\{S_i(x_i)\}_{i \in [n]}$ such that $T_i^m(x_i) \equiv S_i(x_i)$ for every $i \in [n]$ and $m \in [k]$. Therefore, we can define the preprocessing $S(\bar{x}) \triangleq (S_1(x_1), S_2(x_2), \dots, S_n(x_n))$ which satisfies $T^m(\bar{x}) \equiv S(\bar{x})$ for every $m \in [k]$. Furthermore, observe that for every $i \in [n]$

$$\frac{y \cdot g|_{x_i=y}}{S_i(y)} \equiv \frac{w \cdot g|_{x_i=w}}{S_i(w)} = g'_i(\bar{x})$$

for some g'_i such that $x_i \notin \text{var}(g'_i)$ (that is, g'_i does not depend on x_i). It follows that $\frac{x_i \cdot g(\bar{x})}{S_i(x_i)} = g'_i(\bar{x})$ and therefore $x_i \cdot g(\bar{x}) = g'_i(\bar{x}) \cdot S_i(x_i)$. We conclude that for every $i \in [n]$ the polynomial $S_i(x_i)$ is a factor of $g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ and since $x_i \notin \text{var}(g'_i)$ we obtain that:

$$g(\bar{x}) \cdot \mathcal{P}_n(\bar{x}) = c' \cdot S_n(x_n) \cdot S_{n-1}(x_{n-1}) \cdot \dots \cdot S_1(x_1) = c' \cdot \mathcal{P}_n(S(\bar{x}))$$

for some constant $c' \in \mathbb{F}$. Let $Q(\bar{z}) = \sum_{m=1}^k Q_m(\bar{z})$ be a sum of weakly- $\bar{0}$ -justified ROPs. Then:

$$Q(S(\bar{x})) = \sum_{m=1}^k Q_m(S(\bar{x})) = \sum_{m=1}^k F_m = g(\bar{x}) \cdot \mathcal{P}_n(\bar{x}) = c' \cdot \mathcal{P}_n(S(\bar{x}))$$

and consequently: $Q(\bar{z}) = c' \cdot \mathcal{P}_n(\bar{z})$ (see discussion after Definition 3.11). Thus, we have a representation of $c' \cdot \mathcal{P}_n$ as a sum of $k \leq \frac{n}{3}$ ROPs. By Theorem 6.1 we get that $c' = 0$ and hence

$$\sum_{m=1}^k F_m = c' \cdot \mathcal{P}_n(S(\bar{x})) \equiv 0.$$

Case 2: There exist $i \in [n]$ and $m \in [k]$ such that $y \cdot g|_{x_i=y} \cdot T_i^m(w) \not\equiv w \cdot g|_{x_i=w} \cdot T_i^m(y)$. Assume w.l.o.g. that $i = n$ and $m = k$. Since \mathbb{F} is “large enough” Corollary 2.20 implies that there exist $\alpha, \beta \in \mathbb{F}$ such that

$$\alpha \cdot g|_{x_n=\alpha} \cdot T_n^k(\beta) \not\equiv \beta \cdot g|_{x_n=\beta} \cdot T_n^k(\alpha). \quad (3)$$

From the observation that $\frac{\partial(g \cdot \mathcal{P}_n)}{\partial_\alpha x_n} = \alpha \cdot g|_{x_n=\alpha} \cdot \mathcal{P}_{n-1}$ it follows that

$$\sum_{m=1}^k \frac{\partial Q_m}{\partial z_n} \Big|_{\bar{z}=T^m(\bar{x})} \cdot T_n^m(\alpha) = \sum_{m=1}^k \frac{\partial F_m}{\partial_\alpha x_n} = \frac{\partial(g \cdot \mathcal{P}_n)}{\partial_\alpha x_n} = \alpha \cdot g|_{x_n=\alpha} \cdot \mathcal{P}_{n-1} \quad (4)$$

and

$$\sum_{m=1}^k \frac{\partial Q_m}{\partial z_n} \Big|_{\bar{z}=T^m(\bar{x})} \cdot T_n^m(\beta) = \sum_{m=1}^k \frac{\partial F_m}{\partial_\beta x_n} = \frac{\partial(g \cdot \mathcal{P}_n)}{\partial_\beta x_n} = \beta \cdot g|_{x_n=\beta} \cdot \mathcal{P}_{n-1}. \quad (5)$$

We now apply Gaussian Elimination to “get rid” of the last summand. For that purpose we multiply Equation 4 by $T_n^k(\beta)$ and subtract Equation 5 multiplied by $T_n^k(\alpha)$. We get that

$$\begin{aligned} & \sum_{m=1}^k \frac{\partial Q_m}{\partial z_n} \Big|_{\bar{z}=T^m(\bar{x})} \cdot \left(T_n^m(\alpha) \cdot T_n^k(\beta) - T_n^m(\beta) \cdot T_n^k(\alpha) \right) = \\ & \mathcal{P}_{n-1} \cdot \left(\alpha \cdot g|_{x_n=\alpha} \cdot T_n^k(\beta) - \beta \cdot g|_{x_n=\beta} \cdot T_n^k(\alpha) \right). \end{aligned}$$

Let

$$\begin{aligned} F'_m & \triangleq \frac{\partial Q_m}{\partial z_n} \Big|_{\bar{z}=T^m(\bar{x})} \cdot \left(T_n^m(\alpha) \cdot T_n^k(\beta) - T_n^m(\beta) \cdot T_n^k(\alpha) \right) \quad \text{and} \\ g' & \triangleq \alpha \cdot g|_{x_n=\alpha} \cdot T_n^k(\beta) - \beta \cdot g|_{x_n=\beta} \cdot T_n^k(\alpha). \end{aligned}$$

As the $\frac{\partial Q_m}{\partial z_n}$ -s are weakly- $\bar{0}$ -justified ROPs (Lemma 3.8) and $\{T^m(\bar{x})\}_{m \in [k]}$ are standard preprocessing, the F'_m are weakly- $\bar{0}$ -justified PROPs and $g' \not\equiv 0$ (this follows from Equation 3). We thus get that $\sum_{m=1}^{k-1} F'_m = g' \cdot \mathcal{P}_{n-1}$. Hence, we obtain a representation of $g' \cdot \mathcal{P}_{n-1}$ as a sum of $0 < \hat{k} \leq k-1$ weakly- $\bar{0}$ -justified PROPs such that $0 < 3\hat{k} \leq 3(k-1) = 3k-3 < n-1$ which contradicts the induction hypothesis.

We therefore proved that $g \cdot \mathcal{P}_n$ cannot be represented as a sum of less than $\frac{n}{3}$ weakly- $\bar{0}$ -justified PROPs which concludes the proof of the theorem. \square

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 6.3. *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 $A = \{\alpha_1, \alpha_2, \dots, \alpha_n\} \subseteq \mathbb{F}$ be a subset of n distinct elements, such that $\alpha_i \neq 0$. For every $i \in [n]$ let $u_i(w)$ be the i -th Lagrange Interpolation polynomial over A (for more details see Definition 5.1). Consider $\varphi(\bar{x}, t) = (x_1 + t)(x_2 + t) \cdots (x_n + t) - t^n$. Since the degree of t in $\varphi(\bar{x}, t)$ is $n - 1$ we can write:

$$\varphi(\bar{x}, t) = \sum_{i=1}^n u_i(t) \cdot \varphi(\bar{x}, \alpha_i)$$

(that is, interpolate $\varphi(\bar{x}, t)$ as an $n - 1$ degree polynomial in t). Consequently, we obtain

$$\mathcal{P}_n(\bar{x}) = \varphi(\bar{x}, 0) = \sum_{i=1}^n u_i(0) \cdot \varphi(\bar{x}, \alpha_i) = \sum_{i=1}^n F_i(\bar{x}) + c$$

where $F_i(\bar{x}) \triangleq u_i(0) \cdot \varphi(\bar{x}, \alpha_i)$ are $\bar{0}$ -justified ROPs and $c = -\sum_{i=1}^n u_i(0) \cdot \alpha_i^n$. Clearly we can add c to F_n , and so the proof is completed. \square

6.2 Vanishing Theorem

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

Theorem 6.4. *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 ⁸ $d + 1$ 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 $F|_{\mathcal{A}_{3k}^n(W)} \equiv 0$ (recall Definition 2.10).*

Proof. The first direction is trivial. For the second direction we apply induction on n . Our base case is when $n \leq 3k$. In this case F is a polynomial in $n \leq 3k$ variables of degree at most d in each variable and therefore by Lemma 2.19 we get that $F|_{\mathcal{A}_{3k}^n(W)} \equiv 0$ implies that $F \equiv 0$. We now assume that $n > 3k \geq 4$. 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 $F' \triangleq F|_{x_\ell=0}$ and $\{F'_m \triangleq F_m|_{x_\ell=0}\}_{m \in [k]}$ as well. Indeed, the $\{F'_m\}_{m \in [k]}$ are $\bar{0}$ -justified PROPs with individual degrees bounded by d . Moreover, $F'|_{\mathcal{A}_{3k}^{n-1}(W)} = F'|_{\mathcal{A}_{3k}^n(W)} \equiv 0$. 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 2.21). 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 $g(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$. 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 6.2 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.

Corollary 6.5. *In the settings of Theorem 6.4 let \bar{a} be a common justifying assignment for the PROPs F_1, F_2, \dots, F_k . Then $F \equiv 0$ if and only if $F_{\bar{a}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$.*

6.3 The Identity Testing Algorithm - Proof of Theorem 2

We now give an identity testing algorithm for the sum k of preprocessed read-once formulas with individual degrees bounded by d . For the algorithm we assume that $|\mathbb{F}| > d$ (recall that we are allowed to make queries from an extension field).

⁸We implicitly assume that $|\mathbb{F}| > d$.

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

Input: PROFs F_1, \dots, F_k and $F = F_1 + \dots + F_k$ Output: “true” if $F \equiv 0$ and “false” otherwise.

- 1: Let $W \subseteq \mathbb{F}$ be a subset of size $d + 1$, such that $0 \in W$
 - 2: acquire common justifying assignment \bar{a} for F_1, F_2, \dots, F_k {using Algorithm 1}.
 - 3: **return** “true” iff $F_{\bar{a}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$.
-

Lemma 6.6. *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 non-zero 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 $F_{\bar{a}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$, where \bar{a} is common justifying assignment for the PROPs F_1, \dots, F_k . Corollary 6.5 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 $\mathcal{O}(n^4 k^2 d)$ (see Lemma 4.3 with $t = \mathcal{O}(n)$). Verifying that $F_{\bar{a}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$ requires at most $\left(\sum_{i=0}^{3k} \binom{n}{i} \cdot d^i\right) \cdot k$ time. We thus get that the running time is $\mathcal{O}(k \cdot \binom{n}{3k} \cdot d^{3k}) = (nd)^{\mathcal{O}(k)}$. \square

Theorem 2 is an immediate corollary of Lemma 6.6.

6.4 Black-Box PIT for Sum of PROPs

In this section we give a black-box version Algorithm 2 and prove Theorems 3 and 4. The idea is to combine the HOR approach (specifically, Corollary 6.5 which requires a common justifying assignment) and Lemma 4.5 that shows how to obtain a justifying set from black-box PIT algorithm.

Theorem 6.7. *Let F_1, \dots, F_k be PROPs, with individual degree bounded by d , that are computed by a circuit class \mathcal{M} . Let \mathcal{M}' be (another) circuit class such that $\partial\mathcal{M} \subseteq \mathcal{M}'$. Let $\mathcal{G} = (\mathcal{G}^1, \dots, \mathcal{G}^n) : \mathbb{F}^q \rightarrow \mathbb{F}^n$ be a generator for \mathcal{M}' (see Definition 4.4) such that the individual degrees of the \mathcal{G}^i -s are bounded by δ . Let $\mathcal{J}_{\mathcal{G}}^{k,d}$ be a (k, d) -justifying set (as defined in Lemma 4.5). Finally, let $W \subseteq \mathbb{F}$ be of size $|W| = d + 1$, such that $0 \in W$. Then Algorithm 3 determines whether $\sum_{m=1}^k F_m \equiv 0$, and its running time is $(nd\delta)^{\mathcal{O}(k+q)}$.*

Algorithm 3 Black-Box PIT algorithm for sum of k of PROPs

Input: Black-Box holding $F = F_1 + \dots + F_k$, for k PROPs F_1, \dots, F_k .Output: “true” if $F \equiv 0$ and “false” otherwise.

- 1: **return** “true” iff for each $\bar{\gamma} \in \mathcal{J}_{\mathcal{G}}^{k,d}$ $F_{\bar{\gamma}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$ {No non-zero entry was found}
-

Proof. We first show the correctness of the algorithm. Indeed, if the algorithm fails, then F evaluates to a non-zero value which implies that $F \not\equiv 0$. By Lemma 4.5 there exists $\bar{a} \in \mathcal{J}_{\mathcal{G}}^{k,d}$ which is a common justifying assignment for the PROPs F_1, \dots, F_k . If the algorithm outputs “true”, then, in particular $F_{\bar{a}}|_{\mathcal{A}_{3k}^n(W)} \equiv 0$. Corollary 6.5 implies that $F \equiv 0$.

Regarding the running time, by Lemma 4.5 $|\mathcal{J}_G^{k,d}| \leq (kn^2d\delta + 1)^q$. Hence, the total running time is $\mathcal{O}\left(|\mathcal{J}_G^{k,d}| \cdot |\mathcal{A}_{3k}^n(W)|\right) = (kn^2d\delta + 1)^q \cdot (nd)^{\mathcal{O}(k)} = (nd\delta)^{\mathcal{O}(k+q)}$. \square

We now show how to implement Algorithm 3 in several scenarios. We first implement the algorithm for the case that the F_m 's are general PROFs.

Proof of Theorem 3. From Lemma 5.3 we get that for $\ell = \lceil \log_2 n \rceil + 1$ the mapping $G_\ell : \mathbb{F}^{2\ell} \rightarrow \mathbb{F}^n$ is a generator for PROPs. Lemma 3.18 implies that PROPs are closed under partial derivatives (See Definition 2.22). Recall that the individual degrees of the G_ℓ^i -s are bounded by $\delta = n - 1$. Hence, Theorem 6.7 gives a black-box PIT algorithm for the sum of k PROPs of running time $(n^2d)^{\mathcal{O}(k+2\ell)} = (nd)^{\mathcal{O}(k+\log n)}$. \square

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

Proof of Theorem 4. From Lemma 5.11 we get that for $\ell = D + 1$ the mapping $G_\ell : \mathbb{F}^{2\ell} \rightarrow \mathbb{F}^n$ is a generator for PROPs of depth at most D . Lemma 5.9 implies that this circuit class is closed under partial derivatives (See Definition 2.22). Recall that the individual degrees of the G_ℓ^i -s are bounded by $\delta = n - 1$. Hence, Theorem 6.7 gives a black-box PIT algorithm for the sum of k PROPs of depth at most D of running time $(n^2d)^{\mathcal{O}(k+2\ell)} = (nd)^{\mathcal{O}(k+D)}$. \square

7 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 begin by formally defining our models and stating the relevant known results. Then we discuss the properties of $\bar{0}$ -justified polynomials computed in those models and finally we present our results.

Definition 7.1. 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 $P(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$ is a linear product if it can be represented as a product of linear functions e.g. $P(\bar{x}) = \prod_j L_j(\bar{x})$ when $L_j(\bar{x})$ are linear functions.

Definition 7.2. A depth-3 $\Sigma\Pi\Sigma(k)$ circuit C computes a polynomial of the form

$$C(\bar{x}) = \sum_{m=1}^k F_m(\bar{x}) = \sum_{m=1}^k \prod_{j=1}^{d_m} L_{mj}(\bar{x})$$

where the $L_{mj}(\bar{x})$ -s are linear functions. The F_m -s are the multiplication gates of the circuit. Note that the F_m -s are in fact linear products. We denote with $\Sigma\Pi\Sigma(k, d)$ 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 , or equivalently F_m are linear products of degree at most d . A multilinear $\Sigma\Pi\Sigma(k)$ circuit is as above with the additional requirement 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.

We now define a special case of depth-4 circuits: ‘‘Preprocessed $\Sigma\Pi\Sigma(k)$ ’’ circuits. Having a $\Sigma\Pi\Sigma(k)$ circuit at hand we ‘‘create’’ another level by replacing each variable with a univariate polynomial in a similar manner to PROFs and PROPs (recall Definition 3.12).

Definition 7.3. A preprocessed linear function is a polynomial of the form $L(\bar{x}) = \sum_{i=1}^n h_i(x_i)$, where $h_i(x_i)$ are univariate polynomials. A polynomial $F(\bar{x})$ is a preprocessed linear product if it can be represented as a product of preprocessed linear functions.

The intuition behind the definition is that such polynomials can be computed by composition of linear functions on preprocessings.

Definition 7.4. 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. We denote by $P\Sigma\Pi\Sigma(k, d)$ a $P\Sigma\Pi\Sigma(k)$ circuit of a total degree (at most) d (i.e. the total degree of each F_m is at most d).

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 ROP of depth 2. Therefore, preprocessed multilinear $\Sigma\Pi\Sigma(k)$ circuits are actually the sum of k PROPs each of depth 2. Note, each linear function in a preprocessed depth-3 circuit can have a different preprocessing. We can now apply the results of Section 6 (i.e. Theorems 6.2 and 6.4). We note that these results are tight for depth-3 circuits since Lemma 6.3 shows, in fact, a representation of \mathcal{P}_n as a sum of n $\bar{0}$ -justified linear products and a constant⁹. We thus obtain the following result.

Theorem (Restated Theorem 5). *Let F 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 F that runs in time $(nd)^{\mathcal{O}(k)}$.*

Note that preprocessed multilinear depth-3 circuits form a restricted class of depth-4 circuits. The known PIT algorithms for this circuit class are non black-box and cover only 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).

In addition to the above corollary we give a new algorithm for PIT of general $\Sigma\Pi\Sigma(k)$ circuits. Before presenting our algorithm we give several notations and discuss related results.

Definition 7.5. 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 the identically zero polynomial. 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. $\gcd(C) = \gcd(F_1, \dots, F_k)$. We say that circuit is simple if $\gcd(C) = 1$. The simplification of C , denoted by $\text{sim}(C)$, is defined as $C/\gcd(C)$. Note that if C is a $\Sigma\Pi\Sigma(k, d)$ then so is $\text{sim}(C)$. Let $\text{rank}(C)$ 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 “small”. Their bound was recently improved by [SS08].

⁹A slightly more sophisticated argument allows us to “get rid” of the constant.

Theorem 7.6 (Theorem 2 [SS08]). *Let $C \equiv 0$ be a simple and minimal $\Sigma\Pi\Sigma(k, d)$ circuit then $\text{rank}(C) < R(k, d)$, where $R(k, d) = \mathcal{O}(k^3 \log d)$.*

In [KS08] a black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d)$ circuits was given. The running time is $\text{poly}(n) \cdot d^{R(k, d)}$ ¹⁰. Hence, [SS08] got the following corollary:

Theorem 7.7 (Theorem 3 [SS08]). *There is a deterministic black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d)$ circuits that runs in $\text{poly}(n) \cdot d^{R(k, d)} = \text{poly}(n) \cdot d^{\mathcal{O}(k^3 \log d)}$ time.*

On the other hand, the best known *non black-box* PIT algorithm runs in $\text{poly}(n, d^k)$ time.

Theorem 7.8 (Theorem 1.1 [KS07]). *There is a deterministic non black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d)$ circuits that runs in $\text{poly}(n, d^k)$ time.*

7.1 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 $\Sigma\Pi\Sigma(k, d)$ based on the recent result of [SS08] using our hardness of representation approach. The result of [SS08] extends the previous structural theorems, giving an upper bound on the rank of the linear factors of a polynomial that is computed by a simple, minimal and non-zero $\Sigma\Pi\Sigma(k, d)$ circuit.

Definition 7.9. *Let $P(\bar{x}) = h_1(\bar{x}) \cdot h_2(\bar{x}) \cdot \dots \cdot h_t(\bar{x})$ be a non-zero polynomial and its irreducible factors, respectively. We denote by $\text{Lin}(P)$ the set of (non-constant) linear factors of P . We define $\text{Lin}(P) \triangleq \{h_i \mid h_i \text{ is a linear factor of } P\}$.*

Lemma 7.10 (Theorem 5 of [SS08]). *Let $P(x_1, \dots, x_n)$ be a polynomial computed by a simple, minimal and non-zero $\Sigma\Pi\Sigma(k, d)$ circuit then $\text{rank}(\text{Lin}(P)) \leq R(k, d)$.*

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 consider¹¹ non-zero- $\bar{0}$ -justified linear functions and linear products. The following observation shows one reason for that.

Observation 7.11. *Let $F(\bar{x}) = \prod_i L_i(\bar{x})$ be a (preprocessed) linear product. Then F is non-zero- $\bar{0}$ -justified if and only if $F(\bar{0}) \neq 0$. In addition, F is non-zero- $\bar{0}$ -justified if and only if $L_i(\bar{0}) \neq 0$ for each i .*

We now give an efficient algorithm for acquiring a non-zero-justifying assignment, using techniques from Section 4.

Lemma 7.12. *The function $G_1(y_1, z_1)$ (recall Definition 5.1) is a generator for preprocessed linear products.*

Proof. It is sufficient to show that the claim holds for a single non-constant preprocessed linear function $L(\bar{x}) = \sum_{i=1}^n h_i(x_i)$. From the definition, there exists j such that $h_j(x_j)$ is a non-constant polynomial. As $L(G_1(\alpha_j, z_1)) = \sum_{i \neq j} h_i(0) + h_j(z_1)$ (see Observation 5.2) we get that $L(G_1(y_1, z_1))$ is a non-constant polynomial. □

¹⁰At the time of that result a worse bound $R(k, d) = 2^{\mathcal{O}(k^2)} \log^{k-2} d$ was known.

¹¹It is possible to get similar algorithms using the “regular” $\bar{0}$ -justified polynomials, instead of the non-zero ones, however this requires additional technical work that does not contribute to the understanding of the model.

Note that similarly to Lemma 5.3 $G_1(y_1, z_1)$ is a generator regardless of the degree of the preprocessed L . Using this easy lemma, we can obtain a non-zero-justifying set $\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)}$ for $\text{P}\Sigma\Pi\Sigma(k,d)$ circuits. That is, for every $\text{P}\Sigma\Pi\Sigma(k,d)$ circuit $C(\bar{x}) = F_1(\bar{x}) + F_2(\bar{x}) + \dots + F_k(\bar{x})$ the set $\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)}$ contains an assignment \bar{a} which is a common non-zero-justifying assignment for F_1, F_2, \dots, F_k .

Lemma 7.13. *Let C be a $\text{P}\Sigma\Pi\Sigma(k,d)$ circuit over \mathbb{F} and $V \subseteq \mathbb{F}$ be of size $|V| = dnk$. Then $\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)} \stackrel{\Delta}{=} G_1(V^2)$ is a non-zero-justifying set for C and $|\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)}| \leq d^2 n^2 k^2$.*

The proof is carried out in a similar fashion to the proof of Lemma 4.5. We now prove the hardness of representation result for depth-3 circuits.

Theorem 7.14. *For every $g(\bar{x}) \neq 0$ the polynomial $g(\bar{x}) \cdot \mathcal{P}_n(\bar{x})$ cannot be represented as sum of k non-zero- $\bar{0}$ -justified linear products of (a total) degree d when $n > R(k,d)$.*

Proof. Let $C = \sum_{m=1}^k F_m$. Assume the contrary. Let C' be a minimal sub-circuit of C that computes $g(x) \cdot \mathcal{P}_n$. W.l.o.g $C' = \sum_{m=1}^{k'} F_m$ for some $k' \leq k$. Consider $\text{gcd}(C')$. As all the linear functions in C' are non-zero- $\bar{0}$ -justified we obtain that $x_i \notin \text{gcd}(C')$ for every $i \in [n]$. Therefore, $\text{gcd}(C', \mathcal{P}_n) = 1$. Consequently, if we consider the simplification of C' then we obtain:

$$C'' \stackrel{\Delta}{=} \text{sim}(C') = g'(x) \cdot \mathcal{P}_n(\bar{x})$$

where $g' = g/\text{gcd}(C') \neq 0$. That is, the polynomial $g' \cdot \mathcal{P}_n$ is computed by a simple, minimal, non-zero circuit C'' (C'' remains minimal after simplification). In addition, note that $x_i \in \text{Lin}(g' \cdot \mathcal{P}_n)$ for every $i \in [n]$. Hence from Lemma 7.10 we get that

$$n \leq \text{rank}(\text{Lin}(g' \cdot P)) \leq R(k', d) \leq R(k, d)$$

□

We can now establish a vanishing theorem. The proof is similar to the proof of Theorem 6.4.

Theorem 7.15 (Vanishing). *Let $\{F_m(\bar{x})\}_{m \in [k]}$ be non-zero- $\bar{0}$ -justified linear products over \mathbb{F} of total degree at most d and let $W \subseteq \mathbb{F}$ be a subset of size $d+1$, such that $0 \in W$. Let $C(\bar{x}) = \sum_{m=1}^k F_m(\bar{x})$. Then $C \equiv 0$ if and only if $C|_{\mathcal{A}_{R(k,d)}^n(W)} \equiv 0$.*

Using Theorems 7.14 and 7.15 we obtain a new PIT algorithm for $\Sigma\Pi\Sigma(k,d)$ circuits. This immediately implies Theorem 6.

Theorem 7.16. *Let C be $\Sigma\Pi\Sigma(k,d)$ circuit. Let $\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)}$ be as defined in Lemma 7.13 and $W \subseteq \mathbb{F}$ be of size $|W| = d+1$ such that $0 \in W$. Then Algorithm 4 determines whether $C \equiv 0$, and its running time is $(nd)^{\mathcal{O}(R(k,d))} = (nd)^{\mathcal{O}(k^3 \log d)}$.*

As before we assume that $|\mathbb{F}| > ndk$ (as required by Lemma 7.13).

Proof. The proof of correctness of the algorithm is identical to the proof of correctness of Algorithm 3. The running time is easily upper bounded by $\mathcal{O}\left(|\mathcal{J}_{\text{P}\Sigma\Pi\Sigma(k,d)}| \cdot |\mathcal{A}_{R(k,d)}^n(W)|\right) = (nd)^{\mathcal{O}(R(k,d))} = (nd)^{\mathcal{O}(k^3 \log d)}$. □

Algorithm 4 Black-Box PIT algorithm for $\Sigma\Pi\Sigma(k, d)$ circuits.

Input: Black-Box holding a $\Sigma\Pi\Sigma(k, d)$ circuit C

Output: “true” if $C \equiv 0$ and “false” otherwise.

1: **return** true iff for each $\bar{\gamma} \in \mathcal{J}_{\Sigma\Pi\Sigma(k,d)} C_{\bar{\gamma}}|_{\mathcal{A}_{R(k,d)}^n(W)} \equiv 0$ {No non-zero entry was found }

8 Reconstruction of a PROF

In this section we discuss the problem of ROF interpolation - that is, given a black-box containing a ROF f we wish construct a ROF \hat{f} such that $\hat{f} \equiv f$. We extend the existing reconstruction algorithm so it can handle PROFs.

8.1 Reconstruction of a ROF

We first give a very high level description of an algorithm of [HH91, BHH95] that shows how to reconstruct a ROF given a justifying assignment (an adaptation of Algorithm 3.2.2 in [HH91]). We shall later use this algorithm as a subroutine.

Algorithm 5 Reconstructing ROF (f, \bar{a})

Input: ROF f given as a black-box access, justifying assignment \bar{a} of f

Output: ROF \hat{f} such that $\hat{f} \equiv f$.

Step 1: Construct the gates-graph of f (using the justifying assignment)).

Step 2: Construct the skeleton of f (using the justifying assignment).

Step 3: Construct the ROF by recursively constructing its sub-formulas.

The following lemma is an adaptation of Lemma 3 of [BHH95].

Lemma 8.1 (Lemma 3 of [BHH95]). *Given oracle access to a black-box holding a ROF f and a justifying assignment \bar{a} of f there is a polynomial time algorithm A that can reconstruct f , that is construct a (non-degenerate) ROF \hat{f} such that $\hat{f} \equiv f$, in a non-adaptive manner. A high level description of A is given in Algorithm 5.*

8.2 Identifying the Preprocessing

An important phase in extending Algorithm 5 to deal with PROPs is identifying (learning) the preprocessing. Once we know the preprocessing, we can in some sense “simulate” the queries to the ROF “behind” the preprocessing. Let $P(\bar{x}) = Q(T(\bar{x}))$ be a PROP and its decomposition, respectively and let \bar{a} be a justifying assignment of P . Consider

$$P_i(x_i) = P(a_1, a_2, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n).$$

Observation 8.2. $P_i(x_i) = \alpha \cdot T_i(x_i) + \beta$ for some $\alpha, \beta \in \mathbb{F}$, in addition $\alpha \neq 0$ iff $x_i \in \text{var}(P)$.

At this point we, in fact, learn the $\text{var}(P)$ and thus assume w.l.o.g. that $\text{var}(P) = [n']$ for some $n' \leq n$, as we can simply ignore the variables in $[n] \setminus \text{var}(P)$. Furthermore, based on Lemma 3.14 and the (following it discussion) we can use $P_i(x_i)$ “instead” of $T_i(x_i)$ by computing the standard form.

8.3 Extending the algorithm for Reconstruction of a PROF

Here we summarize the extended version of the reconstruction algorithm. The main difference from Algorithm 5 is the (mandatory) Step 0, which is carried out by performing univariate polynomial interpolation on $f_i(x_i) = f(a_1, a_2, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n)$ (defined similar to P_i in Section 8.2). Note, that this step can be carried out in a non-adaptive manner, by simply querying f on $d + 1$ distinct points. Having the preprocessing at hand, it possible, in a sense, to execute the original algorithms as if the T_i 's (the preprocessings) were replaced by variables.

Algorithm 6 Reconstructing PROF (f, \bar{a})

Input: PROF f given as a black-box access, justifying assignment \bar{a} of f

Output: PROF \hat{f} such that $\hat{f} \equiv f$.

Step 0: Learning the preprocessing and $\text{var}(f)$ (using the justifying assignment).

Step 1: Construct the gates-graph of f (using the justifying assignment).

Step 2: Construct the skeleton of f (using the justifying assignment).

Step 3: Construct the ROF by recursively constructing its sub-formulas.

The following lemma summarizes the extend algorithm.

Lemma 8.3. *Given oracle access to a black-box holding a PROF f with individual degrees (of the preprocessing) bounded by d and a justifying assignment \bar{a} of f there is an algorithm that can reconstruct f , that is construct a (non-degenerate) PROF \hat{f} such that $\hat{f} \equiv f$, in a non-adaptive manner. The running time of the algorithm is $\text{poly}(n, d)$. A high level description of the algorithm is given in Algorithm 6.*

The following corollary is immediate.

Corollary 8.4. *There is deterministic algorithm that can reconstruct a PROF f . The algorithm runs in time $(nd)^{\mathcal{O}(\log n)}$. If $|\text{var}(f)| \leq t$ then the running time is $(nd)^{\mathcal{O}(\log t)}$. If $\text{depth}(f) \leq D$ then the running time is $(nd)^{\mathcal{O}(D)}$.*

Proof. We execute Algorithm 1 combined with Theorem 5.4 or Theorem 5.12 to obtain a justifying assignment \bar{a} for f . Then we apply Lemma 8.3. \square

9 Preprocessed Read-Once Testing

In this section we study the relation between the *polynomial identity testing* problem (PIT) and the *preprocessed read-once testing* problem and prove Theorem 8. The PROT problem is, clearly, a generalization of ROT problem discussed in [SV08]. We present a generic scheme which extends and, in some sense, improve the scheme presented in [SV08]. The scheme can be used to strengthen efficient PIT algorithms to yield efficient read-once testing algorithms.

9.1 Generic Scheme

The idea behind the scheme is: “Doveriai, no Proveriai”¹². First, we give a *read-once reconstruction* algorithm (based on Algorithms 1 and 5). That is, an algorithm that given a circuit C computing a PROP, outputs a PROF f such that $f \equiv C$. Then, to preform the preprocessed read-once

¹²“Trust, but Verify” - a translation of a Russian proverb which became a signature phrase of Ronald Reagan during the cold war.

testing, we assume that the given circuit C computes a PROF and run the preprocessed read-once reconstruction algorithm based on this assumption. If the algorithm encounters an error or is unable to run correctly, then we conclude that our assumption was wrong (i.e. C does not compute a PROF) and thus we stop and report a failure. Things are more complicated in the case of success, that is, when the algorithm does output a PROF. The problem is that we do not have a guarantee regarding the correctness of our assumption (that the circuit computes a PROF) and hence the correctness of its output. Moreover, for any circuit C that computes a PROF there exist many circuits (computing different polynomials) “aliasing” C . Meaning, that an execution of our preprocessed read-once reconstruction algorithm on each such circuit $C' \neq C$ will succeed and yield a PROF f such that $f \equiv C \neq C'$. Consequently, to complete the read-once testing we need to *verify* the correctness of the output. For this purpose we need a *verification* procedure (Algorithm 8). Algorithm 7 gives the generic scheme for PROT. The algorithm works both in the black-box and in the non black-box settings, depending on the PIT algorithm for \mathcal{M} at hand.

Algorithm 7 Generic PROT Scheme

Input: A (black-box holding a) circuit C that belongs to \mathcal{M} .

Output: PROF f such that $f \equiv C$, if C computes a PROF,
“failure” otherwise.

- 1: Acquire a justifying assignment \bar{a} using Algorithm 1
 - 2: Given the justifying assignment \bar{a} reconstruct f from C using Algorithm 6.
 - 3: Verify that $f \equiv C$ using Algorithm 8 (given in Subsection 9.2).
-

In Section 9.3 we give the proof of Theorem 8, that basically analyzes the running time of Algorithm 7 given the running times of Algorithms 6, 1 and 8.

9.2 Read-Once Verification

Read-Once Verification is testing whether a given circuit C , from a certain circuit class \mathcal{M} , and a given PROF f compute the same polynomial. Note though, that while the verification might have the nature of a polynomial identity testing, it is somewhat a harder problem since it requires determining the equivalence of polynomials computed by circuits from two different circuit classes. We shall work under the assumption that \mathcal{M} has a PIT algorithm and the PROF f is given to us explicitly (e.g. as a graph of computations). The circuit C , on the other hand, may be given to us as a black-box, depending on the PIT algorithm for \mathcal{M} . Clearly, if $C - f \in \mathcal{M}$ then the verification procedure is trivial (as \mathcal{M} has a PIT algorithm), the general case however is more complicated. We shall present a verification procedure that enables us to take care of the case where $C - f \notin \mathcal{M}$. The idea behind the algorithm is to recursively ensure that every gate v of f computes the same polynomial as a “corresponding” restriction of C . To give a rough sketch, we first find a justifying assignment for f , \bar{a} . Then we consider v , the root of f . It has two sub formulas. W.l.o.g. assume that $f = \alpha \cdot (f_{v_1} \text{ op } f_{v_2}) + \beta$, where f_{v_i} is the PROF computed at v_i and *op* is either $+$ or \times . Assume that the variables of v_i are S_i (S_1 and S_2 are disjoint). Consider the circuit C_1 that equals to C after we substitute the corresponding values from \bar{a} to the variables in S_2 . Similarly we define C_2 , and the PROFs¹³ f_1 and f_2 . We now recursively verify that $C_i \equiv f_i$. The only thing left now is to verify that indeed $C \equiv \alpha \cdot (C_1 \text{ op } C_2) + \beta$. This basically reduces the verification problem to the problem of verifying that $C \equiv C_1 \text{ op } C_2$ where C_1 and C_2 compute variable disjoint PROPs and *op* is either $+$ or \times . Note, that this is a PIT problem for a circuit class that is closely related

¹³In fact, in the algorithm it will be more convenient to have $f_i = f_{v_i}$ and so we will slightly change the definition of C_i .

to \mathcal{M} , although slightly different (e.g. if \mathcal{M} is the class of $\Sigma\Pi\Sigma(k)$ circuits, defined in Section 7, then we need a PIT algorithm for $\Sigma\Pi\Sigma(\mathcal{O}(k^2))$ circuits). Therefore, we shall assume that a slightly larger circuit class has an efficient PIT algorithm (e.g. a class containing $C - C_1 \text{ op } C_2$ for variable disjoint C_1 and C_2). For this we make the following definition of a “verifying class”. The definition uses the notations given in Definition 2.22. Note, that for most circuit classes that have efficient PIT algorithms, there also exists a PIT algorithm for a corresponding verifying class.

Definition 9.1. *A circuit class \mathcal{M}_V is a Verifying Class of \mathcal{M} if $P_1 + P_2 + P_3 \times P_4 \in \mathcal{M}_V$ when $P_1, P_2, P_3, P_4 \in \mathcal{L}(\mathcal{M})$ and P_2, P_3, P_4 are **variable disjoint**.*

Algorithm 8 Verify (C, f)

Input: Circuit C from a circuit class \mathcal{M} ,
The root v of the a PROF f ,
Justifying assignment \bar{a} for f ,
Access to a PIT algorithm for \mathcal{M}_V .

Output: “true” if $C \equiv f$ and “false” otherwise.

- 1: **if** $v.Type = \text{IN OR } v.Type = \text{CONST}$ **then** $\{p_v$ is a univariate or constant polynomial}
 - 2: Check that $C \equiv p_v$ { By querying on $d + 1$ points as two univariate polynomials of degree d }
{Internal Gate}
 - 3: $L \leftarrow \text{var}(p_{v.Left})$
 - 4: $R \leftarrow \text{var}(p_{v.Right})$
{Multiplication Gate}
 - 5: **if** $v.Type = \times$ **then**
 - 6: $C_L \leftarrow (C|_{x_R=\bar{a}_R} - v.\beta) / (v.\alpha \cdot p_{v.Left}(\bar{a}))$
 - 7: $C_R \leftarrow (C|_{x_L=\bar{a}_L} - v.\beta) / (v.\alpha \cdot p_{v.Right}(\bar{a}))$
 - 8: Verify($C_L, v.Left$) **and** Verify($C_R, v.Right$) {Recursively}
 - 9: Check that $C \equiv v.\alpha \cdot (C_L \times C_R) + v.\beta$ {Using the PIT algorithm for \mathcal{M}_V }
{Addition Gate}
 - 10: **if** $v.Type = +$ **then**
 - 11: $C_L \leftarrow (C|_{x_R=\bar{a}_R} - v.\beta) / v.\alpha - p_{v.Left}(\bar{a})$
 - 12: $C_R \leftarrow (C|_{x_L=\bar{a}_L} - v.\beta) / v.\alpha - p_{v.Right}(\bar{a})$
 - 13: Verify($C_L, v.Left$) **and** Verify($C_R, v.Right$) {Recursively}
 - 14: Check that $C \equiv \alpha \cdot (C_L + C_R) + v.\beta$ {Using the PIT algorithm for \mathcal{M}_V }
{Everything is OK}
 - 15: **return** true
-

Theorem 9.2. *Let C and f be a circuit from a circuit class \mathcal{M} , and a PROF correspondingly, such that $\text{var}(C) \subseteq \text{var}(f)$. In addition, let \mathcal{M}_V be a verifying class of \mathcal{M} and \bar{a} a justifying assignment of f . Then given C, \bar{a} and f Algorithm 8 runs in time $\mathcal{O}(nd + n \cdot t)$, when t is the cost of a single PIT algorithm for \mathcal{M}_V , and outputs “true” if and only if $C \equiv f$.*

Proof. We start by analyzing the running time. As described above, the algorithm performs a traversal over the tree of f . The PIT algorithm of \mathcal{M}_V is invoked once for every internal gate (Multiplication, Addition). In each each input gate a single variable query on $d + 1$ points is performed. Hence the total running time is $\mathcal{O}(nd + n \cdot t)$ when t is the cost of a single PIT algorithm for \mathcal{M}_V . We now prove the correctness of the algorithm.

There are two things to prove. First, we show that all the PIT calls that we make are “well defined”. Then, we show that given a PIT algorithm for \mathcal{M}_V the algorithm returns the correct answer.

From the first glance, we might expect “hazards” executing the lines which require an invocation of a PIT algorithm. That is, lines 2, 9 and 14. We make the following observations. In each stage of the algorithm it holds:

1. $C, C_L, C_R \in \mathcal{L}(\mathcal{M})$ (when C_L, C_R defined). This follows (recursively) from the definitions of C_L, C_R . (C_L, C_R are defined as an application of a linear function on a restriction of C).
2. $\text{var}(C) \subseteq \text{var}(p_v)$. By induction. For the base case we are guaranteed that $\text{var}(C) \subseteq \text{var}(f)$. For the step, consider the definition of the set L and R . By Lemma 2.4.

$$\text{var}(C|_{x_R=\bar{a}_R}) \subseteq \text{var}(C) \setminus R \subseteq \text{var}(f) \setminus R = L$$

Therefore, $\text{var}(C_L) \subseteq L = \text{var}(p_{v.\text{Left}})$ and similarly $\text{var}(C_R) \subseteq R = \text{var}(p_{v.\text{Right}})$. It is only left to notice that the above result corresponds with the recursive invocation of the algorithm.

3. $(C_L \cap C_R) \subseteq (\text{var}(p_{v.\text{Left}}) \cap \text{var}(p_{v.\text{Right}})) = \emptyset$. Follows from the definition of PROF.

From Observations 1 and 3 it follows that $C - v.\alpha \cdot (C_L \times C_R) - v.\beta \in \mathcal{M}_V$ and $C - v.\alpha \cdot (C_L + C_R) - v.\beta \in \mathcal{M}_V$. Hence, we can conclude that the identity tests in lines 9, 14 can be carried out using the PIT algorithm of \mathcal{M}_V . Consider line 2. In this case p_v either a constant polynomial or a univariate polynomial of degree at most d . As $\text{var}(C) \subseteq \text{var}(p_v)$ (Observation 2) we obtain that so is $C - p_v$. Hence the test can be carried out by querying C and p_v on (at most) $d + 1$ points.

The correctness of the algorithm’s output can be proven by a simple induction on v . That is, the algorithm outputs “true” iff $C \equiv p_v$. The base case is trivial. The correctness of the step follows from Lemmas 9.4 and 9.5 (and the correctness of the PIT algorithm of \mathcal{M}_V). This completes the proof. \square

Notice, that as f is given to us explicitly we can acquire a justifying assignment \bar{a} for f in $\mathcal{O}(n^4 d)$ time (see Lemma 4.3 with $t = \mathcal{O}(n)$). In addition, as $\partial\mathcal{M} \subseteq \mathcal{M}_V$ we can compute $\text{var}(C)$ by applying Lemma 4.2 and hence check whether $\text{var}(C) = \text{var}(f)$. Clearly, if $\text{var}(C) \neq \text{var}(f)$ then $C \not\equiv f$. The next corollary is immediate.

Corollary 9.3. *In time $\mathcal{O}(n^4 d \cdot t)$ we can verify whether $C \equiv f$, where t is the cost of a single PIT for \mathcal{M}_V .*

The following simple Lemmas from [SV08] are brought for the sake of completeness.

Lemma 9.4. *Let $C(\bar{x}, \bar{y}, \bar{z})$ and $f(\bar{x}, \bar{y}) = \alpha \cdot f_\ell(\bar{x}) \times f_r(\bar{y}) + \beta$ be two polynomials and let (\bar{x}_0, \bar{y}_0) be a justifying assignment of f . Then $C(\bar{x}, \bar{y}, \bar{z}) \equiv f(\bar{x}, \bar{y})$ if and only if the following conditions hold:*

1. $f_\ell(\bar{x}) = \frac{C(\bar{x}, \bar{y}_0, \bar{z}) - \beta}{\alpha \cdot f_r(\bar{y}_0)}$
2. $f_r(\bar{y}) = \frac{C(\bar{x}_0, \bar{y}, \bar{z}) - \beta}{\alpha \cdot f_\ell(\bar{x}_0)}$
3. $C(\bar{x}, \bar{y}, \bar{z}) = \alpha \cdot \frac{C(\bar{x}, \bar{y}_0, \bar{z}) - \beta}{\alpha \cdot f_r(\bar{y}_0)} \times \frac{C(\bar{x}_0, \bar{y}, \bar{z}) - \beta}{\alpha \cdot f_\ell(\bar{x}_0)} + \beta$

Notice that the conditions are well defined since (\bar{x}_0, \bar{y}_0) is a justifying assignment of f and hence $f_\ell(\bar{x}_0), f_r(\bar{y}_0) \neq 0$.

The additive version:

Lemma 9.5. *Let $C(\bar{x}, \bar{y}, \bar{z})$ and $f(\bar{x}, \bar{y}) = \alpha \cdot (f_\ell(\bar{x}) + f_r(\bar{y})) + \beta$ be two polynomials and let (\bar{x}_0, \bar{y}_0) be a justifying assignment of f . Then $C(\bar{x}, \bar{y}, \bar{z}) \equiv f(\bar{x}, \bar{y})$ if and only if the following conditions hold:*

1. $f_\ell(\bar{x}) = \frac{C(\bar{x}, \bar{y}_0, \bar{z}) - \beta}{\alpha} - f_r(\bar{y}_0)$
2. $f_r(\bar{y}) = \frac{C(\bar{x}_0, \bar{y}, \bar{z}) - \beta}{\alpha} - f_\ell(\bar{x}_0)$
3. $C(\bar{x}, \bar{y}, \bar{z}) = \alpha \cdot \left(\frac{C(\bar{x}, \bar{y}_0, \bar{z}) - \beta}{\alpha} - f_r(\bar{y}_0) + \frac{C(\bar{x}_0, \bar{y}, \bar{z}) - \beta}{\alpha} - f_\ell(\bar{x}_0) \right) + \beta$

Proof. Both proofs are preformed by simple substitutions. □

9.3 Proof of Theorem 8

Before giving the proof we show why any “reasonable” model that poses an interest from the PROT point of view can, indeed, compute univariate polynomials. To see that notice that all the univariate PROPs are, in fact, all the univariate polynomials. Now, let C be a circuit from a circuit class \mathcal{M} that computes a non-constant PROP. By considering all the corresponding restrictions we will obtain circuits in \mathcal{M} that compute univariate polynomials.

Theorem 9.6 (Analog of Theorem 7 of [SV08]). *Let C be a size s circuit on n variables from a not trivial, linearly closed circuit class \mathcal{M} . Assume that \mathcal{M}_V has a PIT algorithm of running time $t(s)$, for circuits of size s . Assume further that the individual degrees of the polynomial computed by C are bounded by d . Then there is a deterministic algorithm that runs in time $\text{poly}(n, s, d, t(s))$ that when given access (explicit or via a black-box, depending on the PIT algorithm for \mathcal{M}) to C solves the PROT problem.*

Proof of Theorem 8. We will show that given \mathcal{M} and \mathcal{M}_V satisfying the required conditions we can successfully preform each phase of the “Generic PROT Scheme” described in Algorithm 7.

1. Acquiring a Justifying Assignment

As $\partial\mathcal{M} \subseteq \mathcal{M}_V$ we can acquire a justifying assignment \bar{a} by invoking Algorithm 1. The running time is $\mathcal{O}(n^3 d \cdot T(n, d, s))$.

2. Reconstructing ROF f from C

Given the justifying assignment \bar{a} we can reconstruct f from C using Algorithm 6. The running time is $\text{poly}(n, d, s)$.

3. Verifying that $f \equiv C$

Notice that \mathcal{M}_V satisfies the conditions of Definition 9.1 and hence can serve a verifying class of \mathcal{M} . This allows to invoke the verification procedure described in Algorithm 8.

The running time is $\mathcal{O}(n^4 d \cdot T(n, d, s))$.

The total running time is $\text{poly}(n, s, d, T(n, d, s))$. □

In the next section we use Theorem 8 and the Generic Scheme suggested in Algorithm 7 in order to get PROT algorithms for several restricted models for which PIT is known. Given our result it is necessary that a PIT algorithm is known for the class that we wish to get PROT algorithm

for. However, we note that basically any “reasonable” PIT algorithm yields a ROT algorithm. For example, an immediate conclusion of Theorem 8 is that an efficient PIT algorithm for multilinear circuits (for both black-box and none black-box settings) implies an efficient read-once testing algorithm for multilinear circuits.

Using the same schema and the Schwartz-Zippel result [Zip79, Sch80] as a verification algorithm we can obtain a two-sided randomized algorithm for the PROT problem, while the Schwartz-Zippel result shows that $\text{PIT} \in \text{coRP}$ (i.e. a one-sided randomized algorithm for PIT). Similarly, as a consequence from Theorem 8 we obtain that $\text{ROT} \in \text{P}^{\text{PIT}} \subseteq \text{BPP}$ (in the decision version of the problem).

References

- [AB03] M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *JACM*, 50(4):429–443, 2003. 3
- [Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105, 2005. 3
- [Alo99] N. Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8:7–29, 1999. 9
- [AM07] V. Arvind and P. Mukhopadhyay. The monomial ideal membership problem and polynomial identity testing. In *Proceedings of the 18th International Symposium on Algorithm and Computation (ISAAC)*, pages 800–811, 2007. 3, 5, 30
- [AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the forty ninth Annual FOCS*, pages 67–75, 2008. 3, 5
- [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. 5
- [BHH95] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995. 5, 6, 33
- [BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the 20th Annual STOC*, pages 301–309, 1988. 3
- [CK97] Z. Chen and M. Kao. Reducing randomness via irrational numbers. In *Proceedings of the 29th Annual STOC*, pages 200–209, 1997. 3
- [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. 3, 5, 30
- [DSY08] Z. Dvir, A. Shpilka, and A. Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 741–748, 2008. 3
- [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. 5, 6, 10, 33

- [KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. [3](#)
- [KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual STOC*, pages 216–223, 2001. [3](#)
- [KS07] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007. [1](#), [3](#), [5](#), [31](#)
- [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. [3](#), [4](#), [5](#), [31](#)
- [LV98] D. Lewin and S. Vadhan. Checking polynomial identities over any field: Towards a derandomization? In *Proceedings of the 30th Annual STOC*, pages 428–437, 1998. [3](#)
- [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. [3](#)
- [Raz04] R. Raz. Multilinear $NC_1 \neq$ Multilinear NC_2 . In *Proceedings of the 45th Annual FOCS*, pages 344–351, 2004. [3](#)
- [Raz05] R. Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual STOC*, pages 11–20, 2005. [3](#)
- [RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005. [3](#)
- [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. [3](#)
- [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. [3](#)
- [Sax08] N. Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP (1)*, pages 60–71, 2008. [3](#), [30](#)
- [Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *JACM*, 27(4):701–717, 1980. [3](#), [39](#)
- [Shp09] A. Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. on Computing*, 38(6):2130–2161, 2009. [5](#)
- [SS08] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *CoRR*, abs/0811.3161, 2008. [3](#), [4](#), [5](#), [30](#), [31](#)
- [SV08] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. In *Proceedings of the 40th Annual STOC*, pages 507–516, 2008. [1](#), [3](#), [4](#), [5](#), [6](#), [10](#), [11](#), [12](#), [17](#), [21](#), [34](#), [37](#), [38](#)
- [Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation*, pages 216–226. 1979. [3](#), [39](#)