Electronic Colloquium on Computational Complexity, Report No. 184 (2018)

Uniform, Integral and Feasible Proofs for the Determinant Identities

Iddo Tzameret[*] Stephen A. Cook[†]

Abstract

Aiming to provide weak as possible axiomatic assumptions in which one can develop basic linear algebra, we give a uniform and integral version of the short propositional proofs for the determinant identities demonstrated over $GF(2)$ in Hrubeš-Tzameret [HT15]. Specifically, we show that the multiplicativity of the determinant function and the Cayley-Hamilton theorem over the integers are provable in the bounded arithmetic theory $\mathbf{VNC}^2$; the latter is a first-order theory corresponding to the complexity class $\mathbf{NC^2}$ consisting of problems solvable by uniform families of polynomial-size circuits and $O(\log^2 n)$-depth. This also establishes the existence of uniform polynomial-size $\mathbf{NC^2}$-Frege proofs of the basic determinant identities over the integers (previous propositional proofs hold only over the two element field).

Contents

[*] Department of Computer Science, Royal Holloway, University of London. Iddo.Tzameret@rhul.ac.uk
[†] Department of Computer Science, University of Toronto. sacook@cs.toronto.edu

ISSN 1433-8092

1 Introduction

The complexity of linear algebraic operations such as matrix inverse and the determinant is well studied (cf. Cook [Coo85]). It is well known that many linear algebraic operations like the determinant can be computed quickly in parallel, and specifically are in $\mathbf{NC^2}$, which is the complexity class consisting of all languages that can be decided by uniform families of $O(\log^2 n)$-depth and polynomial-size circuits (ignoring for now the distinction between function and language classes). This complexity class captures fast parallel computation in the sense that a language in it can be decided in time $O(\log^2 n)$ while using polynomially many processors working in parallel. In fact, within the $\mathbf{NC} := \cup_{i=0}^{\infty}\mathbf{NC}^i$ hierarchy, which consists of all polynomial-size circuit families of poly-logarithmic depth, $\mathbf{NC^2}$ is the weakest level known to compute the determinant (formally, the weakest circuit class computing integer determinants is the class DET that lies between $\mathbf{NC^1}$ and $\mathbf{NC^2}$; see below).

Furthermore, the importance of linear algebra in bounded arithmetic and proof complexity has been identified in many works, and it has been conjectured that the determinant identities, and specifically the multiplicativity of the determinant function $\mathrm{DET}(A) \cdot \mathrm{DET}(B) = \mathrm{DET}(AB)$, for two matrices A, B, can be proved in a formal theory that, loosely speaking, reasons with $\mathbf{NC^2}$ concepts (Cook and Nguyen present this specific question in their monograph [CN10]; see also [CF12, BBP95, BP98, Sol01, SC04]). This conjecture is aligned with the intuition that basic properties of many constructions and functions of a given complexity class are provable in logical theories not using concepts beyond that class.

The weakest theory known to date to prove the determinant identities is $\mathbf{PV}$ which corresponds to polynomial-time reasoning; this was shown by Soltys and Cook [SC04] (cf. [CF12, Jeř05]). Quite recently, Hrubeš and Tzameret [HT15] showed that at least in the *propositional* case, the determinant identities expressing the multiplicativity of the determinant over $GF(2)$ can be proved with polynomial-size propositional proofs operating with $\mathbf{NC^2}$-circuits (as well as with quasipolynomial size Frege proofs). However, this does not lend itself immediately to the uniform framework of bounded arithmetic. That is, the fact that a statement admits polynomial-size propositional proofs in a certain proof-system does not imply that the same statement (suitably translated to first-order logic) is provable in the bounded arithmetic theory corresponding to the proof-system. For example, a short propositional proof may be shown to exist, but without knowing whether it could be constructed uniformly, and let alone in a restricted computational model such as uniform-$\mathbf{NC^2}$—making it thus impossible to carry out directly in bounded arithmetic.

Furthermore, [HT15] crucially used in their construction elimination of division gates from algebraic circuits, which we do not know how to do using uniform weak computational models like uniform-$\mathbf{NC^2}$ (since for general division elimination one needs to use the existential statement about field assignments that do not nullify a given polynomial [Str73]).

The main goal of this work is to prove the determinant identities in the theory $\mathbf{VNC}^2$ (corresponding to "$\mathbf{NC^2}$-reasoning"). We will show that similar reasoning as in [HT15] can be carried over to $\mathbf{VNC}^2$, with further complications imposed by uniformity and parallelism. As a result of working in bounded arithmetic it will also become possible to conclude short propositional-proofs over the integers (while the previous propositional proofs worked only over $GF(2)$).

Organization. The preliminaries for this work are somewhat long. *For this reason we begin with a* high-level *overview of the results and their proofs in Section 2* (readers who are unfamiliar with some of the concepts in the overview can consult the preliminaries section for those). The preliminaries themselves are given in Section 3, consisting of basic definitions from bounded arithmetic, the uniform complexity class $\mathbf{NC^2}$, the corresponding theory $\mathbf{VNC}^2$ [CN10], basic definitions of algebraic circuits, as well as proof systems operating with algebraic circuits establishing polynomial identities (PI-proofs [HT09, HT15]). In Section 4 we give a much more detailed guide to the proof of the determinant identities in the theory, while still leaving out many of the technical details and proofs. Section 5 explains in some detail how we encode certain algebraic circuits in the theory. Sections 6 to 10 are dedicated to the construction in uniform $\mathbf{NC^2}$ the PI-proof from [HT15]. Section 11 wraps-up the proof by establishing the reflection principle for Polynomial Identity (PI) proofs, and Section 12 provides $\mathbf{VNC}^2$ proofs of further basic statement in linear algebra. We finish with conclusions and open problems in Section 13. The appendix provides more background details about bounded arithmetic as well as some technical lemmas that do not appear in the main text.

2 Overview

Our goal is to prove the determinant identities inside $\mathbf{VNC}^2$. For the logical setting and $\mathbf{VNC}^2$ see Section 3. Specifically, we want to have a $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ function $\mathrm{DET}(\cdot)$ with input an integer matrix and output an integer represented as a binary string, such that $\mathbf{VNC}^2$ proves:

$$\forall A, B\; n \times n \text{ matrices over } \mathbb{Z},\; \mathrm{DET}(A) \cdot \mathrm{DET}(B) = \mathrm{DET}(AB) \tag{1}$$

and

$$\forall C\; n \times n \text{ triangular matrix over } \mathbb{Z},\;\; \mathrm{DET}(C) = c_{11} \cdots c_{nn}. \tag{2}$$

Note that these two identities can be considered as the *defining* identities of the determinant polynomial, in the sense that every polynomial for which these two identities hold is the determinant polynomial. One way of seeing this is to observe that every square matrix is equal to a product of upper and lower triangular matrices.

Integer numbers are represented as binary strings in the theory, where the least significant bit (lsb) is 0 (resp. 1) when the integer is positive (resp. negative), and where the rest of the string is the binary representation of the absolute value of the integer. An $n \times n$ matrix over $\mathbb{Z}$ is encoded as a two-dimensional array (cf. [CF12]).

It is not hard to show that we can prove *simple* facts about matrices, such as the definability of matrix product AB, the statement expressing associativity and commutativity of matrix products $A(BC) = (AB)C$ and $A + B = B + A$, resp., and so forth (see for example [SC04, CN10] and [HT15, Lemma 28] about these basic identities that can be proved already in the theory $\mathbf{VNC}^1$, that corresponds to $\mathbf{NC^1}$).

All circuit classes discussed in this work (except when otherwise stated) are assumed to be uniform circuit classes. Formally, we require uniformity in the sense that the extended connection language of the circuit family is in $\mathbf{FO}$ (see [CN10, Chapter A.5] for the definition).

Let us now sketch briefly how we define the determinant function in the theory and then how we *prove* its identities (5) and (6) in the theory.

Defining the determinant function in the theory. Given an $n \times n$ integer matrix, the $\mathbf{\Sigma}_1^B$-definable string function (recall that we encode integers as strings) in $\mathbf{VNC}^2$ for the determinant is defined roughly as follows: first, construct an $O(\log^2 n)$-depth algebraic circuit computing the determinant of $n \times n$ integer matrices, and then evaluate the circuit under the input assignment.

More specifically, the determinant function in the theory first constructs a recursive algebraic circuit (or equivalently, a straight-line program) computing the symbolic $n \times n$ determinant *with division gates* ("symbolic" here means that the algebraic circuit computes the determinant as the formal polynomial over n^2 distinct variables). This is done using the standard recursive formula of the block-wise determinant (using "Schur complement"), simulating in a sense Gaussian elimination (cf. [HT15]). Then, *eliminate the division gates* in the determinant circuit using, among other conversions, substitutions of power series in the circuit. Then, *homogenize* the circuit getting rid of high degrees, *balance the circuit* to achieve the squared logarithmic depth, and finally *evaluate* the result under the input integer matrix.

The function that evaluates a balanced algebraic circuit in itself consists of several steps, as follows: given as an input a balanced algebraic circuit, the function: (i) converts it into a layered circuit (namely, a circuit in which each node connects only to the subsequent layer); (ii) transforms it into a *Boolean circuit* computing the same polynomial over the integers (coded as bit-strings) while taking care that the negations appear only in the bottom layer; and finally (iii) evaluates the Boolean circuit using the fact that the Monotone $\mathbf{NC^2}$ Circuit Evaluation Problem is $\mathbf{NC^2}$-complete (under $\mathbf{AC}^0$-reductions [CN10]).

Note that since we show that the determinant function as defined above is Σ_1^B-definable in $\mathbf{VNC}^2$, by [CN10] it means that this function is in uniform-$\mathbf{NC^2}$.

Proving the determinant equalities in the theory. Informally, the basic argument formalized in the theory is that there exists a balanced PI-proof (for *Polynomial Identity proof*), in symbols, a $\mathbb{P}_c(\mathbb{Z})$-proof (as in [HT15]; see Section 3.5), of these identities. Thus, by soundness of balanced $\mathbb{P}_c(\mathbb{Z})$-proofs, which we show is provable in $\mathbf{VNC}^2$, these identities must be true. Informally, a $\mathbb{P}_c(\mathbb{Z})$-proof is a sequence of equations between algebraic circuits over $\mathbb{Z}$, each of which is either an instance of the polynomial-ring axioms or was derived by addition or multiplication of previous equations.

More precisely, we demonstrate a Σ_1^B-definable function in $\mathbf{VNC}^2$ that given an input n in unary, outputs $\mathbb{P}_c(\mathbb{Z})$-proofs of the determinant identities (see equations (5) and (6)). In this $\mathbb{P}_c(\mathbb{Z})$-proof every proof-line is an equation between depth $O(\log^2(n))$ algebraic circuits (without division gates) of a polynomial syntactic-degree. To conclude the argument, we use the soundness of $O(\log^2(n))$-depth $\mathbb{P}_c(\mathbb{Z})$-proofs: using induction on proof-length we argue that for *every assignment of integers*, the determinant identity (equations (5) and (6)) must hold.

One important observation in this work, that is central in constructing the PI-proofs in the theory, is that for some parts in the construction the only properties that the theory is required to express and prove about these PI-proofs are "local and syntactic" properties, namely the fact that the each proof-line follows syntactically from previous ones.

For example, let C be an algebraic circuit of polynomial-size and exponential syntactic-degree; e.g., $(x^2)^{2\cdots 2} - (x^2)^{2\cdots 2} + 1$, where $(x^2)^{2\cdots 2}$ is written as a chain of n product gates. The theory cannot express the fact that C has exponential syntactic-degree (because the theory defines only polynomially bounded number functions). Nevertheless, the theory can prove, for example, that $(x^2)^{2\cdots 2} - (x^2)^{2\cdots 2} + 1 = 1$ has a legal PI-proof, using possibly an axiom of the form $F - F = 0$.

Overall, in our argument, the main "non-syntactic" property we need the theory to express about algebraic circuits is the evaluation of $O(\log^2 n)$-depth circuits over $\mathbb{Z}$. The axiom of $\mathbf{VNC}^2$ is specifically tailored for this purpose (see Section 11.1). We also use the ability to power matrices in $\mathbf{NC^2}$ when balancing the PI-proofs in the theory.

2.1 Technical Challenges

Showing that the long and nontrivial constructions from [HT15] can be carried out in $\mathbf{VNC}^2$ requires quite a lot of work. The main technical obstacles that we face are *parallelism* and *uniformity* as we explain in what follows.

Parallelism here means that the construction of the original PI-proofs from [HT15] *must be done by itself in* $\mathbf{NC^2}$. The construction in [HT15] is quite involved, and to make it parallel we need to devise several $\mathbf{AC}^0$- and $\mathbf{NC^2}$-algorithms (all Σ_1^B-definable in $\mathbf{V}^0$ and $\mathbf{VNC}^2$, respectively). In fact we show that most parts of the construction can be carried out already in $\mathbf{AC}^0$ (or its functional version $\mathbf{FAC}^0$), namely we carry out the construction in $\mathbf{V}^0$. Among the algorithms we devise are the following ones:

(**i**) Division normalization: converting algebraic circuits with division gates into circuits with a single division gate at the output gate (in $\mathbf{FAC}^0$); This follow Strassen's algorithm [Str73]. (**ii**) Converting algebraic circuits C into the sum of their syntactic-homogeneous components, given as input an upper bound on the syntactic-degree of C; i.e., each summand $C^{(i)}$ is a syntactic-homogeneous circuit computing the degree i homogeneous component of C (in $\mathbf{FAC}^0$); This also follows Strassen's algorithm [Str73], only that we show that for most purposes there is no need to compute syntactic-degrees of nodes, rather *upper bounds* on syntactic-degrees suffice. Such upper bounds are easy to compute in $\mathbf{AC}^0$. (**iii**) An $\mathbf{FNC}^2$ algorithm for balancing an algebraic circuit of size s and syntactic-degree d into a poly(s, d)-size algebraic circuit of depth $O(\log s \cdot \log d + \log^2 d)$, given as input an upper bound on the syntactic-degree of C. This part combines the original balancing algorithm by Valiant *et al.* [VSBR83] with ideas from Miller *at al.* [MRK88], and further new ideas entailed by the need to work in $\mathbf{FNC}^2$. Specifically, we use matrix powering to power adjacency matrices of graphs to find out, for example, whether a node has a directed path to another node, as well as to compute coefficients of linear polynomials computed by circuits with syntactic-degree 1.

By first balancing an input circuit and then evaluating it (both in $\mathbf{FNC}^2$) our results give rise to: (**iv**) an F$\mathbf{NC^2}$ *evaluation procedure for algebraic circuits of any depth* (given as input an upper bound on their syntactic-degree and assuming the syntactic degree of the circuit is polynomial[1]) that is different from the previously known algorithm by Miller *et al.* [MRK88] (their algorithm does not require the syntactic-degree as input) and that of Allender *et al.* [AJMV98] (which is implicit in that work, and can be extracted from the text [All18]; see also Vinay [Vin91]).

Proving parallel algorithms for structural results on algebraic circuits is however not enough. We further need to show that the correctness of these algorithms can be formalized efficiently with PI-proofs and that these proofs are constructible in $\mathbf{V}^0$ and $\mathbf{VNC}^2$, in order to conclude that $\mathbf{VNC}^2$ proves the existence of a (uniform $\mathbf{NC^2}$) function that constructs the low depth PI-proofs of the determinant identities.

Uniformity here means that we need the whole proof to be constructible in uniform-$\mathbf{NC^2}$. For instance, we need to eliminate division gates from certain algebraic circuits and proofs. To eliminate division gates like u/v (for two nodes u, v), one needs to find an assignment to the variables in which the polynomial computed at node v is nonzero. In general we do *not* know how to do this in the theory. Nevertheless, we show that for our purposes it is enough to eliminate only those division gates that occur in some specific circuits. In order to eliminate division gates we will also need to find 'inverse elements' in the ring of integers, and hence we will have to show that for our purposes it is enough to consider only the inverse of 1 in $\mathbb{Z}$.

Apart from uniformity and parallelism, working in bounded arithmetic allows us to work more easily over the integers, where previously short $\mathbf{NC^2}$-Frege proofs of the determinant identities were known only over $GF(2)$ [HT15].

[1] Formally, we need to assume that the syntactic-degree of every node in the circuit when constant nodes are replaces by corresponding variables is polynomially bounded.

2.2 Note on the Choice of Theory

It is interesting to consider whether the theory in which the determinant identities is proved can be pushed even further down to a theory that corresponds to a complexity class that lies somewhere between $\mathbf{NC^1}$ and $\mathbf{NC^2}$.

Cook and Fontes [CF12] developed a bounded arithmetic theory $V\#L$, corresponding to DET, where DET is the class of functions that can be computed by uniform families of polynomial-size constant-depth Boolean circuits with oracle access to the determinant over $\mathbb{Z}$ (where integer entries of matrices are presented in binary). In other words, DET is the $\mathbf{AC}^0$-closure of integer determinants. Complete problems for the class DET include computing matrix powers and the determinant itself. We have the following class inclusions (we ignore here the distinction between function and decision classes): $\mathbf{NC^1} \subseteq \mathsf{DET} \subseteq \mathbf{NC^2}$, to which the theories $\mathbf{VNC}^1 \subseteq V\#L \subseteq \mathbf{VNC}^2$ correspond.

Our argument cannot be carried out in $V\#L$ since the evaluation of algebraic circuits, even those with squared logarithmic depth (or those in algebraic-$\mathbf{AC}^1$) over the integers, which is crucial to our argument, is apparently not definable in $V\#L$. Note that excluding the evaluation of low-depth algebraic circuits all our arguments seem to carry over to $V\#L$. This also includes for example our algorithm for balancing algebraic circuits.[2]

Note also that the two classes $\#\mathbf{SAC}^1 \subseteq \mathbf{TC}^1$ that are above DET but below $\mathbf{NC^2}$, can compute the required depth reduction and the evaluation of algebraic circuits. *We believe that our construction can be carried out more or less the same in theories corresponding to these classes*. However, for these two classes we are not aware of established bounded arithmetic theories, hence we shall work in $\mathbf{VNC}^2$.

3 Preliminaries

In this section we present some of the necessary background from bounded arithmetic as well as algebraic circuit complexity. Specifically, we describe the two-sorted bounded arithmetic theory $\mathbf{VNC}^2$ as developed by Cook and Nguyen [CN10] and show how to define the evaluation of algebraic circuits over the integers in the theory, and then define algebraic circuits computing formal polynomials and proof systems for polynomial identities [HT09, HT15] (cf. [PT16] for a survey). We start with an exposition of bounded arithmetic.

Bounded arithmetic is a general name for weak formal systems of arithmetic, namely, fragments of Peano Arithmetic. The bounded arithmetic theories we use are first-order two-sorted theories, having a first-sort for natural numbers and a second-sort for finite sets of numbers, representing bit-strings via their characteristic functions (for the original *single-sort* treatment of theories of bounded arithmetic see [Bus86, HP93, Kra95]). The theory $\mathbf{V}^0$ corresponds to the complexity class uniform-$\mathbf{AC}^0$, and $\mathbf{VNC}^2$ corresponds to uniform-$\mathbf{NC^2}$. The complexity classes $\mathbf{AC}^0$, $\mathbf{NC^2}$, and their corresponding function classes $\mathbf{FAC}^0$ and $\mathbf{FNC}^2$ are defined using a two-sorted universe (specifically, the first-ordered sort [numbers] are given to the machines in unary representation and the second-sort as binary strings). See Section 3.2 below for the definitions of $\mathbf{NC^2}$ and $\mathbf{FNC}^2$, and Definition A.6 in the appendix for $\mathbf{AC}^0$ and $\mathbf{FAC}^0$.

Definition 3.1 (Language of two-sorted arithmetic $\mathcal{L}_A^2$)**.** *The language of two-sorted arithmetic, de-*

[2] It is possible also to balance algebraic circuits to squared logarithmic depth in DET using some variants of the algorithm in [AJMV98], as we were informed by Eric Allender [All18].

noted $\mathcal{L}_A^2$, consists of the following relation, function and constant symbols:

$$\{+, \cdot, \leq, 0, 1, |\,|, =_1, =_2, \in\}.$$

We describe the intended meaning of the symbols by considering the standard model $\mathbb{N}_2$ of two-sorted Peano Arithmetic. It consists of a first-sort universe $U_1 = \mathbb{N}$ and a second-sort universe U_2 of all finite subsets of $\mathbb{N}$, which are thought of as strings. The constants 0 and 1 are interpreted in $\mathbb{N}_2$ as the appropriate natural numbers zero and one, respectively. The functions $+$ and $\cdot$ are the usual addition and multiplication on the universe of natural numbers, respectively. The relation $\leq$ is the appropriate "less or equal than" relation on the first-sort universe. The function $|\cdot|$ maps a finite set of numbers to its largest element plus one. The relation $=_1$ is interpreted as equality between numbers, $=_2$ is interpreted as equality between finite sets of numbers. The relation $n \in N$ holds for a number n and a finite set of numbers N if and only if n is an element of N.

We denote the first-sort (number) variables by lower-case letters $x, y, z, \ldots$, and the second-sort (string) variables by capital letters $X, Y, Z, \ldots$.

We build formulas in the usual way, using two sorts of quantifiers: number quantifiers and string quantifiers. A number quantifier is said to be *bounded* if it is of the form $\exists x(x \leq t \wedge \ldots)$ or $\forall x(x \leq t \rightarrow \ldots)$, respectively, for some *number term* t that does not contain x. We abbreviate $\exists x(x \leq t \wedge \ldots)$ and $\forall x(x \leq t \rightarrow \ldots)$ by $\exists x \leq t$ and $\forall x \leq t$, respectively. A string quantifier is said to be *bounded* if it is of the form $\exists X(|X| \leq t \wedge \ldots)$ or $\forall X(|X| \leq t \rightarrow \ldots)$ for some *number term* t that does not contain X. We abbreviate $\exists X(|X| \leq t \wedge \ldots)$ and $\forall X(|X| \leq t \rightarrow \ldots)$ by $\exists X \leq t$ and $\forall X \leq t$, respectively.

A formula is in the class of formulas $\mathbf{\Sigma}_0^B$ or $\mathbf{\Pi}_0^B$ if it uses *no string quantifiers* and all number quantifiers are bounded. A formula is in $\mathbf{\Sigma}_{i+1}^B$ or $\mathbf{\Pi}_{i+1}^B$ if it is of the form $\exists X_1 \leq t_1 \ldots \exists X_m \leq t_m \psi$ or $\forall X_1 \leq t_1 \ldots \forall X_m \leq t_m \psi$, where $\psi \in \mathbf{\Pi}_i^B$ and $\psi \in \mathbf{\Sigma}_i^B$, respectively, and t_i does not contain X_i, for all $i = 1, \ldots, m$. We write $\forall \mathbf{\Sigma}_0^B$ to denote the universal closure of $\mathbf{\Sigma}_0^B$ (i.e., the class of $\mathbf{\Sigma}_0^B$-formulas that possibly have [not necessarily bounded] universal quantifiers on their front [left]). We write $T(t)$ to abbreviate $t \in T$, for a number term t and a string term T.

As mentioned before, a finite set of natural numbers N represents a finite string $S_N = S_N^0 \ldots S_N^{|N|-1}$ such that $S_N^i = 1$ if and only if $i \in N$. We will abuse notation and identify N and S_N.

3.1 The Theory $\mathbf{V}^0$

The base theory $\mathbf{V}^0$, which corresponds to the computational class $\mathbf{AC}^0$, consists of the following axioms:

Basic 1. $x + 1 \neq 0$ **Basic 2**. $x + 1 = y + 1 \rightarrow x = y$

Basic 3. $x + 0 = x$ **Basic 4**. $x + (y + 1) = (x + y) + 1$

Basic 5. $x \cdot 0 = 0$ **Basic 6**. $x \cdot (y + 1) = (x \cdot y) + x$

Basic 7. $(x \leq y \wedge y \leq x) \rightarrow x = y$ **Basic 8**. $x \leq x + y$

Basic 9. $0 \leq x$ **Basic 10**. $x \leq y \vee y \leq x$

Basic 11. $x \leq y \leftrightarrow x < y + 1$

Basic 12. $x \neq 0 \rightarrow \exists y \leq x(y + 1 = x)$

L1. $X(y) \rightarrow y < |X|$ **L2**. $y + 1 = |X| \rightarrow X(y)$

$$\textbf{SE.}\ (|X| = |Y| \wedge \forall i \leq |X| \, (X(i) \leftrightarrow Y(i))) \rightarrow X = Y$$

$$\mathbf{\Sigma_0^B}\textbf{-COMP.}\ \exists X \leq y \forall z < y \, (z \in X \leftrightarrow \varphi(z)) \,, \text{ for all } \varphi \in \mathbf{\Sigma}_0^B$$
$$\text{where } X \text{ does not occur freely in } \varphi \,.$$

Here, the axioms **Basic 1** through **Basic 12** are the usual axioms used to define Peano Arithmetic without induction (PA^-), which settle the basic properties of addition, multiplication, ordering, and of the constants 0 and 1. The Axiom **L1** says that the length of a string coding a finite set is an upper bound to the size of its elements. **L2** says that $|X|$ gives the largest element of X plus 1. **SE** is the axiom for strings which states that two strings are equal if they code the same sets. Finally, $\mathbf{\Sigma_0^B}$**-COMP** is the comprehension axiom *scheme* for $\mathbf{\Sigma}_0^B$-formulas (i.e., it is an axiom for each such formula) and implies the existence of all sets which contain exactly the elements that fulfill any given $\mathbf{\Sigma}_0^B$ property.

Proposition 3.2 (Corollary V.1.8. [CN10])**.** *The theory* $\mathbf{V}^0$ *proves the* number *induction axiom scheme for* $\mathbf{\Sigma}_0^B$*-formulas* Φ*:*

$$(\Phi(0) \wedge \forall x \, (\Phi(x) \rightarrow \Phi(x+1))) \rightarrow \forall z \, \Phi(z).$$

In the above induction axiom, x is a number variable and Φ can have additional free variables of both sorts.

We seek to define the determinant function in a theory via a $\mathbf{\Sigma}_1^B$-formula, where a function is said to be *defined in a theory* if the theory can prove that given an input to the function there always exists a unique output. For the exact definition of definability of functions in $\mathbf{V}^0$ (and $\mathbf{VNC}^2$) consult the appendix (Section A). Note that the $\mathbf{\Sigma}_1^B$-definable functions of $\mathbf{V}^0$ (equivalently, the $\mathbf{\Sigma}_0^B$-definable functions of $\mathbf{V}^0$) are precisely the $\mathbf{FAC}^0$ functions, and that the $\mathbf{\Sigma}_1^B$-definable functions of $\mathbf{VNC}^2$ are precisely the $\mathbf{FNC}^2$ functions (see Theorem 3.5 below).

3.2 The Complexity Class $\mathbf{NC^2}$

The uniform complexity class $\mathbf{NC^2}$ is defined using an alternating time-space (nondeterministic) Turing machine.

Alternating Turing machines. An alternating Turing machine is a *nondeterministic* Turing machine in which every state, except the halting states, is either an *existential state* or a *universal state*. A *computation* in such a machine can be viewed as an (unbounded fan-in) tree of configurations as follows. A configuration is said to be *existential* (resp. *universal*) if its state is existential (resp. universal). In a computation tree of an alternating Turing machine every *existential* configuration has one or more children, such that each child is a configuration reachable in one step from the configuration in the parent node; and every universal configuration has as its set of children *all* configurations reachable in one step from the configuration on the parent in node. We say that a computation of an alternating Turing machine is *accepting* when all the leaves of the computation tree are accepting configurations. We say that an alternating Turing machine *accepts an input* x if there *exists* an accepting computation tree whose root is the initial configuration with the input x.

A computation tree is said to have k *alternations* if the number of alternations between existential and universal states in every branch of the tree is at most k. An alternating Turing machine is said to *work in* $f(n)$ *alternations* if for every input x of length n the number of alternations in *every* computation tree of x is at most $f(n)$. A computation tree is said to have *space* s if the working space used in every configuration of the tree is at most s. An alternating Turing machine is said to *work in space* $g(n)$ if for every input x of length n the space of every computation tree of x is at most $g(n)$.

Definition 3.3 (Uniform $\mathbf{NC^2}$)**.** *The uniform complexity class* $\mathbf{NC^2}$ *is defined to be the class of languages that can be decided by alternating Turing machines with* $O(\log n)$ *space and* $O(\log^2 n)$ *time.*

We define the *function* class $\mathbf{FNC}^2$ as the function class containing all number functions $f(\vec{x}, \vec{X})$ and string functions $F(\vec{x}, \vec{X})$, where $\vec{x}$ and $\vec{X}$ are number and string variables, respectively, such that the relation of the function is defined (resp. bit-defined; see Definition A.5) in $\mathbf{NC^2}$ (a binary relation R is *defined in* $\mathbf{NC^2}$ if the language containing the set of pairs in R is decidable in $\mathbf{NC^2}$).

$\mathbf{NC^2}$ Boolean circuit families. Let $\{C_n\}_{n=1}^{\infty}$ be a family of Boolean circuits (with fan-in at most two $\vee, \wedge, \neg$ gates). We say that this family is an $\mathbf{NC^2}$ *circuit family* if every circuit C_n in the family has depth $O(\log^2 n)$ and size $n^{O(n)}$. A circuit taken from a given Boolean $\mathbf{NC^2}$ circuit family is said to be an $\mathbf{NC^2}$*-circuit*. It is known that the $\mathbf{NC^2}$ circuit value problem is complete under $\mathbf{AC}^0$-reductions for the class $\mathbf{NC^2}$ (Definition 3.3). We say that $\{C_n\}_{n=1}^{\infty}$ is a ***uniform*** $\mathbf{NC^2}$***-circuit family*** if its extended connection language is in $\mathbf{FO}$ (we refer the reader to [CN10, page 455] for the definitions). This definition coincides with Definition 3.3.

For the definition of uniform $\mathbf{NC^1}$ (and $\mathbf{AC}^1$) we also refer the reader to [CN10].

3.3 The Theory $\mathbf{VNC}^2$

Here we define the theory $\mathbf{VNC}^2$ as developed in [CN10]. It is an extension of $\mathbf{V}^0$ over the language $\mathcal{L}_A^2$ where we add the axiom stating the existence of a sequence of values that represent the evaluation of monotone Boolean circuits of $O(\log^2(n))$-depth. It is known (cf. [CN10]) that the Monotone Boolean Circuit Value problem for circuits of $O(\log^2(n))$-depth is complete under $\mathbf{AC}^0$-reductions for $\mathbf{NC^2}$.

The $\mathbf{NC^2}$ circuit value problem is the problem that determines the value computed by a Boolean $\mathbf{NC^2}$-circuit, given a 0-1 assignment to its input variables. An input circuit to the problem is encoded as a *layered circuit* with $d+1$ layers, namely, a circuit in which every node in layer j is connected only to zero or more nodes in layer $j+1$. The actual evaluation of such an ($\mathbf{NC^2}$) circuit within the class $\mathbf{NC^2}$ is done in stages, where we start from layer 0 and "compute" (using alternations and nondeterminism) the values of every node in every layer. Formally, we define this evaluation process as follows (see also [CN10, Chap. IX.5.6]).

The layered monotone Boolean circuit with $d+1$ layers is encoded with a string variable I, with $|I| \leq n$, which defines the (Boolean) input gates to the circuit. Then we have a string variable G such that $G(x, y)$, for $x \in [d]$, holds iff the yth gate in layer x is $\wedge$, and is $\vee$ otherwise. Also the wires of C are encoded by a three-dimensional array, namely a string variable E such that $E(z, x, y)$ holds iff the output of gate x on layer z is connected to the input of gate y on layer $z+1$. To compute the value of each of the gates in the circuit C on input I, simply compute the values of the gates in each layer, starting from the input layer, in $d+1$ stages, using the values of the previous layer. The formula $\delta_{LMCV}(n, d, E, G, I, Y)$ below formalizes this evaluation procedure (where *LMCV* stands for "layered monotone circuit value"). The two-dimensional array Y stores the result of computation, namely the evaluation string: for $1 \leq z \leq d$, row $Y^{[z]}$ contains the gates on layer z that output 1.

$$\begin{aligned}\delta_{LMCV}(n,d,E,G,I,Y) \equiv \\ \forall x < n \forall z < d \Big((Y(0,x) \leftrightarrow I(x)) \wedge \\ \big(Y(z+1,x) \leftrightarrow \big(\big(G(z+1,x) \wedge \forall u < n, E(z,u,x) \rightarrow \\ Y(z,u)\big) \vee (\neg G(z+1,x) \wedge \exists u < n, E(z,u,x) \wedge Y(z,u))\big)\big)\Big). \qquad (3)\end{aligned}$$

The following formula states that the circuit with underlying graph (n, d, E) has fan-in two:

$$\begin{aligned}Fanin2(n,d,E) \equiv \\ \forall z < d \forall x < n \exists u_1 < n \exists u_2 < n \exists v < n (E(z,v,x) \rightarrow \\ (v = u_1 \vee v = u_2)). \qquad (4)\end{aligned}$$

Finally, we arrive at the definition of $\mathbf{VNC}^2$:

Definition 3.4 ($\mathbf{VNC}^2$)**.** *The theory* $\mathbf{VNC}^2$ *has vocabulary* $\mathcal{L}_A^2$ *and is axiomatized by the axioms of* $\mathbf{V}^0$ *and the axiom:*

$$Fanin2(n, |n|^2, E) \rightarrow \exists Y \le \langle |n|^2 + 1, n\rangle \delta_{LMCV}(n, |n|^2, E, G, I, Y).$$

In this definition $\langle \cdot \rangle$ is the pairing function, and $\langle |n|^2 + 1, n\rangle$ is an upper bound on the length needed for the two-dimensional array Y. Also, note that given a natural number n the binary representation length of n, denoted $|n|$, that is, $\lceil \log_2(n+1) \rceil$, is an $\mathbf{AC}^0$ function of n (see [CN10, Exercise III.3.30]).

Recall the concept of a $\mathbf{\Sigma}_1^B$-definable function in a theory (see the appendix Section A). The following is the main theorem for $\mathbf{V}^0$ and $\mathbf{VNC}^2$:

Theorem 3.5. ([CN10, Corollaries V.5.2 and IX.5.31]) A function is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ iff it is $\mathbf{\Sigma}_0^B$-definable in $\mathbf{V}^0$ iff it is in $\mathbf{FAC}^0$. A function is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ iff it is in $\mathbf{FNC}^2$.

Note that the fact that a function is defined in the theory does not mean that we can prove all of its properties, or even anything interesting about it. To actually prove statements about a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{VNC}^2$, for example, we need to carefully consider the $\mathbf{\Sigma}_1^B$-formula defining it, formulate the property that we want to prove in the theory as a formula in the language $\mathcal{L}_A^2$, and verify that indeed the formula is provable in the theory.

3.4 Polynomials and Algebraic Circuits

For a good monograph on algebraic circuits and their complexity see Shpilka and Yehudayoff [SY10]. Let $\mathbb{G}$ be a ring. Denote by $\mathbb{G}[X]$ the ring of (commutative) polynomials with coefficients from $\mathbb{G}$ and variables $X := \{x_1, x_2, \ldots\}$. A *polynomial* is a formal linear combination of monomials, where a *monomial* is a product of variables. Two polynomials are *identical* if all their monomials have the same coefficients. The *degree* of a polynomial is the maximal total degree of a monomial in it.

Algebraic circuits and formulas over the ring $\mathbb{G}$ compute polynomials in $\mathbb{G}[X]$ via addition and multiplication gates, starting from the input variables and constants from the field. More precisely, an *algebraic circuit* C is a finite directed acyclic graph (DAG) with *input nodes* (i.e., nodes of in-degree zero) and a single *output node* (i.e., a node of out-degree zero). Input nodes are labeled with either a variable or a field element in $\mathbb{F}$. All the other nodes have in-degree two (unless otherwise stated) and are labeled by either an addition gate $+$ or a product gate $\times$. An input node is said to *compute* the variable or scalar that labels itself. A $+$ (or $\times$) gate is said to compute the addition (product, resp.) of the (commutative) polynomials computed by its incoming nodes. An algebraic circuit is called a *formula*, if the underlying directed acyclic graph is a tree (that is, every node has at most one outgoing edge). The *size* of a circuit C is the number of nodes in it, denoted $|C|$, and the *depth* of a circuit is the length of the longest directed path in it.

We say that a polynomial is *homogeneous* whenever every monomial in it has the same (total) degree.

Definition 3.6 (Syntactic-degree $d(\cdot)$)**.** *Let C be a circuit and v a node in C. The* syntactic-degree *$d(v)$ of v is defined as follows:*

1. *If v is a field element or a variable, then $d(v) := 0$ and $d(v) := 1$, respectively;*
2. *If $v = u + w$ then $d(v) := \max\{d(u), d(w)\}$;*
3. *If $v = u \cdot w$ then $d(v) := d(u) + d(w)$.*

An algebraic circuit is said to be *syntactic-homogeneous* if for every plus gate $u + v$, $d(u) = d(v)$.

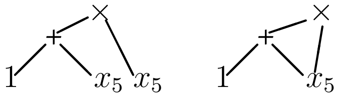
Given a circuit F and a node u in F, F_u denotes the subcircuit of F with output node u. If F, G are two circuits then

$$F \oplus G \text{ and } F \otimes G$$

denotes *any* circuit H whose output node is $u + v$ or $u \times v$, respectively, where H_u is the circuit F and H_v the circuit G. In other words, $F \oplus G$ denotes a circuit with output node $+$ with the two incoming subcircuits F and G, where F and G *may not be disjoint* (so $F \oplus G$ is a set of possible different circuits, from which we assume one is picked; the two subcircuits F, G of the output node of $F \oplus G$ are *identical* to F, G, respectively). Furthermore,

$$F + G \text{ and } F \times G$$

denote the *unique* circuit of the form $F' \oplus G'$ and $F' \otimes G'$, respectively, where F', G' are disjoint copies of F and G. In particular, if F and G are formulas then so are $F + G$ and $F \times G$. For example, $(1 + x_5) \otimes x_5$ can be any of the following two circuits:



3.5 Polynomial Identity (PI-) Proofs

In this section we give the necessary background on the PI-proof system $\mathbb{P}_c$. This proof-system was first introduced in [HT09] (under the name "arithmetic proofs" and for algebraic formulas instead of algebraic circuits), and was subsequently studied in [HT15].

PI-proofs, as originally introduced in [HT09], denoted $\mathbb{P}_c$ (and $\mathbb{P}_c(\mathbb{G})$ when we wish to be explicit about the ring $\mathbb{G}$), are sound and complete proof systems for the set of polynomial identities of $\mathbb{G}$, written as equations between algebraic circuits. A PI-proof starts from axioms like associativity, commutativity of addition and product, distributivity of product over addition, unit element axioms, etc., and derives new equations between algebraic circuits $F = G$ using rules for adding and multiplying two previous identities. The axioms of $\mathbb{P}_c$ express reflexivity of equality, commutativity and associativity of addition and product, distributivity, zero element, unit element, and true identities in the field.

Algebraic circuits in PI-proofs are treated as purely syntactic objects (similar to the way a propositional formula is a syntactic object in propositional proofs). Thus, simple computations such as multiplying out brackets, are done explicitly, step by step.

Definition 3.7 (***PI-proofs***; System $\mathbb{P}_c(\mathbb{G})$, [HT09, HT15]). *The system $\mathbb{P}_c(\mathbb{G})$ proves equations of the form $F = G$ over the ring $\mathbb{G}$, where F, G are algebraic circuits over $\mathbb{G}$. The inference rules of $\mathbb{P}_c$ are (with F, G, H ranging over algebraic circuits, and where an equation below a line can be derived from the one above the line):*

$$\begin{array}{llll}
\text{R1} & \dfrac{F = G}{G = F} & \text{R2} & \dfrac{F = G \qquad G = H}{F = H} \\[2ex]
\text{R3} & \dfrac{F_1 = G_1 \qquad F_2 = G_2}{F_1 + F_2 = G_1 + G_2} & \text{R4} & \dfrac{F_1 = G_1 \qquad F_2 = G_2}{F_1 \cdot F_2 = G_1 \cdot G_2}.
\end{array}$$

The axioms are equations of the following form, with F, G, H circuits:

A1 $F = F$

A2 $F + G = G + F$

A3 $F + (G + H) = (F + G) + H$

A4 $F \cdot G = G \cdot F$

A5 $F \cdot (G \cdot H) = (F \cdot G) \cdot H$

A6 $F \cdot (G + H) = F \cdot G + F \cdot H$

A7 $F + 0 = F$

A8 $F \cdot 0 = 0$

A9 $F \cdot 1 = F$

A10 $a = b + c\,,\ a' = b' \cdot c'$ *(if $a, b, c, a', b', c' \in \mathbb{G}$,*
are such that the equations hold in $\mathbb{G}$);

C1 $F \oplus G = F + G$

C2 $F \otimes G = F \cdot G$

A $\mathbb{P}_c$($\mathbb{G}$)-proof *is a sequence of equations, called* ***proof-lines****,* $F_1 = G_1,\ F_2 = G_2, \ldots, F_k = G_k$*, with F_i, G_i circuits, such that every equation is either an axiom or was obtained from previous equations by one of the inference rules. The* ***size*** *of a proof is the total size of all circuits appearing in the proof. The* number of steps *in a proof is the number of proof-lines in it.*

A PI-proof can be easily verified for correctness in deterministic polynomial-time (assuming the field (or ring) has efficient representation; e.g., the field of rational numbers or the the ring $\mathbb{Z}$), simply by syntactically checking that each proof line is derived from previous lines by one of the inference rules.

3.6 Circuits and Proofs with Division

We denote by $\mathbb{G}(X)$ the field of formal rational functions in the variables X, where a formal rational fraction is a fraction of two formal polynomials with coefficients from $\mathbb{G}$. In this work we will consider $\mathbb{G}$ to be the ring of integers $\mathbb{Z}$. We will not be interested in 'inverse elements' in $\mathbb{Z}$ (excluding the element 1), nor much in the completeness or soundness of proof systems for rational functions (like $\mathbb{P}_c^{-1}(\mathbb{Z})$ described below), because the theory will only prove *syntactical* properties of these proof systems (hence, no actual 'division' is performed over the integers).

It is possible to extend the notion of a circuit so that it computes rational functions in $\mathbb{G}(X)$ ([HT15]). This is done in the following way: a ***circuit with division*** F is an algebraic circuit which may contain an additional type of gate with fan-in 1, called an *inverse* or a *division* gate, denoted $(\cdot)^{-1}$. A division gate v^{-1} (i.e., a division gate whose incoming circuit is v) computes the rational function $1/\widehat{v} \in \mathbb{G}(X)$, assuming v does not compute the zero polynomial. If the circuit with division F contains some division gate v^{-1} such that v computes the zero polynomial, then we say that the circuit F is *not well-defined*, and is otherwise *well-defined*. Note, for instance, that the circuit $(x^2 + x)^{-1}$ over $GF(2)$ is well-defined, since $x^2 + x$ is not the zero polynomial (although it vanishes as a function over $GF(2)$, for example).

We define the system $\mathbb{P}_c^{-1}(\mathbb{G})$, operating with equations $F = G$ where F and G are circuits with division [HT15], as follows: first, we extend the axioms of $\mathbb{P}_c(\mathbb{G})$ to apply to well-defined circuits with division. Second, we add the following new axiom:

$$\text{D} \qquad F \cdot F^{-1} = 1\,, \text{ provided that } F^{-1} \text{ is well-defined.}$$

Note that if F^{-1} is well-defined then both F is well-defined and $F \neq 0$. We sometimes call the $\mathbb{P}_c^{-1}$-system *PI-proof* as well (although it operates with rational functions and not merely polynomial).

We say that a $\mathbb{P}_c^{-1}$-proof is ***syntactically correct*** if it is a correct $\mathbb{P}_c^{-1}$-proof except that in the axiom D above F^{-1} is *not* necessarily well-defined. Since we do not know how to check in uniform $\mathbf{NC^2}$ that a circuit is well-defined, we do not know how to express the full correctness of $\mathbb{P}_c^{-1}$-proofs in the $\mathbf{VNC}^2$. For our purposes it is sufficient that $\mathbf{VNC}^2$ expresses only the syntactic correctness of $\mathbb{P}_c^{-1}$-proofs.

The *syntactic-degree of a circuit C with division* is defined as

$$d(C) := d(\text{Num}(C)) + d(\text{Den}(C)).$$

4 Carrying the Proof in the Theory: Overview

Here we provide a detailed overview of the proof of the determinant identities in the theory, as highlighted before in Section 2.

We assume all polynomials are over the ring of integers $\mathbb{Z}$. We reason inside $\mathbf{VNC}^2$ (and $\mathbf{V}^0$) about $\mathbb{P}_c^{-1}(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-proofs (Definition 3.7 and Section 3.6). We use the following *reflection principle*, stating that if an equation has a proof then the equation is true:

Theorem 4.1 ($\mathbb{P}_c(\mathbb{Z})$-reflection principle; In $\mathbf{VNC}^2$**).** *Let π be an $O(\log^2 n)$-depth $\mathbb{P}_c(\mathbb{Z})$-proof of the equation $F = G$. Then $F = G$ is* true *in $\mathbb{Z}$; that is, $\forall \alpha \in \mathbb{Z}^n(F(\alpha) = G(\alpha))$.*

Theorem 4.1 is proved as follows. We define the *evaluation function* for $O(\log^2 n)$-depth algebraic circuits over $\mathbb{Z}$ as the function that receives an integer assignment A and an $O(\log^2 n)$-depth algebraic circuit C. The algorithm then converts C into a layered and monotone *Boolean* $\mathbf{NC^2}$ circuit, where the inputs are the bit-strings corresponding to A. And then evaluates the Boolean circuit using evaluation of $\mathbf{NC^2}$ circuits ($\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$), and finally outputs the result (see Section 11.1).

We also need to show in $\mathbf{VNC}^2$ that the rules and axioms of $O(\log^2 n)$-depth $\mathbb{P}_c(\mathbb{Z})$ are sound with respect to the above evaluation function. This is proved by inspection of each of the axioms and rules.

Note that we do *not* know how to prove the soundness of $\mathbb{P}_c^{-1}$-proofs in $\mathbf{VNC}^2$. This is because the division axiom $F \cdot F^{-1} = 1$ requires that $\widehat{F} \neq 0$, and we do not know how to check in $\mathbf{NC^2}$ that a circuit does not compute the zero polynomial. However, we observe that for our purposes it is enough to show that given a specific object (a $\mathbb{P}_c^{-1}$-proof for the determinant identities) it is possible to obtain from this object a new legitimate $\mathbb{P}_c$-proof of the determinant identities. See the example in the *Eliminating division gates* part below.

The determinant function DET **in the theory.** We now describe slightly informally the (uniform-$\mathbf{NC^2}$) determinant function DET defined in the theory. Essentially, each step in the algorithm corresponds to a (more involved) step in the construction of the final PI-proof of the determinant identities in the theory (as described after the algorithm below).

Algorithm DET (in $\mathbf{VNC}^2$)

Input: an $n \times n$ integer matrix A.
Output: $z \in \mathbb{Z}$, where z is the determinant of A.

1. Write down an unbalanced algebraic circuit $\mathsf{Det}_{circ^{-1}}(X)$ with division that computes the symbolic $n \times n$ determinant polynomial, over the variables $X = \{x_{ij}\}_{i,j\in[n]}$. This circuit captures the standard recursive block-wise formula for computing the determinant of matrices, using Schur complement (intuitively, it captures the Gaussian elimination procedure). For details see Section 5.1.2.

2. Consider the circuit $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ as computing a univariate polynomial in the new variable z. Using this circuit, construct a new circuit $\mathsf{Det}_{Taylor}(X)$ computing the nth term of the Taylor expansion of $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ around $z = 0$. This is a circuit *with* a division gate, of exponential syntactic-degree, that computes the determinant *as a polynomial*. For details see Section 6.2

3. Convert the circuit $\mathsf{Det}_{Taylor}(X)$ into a syntactic homogeneous circuit without division of syntactic-degree n, denoted $\mathsf{Det}^{\#}_{Taylor}(X)$. For details see Section 6.3.

4. Make sure that constant leaves in $\mathsf{Det}^{\#}_{Taylor}(X)$, when treated as if they are variables, do not lead up to nodes of exponential syntactic-degree in $\mathsf{Det}^{\#}_{Taylor}(X)$. This is done by constructing the circuit $\mathsf{Det}^{\star}_{Taylor}(X)$, based on $\mathsf{Det}^{\#}_{Taylor}(X)$. For details see Section 10.1.1.

5. Balance $\mathsf{Det}^{\star}_{Taylor}(X)$ via a (uniform) balancing algorithm, to yield a polynomial size and $O(\log^2 n)$-depth circuit without division denoted $\mathsf{Det}_{balanced}(X)$ that computes the determinant polynomial. For details see Section 10.

6. Evaluate the circuit $\mathsf{Det}_{balanced}(X)$ with the input assignment A, using the algebraic circuit evaluation function for $O(\log^2 n)$-depth circuits, and output the resulting integer in binary. For details see Section 11.

Since we show that all the parts in the algorithm above are $\mathbf{\Sigma}_1^B$-definable functions in $\mathbf{VNC}^2$, the determinant function as defined above is $\mathbf{\Sigma}_1^B$-definable in the theory (namely, totally recursive).

Given the function DET we now sketch the proof in $\mathbf{VNC}^2$ of the two determinant identities (5), (6) below.

Step 1: Existence of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs with division gates. We show in $\mathbf{V}^0$ a $\mathbf{\Sigma}_0^B$-definable function that given a natural number n outputs a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π_0 of the following equations

$$\mathsf{Det}_{circ^{-1}}(X) \cdot \mathsf{Det}_{circ^{-1}}(Y) = \mathsf{Det}_{circ^{-1}}(XY) \tag{5}$$

$$\mathsf{Det}_{circ^{-1}}(Z) = z_{11} \cdots z_{nn}, \tag{6}$$

for X, Y symbolic $n \times n$ matrices; that is, the (i,j)th entry of X and Y are the variables x_{ij} and y_{ij}, respectively, and Z a lower (equivalently, upper) triangular symbolic matrix in which the variable z_{ij} is the (i,j)th entry of Z iff $i \geq j$, and 0 otherwise.

These are equations between algebraic circuits over $\mathbb{Z}$. This is a proof in which circuits have exponential syntactic-degrees (though the theory cannot express this fact). The circuits in the proof are not necessarily homogeneous, and have division gates. The theory can also only prove that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is syntactically correct (Section 3.6). Note that $\mathsf{Det}_{circ^{-1}}(X)$ computes the determinant as a rational function and not as a polynomial. The construction of the proofs uses only the $\mathbf{\Sigma}_0^B$**-COMP** axiom and thus is done already in $\mathbf{V}^0$. See Section 5.2.1 for details.

Step 2: From the determinant polynomial to a rational function. For technical reasons relating to eliminating both division gates and high syntactic-degrees, we will need to construct in the theory a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities in which the determinant circuits appearing in the identities that are proved are, firstly, written as polynomials and not as rational functions, that is, as circuits without division; and secondly, have small syntactic-degree. Nevertheless, note that some intermediate $\mathbb{P}_c^{-1}(\mathbb{Z})$ proof-lines will contain the determinant written with division gates and having high syntactic-degree. The first task is achieved in the current step, and the second task in the next step.

Let $F = F(\overline{x}, z)$ be a circuit with division of syntactic-degree d. Similar to [HT15], we define $\mathrm{coeff}_{z^k}(F)$ as a circuit in the variables $\overline{x}$, computing the coefficient of z^k in F, when F is written as a power series at $z = 0$. In other words, $\sum_{i=0}^{d} \mathrm{coeff}_{z^i}(F) \cdot z^i$ are the first $d+1$ terms in the Taylor expansion of F at $z = 0$.

Let $\mathsf{Det}_{Taylor}(X) := \mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ be the circuit computing the nth term of the Taylor expansion of $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ about $z = 0$. It is easy to see that $\mathsf{Det}_{Taylor}(X)$ computes the determinant function: since every variable x_{ij} is multiplied by z, the coefficient of z^n is precisely the determinant.

By construction, $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ will compute the determinant *as a polynomial* and will have only one division gate (this is where we differ from [HT15]; due to the fact that we cannot simply substitute division gates u^{-1} that compute 1 by the node 1, because the theory needs to express the correctness of this substitution in some way). Furthermore, since we work over $\mathbb{Z}$ we need to make sure that the only invertible ring element needed to be used is the element 1.

We then show in $\mathbf{V}^0$ the existence of a function that given a natural positive number n outputs a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{Taylor}(X) = \mathsf{Det}_{circ^{-1}}(X)$, for the $n \times n$ symbolic matrix X. Combined with the previous step, $\mathbf{V}^0$ proves the existence of a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, denoted π_1, of the determinant identities (5), (6), in which the determinant circuit in (5), (6) is replaced by Det_{Taylor}. See Section 6.2.

Step 3: Reducing the syntactic-degree of the determinant polynomial. The circuit $\mathsf{Det}_{Taylor}(X)$ has exponential syntactic-degree (here we once more differ from [HT15], since we do not know how to formulate and prove the correctness of an $\mathbf{NC^2}$-algorithm that eliminates 0 nodes in general algebraic circuits, or nodes of high syntactic-degree that compute the zero polynomial). However, for the next step, we need $\mathsf{Det}_{Taylor}(X)$ to have a *polynomial* syntactic-degree. We show in $\mathbf{V}^0$ that there exists a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{Taylor}(X) = \mathsf{Det}^{\#}_{Taylor}(X)$, where $\mathsf{Det}^{\#}_{Taylor}(X)$ has syntactic-degree n and no division. This is done simply by a direct construction of such a proof using the $\mathbf{\Sigma}_0^B$-**COMP** axiom, and thus is carried out in $\mathbf{V}^0$.

Therefore, by previous steps, $\mathbf{V}^0$ proves the existence of a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities (5), (6), where the determinant in these two equations is replaced by $\mathsf{Det}^{\#}_{Taylor}$ which is an algebraic circuit with no division gates and of syntactic-degree n. Denote this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof by π_2 (this proof does contain the determinant circuit written as $\mathsf{Det}_{circ^{-1}}$ and Det_{Taylor} but only in intermediate proof-lines). See Section 6.3.

Step 4: Bringing division gates to the top. We say that a circuit C *has a division at the top* whenever C is of the form $F \cdot (G)^{-1}$ or $(G)^{-1} \cdot F$, for two circuits F, G. If F, G do not have division gates we say that C *has a single division gate at the top*. We need our circuits to have a single division gate at the top, because in the next step we need to replace division gates by an "approximating" power series, but we do not know how to do it with nested divisions.

We devise an $\mathbf{FAC}^0$ algorithm that takes an algebraic circuit with division, *of any depth*, and outputs an algebraic circuit computing the same rational function that has a *single* division gate at the top. Using this algorithm, we show in $\mathbf{V}^0$ how to convert the $\mathbb{P}_c^{-1}(\mathbb{Z})$*-proof* π_2 into a proof in which every circuit has a single division gate at the top or is division free. Denote the resulted proof by π_3. This step is shown in Section 7.

Step 5: Eliminating division gates. We now wish to eliminate the division gates from the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π_3. Standard division elimination by Strassen [Str73] requires finding a total assignment to the variables, such that no division gate in the circuit equals zero under this assignment. However, we do not know how to uniformly find such assignments in uniform-$\mathbf{NC^2}$, and so we do not know how to uniformly eliminate division gates from general algebraic circuits in $\mathbf{VNC}^2$. We solve this by working out the division elimination only for those circuits in π_3.

In fact, the only properties of the proof-sequence π_3, as well as the proof-sequence π_4 constructed during this stage, that we need to express in the theory are about the proof-sequences having a good "local" behaviour, namely that proof-lines in the resulted $\mathbb{P}_c(\mathbb{Z})$-proof obtained after division elimination, are derived syntactically correct from previous lines according to the rules of $\mathbb{P}_c(\mathbb{Z})$.

We start with a simple example to illustrate the main idea in this and the next step, and then describe the current step in more detail.

Example: Recall that the theory expresses only the syntactic correctness of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs (due to the fact that we cannot verify that division gates do not compute the zero polynomial). In $\mathbf{V}^0$ we can reason as follows about division elimination. Start with the following $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof: $x \cdot x^{-1} = 1$.

Then, apply the linear transformation $x \mapsto 1 - x$ which yields $(1-x)\cdot(1-x)^{-1}$. Substitute $1-x$ by the term $\mathrm{Inv}_k(1-x)$ defined as:

$$\mathrm{Inv}_k(1-x) := 1 + x + x^2 + \cdots + x^k,$$

which serves to "approximate" the inverse of $1-x$ up to degree k, in the sense that $(1-x)\cdot\mathrm{Inv}_k(1-x) = 1 - x^{k+1}$. For a circuit F denote by $F^{(i)}$ the syntactic-homogeneous component of degree i of F, which computes the sum of all (syntactic-)degree i monomials in F. Then, $\mathbf{V}^0$ can prove the following statement:

> *"Let $k \geq 1$ be a natural number. Given $x \cdot x^{-1} = 1$, substitute $1-x$ for x, and then substitute the circuit $\mathrm{Inv}_k(1-x)$ for $(1-x)^{-1}$. Assuming $(1-x)^{(0)} = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof, there exist $\mathbb{P}_c(\mathbb{Z})$-proofs of syntactic-degree at most k for the following equations:*
>
> $$((1-x)\cdot\mathrm{Inv}_k(1-x))^{(0)} = 1,$$
> $$((1-x)\cdot\mathrm{Inv}_k(1-x))^{(i)} = 0, \quad \textit{for } 1 \leq i \leq k."$$

We now describe how to eliminate division gates in more detail. Similar to $\mathrm{coeff}(F)$, the use of $\mathrm{Inv}_k(F)$, for a circuit F, involves using the inverse of the *constant term of* F, namely, the inverse of $F^{(0)}$. This is why we need to make sure that the only invertible ring element to be used is 1 (and thus it has an inverse in $\mathbb{Z}$). For this purpose we show that the assignment of identity matrices to the matrix-entry variables $X = \{x_{ij}\}, Y = \{y_{ij}\}$ and $Z = \{z_{ij}\}$, for $i, j \in [n]$, in π_3 will result in all division gates u^{-1} computing polynomials with a constant term 1 (though this statement is not expressed in the theory).

Assuming for simplicity that r_i (for $i \in J$) are all the variables appearing in π_3 and b is the assignment of identity matrices to the variables in π_3, substitute in π_3 the term $(b_i - w_i)$ for each r_i (for all $i \in J$) denoting the obtained proof by π_3'. Then, by our assumption about identity matrices assignment, the all zero assignment $\overline{0}$ to the w_i variables in π_3' does not nullify any division gate in π_3'. Furthermore, we show that under this assignment every division gate provably in $\mathbb{P}_c^{-1}(\mathbb{Z})$ computes the polynomial 1. Therefore, in the theory, we construct this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π_3' (which is simply a substitution instance of π_3).

As exemplified above, let $\mathrm{Inv}_n(H)$ be the truncated power series of H^{-1} over the point determined by the identity matrices to the variables of the entries of the matrices X, Y, Z. Loosely speaking, this truncated power series serves as the inverse polynomial of H "up to the nth power". Specifically, $\widehat{H} \cdot \widehat{\mathrm{Inv}_n(H)} = 1 + [\text{terms of degree} > n]$ (note again that $\mathbf{VNC}^2$ cannot necessarily prove this equality, since general evaluation of (unrestricted depth) algebraic circuits is not known to be defined in the theory). For every circuit C with a top division gate H^{-1}, $\mathbf{V}^0$ proves there exists a corresponding division-free circuit C', obtained by replacing the division gate H^{-1} in C by $\mathrm{Inv}_n(H)$.

Let π_4 be the corresponding division-free proof-sequence obtained from π_3' by replacing every circuit with the corresponding division-free circuit as above. By itself π_4 is not a legal $\mathbb{P}_c(\mathbb{Z})$-proof, since the axiom of division in $\mathbb{P}_c^{-1}(\mathbb{Z})$ does not translate into an axiom in $\mathbb{P}_c(\mathbb{Z})$. In other words, the axiom D of division: $F \cdot F^{-1} = 1$ (provided that for every division node u^{-1} in F^{-1}, including F^{-1} itself, u does not compute the zero polynomial; see Definition 3.7), translates into $F \cdot \mathrm{Inv}_n(F) = 1$, which is neither a legal axiom, nor a true identity (since $F \cdot \mathrm{Inv}_n(F) = 1 + [\text{terms of degree} > n]$). We fix this problem as follows: first, we break this equation into its lower $n+1$ syntactic homogeneous components, thereby also getting rid of terms of syntactic-degree $> n$. Second, we will need to construct explicitly $\mathbb{P}_c(\mathbb{Z})$-proofs of $(F \cdot \mathrm{Inv}_n(F))^{(0)} = 1$ (using the notion of *provably good division gates*; see Section 8, as well as Lemma 6.1).

Step 5 is shown in details in section 8.

Step 6: Eliminating high degrees. Here we eliminate the high syntactic-degree ($> n$) parts in the circuits appearing in π_4 (these high syntactic-degree circuits appear in the middle of the proof, and not in the identities proved). This is done by homogenizing the proof π_4. Specifically, we show a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ that receives an algebraic circuit G of syntactic-degree k and converts it into a sum of $k+1$ syntactic-homogeneous circuits $\sum_{i=0}^{k} G^{(i)}$ (computing the same polynomial), in which every node is labeled with an *upper bound* on its syntactic-degree. We show that for our purposes it is enough to work with upper bounds on syntactic-degrees rather than the syntactic-degrees themselves.

More generally, we show a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ that given a $\mathbb{P}_c(\mathbb{Z})$-proof of an equation $F = G$ of syntactic-degree n, decomposes the proof into $n+1$ $\mathbb{P}_c(\mathbb{Z})$-proofs of $F^{(i)} = G^{(i)}$, for $i = 0, \ldots, n$, each proof having syntactic-degree at most i. Combining these proofs gives a low syntactic-degree version of π_4.

This also fixes the problem caused by division elimination described at the end of the previous step. We thus obtain a $\mathbb{P}_c(\mathbb{Z})$-proof, denoted π_5, of equations (5) and (6), where in these two equations the determinant is written as $\mathrm{Det}^{\#}_{Taylor}$.

See Section 9 for more details.

Step 7: Balancing algebraic circuits in the theory. We $\mathbf{\Sigma}_1^B$-define in $\mathbf{VNC}^2$ a function that receives an algebraic circuit C with size s and a number d which stands for an upper bound on the syntactic-degree of C, and outputs a circuit denoted $[C]$ computing $\widehat{C}$ with depth $O(\log s \cdot \log d + \log^2 d)$ and size $\mathrm{poly}(s, d)$. As mentioned before this $\mathbf{FNC}^2$-algorithm provides an $\mathbf{FAC}^0$-implementation of most parts of the classic Valiant *et al.* [VSBR83] algorithm, combining it with ideas from the Miller *et al.* [MRK88] algorithm and usages of matrix powering (which then entails working in $\mathbf{VNC}^2$).

More generally, we show a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{VNC}^2$ that receives a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ with syntactic-degree d, and outputs a $\mathbb{P}_c(\mathbb{Z})$-proof of $[F] = [G]$ in which every circuit is of depth $O(\log s \cdot \log d + \log^2 d)$ and the size of the proof is $\mathrm{poly}(s, d)$.

Applying this function to π_5, we obtain a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{VNC}^2$, which given n in unary outputs a depth $O(\log^2 n)$ $\mathbb{P}_c(\mathbb{F})$-proof π_6 of the determinant identities (5), (6), where the determinant in the two identities is replaced by the appropriate balanced division free circuit of syntactic-degree n computing the determinant, denoted $\mathrm{Det}_{balanced}$. Note that the theory will now express the fact that the PI-proof obtained is indeed a legitimate PI-proof.

See Section 10 for more details.

Step 8: Applying the reflection principle. We now reason in $\mathbf{VNC}^2$ as follows: for every n and every pair of matrices A, B over $\mathbb{Z}$ of dimension $n \times n$, by the definition of the function DET in the theory, $\mathrm{DET}(AB)$, $\mathrm{DET}(A)$ and $\mathrm{DET}(B)$ equals the value of applying the evaluation function to the circuit $\mathrm{Det}_{balanced}$ with the input assignment AB, A, B, respectively.

By the arguments above, there exists a depth $O(\log^2 n)$ $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Det}_{balanced}(XY) = \mathrm{Det}_{balanced}(X) \cdot \mathrm{Det}_{balanced}(Y)$ for the two symbolic matrices X, Y of dimension $n \times n$. But by the reflection principle for depth $O(\log^2 n)$ $\mathbb{P}_c(\mathbb{Z})$-proof from Theorem 4.1 this means that for every input matrices over $\mathbb{Z}$, $\mathsf{Det}_{balanced}(AB) = \mathsf{Det}_{balanced}(A) \cdot \mathsf{Det}_{balanced}(B)$. We therefore conclude $\mathrm{DET}(AB) = \mathrm{DET}(A) \cdot \mathrm{DET}(B)$.

The same argument applies to the proof of the determinant identity (6), when using a symbolic triangular (lower or upper) matrix and then using Theorem 4.1.

5 Encoding Circuits and PI-Proofs in the Theory

Here we explain how to encode algebraic circuits and PI-proofs in the theory. Specifically, in Section 5.1.2 we describe the circuit $\mathsf{Det}_{circ^{-1}}$, namely a circuit with division for the determinant. In Section 5.1.3 we explain how to construct $\mathsf{Det}_{circ^{-1}}$ in $\mathbf{V}^0$. Finally, in Section 5.2.1 we complete Step 1 (in Section 4) in the construction of the proof in the theory: we construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities, where the determinant is written as $\mathsf{Det}_{circ^{-1}}$. The construction in the theory is syntactic in nature, and is done in $\mathbf{V}^0$ using the $\mathbf{\Sigma}_0^B$-**COMP** axiom.

5.1 Encoding Circuits

In order to talk about algebraic circuits, Boolean circuits and PI-proofs in the theory we need to fix an encoding scheme for these objects. Basically, $\mathbf{VNC}^2$ (in fact, already $\mathbf{V}^0$) is rich enough to let us encode syntactic objects in a rather natural way. Since every uniform $\mathbf{AC}^0$ function is definable in $\mathbf{V}^0$ we can assume basic encoding functions to be defined in the theory.

We show below how to construct $\mathsf{Det}_{circ^{-1}}$ in the theory. Encoding and constructing PI-proofs in the theory follows similar lines, we will not always define all the encoding details explicitly when these objects are already constructible in $\mathbf{V}^0$.

5.1.1 Encoding of Algebraic Circuits in the Theory

Algebraic circuits are encoded using strings in the theory as follows: (i) a string of nodes V (in which we assign natural numbers to nodes; this is convenient for our encoding schemes); (ii) a string of gates G, where each gate is a natural number interpreted as a pair of natural numbers (v, t) (using the pairing function) where v is a node in V and t is a natural number that expresses that the gate v is either $+$, $\times$ or $(\cdot)^{-1}$ (plus gate, times gate or a division gate, respectively) or is the ith input (the first two connectives are binary and the third is unary); (iii) a two-dimensional string of input gates I, where, if the first (lsb) of the ith string is 0, the ith string of I encodes a variable x_j, and otherwise it is a binary string representing an integer scalar—the index j of an input variable x_j is represented using the binary representation of j; and finally (iv) a string of directed edges E between two nodes, where $(u, v) \in E$ means that there is an incoming edge to $v \in V$ emanating from $u \in V$.

We show in Section 11.1 that there is a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{VNC}^2$ that converts a non-layered algebraic circuit into a layered circuit in $\mathbf{VNC}^2$, where a layered circuit is a circuit in which each node belongs to a specific layer and nodes in layer i have an outgoing edge only to nodes in layer $i + 1$. This will enable us to convert algebraic circuits to layered Boolean circuits as required by the evaluation axiom of $\mathbf{VNC}^2$ (Definition 3.4).

5.1.2 Circuit with Division for the Determinant

First we need to define the determinant circuit *with division* denoted $\mathsf{Det}_{circ^{-1}}$. Similar to [HT15], this is done using block-wise inversion and Schur complement and can be viewed as performing a block Gaussian elimination: by considering the symbolic matrix $X = \{x_{ij}\}_{i,j\in[n]}$, consisting of n^2 distinct variables, defining the matrix inverse X^{-1} of X and then, by partitioning X into blocks, we formulate a recursive definition of the determinant, using matrix inverse.

Formally, we define an $n \times n$ matrix X^{-1} whose entries are circuits with divisions, computing the inverse of X, as follows:

1. If $n = 1$, let $X^{-1} := (x_{11}^{-1})$.
2. If $n > 1$, write X as follows:
$$X = \begin{pmatrix} X_1 & v_1^t \\ v_2 & x_{nn} \end{pmatrix}, \tag{7}$$
where $X_1 = \{x_{ij}\}_{i,j\in[n-1]}$, $v_1 = (x_{1n}, \dots, x_{(n-1)n})$ and $v_2 = (x_{n1}, \dots, x_{n(n-1)})$. Assuming we have constructed X_1^{-1}, let the *Schur complement* be defined as
$$\delta(X) := x_{nn} - v_2 X_1^{-1} v_1^t\,. \tag{8}$$
Since $\delta(X)$ computes a single non-zero rational function, $\delta(X)^{-1}$ is well-defined. Finally, let
$$X^{-1} := \begin{pmatrix} X_1^{-1}\left(I_{n-1} + \delta(X)^{-1} v_1^t v_2 X_1^{-1}\right) & -\delta(X)^{-1} X_1^{-1} v_1^t \\ -\delta(X)^{-1} v_2 X_1^{-1} & \delta(X)^{-1} \end{pmatrix}. \tag{9}$$

The circuit $\mathsf{Det}_{circ^{-1}}(X)$ is defined as follows:

1. If $n = 1$, let $\mathsf{Det}_{circ^{-1}}(X) := x_{11}$.
2. If $n > 1$, partition X as in (7) and let $\delta(X)$ be as in (8). Let
$$\mathsf{Det}_{circ^{-1}}(X) := \mathsf{Det}_{circ^{-1}}(X_1) \cdot \delta(X) = \mathsf{Det}_{circ^{-1}}(X_1) \cdot (x_{nn} - v_2 X_1^{-1} v_1^t)\,. \tag{10}$$

The definition in (9) should be understood as a circuit with n^2 outputs which takes $X_1^{-1}, v_1, v_2, x_{nn}$ *as inputs* and moreover, such that the inputs from X_1^{-1} *occur exactly once*. Altogether, we obtain a polynomial-size circuit for X^{-1} and the determinant function of X. The circuits obtained are unbalanced, have division gates and are of exponential syntactic-degree (see Definition 3.6). The fact that $\mathsf{Det}_{circ^{-1}}(X)$ indeed computes the determinant (as a rational function) stems, e.g., from the fact (shown in this work, or in [HT15]) that $\mathbb{P}_c^{-1}(\mathbb{Z})$ can prove the two identities that characterize the determinant. That X^{-1} computes matrix inverse is also proved in the theory.

5.1.3 Constructing the Circuit $\mathsf{Det}_{circ^{-1}}$ in $\mathbf{V}^0$

Here we show a $\mathbf{\Sigma}_0^B$-definable in $\mathbf{V}^0$ function, denoted $\mathsf{write}_{X^{-1}}(n)$, that outputs the multi-output circuit X^{-1} ((9) above) given as input a unary integer n. From X^{-1}, in a similar manner we can construct (the single-output circuit) $\mathsf{Det}_{circ^{-1}}(X)$ using (10) above. Note that the definition in (9) is implicitly a construction that uses $\mathbf{\Sigma}_1^B$-induction (that is, the number induction axiom as in Proposition 3.2 in which we use $\mathbf{\Sigma}_1^B$ instead of $\mathbf{\Sigma}_0^B$): given that there exists a circuit for X_1^{-1} of dimension $(n-1) \times (n-1)$, we construct X^{-1} of dimension $n \times n$. However, since we do not have in $\mathbf{V}^0$, nor in $\mathbf{VNC}^2$, the number induction axiom for $\mathbf{\Sigma}_1^B$-formulas we will need to construct the circuit "syntactically" using only the $\mathbf{\Sigma}_0^B$-**COMP** axiom by utilizing a natural encoding scheme. This idea and similar encoding is then used in the sequel to construct all the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in the theory. For getting the final division free $\mathbb{P}_c(\mathbb{Z})$-proofs using homogenization and balancing we need to consider different arguments, including the axioms of $\mathbf{VNC}^2$, e.g., to be able to compute matrix powering (see Sections 9, 10).

The circuit for X^{-1} is encoded as follows. It is a multi-output circuit. The string V encodes the nodes in the circuit, as natural numbers, where a node number is interpreted as a tuple of natural numbers as shown below (using the $\mathbf{\Sigma}_0^B$-definable in $\mathbf{V}^0$ tupling number function). For each *inductive level* $d = 1, \dots, n$ in the inductive definition of X^{-1} in (9), corresponding to the construction of a $d \times d$ inverse matrix, we have a set of nodes $(d, (i, j), \ell) \in V$, each interpreted as a three-tuple of numbers where the second number is a pair of numbers in itself. In $(d, (i, j), \ell) \in V$, the pair (i, j), for $i, j \in [d]$, is an entry in a $d \times d$ matrix, meaning that the node $(d, (i, j), \ell)$ is part of a sub-circuit of X^{-1} that computes the (i, j)th entry in the dth inductive-step; ℓ is the running index of the nodes in that part, where $\ell = 0$ iff the node is what we consider an *output node of the given level* d *and the given entry* (i, j). Nodes of the form $(0, (i, j), 0)$ stand for the *input node* corresponding to the variable x_{ij} (or scalar) in the input string I.

For example, $(1, (1, 1), 0)$ is the node computing x_{11}^{-1}, because the first coordinate $d = 1$ refers to "inductive" level 1 in (9), the second coordinate is $(1, 1)$, meaning the $(1, 1)$-entry from the circuit computing the inverse of x_{11}, and the last coordinate is 0, meaning this is the *output* node of the inverse of x_{11}. Note that we use the numbers on the nodes in V to denote information on the structure of the circuit, namely information about the edges in E and whether a gate is an input node (this information is expressed also in G). This makes the construction of the corresponding E and G easier.

Additionally, we have a string G of natural numbers, each interpreted as a four-tuple encoding the gate-type of each node in V, excluding the input nodes $(0, (i, j), 0)$. That is, $(d, (i, j), \ell, g) \in G$ means that node $(d, (i, j), \ell) \in V$ is of type $+$ if $g = 0$, $\times$ if $g = 1$ and division $(\cdot)^{-1}$ if $g = 2$, and an input variable x_{ij} if $g = (i, j)$, where, again, $(\cdot, \cdot)$ is the pairing function (note that the pairing function (cf. [CN10]) is monotone increasing and that $(1, 1) > 2$, so we can distinguish between the case of an arithmetic gate and an input gate). Finally, the string E encodes the edges between nodes in the circuit. That is, $(d, (i, j), \ell, d', (i', j'), \ell')$ means that there is a directed edge from node $(d, (i, j), \ell)$ to node $(d', (i', j'), \ell')$.

Using the above encoding scheme it is possible now to bit-define the string function $\mathsf{write}_{X^{-1}}$ as a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$. We only need to construct, given some level d, (i, j), the sub-circuits whose nodes will be $(d, (i, j), \ell)$, for some ℓ, according to the definition in (9). We will use the following notation and functions in the theory.

Notations and basic functions for constructing sub-circuits. Let F be some "primitive" arithmetic function, such as inner product of two n-element vectors over the integers, or one of the functions in (9) used to define a minor or the matrix inverse X^{-1}, such as $\delta(X)^{-1}$ (we use the term *minor* to refer to a sub-matrix). We will denote by $\mathsf{write}_F(n, d, \ell, \overline{I}, \overline{O})$ the following string function: the input to this string function are $\overline{I}$, serving as the input nodes to the circuit and $\overline{O}$ the output nodes of the circuit for F, d is the index "level" (used to record the induction-level of the inductive circuit constructions as in (9)) and ℓ is the "running index" of a node in a given level d, and n stands for the "dimension" of the operation defined by F (e.g., inner product of vectors of size n, or matrix product of two $n \times n$ matrices has dimension n). The output is a string, but we abuse notation and assume it is *three* separate strings encoding the (output) circuit, for simplicity, as follows: E, V, G as described above.

More formally, we define $\mathsf{write}_F(n, d, \ell, \overline{I}, \overline{O}) = (E, V, G)$ as follows (similar to the above notation): V is a string describing the vertices in an algebraic circuit. E is a string describing the edges between vertices in V. G is a string describing the gate-types of vertices in V. Every vertex is of the form $(d, (i, j), \ell)$ with d the recursive level in the definition of X^{-1} in (9), (i, j) means that the node is in the (i, j)'s part of the definition of X^{-1}, and ℓ is the running index of nodes in the same level d and same part (i, j), where $\ell = 0$ iff the node is an output node of *that level* d (it is not necessarily the output node of the whole circuit). Assume that $F(\overline{I})$ is some algebraic function with m_0 integer inputs $\overline{I}$ and m_1 integer outputs $\overline{O}$. Then, we supply $\mathsf{write}_F(n, d, \ell, \overline{I}, \overline{O})$ with the node indices (as encoded in V) to be used as input nodes and output nodes for the (sub-)circuit computing F. Here is an example of the input and output nodes of F_1.

Example: Consider the multi-output circuit $F_1 := X_1^{-1}(I_{n-1} + \delta(X)^{-1} v_1^t v_2 X_1^{-1})$ from (9). We want to construct the circuit F_1 in $\mathbf{V}^0$. Note that F_1 is a recursive function in the sense that it uses as inputs the outputs X_1^{-1} which are computed in the previous recursive level $d-1$, together with the "new" nodes in row d and column d in X. Therefore, the inputs of F_1 are the following nodes: $(d-1)^2$ input nodes for X_1^{-1}, $2(d-1)$ input nodes for v_1^t and v_2, and finally one input node x_{dd} (needed for computing $\delta(X)^{-1}$), which sums up to d^2 input nodes in total. The number of output nodes for F_1 is $(d-1)^2$, as it defines a $(d-1) \times (d-1)$ minor of X^{-1}. Therefore, in our encoding scheme, the input nodes for F_1 (viewed as a $d \times d$ matrix) are:

$$\begin{pmatrix} (d-1,(1,1),0) & \dots & (d-1,(1,d-1),0) & (0,(1,d),0) \\ \vdots & \ddots & \vdots & \vdots \\ (d-1,(d-1,1),0) & \dots & (d-1,(d-1,d-1),0) & (0,(d-1,d),0) \\ (0,(d,1),0) & \dots & (0,(d,d-1),0) & (0,(d,d),0) \end{pmatrix}$$

and the output nodes (viewed as a $(d-1) \times (d-1)$ matrix) are:

$$\begin{pmatrix} (d,(1,1),0) & \dots & (d,(1,d-1),0) \\ \vdots & \ddots & \vdots \\ (d,(d-1,1),0) & \dots & (d,(d-1,d-1),0) \end{pmatrix}.$$

Let F_2, F_3, F_4 be the other three functions used in the definition of X^{-1} (9) (for the other simpler three minors). We shall define similarly write_{F_i} functions for these F_i's.

To show that $\mathsf{write}_{X^{-1}}$ is a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ we need to demonstrate how to bit-define this function using a $\mathbf{\Sigma}_0^B$-formula (see Definition A.4 in Section A for bit-definability). In our case we need to show how to bit-define $\mathsf{write}_{v \cdot u}$ using a $\mathbf{\Sigma}_0^B$-formula, given two n-element vectors of integers v, u representing *nodes* in the circuit. This is quite easy to do: simply output a binary tree with the appropriate plus and products nodes, and plug the input nodes v, u to the leaves accordingly. We provide a proof of this in the Appendix A.3. Here we denote the nodes in the circuit computing the inner-product $v \cdot u$ in level d using the running index: every node excluding the output nodes of this level d (which are unique for every fixed d and (i, j)) has a different running index $\ell > 0$, namely has the tuple $(d, (i, j), \ell)$ associated with level d and the (i, j) entry in the matrix computed at level d.

Similarly, we have $\mathbf{\Sigma}_0^B$-formulas for constructing other formulas like write_{vA} and write_{Av^t}, given the input nodes for an $n \times n$ matrix A, and the input nodes for an n-elements vector v. Also, given a node z it is immediate to output a circuit computing z^{-1} or $-z$, and given two matrices A, B (i.e., $2n^2$ nodes) it is easy to give a $\mathbf{\Sigma}_0^B$ bit-definition of write_{A+B} in $\mathbf{V}^0$.

Now that we set up the notation and the functions for constructing sub-circuits, we can bit-define with a $\mathbf{\Sigma}_0^B$-formula $\mathsf{write}_{X^{-1}}$ in $\mathbf{V}^0$ as follows. First, for $i = 1, \ldots, 4$, define $\mathrm{Inp}_{F_i}(d)$ and $\mathrm{Out}_{F_i}(d)$ to be the string functions that output the sequence of input and output nodes of the dth recursive level of X^{-1} for each of the F_i's, respectively, as shown for F_1 in the example above. They are all $\mathbf{\Sigma}_0^B$-definable string-functions in $\mathbf{V}^0$. The bit-definition of $\mathsf{write}_{X^{-1}}$ is

$$\begin{aligned}
&\mathsf{write}_{X^{-1}}(n)(i) \equiv \\
&\qquad \exists 2 \le d \le n \Big(\exists 1 \le j \le 4, \mathsf{write}_{\mathsf{level}(X^{-1})} \Big(n, d, 1, \mathrm{Inp}_{F_j}(d), \mathrm{Out}_{F_j}(d)\Big)\,(i)\Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \vee \,\mathsf{write}_{x_{11}^{-1}}\,(n, 1, 0, ((0, (1, 1), 0)), ((1, (1, 1), 0)))\,(i)\big)\,,
\end{aligned}$$

where $\mathsf{write}_{\mathsf{level}(X^{-1})}(n, d, \ell, \overline{I}, \overline{O})$ outputs (E, V, G) encoding a (sub-)circuit that is the dth inductive level of X^{-1}, and $\mathsf{write}_{x_{11}^{-1}}\big(n, 1, 0, ((0, (1, 1), 0)), ((1, (1, 1), 0))\big)$ is the string function that outputs the encoding of the circuit "x_{11}^{-1}".

In the sequel we will be less formal about encoding in $\mathbf{V}^0$ circuits in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in the theory.

5.2 Encoding and Witnessing PI-proofs

Recall that $\mathbb{P}_c^{-1}$ is a PI-proof system with division gates and with the division axiom D added (Section 3.6). Also, recall from Step 5 (Section 4) that we are not going to express in the theory the full correctness of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in the sense that the theory will express only the syntactic-correctness of a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (see definition in Section 3.6).

$\mathbb{P}_c^{-1}(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-proofs are encoded as a two dimensional array S (that is, a string encoding an array of strings), in which the ith string $S^{[i]}$, also called *the ith row of S*, is the ith equation in the proof, written as a pair of circuits with division (and where circuits encoding is done as described in Section 5.1.3).

Furthermore, the encoding of PI-proofs will always consist of additional ***witnesses for syntactic correctness***, as follows:

1. Each row $S^{[i]}$ specifies whether it is an axiom, and if not specifying the proof-lines from which it was derived as well as the rule by which it was derived.

2. For the four rules R1-R4, we have the following convention to witness the correctness of applying the rule: the encoding of the circuits F, G, H and F_1, F_2, G_1, G_2 in the antecedent and consequence of the rules are identical, that is, with the same node numbers in their respective sets of nodes V. In other words, the respective strings encoding $F, G, H, F_1, F_2, G_1, G_2$ in the antecedent and consequence are identical.

3. For the axioms A1-A9, and the axiom D in $\mathbb{P}_c^{-1}$, the circuits F, G, H in both sides of the equations are encoded identically, as in part 2 above.

4. The scalar axioms A10 is encoded as a circuit with scalar inputs as usual. Only that we will not verify their correctness, as this will not be needed.

5. The axioms C1, C2 needs a special treatment. Consider $F_1 \oplus F_2 = F_1 + F_2$, and let V be the set of node (numbers) belonging to $F_1 \oplus F_2$. Every node $u \in V$, excluding the plus at the root, occurs as two different nodes u_1, u_2 in $F_1 + F_2$. To witness this rule we add a string that stores (as an array of number pairs) the mapping from the nodes of F_1 in $F_1 + F_2$ to the nodes of F_1 in $F_1 \oplus F_2$, and similarly for F_2. Given such a witness it is immediate to verify (with a $\mathbf{\Sigma}_0^B$-formula) that the C1 axiom is applied correctly[3]. C2 is treated similarly.

When we talk about a PI-proof in the theory, unless otherwise stated, we assume that the proof encoding includes its witness for syntactic correctness as above. When we talk about $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs specifically, we shall say that "the theory proves the existence of a syntactic correctness $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof" to mean that the proof is encoded with the witness as above, only that we emphasize that the proof and witness only ensure syntactic correctness (since division by zero may occur in such proofs).

5.2.1 Existence of Proofs with Division for the Determinant Identities

Here we complete Step 1 of the argument (Section 4), by demonstrating that there is a $\mathbf{\Sigma}_0^B$-formula with n as a number parameter that defines the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the two determinant identities (5), (6).

Proposition 5.1 (in $\mathbf{V}^0$). *Given a positive natural number n there exists a syntactic correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities (5) and (6) for $n \times n$ matrices, where the determinant in (5) and (6) is written as the division free circuit $\mathsf{Det}_{circ^{-1}}(A)$, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z.*

Proof. It is enough to show that there exists a $\mathbf{\Sigma}_0^B$-formula denoted $\mathsf{write}_{\mathbb{P}_c^{-1}(\mathbb{Z})\vdash(5),(6)}(n, W)$, that holds iff W is the syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the identities (5) and (6) for $n \times n$ matrices. We construct this $\mathbf{\Sigma}_0^B$-formula defining the polynomial-size $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof demonstrated in [HT15, Section 7.1] for the identities (5), (6).

Given a pair of $n \times n$ matrices X, Y, the expressions $XY = A$ in the context of $\mathbb{P}_c^{-1}$ is an abbreviation of a sequence of n^2 equalities between the appropriate entries. Note however that whereas before we treated X^{-1} as a single multi-output circuit, here X^{-1} is a set of n^2 separate circuits for each of the entries in X^{-1} (this is achieved simply by taking the same single multi-output circuit for X^{-1} as before, and duplicating each of the output gates together with its sub-circuit).

We exemplify our $\mathbf{V}^0$-construction with the proof of $X \cdot X^{-1} = I_n$ below. This can potentially be constructed by induction on n. However, similar to the construction of $\mathsf{Det}_{circ^{-1}}$ in the theory (Section 5.1.3), we cannot use (number) induction on $\mathbf{\Sigma}_1^B$-formulas in $\mathbf{VNC}^2$, and thus we need to work out an encoding of the proof that can be constructed using a $\mathbf{\Sigma}_0^B$-formula.

If $n = 1$, we have $x_{11} \cdot x_{11}^{-1} = x_{11}^{-1} \cdot x_{11} = 1$ which is a $\mathbb{P}_c^{-1}$ axiom. Otherwise, let $n > 1$ and X be as in (7). We want to construct a polynomial-size proof of $X \cdot X^{-1} = I_n$ from the assumption $X_1 \cdot X_1^{-1} = I_{n-1}$.

[3] One can also use an ***NL*** algorithm, formalizable in $\mathbf{VNC}^2$ (since ***NL*** $\subseteq \mathbf{NC^2}$), to verify that both sides of the axiom C1 are different representation of the same circuit. However, it will not be easy to prove for our $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs that they are correct with such a predicate of correctness.

Denote $a := \delta(X)$, and $A := I_{n-1} + a^{-1}v_1^t v_2 X_1^{-1} - a^{-1}v_1^t v_2 X_1^{-1}$, and $B := v_2 X_1^{-1} + a^{-1}(v_2 X_1^{-1} v_1^t - x_{nn})v_2 X_1^{-1}$. Taken verbatim from [HT15, Proposition 7.2], using some rearrangements, and the definition of a, we have:

$$
\begin{aligned}
X \cdot X^{-1} &= \begin{pmatrix} X_1 & v_1^t \\ v_2 & x_{nn} \end{pmatrix} \cdot \begin{pmatrix} X_1^{-1}(I_{n-1} + a^{-1}v_1^t v_2 X_1^{-1}) & -a^{-1}X_1^{-1}v_1^t \\ -a^{-1}v_2 X_1^{-1} & a^{-1} \end{pmatrix} \\
&= \begin{pmatrix} A & -a^{-1}v_1^t + a^{-1}v_1^t \\ B & a^{-1}(-v_2 X_1^{-1} v_1^t + x_{nn}) \end{pmatrix} \\
&= \begin{pmatrix} I_{n-1} & 0 \\ v_2 X_1^{-1} - a^{-1} a v_2 X_1^{-1} & a^{-1}a \end{pmatrix} \\
&= \begin{pmatrix} I_{n-1} & 0 \\ 0 & 1 \end{pmatrix}.
\end{aligned}
$$

To encode this we do the following: for every $k = 1, \ldots, n$, we use $\mathbf{\Sigma}_0^B$-**COMP** to define the above $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof sequence, where k replaces n, namely, we construct the above $\mathbb{P}_c^{-1}(\mathbb{Z})$-sequence for $X \cdot X^{-1}$ of dimension $k \times k$ for each $k = 1, \ldots, n$. Such a sequence can be $\mathbf{\Sigma}_0^B$-defined similar to the encoding shown in Section 5.1.3: since we simply need to compose primitive constructions of matrix multiplications (which are written as k^2 separate equations for each entry), dot products, plus and minus, construction of the identity matrix of dimension k, and X_1^{-1} (which we encoded explicitly in Section 5.1.3). The proof sequence for k uses the proof of $X_1 \cdot X_1^{-1} = I$, where X_1 has dimension $(k-1) \times (k-1)$, and this is done by specifying $X_1 \cdot X_1^{-1} = I$ as a previous proof-line from which we derive our new proof-line (in particular, no (string-)induction is needed for this).

Having $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $X \cdot X^{-1} = I_n$ and $X^{-1} \cdot X = I_n$, we can now proceed to construct the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of (5), (6) in the theory. This is done in exactly the same manner, by formalizing directly in $\mathbf{V}^0$ the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof shown in [HT15, Section 7.1]. However, for constructing in the theory the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (6) in [HT15, Section 7.1] we make sure that the diagonal elements of the triangular matrix Z are in fact *variables* and not scalars. This is needed for us to be able to eliminate division gates later: to eliminate division gates we need every division gate to be provably good under the identity matrix assignment (see Section 8), and specifically, these gates cannot compute the zero polynomial. When the diagonal elements are not definable, namely contain division gates that compute the zero polynomial, we will not be able to establish that they are provably good. □

6 From a Rational Function to the Determinant as a Polynomial

Here we complete Step 2 (Section 4) in the construction of the proof in the theory: we construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities (5), (6), where the determinant is written as Det_{Taylor}, which is a polynomial-size algebraic circuit without division gates for the determinant (of exponential syntactic-degree). We first need rst to provide the preliminaries for division elimination and truncated Taylor expansions that we shall use in this and the next sections.

6.1 Preliminaries for Division Elimination

Let F be a *division-free* circuit and let $F^{(0)}$ be the syntactic homogeneous division free circuit that computes the constant term of F at the point $\overline{0}$; we show that such a syntactic homogeneous circuit can be constructed in $\mathbf{V}^0$ in Section 9. We define the circuit $\mathrm{Inv}_k(F)$ that will serve as an inverse of F modulo high degree monomials, in the sense that

$$F \cdot \mathrm{Inv}_k(F) = 1 + [\text{monomials of degree greater than } k]\,. \tag{11}$$

Note that because we work over $\mathbb{Z}$ the only way for (11) to hold (for $k > 1$) is when $\widehat{F^{(0)}} = 1$, since if the constant term in F is not 1, no product of F can compute the constant term 1. In general, the division elimination (as in Strassen [Str73]) needs to work over a field and have an inverse element for $\widehat{F^{(0)}}$. But in our application inside the theory we will always have $F^{(0)}$ as a circuit without division, possibly with variables, that computes the polynomial 1, and hence we do not require an inverse of $\widehat{F^{(0)}}$.

Assume that F is good under the zero assignment, in symbols $\widehat{F^{(0)}} = 1$, and define the circuit $\mathrm{Inv}_k(F)$ as

$$\mathrm{Inv}_k(F) := \left(1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k\right),$$

where powers like $(1-F)^k$ are abbreviation of $(1-F)\cdots(1-F)$, k times (written for instance as a logarithmic in k depth circuit; since k will always be polynomial in our application this will be enough for our purposes). The following lemma demonstrates that $\mathrm{Inv}_k(F)$ can *provably* serve as the inverse polynomial of F "up to the k-th degree".

Lemma 6.1 (in $\mathbf{V}^0$). *Let F be a size s circuit without division, let η be a $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(0)} = 1$, and let $k \geq 1$ be a natural number. Then there exists $\mathbb{P}_c(\mathbb{Z})$-proofs of size $s \cdot \mathrm{poly}(k)$ of the following equations, in which every node in every circuit in the proofs appears with its syntactic-degree upper bound (see Definition 9.1):*

$$\begin{aligned} (F \cdot \mathrm{Inv}_k(F))^{(0)} &= 1 && (12)\\ (F \cdot \mathrm{Inv}_k F)^{(i)} &= 0, \quad \textit{for } 1 \leq i \leq k\,. && (13) \end{aligned}$$

Proof. Denote $a = F^{(0)}$. We construct the following simple $\mathbb{P}_c(\mathbb{Z})$-proof sequences. We have $a(1-(1-F)) = aF$, and by assumption that $a = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof η, we get a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = (1-(1-F))$.

By definition $\mathrm{Inv}_k(F) = (1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k)$. By elementary rearrangement, we prove in $\mathbb{P}_c(\mathbb{Z})$:

$$\begin{aligned} F \cdot \mathrm{Inv}_k(F) &= (1-(1-F)) \cdot \left(1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k\right) \\ &= 1 + (1-F) + \cdots + (1-F)^k \\ &\qquad - (1-F) \cdot \left(1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k\right) \\ &= 1 + (1-F) + \cdots + (1-F)^k \\ &\qquad\qquad - (1-F) - (1-F)^2 - \cdots - (1-F)^{k+1} \\ &= 1 - (1-F)^{k+1}\,. \end{aligned} \tag{14}$$

From (14) and Theorem 9.2 in Section 9, we construct the $\mathbb{P}_c(\mathbb{Z})$-proof of

$$(F \cdot \mathrm{Inv}_k(F))^{(0)} = 1 - ((1-F)^{k+1})^{(0)},$$

wherein every node in every circuit in the proof appears with its syntactic-degree upper bound. Specifically, from Theorem 9.2 we have $\mathbb{P}_c(\mathbb{Z})$-proofs of $\left((1-F)^{k+1}\right)^{(0)} = \left((1-F)^{(0)}\right)^{k+1} =$

$(1 - F^{(0)})^{k+1}$, and using η we have that the rightmost term is $(1-1)^{k+1} = 0$, and we conclude (12).

To conclude (13) we proceed as follows. From (14) and Theorem 9.2 we construct $\mathbb{P}_c(\mathbb{Z})$-proofs $(F \cdot \mathrm{Inv}_k(F))^{(i)} = (1-(1-F)^{k+1})^{(i)}$, for all $1 \le i \le k$, with all nodes appear with their syntactic-degree upper bounds. From Lemma 9.3 in Section 9, we prove $(1-(1-F)^{k+1})^{(i)} = 1^{(i)} - ((1-F)^{k+1})^{(i)} = 0 - ((1-F)^{k+1})^{(i)}$.

To construct the $\mathbb{P}_c(\mathbb{Z})$-proof of $((1-F)^{k+1})^{(i)} = 0$, for $1 \le i \le k$, in the theory we use again Lemma 9.3. We omit the details. (Note that since $F^{(0)} = 1$ by assumption, we have $(1-F)^{(0)} = 0$, meaning that all monomials in $(1-F)$ are of positive total degree. Therefore $(1-F)^{k+1}$ can only have monomials of degree greater than k, and so $((1-F)^{k+1})^{(i)} = 0$ is a true identity for all $i \le k$.) □

6.1.1 Extracting the Numerators and Denominators of Circuits with Division

We also need to show how to extract the denominator and numerator of circuits with divisions. For every node v in a circuit F with division we introduce two nodes $\mathrm{Den}(v)$ and $\mathrm{Num}(v)$ that will compute *as polynomials* (that is, they will be circuits with no division) the numerator and denominator of the rational function computed by v, respectively, as follows:

1. If v is an input node of F, let $\mathrm{Num}(v) := v$ and $\mathrm{Den}(v) := 1$.
2. If $v = u^{-1}$, let $\mathrm{Num}(v) := \mathrm{Den}(u)$ and $\mathrm{Den}(v) := \mathrm{Num}(u)$.
3. If $v = u_1 \cdot u_2$, let $\mathrm{Num}(v) := \mathrm{Num}(u_1) \cdot \mathrm{Num}(u_2)$ and $\mathrm{Den}(v) := \mathrm{Den}(u_1) \cdot \mathrm{Den}(u_2)$.
4. If $v = u_1 + u_2$, let $\mathrm{Num}(v) := \mathrm{Num}(u_1) \cdot \mathrm{Den}(u_2) + \mathrm{Num}(u_2) \cdot \mathrm{Den}(u_1)$ and $\mathrm{Den}(v) := \mathrm{Den}(u_1) \cdot \mathrm{Den}(u_2)$.

Let $\mathrm{Num}(F)$ and $\mathrm{Den}(F)$ be the circuits with the output node $\mathrm{Num}(w)$ and $\mathrm{Den}(w)$, respectively, where w is the output node of F. In Section 7 we show that given a circuit with division F, there is a Σ_0^B-definable function in $\mathbf{V}^0$ that constructs the circuit $\mathrm{Num}(F) \cdot (\mathrm{Den}(F))^{-1}$.

6.2 A PI-Proof Reducing the Determinant from Rational Function to Polynomial

Like [HT15], in order to write the determinant as a polynomial instead of a rational function, we will write the determinant as the coefficient of z^n in the Taylor expansion of a certain circuit with division. The coefficients of a Taylor expansion are defined as follows:

Definition 6.2 (Taylor expansion)**.** *Let $F = F(\overline{x}, z)$ be a circuit with division. Define $\mathrm{coeff}_{z^k}(F)$ as a circuit in the variables $\overline{x}$, computing the coefficient of z^k in F, when F is written as a Taylor power series at $z = 0$, in the following way:*

Case 1: *Assume that no division gate in F contains the variable z. Then we define $\mathrm{coeff}_{z^k}(F)$ by induction on the structure of F as follows:*

1. $\mathrm{coeff}_z(z) := 1$ *and* $\mathrm{coeff}_{z^k}(z) := 0$, *if* $k > 1$.
2. *If F does not contain z, then* $\mathrm{coeff}_{z^0}(F) := F$ *and* $\mathrm{coeff}_{z^k}(F) := 0$, *for* $k > 0$.

3. $\mathrm{coeff}_{z^k}(F+G) := \mathrm{coeff}_{z^k}(F) + \mathrm{coeff}_{z^k}(G)$.
4. $\mathrm{coeff}_{z^k}(F \cdot G) := \sum_{i=0}^{k} \mathrm{coeff}_{z^i}(F) \cdot \mathrm{coeff}_{z^{k-i}}(G)$.

Case 2: *Assume that z occurs in the scope of some division gate in F. We let F_0 be the denominator of the rational function computed by F when $z=0$:*

$$F_0 := \big(Den(F)\big)(z/0).$$

Note that F_0 is not necessarily a constant, as it may contain variables different from z. If $\widehat{F_0} = 0$ then coeff *is* undefined. *Assume that $\widehat{F_0} \neq 0$ and denote by G the circuit $(1 - F_0^{-1} \cdot Den(F))$. We let*

$$\mathrm{coeff}_{z^k}(F) := F_0^{-1} \cdot \mathrm{coeff}_{z^k}\left(Num(F) \cdot \left(1 + G + G^2 + \cdots + G^k\right)\right). \tag{15}$$

Note that z does not occur in any division gate inside $\mathrm{Num}(F) \cdot \left(1 + G + G^2 + \cdots + G^k\right)$, and so $\mathrm{coeff}_{z^k}(F)$ is well-defined. In our applications, when using $\mathrm{coeff}_{z^k}(F)$ we will need to make sure that $\widehat{F_0} = 1$, and that we can prove in $\mathbb{P}_c^{-1}(\mathbb{Z})$ that $F_0^{-1} = 1$.

The following are the main properties of coeff_{z^k} that we can use already in $\mathbf{V}^0$ (similar to [HT15]).

Lemma 6.3 (in $\mathbf{V}^0$). 1. *If $F_0, \ldots, F_k$ are circuits* with *division not containing the variable z, then $\mathrm{coeff}_{z^j}\left(\sum_{i=0}^{k} F_i z^i\right) = F_j$ has a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, for each $j \leq k$.*

2. *Assume that F, G are circuits* with division *such that $F = G$ has a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π of size s. Then, $\mathrm{coeff}_{z^k}(F) = \mathrm{coeff}_{z^k}(G)$ has a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of size $s \cdot \mathrm{poly}(k)$, for every natural number k.*

3. *Let F be a circuit* without *division, together with a witness for the syntactic-degree of all nodes in F, where $d(F) = d$. Then $F = \sum_{i=0}^{d} \mathrm{coeff}_{z^i}(F) \cdot z^i$ has a $\mathbb{P}_c(\mathbb{Z})$-proof.* [4]

In both parts 1, 2 above the inverse elements of an integer a is written as a^{-1} (i.e., with an explicit division gate. This will not be a problem for us since in our case $\mathbf{V}^0$ can prove that $a=1$).

Proof. The proofs of parts 1, 2 are almost identical to the proof of Theorem 9.2 in Section 9 for eliminating high syntactic-degrees (namely, homogenizing $\mathbb{P}_c(\mathbb{Z})$-proofs), and we omit the details. The proof of part 3 is given in the appendix (Lemma C.2). □

The determinant as a polynomial. We are now ready to define the circuit computing the determinant as a polynomial. Let

$$\mathsf{Det}_{Taylor}(X) := \mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX)).$$

As explained in Section 4, $\mathsf{Det}_{Taylor}(X)$ is a circuit of polynomial-size in n that computes the determinant polynomial. This is because every variable from X in the circuit $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ occurs in a product with z, and thus $\mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ computes the nth homogeneous part of the determinant of $I_n + X$, which is simply the determinant of X. By the definition of coeff_{z^n}, the circuit $\mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I + zX))$ *contains exactly one inverse gate*, namely the inverse of $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ at the point $z=0$. $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))(z/0)$ has an exponential syntactic-degree which will be dealt with below in Section 6.3.

We can now use Lemma 6.3 to construct the desired $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof:

[4] The only place where we need this part is the proofs of the Cayley-Hamilton theorem in Section 12. For witnesses for syntactic-degree see the appendix Section C.

Lemma 6.4 (in $\mathbf{V}^0$). *Let A be an $n \times n$ symbolic matrix X of distinct variables or the product of two symbolic matrices XY or a triangular $n \times n$ symbolic matrix Z. Then there exists a polynomial-size $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of* $\mathsf{Det}_{Taylor}(A) = \mathsf{Det}_{circ^{-1}}(A)$.

Proof. Using Lemma 6.3 parts 1 and 2 we construct in $\mathbf{V}^0$ the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof demonstrated in [HT15, Proposition 7.9] (see also Lemma 8.6). (Since we do not use part 3 of Lemma 6.3 we do not need to construct a witness for the syntactic-degrees of any circuit in this case.)

□

6.3 Reducing the Syntactic-Degree of the Determinant Polynomial

We need to reduce the syntactic-degree of $\mathsf{Det}_{Taylor}(X) := \mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$, which is exponential as we now explain. Similar reasoning applies to $\mathsf{Det}_{Taylor}(A)$, where A is an $n \times n$ symbolic matrix X of distinct variables or the product of two symbolic matrices XY or a triangular $n \times n$ symbolic matrix Z.

Let $F(X) = F(x_1, , \dots, x_n)$ be a circuit with division in the displayed input variables. Assume that every input variable is now multiplied by a new variable z, to get $F(zx_1, \dots, zx_n)$, which we denote by F'. Consider $\mathrm{coeff}_{z^k}(F') := F_0'^{-1} \cdot \mathrm{coeff}_{z^k}\left(\mathrm{Num}(F') \cdot \left(1 + G + G^2 + \dots + G^k\right)\right)$ for $G := (1 - F_0'^{-1} \cdot \mathrm{Den}(F'))$, and $F_0' := \left(\mathrm{Den}(F')\right)(z/0)$. By induction on the size of F', inspecting the construction of $\mathrm{coeff}_{z^k}(\cdot)$, it is easy to show that every subcircuit in $\mathrm{coeff}_{z^k}(F')$ has a syntactic-degree at most k, excluding occurrences of the subcircuit $F_0'^{-1}$ that has syntactic-degree $d(\mathrm{Den}(F'))$ that may be greater than k.

Accordingly, $\mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ contains occurrences of $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(z/0)$, that are of syntactic-degree greater than n. To remedy this we define $\mathsf{Det}^{\#}_{Taylor}(X)$ as the circuit $\mathsf{Det}_{Taylor}(X)$ in which we replace the subcircuit $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(z/0)$ by the constant 1 (note that indeed $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(z/0)$ computes the polynomial 1):

$$\begin{aligned}\mathsf{Det}^{\#}_{Taylor}(X) := 1 \cdot \mathrm{coeff}_{z^n}\Big(&\mathrm{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))\cdot \\ &\Big(1 + (1 - 1 \cdot \mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))) + (1 - 1 \cdot \mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX)))^2 + \\ &\qquad \dots + (1 - 1 \cdot \mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX)))^n\Big)\Big)\end{aligned} \tag{16}$$

Lemma 6.5 (in $\mathbf{V}^0$). *Let n be a positive natural number and A be an $n \times n$ symbolic matrix X of distinct variables, or the product of two symbolic matrices XY, or a triangular $n \times n$ symbolic matrix Z. Then, there exists a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of* $\mathsf{Det}^{\#}_{Taylor}(A) = \mathsf{Det}_{Taylor}(A)$.

Proof. We assume that $A = X$. The other cases are similar.

By previous constructions we can construct the circuits $\mathsf{Det}^{\#}_{Taylor}(X)$ and $\mathsf{Det}_{Taylor}(X)$ in $\mathbf{V}^0$.

We first construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(z/0) = 1$. We use Theorem 7.1 in Section 7 that will be proved in the sequel. Initially, construct directly a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right))(z/0) = \mathsf{Det}_{circ^{-1}}\left(I_n\right)$, by substituting z by 0 and gradually replacing in the proof $0 \cdot u$ to 0 and $0 + u$ to u, for u any subcircuit. This is done using $\mathbf{\Sigma}_0^B$-**COMP** and our encoding scheme in Section 5.2.[5]

[5] Note that this is not done for an *arbitrary* circuit, namely, we do not know of an $\mathbf{NC^2}$ algorithm that receives a circuit C, such that $\widehat{C} \neq 0$, and discards in such a way every 0 constant in the circuit. We only build a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof that witnesses such a gradual procedure for discarding 0's from the *specific* circuit $(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right))(z/0)$.

Now, construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(I_n) = 1$. This follows from the proof of Lemma 8.3. Another way to construct this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is as follows: using Proposition 5.1 for constructing the proof of $\mathsf{Det}_{circ^{-1}}(Z) = z_{11} \cdots z_{nn}$, when Z is a symbolic triangular matrix: we can construct a syntactically-correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(I_n) = 1$. Now using Theorem 7.1 part (ii), we construct the proof of $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}(I_n + zX)\right)(z/0) = 1$.

Using the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}(I_n + zX)\right)(z/0) = 1$ and the $\mathbf{\Sigma}_0^B$-**COMP** axiom we can show the existence of a $\mathbb{P}_c^{-1}(Z)$-proof of $\mathsf{Det}^{\#}_{Taylor}(X) = \mathsf{Det}_{Taylor}(X)$. This is done by constructing a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof that gradually substitutes each occurrence of the subcircuit $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}(I_n + zX)\right)(z/0)$ in $\mathsf{Det}_{Taylor}(X)$ by the constant 1. □

From Proposition 5.1 and Lemma 6.5 we get:

Corollary 6.6 (in $\mathbf{V}^0$)**.** *Given a positive natural number n there exists a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities* (5) *and* (6) *for $n \times n$ matrices, where the determinant in* (5) *and* (6) *is written as the division free circuit $\mathsf{Det}^{\#}_{Taylor}(A)$, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z.*

7 Bringing Division Gates to the Top

Here we show a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ that receives an algebraic circuit F with division gates and *normalizes* it as shown in Section 6.1.1, that is, converts it into a circuit with a single division gate "at the top", of the form $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1}$ (formally, the output gate is a product gate with one of its child being a division gate). We need to normalize circuits in such a way in order to be able to eliminate division gates: as seen in the next section, we replace the gate G^{-1} with $\mathrm{Inv}_k(G)$; but for $\mathrm{Inv}_k(G)$ to be defined we need G to be a division free circuit (and can be guaranteed only if there is only one division gate at the top).

We wish to show the following:

Theorem 7.1 (in $\mathbf{V}^0$)**.** *(i) If F is a circuit with division, then $F = \mathit{Num}(F) \cdot \mathit{Den}(F)^{-1}$ has a $\mathbb{P}_c^{-1}(\mathbb{F})$-proof. (ii) Let F, G be circuits with division. Assume that $F = G$ has a $\mathbb{P}_c^{-1}(\mathbb{F})$-proof. Then $\mathit{Num}(F) \cdot \mathit{Den}(F)^{-1} = \mathit{Num}(G) \cdot \mathit{Den}(G)^{-1}$ has a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{F})$-proof such that every division gate in every circuit in the proof occurs only at the top.*

To prove this we first argue that division normalization is doable in $\mathbf{AC}^0$. This follows the similar reasoning as shown in Section 5.1.3: although the division normalization procedure as described in Section 6.1.1 is defined by induction on the size of the circuit (and thus implicitly needs $\mathbf{\Sigma}_1^B$-induction) it is uniform enough to be defined $\mathbf{\Sigma}_0^B$-formula (using $\mathbf{\Sigma}_0^B$-**COMP** axiom) as follows.

Every internal node u in the input circuit F is duplicated into two copies denoted $\mathrm{Num}(u)$ and $\mathrm{Den}(u)$. We then construct (in parallel) the division normalized circuit by wiring the nodes $\mathrm{Num}(u)$ and $\mathrm{Den}(u)$ for each u in F as in the original definition: **(i)** If v is an input node of F, let $\mathrm{Num}(v) := v$ and $\mathrm{Den}(v) := 1$; **(ii)** if $u = v + w$ we construct $\mathrm{Num}(u) := \mathrm{Num}(v) \cdot \mathrm{Den}(w) + \mathrm{Num}(w) \cdot \mathrm{Den}(v)$ and $\mathrm{Den}(u) := \mathrm{Den}(v) \cdot \mathrm{Den}(w)$; **(iii)** if $v = u^{-1}$, let $\mathrm{Num}(v) := \mathrm{Den}(u)$ and $\mathrm{Den}(v) := \mathrm{Num}(u)$; and finally **(iv)** if $v = u_1 \cdot u_2$, let $\mathrm{Num}(v) := \mathrm{Num}(v_1) \cdot \mathrm{Num}(v_2)$ and $\mathrm{Den}(v) := \mathrm{Den}(v_1) \cdot \mathrm{Den}(v_2)$.

Since the nodes $\mathrm{Num}(u)$ and $\mathrm{Den}(u)$ are known in advance for every node u, this construction is doable in parallel, and specifically in $\mathbf{FAC}^0$.

The proof of Theorem 7.1 is similar to the proof of Theorem 9.2 that is presented in details in Section 9. We thus omit the details.

From Corollary 6.6 and Theorem 7.1 we get:

Corollary 7.2 (in $\mathbf{V}^0$). *Given a positive natural number n there exists a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities (5) and (6) for $n \times n$ matrices, where the determinant in (5) and (6) is written as the division free circuit $\mathsf{Det}^{\#}_{Taylor}(A)$, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z in which every circuit with division has only a single division gate at the top.*

8 Eliminating Division Gates

In this section we show in detail step 5 (in Section 4) which eliminates in the theory division gates from the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof we obtained in Corollary 7.2.

A $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is said to be *division axiom free* if it does not use the axiom D of division: $F \cdot F^{-1} = 1$. The *identity matrices assignment* ρ is defined to be the assignment of 0 and 1 elements to the variables in (5), (6) such that $X = Y = Z = I_n$. We say that a division gate u is ***provably good under an assignment*** ρ to its variables whenever $u \restriction \rho = 1$ has a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof. In this case we also say that ρ *is provably good for* u. Accordingly, if ρ is provably good for all the division gates in a circuit C we say that ρ *is provably good for* C.

To eliminate division gates we first make sure that every division gate u^{-1} that appears in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good under ρ. This will allow us to carry out the proof in the theory as follows: by induction on circuit size it is easy to see that if every division gate in a circuit F is good under ρ then $\mathrm{Den}(F)$ (Section 6.1.1) is good under ρ as well (however, we cannot use such $\mathbf{\Sigma}_1^B$-induction in the theory to conclude this fact; see below). We then proceed as follows. After normalizing division gates (Section 7) in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof obtained in Corollary 7.2, every circuit with division in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof has a single division gate and is of the form $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1}$. We first linearly shift the variables by ρ, that is, replace every variable r_i in the proof by $\rho(r_i) - w_i$ (the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is still correct under this shift). Denote the shift $r_i \mapsto \rho(r_i) - w_i$ by σ. Then we replace every subcircuit $\mathrm{Den}(F \restriction \sigma)^{-1}$ in the shifted proof with $\mathrm{Inv}_k(\mathrm{Den}(F \restriction \sigma))$, for a suitable k. We now need to simulate the axiom $F \cdot F^{-1} = 1$ that was replaced by $F \restriction \sigma \cdot \mathrm{Inv}_k(F \restriction \sigma) = 1$ in order to obtain a correct $\mathbb{P}_c(\mathbb{Z})$-proof. This is done by using in Section 9 Lemma 6.1; for this lemma we need to show that $(\mathrm{Den}(F \restriction \sigma))^{(0)} = 1$ is $\mathbb{P}_c(\mathbb{Z})$-provable (note that $(\mathrm{Den}(F \restriction \sigma))^{(0)}$ computes precisely the value of $\mathrm{Den}(F)$ under the identity matrices assignment, and that since $\mathrm{Den}(F)$ is good under ρ, as we mentioned above, $(\mathrm{Den}(F \restriction \sigma))^{(0)}$ computes the polynomial 1).

The construction in $\mathbf{V}^0$ of the $\mathbb{P}_c(\mathbb{Z})$-proofs of $(\mathrm{Den}(F \restriction \sigma))^{(0)} = 1$, for every F as above, is done as follows. As a first attempt, observe that we can construct this $\mathbb{P}_c(\mathbb{Z})$-proof by induction on the size of $F \restriction \sigma$: for every gate u in $F \restriction \sigma$ we construct the proof of $\mathrm{Den}(u)^{(0)} = 1$, using the fact that every division gate in $F \restriction \sigma$ evaluates to 1 under the zero assignment (this is because σ shifts the variables to ρ, and the fact that every division gate is good under ρ). However, since we do not have the $\mathbf{\Sigma}_1^B$-induction axiom in $\mathbf{V}^0$, in Lemma 8.1 we will use a parallel construction of the $\mathbb{P}_c(\mathbb{Z})$-proof, as follows: using the $\mathbf{\Sigma}_0^B$-**COMP** axiom, for every gate u in $F \restriction \sigma$, construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(u)^{(0)} = 1$. This is done simultaneously for all gates u, using pointers to "previous" proof-lines and direct constructions of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $u^{(0)} = 1$ that do not use the division axiom D, for every gate u in $F \restriction \sigma$.

8.1 Identity Matrices are Provably Good Assignments

Here we explicitly inspect all the division gates in our $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs, making sure that they are all provably good under ρ. We first make sure that this is sufficient for our purpose of eliminating division (that is, applying Lemma 6.1).

Lemma 8.1 (in $\mathbf{V}^0$). *Let F be a circuit with division and assume that for every division gate u^{-1} in F there exists a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$. Then, there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of Den$(F \restriction \sigma)^{(0)} = 1$.*

Proof. Simultaneously, for every node v in F we construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$, by which we conclude that $\mathrm{Den}(F \restriction \rho) = 1$ is provable in $\mathbb{P}_c(\mathbb{Z})$. Before showing this construction we show how to get a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(F \restriction \sigma)^{(0)} = 1$ from a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(F \restriction \rho) = 1$.

By the definition of σ and by basic rearrangements in $\mathbb{P}_c(\mathbb{Z})$ we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(F \restriction \rho) = (\mathrm{Den}(F \restriction \sigma)) \restriction \overline{0}$ (where $C \restriction \overline{0}$ means substituting 0 for each variable in C). By the construction of syntactic-homogeneous circuits (Section 9) the underlying graph of a (division free) circuit $C \restriction 0$ and the structure of $C^{(0)}$ are *identical*: constant gates stays the same; variables x_i turns into 0 nodes; plus gates $w = v_1 + v_2$ add the zero copies, $[v_1, 0], [v_2, 0]$ of v_1, v_2, respectively; and product gates $w = v_1 \cdot v_2$ multiply the zero-copies $[v_1, 0], [v_2, 0]$ of v_1, v_2, respectively (in particular, no plus gates are added to the circuit). Hence, $(\mathrm{Den}(F \restriction \sigma)) \restriction \overline{0} = (\mathrm{Den}(F \restriction \sigma))^{(0)}$ is $\mathbb{P}_c(\mathbb{Z})$-provable.

We now come back to the construction of a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$, for every node v in F. This is done by cases as follows.

Case 1: v is an input node. Hence $\mathrm{Den}(v) = 1$ by definition of Den.

Case 2: $v = w_1 \circ w_2$, for $\circ \in \{+, \cdot\}$, then $\mathrm{Den}(v \restriction \rho) := \mathrm{Den}(w_1 \restriction \rho) \cdot \mathrm{Den}(w_2 \restriction \rho)$. Thus we construct the proof of $\mathrm{Den}(w_1 \restriction \rho) \cdot \mathrm{Den}(w_2 \restriction \rho) = 1$ by pointing to the proofs of $\mathrm{Den}(w_1 \restriction \rho) = 1$ and $\mathrm{Den}(w_2 \restriction \rho)$ and using basic rules of $\mathbb{P}_c(\mathbb{Z})$ such as $1 \cdot 1 = 1$.

Case 3: $v = u^{-1}$. This is the relatively more difficult case. We need to use the following claim:

Claim 8.2 (in $\mathbf{V}^0$). *If there exists a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$ in which we do not use the axiom D, then there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of Num$(u \restriction \rho) =$ Den$(u \restriction \rho)$.*

Proof of claim: This is proved in a similar way to the division gates normalization Theorem 7.1. We omit the details. $\blacksquare_{\text{Claim}}$

By definition $\mathrm{Den}(v) = \mathrm{Num}(u)$. Since we have by assumption a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$, by the claim we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) = \mathrm{Den}(u \restriction \rho)$, and from that, using pointers to the $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(u \restriction \rho) = 1$, we get a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. □

We now need to inspect and provide appropriate division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs for all division gates in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities, denoted π_3 (according to the notation in Section 4). The proof π_3 consists of: (**i**) the proof obtained from Proposition 5.1—we deal with this part of the proof in Lemma 8.3; (**ii**) the proof obtained from Theorem 6.4, using the transformation from Lemma 6.3—we deal with this in Lemma 8.6; (**iii**) the proof obtained from Proposition 6.5 in Section 6.3—which is dealt with in Lemma 8.7.

Lemma 8.3 (in $\mathbf{V}^0$). *All division gates F^{-1} in the proof obtained from Proposition 5.1 are provably good under the identity matrices assignment.*

Proof. This is proved by direct construction of the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs.

We need to show that every division gate F^{-1} in this proof is provably good under ρ. Let $\mathcal{U}$ be the set of all the subcircuits u such that u^{-1} occurs in some circuit in π. We show that:

> ⁜ *An assignment is provably good for $\mathcal{U}$ iff the assignment is provably good for the equations* (5), (6).

By Claim 8.5 ρ is provably good for equations (5) and (6) which concludes the proof.

We only check ⁜ for the part of the proof π that proves equation (6), where C is a triangular matrix (see [HT15, Proposition 7.6 (i)]). The inspection of the other parts is similar.

Assume that U is an (upper or lower) triangular $n \times n$ matrix. We show that in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π of

$$\mathsf{Det}_{circ^{-1}}(U) = u_{11} \cdots u_{nn}\,, \tag{17}$$

all division gates w^{-1} are provably good under the identity matrices assignment iff u_{ii}^{-1} in (17) are provably good under this assignment, for all $i \in [n]$.

For a matrix A with entries a_{ij} we let $A[k] := \{a_{ij}\}_{i,j\in[k]}$. Assume that U is a lower triangular matrix (the case for an upper triangular matrix is similar). Denote by $\overline{0}$ the zero vector of the appropriate dimension (depending on the context). We write U as

$$U := \begin{pmatrix} U[n-1] & \overline{0}^t \\ v & u_{nn} \end{pmatrix}.$$

The $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (17) (in [HT15, Proposition 7.6.(i)]) proceeds, using the construction of $\mathsf{Det}_{circ^{-1}}(U)$ as follows (the explicit proof is omitted in [HT15]):

$$\begin{aligned}
\mathsf{Det}_{circ^{-1}}(U) &= \mathsf{Det}_{circ^{-1}}(U[n-1]) \cdot (u_{nn} - v(U[n-1])^{-1}\overline{0}^t) \\
&= \mathsf{Det}_{circ^{-1}}(U[n-1]) \cdot u_{nn} \\
&= \mathsf{Det}_{circ^{-1}}(U[n-2]) \cdot (u_{n-1,n-1} - v'(U[n-2])^{-1}\,\overline{0}^t) \cdot u_{nn} \\
&= \mathsf{Det}_{circ^{-1}}(U[n-2]) \cdot u_{n-1,n-1} \cdot u_{nn} \\
&= \cdots = \prod_{i=1}^{n} u_{ii}
\end{aligned}$$

where v' is v excluding the $(n-1)$th coordinate.

Inspecting the proof sequence above we see that all the division gates in the proof appear in the circuits computing the matrices $(U[n-1])^{-1}, \ldots, (U[1])^{-1}$. It thus suffices to show that all division gates appearing in the circuits $(U[n-1])^{-1}, \ldots, (U[1])^{-1}$ are provably good under ρ iff all u_{ii}^{-1}, for $i \in [n]$, are. Specifically, we show the following claim, from which it is immediate to conclude that all these division gates u have division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $u \restriction \rho = 1$ (using the $\mathbb{P}_c(\mathbb{Z})$-axiom $0 \cdot F = 0$):

Claim 8.4. *Every division gate in the circuits $(U[n-1])^{-1}, \ldots, (U[1])^{-1}$ is of the form*

$$(u_{ii} - \overline{0} \cdot \overline{h})^{-1},$$

for some (possibly empty) vector of circuits with division $\overline{h}$ and some $i \in [n]$.

Proof of claim: For brevity, let $U_i := U[i]$, for $i = 1, \dots, n-1$, and let $v_{(i)} := (v_1, \dots, v_i)$, and denote by I_n the identity matrix of dimension n. Denote

$$A = U_{i-1}^{-1}\left(I_{i-1} + \delta(U_i)^{-1} v_{(i-1)}^t \overline{0} U_{i-1}^{-1}\right).$$

Then, by construction

$$U_i^{-1} = \begin{pmatrix} A & -\delta(U_i)^{-1} U_{i-1}^{-1} v_{i-1}^t \\ -\delta(U_i)^{-1}\overline{0}U_{i-1}^{-1} & \delta(U_i)^{-1} \end{pmatrix} \tag{18}$$

and

$$\delta(U_i) = u_{ii} - \overline{0}U_{i-1}^{-1} v_{(i-1)}^t\,. \tag{19}$$

We proceed by induction on i to show that all division gates in $(U_{n-1})^{-1}, \dots, (U_1)^{-1}$ are of the required form. The base case $i = 1$ holds by definition, since $(U_1)^{-1} = u_{11}^{-1}$.

The induction step $(U_i)^{-1}$ is proved as follows. By (18) all the division gates in U_i appear either in $(U_{i-1})^{-1}$ or in $\delta(U_i)^{-1}$. By induction hypothesis the division gates in $(U_{i-1})^{-1}$ are all of the form $(u_{jj} - \overline{0} \cdot \overline{h})^{-1}$, for some vector of circuits $\overline{h}$ and some $j = 1, \dots, i-1$. Moreover, the single-output circuit $\delta(U_i)^{-1}$ contributes the outermost division gate $(u_{ii} - \overline{0} \cdot \overline{h})^{-1}$, to $\overline{h} = U_{i-1}^{-1} v_{(i-1)}^t$, which is of the correct form; as well as all the division gates in $\overline{h}$ itself. But all the division gates in $\overline{h}$ appear in U_{i-1}^{-1}, which by induction hypothesis all have the required form. ■Claim

It remains to show the following:

Claim 8.5 (in $\mathbf{V}^0$). *All division gates in equations* (5), (6) *are provably good under* ρ.

Proof of claim: For equation (6) above, this is immediate from the above, since every inverse gate that appears in the proof (and equivalently in equation (17)) is one of $(u_{ii} - \overline{0} \cdot \overline{h})^{-1}$, for some $i \in [n]$, which is provably good under the identity matrices assignment.

For equation (5), namely, $\mathsf{Det}_{circ^{-1}}(X)\mathsf{Det}_{circ^{-1}}(Y) = \mathsf{Det}_{circ^{-1}}(XY)$, by definition and notation in (7) we have $\mathsf{Det}_{circ^{-1}}(X) := \mathsf{Det}_{circ^{-1}}(X_1) \cdot \delta(X) = \mathsf{Det}_{circ^{-1}}(X_1) \cdot (x_{nn} - v_2 X_1^{-1} v_1^t)$, and (by induction on n the dimension of X), similarly to Claim 8.4 above, we can verify that every division gate in $\mathsf{Det}_{circ^{-1}}(X)$ is provably good under ρ. Similar reasoning applies to $\mathsf{Det}_{circ^{-1}}(Y)$ and $\mathsf{Det}_{circ^{-1}}(XY)$. We omit the details. ■Claim

This concludes the proof of Lemma 8.3. □

Lemma 8.6. *Every division gate F^{-1} that appears in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in Lemma 6.4 is provably good under ρ.*

Proof. We wish to show that every division gate F^{-1} that appears in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in Lemma 6.4 of

$$\mathsf{Det}_{circ^{-1}}(X) = \mathsf{Det}_{Taylor}(X), \tag{20}$$

where $\mathsf{Det}_{Taylor}(X) := \mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I - zX))$, is provably good under ρ. The cases where A in Lemma 6.4 is XY or Z are similar.

We shall characterize all division gates in this proof. The goal of this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof as shown in [HT15, Proposition 7.9] is to show that $\mathsf{Det}_{circ^{-1}}(X)$ is equal to $z^n \cdot \mathsf{Det}_{Taylor}(X) + \sum_{i<n} z^i \cdot Q_i$

and then apply Lemma 6.3 to extract the equality between the coefficients of z^n. For this purpose, the proof of (20) proceeds by first proving

$$\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1}) = z^n + \sum_{i=0}^{n-1} z^i \cdot Q_i \tag{21}$$

where the Q_i's are circuits with division, and such that z does not occur in the Q_i's. We then use the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathrm{Det}_{circ^{-1}}(X)\mathrm{Det}_{circ^{-1}}(Y) = \mathrm{Det}_{circ^{-1}}(XY)$ proved in Section 5.2.1 to show that $\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1}) \cdot \mathrm{Det}_{circ^{-1}}(X) = \mathrm{Det}_{circ^{-1}}(zX + I_n)$. By this and (21), we get that

$$\mathrm{Det}_{circ^{-1}}(zX + I_n) = \mathrm{Det}_{circ^{-1}}(X) \cdot (z^n + \sum_{i=0}^{n-1} Q_i). \tag{22}$$

We use Lemma 6.3 to extract the coefficient of z^n in (22) to get $\mathrm{coeff}_{z^n}(\mathrm{Det}_{circ^{-1}}(zX + I_n)) = \mathrm{Det}_{circ^{-1}}(X)$, which is our desired identity.

We will first check that the division gates in $\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1})$ are provably good under ρ. The division gates in $\mathrm{Det}_{circ^{-1}}(zI_n+X^{-1})$ include, among other gates, the gates in the circuit X^{-1}. But by the proof of Lemma 8.3 the division gates in X^{-1} are all provably good under ρ. The other division gates in $\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1})$ occur also in $\mathrm{Det}_{circ^{-1}}(X)$. More precisely, let u^{-1} be a division gate in $\mathrm{Det}_{circ^{-1}}(X)$; then u^{-1} occurs also in the substitution instance $\mathrm{Det}_{circ^{-1}}(zI_n+X^{-1})$ of $\mathrm{Det}_{circ^{-1}}(X)$. Such a division gate u^{-1} in $\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1})$, by definition (10) of $\mathrm{Det}_{circ^{-1}}$, will occur in the following term $\mathrm{Det}_{circ^{-1}}(zI_n + X^{-1}) = z \cdot 1 + \delta(X)^{-1} - (-\delta(X)^{-1}v_2X_1^{-1}) \cdot (X^{-1})_1^{-1} \cdot (-\delta(X)^{-1}X_1^{-1}v_1^t)$ (where $(X^{-1})_1^{-1}$ is $(X^{-1})^{-1}$ without the rightmost column and without the lowest row). Note that in this term *we have only the division gate* $\delta(X)^{-1}$ (terms like $(X^{-1})_1^{-1}$ contain division gates, but they do not stand themselves as division gates, since X^{-1} is a notation for a matrix, not a circuit whose root is a division gate).

We thus need to construct a free division axiom $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof for the fact that the division gate $\delta(X)^{-1}$ is 1, under ρ. But this already stems from the proof of Lemma 8.3 above.

Precisely the same reasoning shows that for every circuit F in the $\mathbb{P}_c^{-1}$-proofs of $\mathrm{Det}_{circ^{-1}}(X) = \mathrm{Det}_{Taylor}(X)$ in Lemma 6.4 we have: if z occurs in the scope of a division gate in F then $\big(\mathrm{Den}(F)\big)(z/0)$ is provably good under ρ. □

Lemma 8.7. *Every division gate u^{-1} that appears in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in Lemma 6.5 is provably good under ρ.*

Proof. We consider the case where A in Lemma 6.5 is X. The cases for XY, Z are similar.

By Section 6.3 every division gate in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of Lemma 6.5 is of the form $(\mathrm{Den}\,(\mathrm{Det}_{circ^{-1}}\,(I_n + zX))\,(z/0))^{-1}$. Identical to the proof of Lemma 6.5, start by constructing directly a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(\mathrm{Det}_{circ^{-1}}\,(I_n + zX))(z/0) = \mathrm{Det}_{circ^{-1}}\,(I_n)$, by substituting z by 0 and gradually replacing in the proof $0 \cdot u$ to 0 and $0 + u$ to u, for u any subcircuit. Now, by the proof of Lemma 8.3, all the division gates in $\mathrm{Det}_{circ^{-1}}\,(I_n)$ are provably good under ρ. By Lemma 8.1 we get that $(\mathrm{Den}\,(\mathrm{Det}_{circ^{-1}}\,(I_n + zX))\,(z/0)) = 1$ has a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (in fact, already a $\mathbb{P}_c(\mathbb{Z})$-proof). □

In order to be able to apply Lemma 6.1 in Section 9 we need to show how to construct in the theory $\mathbb{P}_c(\mathbb{Z})$-proofs of the fact that the division gates from Corollary 7.2, are equal 1.

Corollary 8.8 (in $\mathbf{V}^0$)**.** *Let π be the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of* (5) *and* (6) *from Corollary 7.2, in which every division gate in a circuit in π appears at the top as $Den(F)^{-1}$, for some circuit F, and where the determinant in* (5) *and* (6) *is written as $\mathsf{Det}^{\#}_{Taylor}$. For every such F in π there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $(Den(F \restriction \sigma))^{(0)} = 1$ (where σ is the linear shift as defined above).*

Proof. This follows from Lemma 8.1 and the fact that all division gates in π were shown to be provably good under ρ in the lemmas that follow Lemma 8.1. □

8.2 Eliminating Division

We are now ready to construct the division free PI-proof of the determinant identities (with some restrictions). Note that we will need to use the fact that division gates are provably good in the next section (Section 9).

We first define the following:

Definition 8.9 (Correct up to degree k $\mathbb{P}_c(\mathbb{Z})$-proof)**.** *Let k be a natural number. We say that a $\mathbb{P}_c(\mathbb{Z})$-proof sequence π is* correct up to degree k *if (i) every proof-line in π is an equation between algebraic circuits that was derived by one of the derivation rules of $\mathbb{P}_c(\mathbb{Z})$ from previous lines; or (ii) is a variant of the division axiom* D*, where instead of $F \cdot F^{-1} = 1$ we have the line $F \cdot \mathrm{Inv}_k(F) = 1$; or (iii) is an axiom of $\mathbb{P}_c(\mathbb{Z})$ different from* D*.*

The witness for syntactic correctness of a correct up to degree k $\mathbb{P}_c(\mathbb{Z})$-proof is similar to that in Section 5.2. Note that we do not need to witness the syntactic-degree of nodes in circuits in a correct up to degree k $\mathbb{P}_c(\mathbb{Z})$-proof. In other words, there exists a Σ_0^B-formula $\psi(Z, k)$ that holds if Z is a correct up to degree k $\mathbb{P}_c(\mathbb{Z})$-proof (where Z contains also the syntactic correctness witness as in Section 5.2). The formula $\Psi(Z, k)$ only needs to verify that in the division axiom D we have $F \cdot \mathrm{Inv}_k(F) = 1$, and checking whether a circuit is $\mathrm{Inv}_k(F)$ is done without the need to witness the syntactic-degree of F or $\mathrm{Inv}_k(F)$.

Corollary 8.10 (in $\mathbf{V}^0$)**.** *Given a positive natural number n, there exists a* correct up to degree $2n$ *$\mathbb{P}_c(\mathbb{Z})$-proof sequence of the determinant identities* (5) *and* (6) *for $n \times n$ matrices where the determinant in* (5) *and* (6) *is written as the division free circuit $\mathsf{Det}^{\#}_{Taylor}(A)$ of syntactic-degree n, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z.*

Proof. By Corollary 7.2 there exists a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π_3 of the determinant identities where the determinant in (5) and (6) is written as $\mathsf{Det}^{\#}_{Taylor}$, and in which each circuit appearing in the proof has at most one division gate that appears as the root of the circuit.

The rest of the proof follows the sketch in Step 5 (Section 4). Assume that r_i (for $i \in J$) are all the variables appearing in π_3 and b is the assignment of identity matrices to the variables in π_3 (that is, the assignment of I_n to the (variables belonging to the) matrices X, Y, Z in π_3).

Substitute in π_3 the term $(b_i - w_i)$ for each r_i (for all $i \in J$) denoting the obtained proof by π_3'. The substitution is performed as follows: given π_3 there is a Σ_0^B-formula $\Psi(\pi_3, v)$ that is true iff v is a node in a circuit in π_3 that is labeled by a variable (more formally, v is a number in the set of nodes V in a circuit in the proof and $(v, t) \in G$, where G is the string specifying the gates of V, and t is a number that specifies one of the input variables; recall the encoding of circuits in Section 5.1.1). Using $\Psi(\pi_3, v)$ we can define a Σ_0^B-formula $\Phi(\pi_3, i)$ such that $\Phi(\pi_3, i)$ is true iff $i \in \pi_3'$, where π_3' is a string that encodes the desired substitution instance of π_3. This is done using a bounded number universal quantifier over nodes v in π_3 and over all indices of circuits in π_3, so that when $\Psi(\pi_3, v)$ holds, $(v,t) \in G$ is replaced by "$(b_i - w_i)$" as follows: (v,t) is replaced by (v,t') where t' specify a $+$ gate; then we add four new nodes in V, new edges in E, new gates in G, and new inputs to the input string I, encoding that the product gate v has two children: b_i and $-1 \cdot w_i$.

Accordingly, when we define the substitution instance above we also need to update the witnesses for the proofs. The witnesses all stay the same except the witnesses for the $\mathbb{P}_c^{-1}$ axioms C1 and C2, that is, $F_1 \oplus F_2 = F_1 + F_2$ and $F_1 \otimes F_2 = F_1 \times F_2$, respectively. For these two axioms we specify the new mappings of F_1, F_2 in the right hand side of the equation to the circuits in the left hand side of the equation under the substitution. Since the substitution is defined by replacing some leaves in F_i by some circuits then we only need to add new pairs to the original mapping to update it appropriately.

By Lemma 8.8 the all zero assignment $\overline{0}$ to the w_i variables in π_3' gives every division gate the value 1.

For every circuit C with a top division gate H^{-1}, let $\mathrm{Inv}_{2n}(H)$ be the truncated power series of H^{-1} over the zero assignment (we use $2n$ because this is the degree of the determinant identities). $\mathbf{V}^0$ proves there exists a corresponding division-free circuit C', obtained by replacing the division gate H^{-1} in C by $\mathrm{Inv}_{2n}(H)$. This is done like the substitution of variables describes above, where now the nodes we define using a $\mathbf{\Sigma}_0^B$-formula $\Psi(\pi_3, v)$ are the inverse nodes u^{-1}, and the substitution adds nodes and wirings while also using the subcircuit u itself (as needed to construct $\mathrm{Inv}_{2n}(u)$).

Let π_3'' be the corresponding division-free proof-sequence obtained from π_3' by replacing every circuit C with a division gate at the top by C' as above.

In π_3'' every occurrence of the axiom D of division $F \cdot F^{-1} = 1$ is replaced by $F \cdot \mathrm{Inv}_{2n}(F) = 1$. Now substitute back the original variables X, Y, Z instead of the terms $b_i - w_i$. The resulting sequence is a correct up to degree $2n$ $\mathbb{P}_c(\mathbb{Z})$ proof sequence of the desired determinant identities. □

9 Eliminating High Degrees: Constructing PI-Proofs with Polynomial Syntactic-Degrees

From now on all our algebraic circuits are division free. Let C be an algebraic circuit. Recall that $d(C)$ is the syntactic-degree of C defined to be the maximal syntactic-degree of a node in C (Definition 3.6). Also, recall that $C^{(i)}$ is a syntactic-homogeneous circuit computing the degree i homogeneous part of $\widehat{C}$ (see Section 3.4). For a node v in C, C_v denotes the circuit rooted at v.

We now show how to eliminate, within $\mathbf{V}^0$, all nodes of syntactic degrees exceeding a given number d, from circuits in a PI-proof of the equation $F = G$ assuming the syntactic-degree of F, G is at most d.

We first note that we do not know of an $\mathbf{FAC}^0$ algorithm that computes the syntactic-degrees of nodes in a given circuit (see also Section C). However, for our purposes it will be sufficient to input an upper bound on the syntactic-degree of the circuit to be homogenized (except for the proof of the Cayley-Hamilton theorem in Section 12 wherein we need to witness explicitly the syntactic-degree of nodes in a proof in order to use Lemma 6.3 part (3); see Section C for the definition of syntactic-degree witness).

Our algorithms will receive a circuit C and a number d that will serve as an upper bound on the syntactic-degree of the circuit (the theory does not need to verify the correctness of this syntactic-degree).

Homogenization Algorithm in Uniform $\mathbf{FAC}^0$

Input: an arithmetic circuit C of size s and a natural number d.

Optional input 1: A syntactic-degree witness for all the nodes in C (including the root that has syntactic-degree d).

Optional input 2: A natural number i.

Output:

1. An arithmetic circuit C' of size $O(d^2 \cdot s)$ computing the polynomial $\widehat{C}$ such that C' is a sum of syntactic homogeneous circuits $C' = C^{(0)} + \cdots + C^{(d)}$ ($C^{(i)} = 0$ for $i > d(C)$).
2. If optional input 1 was supplied, then for every gate v in C', the duplicate gate $[v, j]$ for $j > d(v)$, is the circuit 0 (see below for the notation "$[u, j]$").
3. If the input C is (declared to be)[6] a (sum of) syntactic homogeneous circuits $\sum_{i \in I} C^{(i)}$ for $I \subseteq \{0, \dots, d\}$ then output C, augmented with the nodes $[u, j] = 0$, for all nodes $u \in C$ and all $j \in \{0, \dots, d\} \setminus I$.
4. If optional input 2 was supplied, then $C' = C^{(i)}$, namely the ith homogeneous component. If moreover the input circuit C is already a syntactic homogeneous circuits $C^{(j)}$ then the output is the circuit 0 if $j \neq i$ and is $C^{(i)}$ if $j = i$.

Algorithm: We follow the standard Strassen [Str73] algorithm, but instead of building the circuits by induction from leaves to root we construct all nodes simultaneously as follows.

(**1**) Assume we do not have the witness for syntactic-degrees of all the nodes (namely, the witness was not supplied as an input). Every node v is duplicated $d+1$ times into the nodes $[v, 0], \dots, [v, d]$. For a node $[v, i]$ we call i the ***syntactic-degree upper bound of*** $[v, i]$, denoted as

$$d_{\text{ub}}([v, i]) := i.$$

The node $[v, i]$ is (the root of) a syntactic-homogeneous circuit of syntactic-degree at most i computing *either* 0 or the degree i homogeneous part of the polynomial $\widehat{C_v}$. The algorithm is doable in $\mathbf{FAC}^0$ because every new node $[v, i]$ depends only on the copies of the two nodes u, w that goes into v, and these nodes are already known from the input circuit, namely, they are $[u, i]$, $[w, i]$, for $i = 0, \dots, d+1$, where $v = u + w$ or $v = u \cdot w$ in C. Hence, the wiring of the new circuit is done in parallel for each of the new nodes as follows:

Case 0: v is a leaf in C. If v is a constant α, then define $[v, 0] = \alpha$, and $[v, i] = 0$ for all $i = 1, \dots, d$. Otherwise, v is a variable x_j, and we define $[v, 1] = x_j$, and $[v, i] = 0$ for all $1 \neq i \in \{0, \dots, d\}$.
Case 1: $v = u + w$ in C. Define $[v, i] := [u, i] + [w, i]$ for every $i = 0, \dots, d$.
Case 2: $v = u \times w$ in C. Define $[v, i] := \sum_{\substack{j+k=i \\ j,k=0,\dots,d}} [u, j] \times [w, k]$.
Finally, $C^{(i)} := r^{(i)}$, for r the root of C, for all $i = 0, \dots, d$.

[6] The algorithm does not check for correctness of C being a (sum of) syntactic homogeneous circuits.

(**2**) Otherwise, assume that a witness for the syntactic-degree $d(v)$ for every node v in C was supplied as an input. In this case the algorithm is the same as above, except that the ith duplicate $[v, i]$ of a node v is defined to be the circuit 0 whenever $i > d(v)$. More precisely:
Case 0: v is a leaf in C. If v is a constant α, then define $[v, 0] = \alpha$. Otherwise, v is a variable x_j, and we define $[v, 1] = x_j$, and $[v, j] = 0$, for all $0 \le j \le d$ and $j \neq 1$.
Case 1: $v = u + w$ in C. Define $[v, i] := [u, i] + [w, i]$ for every $i = 0, \dots, d(v)$, and $[v, i] := 0$ for $i = d(v) + 1, \dots, d$.
Case 2: $v = u \times w$ in C. Define $[v, i] := \sum_{\substack{j+k=i \\ j,k=0,\dots,d}} [u, j] \times [w, k]$, for every $i = 0, \dots, d(v)$, and $[v, i] := 0$ for $i = d(v) + 1, \dots, d$.[7]

Finally, $C^{(i)} := r^{(i)}$, for r the root of C, for all $i = 0, \dots, d$.

Note on syntactic-degree upper bounds. Notice that if we do not have a witness for syntactic-degrees and assuming we get as input a correct upper bound, that is, $d \ge d(C)$, the above algorithm produces a syntactic homogeneous circuit in which each node $[u, i]$ is of syntactic-degree i, *except* that for $\widehat{[u, i]} = 0$, the syntactic degree of the circuit rooted at $[u, i]$ can be *smaller* than i (but not bigger). This means that the circuit contains in itself *a witness for the upper bound of the syntactic-degree of each node*.

We will manage to work out our argument without the need to compute syntactic-degrees except for the Cayley-Hamilton theorem shown in Section 12.

On the other hand, if we receive a syntactic-degree witness as an input then assuming $[u, i]$ is not the circuit 0, u has syntactic-degree at least i. Moreover, if the input to the algorithm is already a sum of syntactic homogeneous circuits $\sum_{i=0}^{k} C^{(i)}$ then $[u, i] = \emptyset$ for $i > k$ for every node u in C.

We are going to construct $\mathbb{P}_c(\mathbb{Z})$-proofs which witness the syntactic-degree of every node in:

Definition 9.1. *Given a $\mathbb{P}_c(\mathbb{Z})$-proof π we say that every node in every circuit in the proof* appears with its syntactic-degree upper bound d_{ub} *if every such node u is a pair of numbers $[u, i]$ for $d_{\text{ub}}(u) = i$, according to the construction in the homogenization algorithm above.*

We have the following main theorem about homogenization of proofs:

Theorem 9.2 (in $\mathbf{V}^0$)**.** *Let d be a natural number and assume that $F = G$ has a correct up to degree d $\mathbb{P}_c(\mathbb{Z})$-proof. Then, for every $k = 0, \dots, d$, there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(k)} = G^{(k)}$ of syntactic-degree at most k, in which every circuit is a syntactic homogeneous circuit in which every node u appears with its syntactic-degree upper bound, and $d_{\text{ub}}(u) \le k$.*

We need the following lemmas before concluding this theorem.

Lemma 9.3 (in $\mathbf{V}^0$)**.** *Let $F_1 \oplus F_2$ and $F_1 \oplus F_2$ be two circuits and let k be a natural number*[8]*. The following have $\mathbb{P}_c(\mathbb{Z})$-proofs, in which every circuit is a sum of syntactic homogeneous circuits in which every node u appears with its syntactic-degree upper bound:*

1. $(F_1 \oplus F_2)^{(k)} = F_1^{(k)} + F_2^{(k)}$;

[7] Note that this means that provably in $\mathbf{V}^0$ there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $[v, i] := \sum_{\substack{j+k=i \\ 0 \le j \le d(u), 0 \le k \le d(k)}} [u, j] \times [w, k]$, for every $i = 0, \dots, d$, since $[u, j] = 0$ for $j > d(u)$ and $[w, k] = 0$ for $k > d(u)$.

[8] Recall that we mean here that we pick one such circuit out of all possible circuits of this form.

2. $(F_1 \otimes F_2)^{(k)} = \sum_{i=0}^{k} F_1^{(i)} \cdot F_2^{(k-i)}$.

Proof. Using the homogenization $\mathbf{FAC}^0$-algorithm above we construct $(F_1 \oplus F_2)^{(k)}$. By definition, $(F_1 \oplus F_2)^{(k)} := F_1^{(k)} \oplus F_2^{(k)}$, and by axiom C1, $F_1^{(k)} \oplus F_2^{(k)} = F_1^{(k)} + F_2^{(k)}$. We thus construct this one-line proof, adding to the proof a witness for the application of axiom C1. Note that given a circuit $A \oplus B$ we can construct in $\mathbf{V}^0$ a witness for the correctness of applying C1 to get $A \oplus B = A + B$. The witness will say that A in $A \oplus B$ is identical to A in $A + B$, by an explicit mapping of nodes from the former to the latter copy; and similarly for B. Furthermore, note that in this one-line $\mathbb{P}_c(\mathbb{Z})$-proof every node u in every circuit appears with its syntactic-degree upper bound: in $(F_1 \oplus F_2)^{(k)}$ this is true by construction of $(\cdot)^{(k)}$ and in $F_1^{(k)} + F_2^{(k)}$ we simply specify every node u in $F_1^{(k)} + F_2^{(k)}$ to have the same syntactic-degree upper bound as its origin node in $(F_1 \oplus F_2)^{(k)}$ (note that this is indeed a true upper bound on the syntactic-degree of u).

This concludes 1. Part 2 is similar using the homogenization algorithm above. □

We now conclude the proof of the theorem:

Proof of Theorem 9.2. For every $k = 0, \ldots, d$, we devise a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ that produces a $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(k)} = G^{(k)}$ with every node u in every circuit in the proof appears with its syntactic-degree upper bound $d_{\text{ub}}(u)$ (that is, it appears as $[u, i]$ where $i = d_{\text{ub}}(u)$) and such that $d_{\text{ub}}(u) \le k$. This is done in a manner resembling the algorithm above for homogenizing circuits. Specifically, for every $k = 0, \ldots, d$ and every line $S = T$ in π, we construct in parallel a part of the proof of $S^{(k)} = T^{(k)}$ (that taken collectively would amount to a proof of $S^{(k)} = T^{(k)}$).

Though some proof-lines $S = T$ possess syntactic-degree witnesses while some are already syntactic homogeneous, and some proof-lines do not fall into the two former categories, the proof we show works for all these three cases; this stems from the way we defined the homogenization algorithm: this algorithm constructs the nodes $[u, i]$ for every original node u in its input and every $i = 0, \ldots, d$ (even in the case where i exceeds the specified syntactic-degree of u in the syntactic-degree witnesses input; and similar for the output of the homogenization algorithm part 3).

Case 1: $S = T$ is an axiom of $\mathbb{P}_c(\mathbb{Z})$ of size s. We construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $S^{(k)} = T^{(k)}$ with size $s \cdot \text{poly}(k)$ (and syntactic degree $\le k$).

Lemma 9.3 gives a proof $(F_1 \oplus F_2)^{(k)} = (F_1 + F_2)^{(k)}$ and $(F_1 \otimes F_2)^{(k)} = (F_1 \cdot F_2)^{(k)}$, as required for the axioms C1 and C2.

Axioms A1 and A10 are immediate. For the other axioms, consider for example the axiom $F_1 \cdot (F_2 \cdot F_3) = (F_1 \cdot F_2) \cdot F_3$, where the circuits have size $\le s$. We have to construct a proof of

$$(F_1 \cdot (F_2 \cdot F_3))^{(k)} = ((F_1 \cdot F_2) \cdot F_3)^{(k)} \,. \tag{23}$$

By part (ii) of Lemma 9.3 the equations

$$(F_1 \cdot (F_2 \cdot F_3))^{(k)} = \sum_{i=0}^{k} F_1^{(i)} \left(\sum_{j=0}^{k-i} F_2^{(j)} F_3^{(k-i-j)} \right) \tag{24}$$

$$((F_1 \cdot F_2) \cdot F_3)^{(k)} = \sum_{i=0}^{k} \left(\sum_{j=0}^{i} F_1^{(j)} F_2^{(i-j)} \right) \cdot F_3^{(k-i)} \,, \tag{25}$$

can be proved in $\mathbb{P}_c(\mathbb{Z})$. In $\mathbb{P}_c(\mathbb{Z})$, the right hand sides of both (24) and (25) can be written as $\sum_{i+j+l=k} F_1^{(i)} F_2^{(j)} F_3^{(l)}$ by a proof of size roughly $s(k+1)^4$. (This gives the proof of (23) of size $s \cdot \text{poly}(k)$.)

Case 2: If the line $S = T$ is $S_1 \cdot S_2 = T_1 \cdot T_2$, and it was derived using the rule R4 as follows:

$$\frac{S_1 = T_1 \qquad S_2 = T_2}{S_1 \cdot S_2 = T_1 \cdot T_2}\,. \tag{26}$$

From previous lines $S_1 = T_1$ and $S_2 = T_2$, we construct the derivation of $S^{(k)} = T^{(k)}$ by using the lines ${S_1}^{(i)} = {T_1}^{(i)}$ and ${S_2}^{(i)} = {T_2}^{(i)}$, for all $i = 0, \dots, k$, as follows: by Lemma 9.3, we can construct the proofs of $(S_1 \cdot S_2)^{(k)} = \sum_{i=0,\dots,k} S_1^{(i)} \cdot S_2^{(k-i)}$ and $(T_1 \cdot T_2)^{(k)} = \sum_{i=0,\dots,k} T_1^{(i)} \cdot T_2^{(k-i)}$. Hence, $(S_1 \cdot S_2)^{(k)} = (T_1 \cdot T_2)^{(k)}$ can be proved from the assumptions $S_1^{(i)} = T_1^{(i)}$, $S_2^{(i)} = T_2^{(i)}$, $i = 0, \dots k$. (The proof has size roughly $s \cdot (k+1)^c (k+1)$.)

Note that this is done independently and simultaneously for each proof-line (and specifically, not by induction on proof-length). That is, given the rule application (26), we construct the (partial) proof of $(S_1 \cdot S_2)^{(k)} = (T_1 \cdot T_2)^{(k)}$ using only the lines ${S_1}^{(i)} = {T_1}^{(i)}$ and ${S_2}^{(i)} = {T_2}^{(i)}$, for all $i = 0, \dots, k$; and in addition, since we also want to record the information about which line was derived from which previous lines we add pointers to previous lines. The latter can be defined via a $\mathbf{\Sigma}_0^B$-definable $\mathbf{V}^0$ number function given as input the line-numbers of $S_1 = T_1$ and $S_2 = T_2$. Hence the whole construction is in $\mathbf{V}^0$.

Case 3: $S = T$ is the rule R1-R3. This is similar to the case for rule R4.

The fact that in the $\mathbb{P}_c(\mathbb{Z})$-proof we constructed every node u in every circuit appears with its syntactic-degree upper bound is clear from the construction, since we used the homogenization algorithm to produce the syntactic homogeneous circuits and Lemma 9.3 □

We need the following claims:

Claim 9.4 (in $\mathbf{V}^0$). *Given a syntactic homogeneous circuit $F^{(d)}$ there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(d)} = \sum_{i=0}^{d} \left(F^{(d)}\right)^{(i)}$ in which every node u in every circuit appears with its syntactic-degree upper bound.*

Proof of claim: This is by the definition of the homogenization algorithm: if the input to the algorithm is the (already) syntactic homogeneous circuit $F^{(d)}$ then $\left(F^{(d)}\right)^{(i)} = \emptyset$, for all $i \neq d$, and $\left(F^{(d)}\right)^{(i)} = F^{(d)}$, for $i = d$. $\blacksquare_{\text{Claim}}$

Claim 9.5 (in $\mathbf{V}^0$). *Let $F(x_1, , \dots, x_m)$ be a circuit without division, in the displayed input variables. Assume that every input variable is now multiplied by a new variable z to get $F(zx_1, \dots, zx_m)$, which we denote by F_z. Then, there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{coeff}_{z^i}(F_z) = F^{(i)}$ in which every node u in every circuit appears with its syntactic-degree upper bound.*

Proof of claim: This is by construction of $\mathrm{coeff}_{z^i}(\cdot)$ and $(\cdot)^{(i)}$. Consider the construction of $\mathrm{coeff}_{z^i}(\cdot)$ in $\mathbf{V}^0$ as shown in the appendix Section B. Then, having each variable x_j multiplied directly by z in the circuit F' means that $\mathrm{coeff}_{z^i}(F') = F^{(i)}$ are syntactically identical except for the bottom level of the circuits, namely $\mathrm{coeff}_{z^1}(z \cdot x_j) = 1 \cdot x_j$ (ignoring zero terms, and applying basic rearrangements), and $\mathrm{coeff}_{z^r}(z \cdot x_j) = 0$ for all $1 \neq r \leq k$ and all $j \in [n]$. And accordingly, ${x_j}^{(1)} = x_j$ and ${x_j}^{(r)} = 0$, for all $1 \neq r \leq k$ and all $j \in [n]$. $\blacksquare_{\text{Claim}}$

Corollary 9.6 (in $\mathbf{V}^0$). *Given a positive natural number n there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of the determinant identities (5) and (6), where the determinant in (5) and (6) is written as the division free circuit*

$\mathsf{Det}^{\#}_{Taylor}(A)$*, for* A*, the* $n \times n$ *symbolic matrix* X *or* Y*, or their product* XY*, or a symbolic triangular matrix* Z*. Moreover, in this proof every circuit is a sum of syntactic homogeneous circuits in which every node appears with its syntactic-degree upper bound* $d_{\rm ub}$*.*

Proof. By Corollary 8.10 there exists a *correct up to degree* $2n$ $\mathbb{P}_c(\mathbb{Z})$ proof-sequence π of the determinant identities (5) and (6) where the determinant in (5) and (6) is written as $\mathsf{Det}^{\#}_{Taylor}$. By Lemma 6.1 and Corollary 8.8 we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $(F \cdot \mathrm{Inv}_{2n}(F))^{(i)} = 0$, for every $1 \le i \le 2n$, and a $\mathbb{P}_c(\mathbb{Z})$-proof of $(F \cdot \mathrm{Inv}_{2n}(F))^{(0)} = 1$, for every proof-line $F \cdot \mathrm{Inv}_{2n}(F) = 1$ appearing in π, and these proofs also contain the syntactic-degree upper bound for every node in every circuit. Therefore, by Theorem 9.2 part (i) we have a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$\left(\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y)\right)^{(k)} = (\mathsf{Det}^{\#}_{Taylor}(XY))^{(k)}, \tag{27}$$

for every $k = 0, \ldots, 2n$, and similarly for (6), wherein every node appearing in the proof has a specified syntactic-degree upper bound.

Finally, we can conclude the corollary by reasoning as follows.

Recall the definition of $\mathsf{Det}^{\#}_{Taylor}(X)$ from (16). Then $\mathsf{Det}^{\#}_{Taylor}(X)$ is a division free circuit such that every X variable in it is a product of z. Thus, by Claim 9.5 (and its proof) we can assume that $\mathsf{Det}^{\#}_{Taylor}(X)$ is specified already as a syntactic homogeneous circuit when given as input to the homogenization algorithm, meaning that the algorithm will output the same syntactic homogeneous circuit it got as an input (or a single homogeneous component as in part 3 of the output). Thus, by Claim 9.4 there exist $\mathbb{P}_c(\mathbb{Z})$-proofs of:

$$(\mathsf{Det}^{\#}_{Taylor}(X))^{(i)} = 0, \text{ for all } i < n, \tag{28}$$

$$(\mathsf{Det}^{\#}_{Taylor}(X))^{(n)} = \mathsf{Det}^{\#}_{Taylor}(X)\,. \tag{29}$$

The same argument works for $\mathsf{Det}^{\#}_{Taylor}(Y)$. We proceed as follows.

By Lemma 9.3 we have a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$\sum_{i=0}^{2n} \left(\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y)\right)^{(i)} = \sum_{i=0}^{2n} \sum_{\substack{l+j=i \\ 0\le l\le 2n, 0\le j\le 2n}} \mathsf{Det}^{\#}_{Taylor}(X)^{(l)} \cdot \mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}\,. \tag{30}$$

This equals

$$\sum_{i=0}^{2n} \sum_{\substack{l+j=i \\ 0\le l\le n, 0\le j\le n}} \mathsf{Det}^{\#}_{Taylor}(X)^{(l)} \cdot \mathsf{Det}^{\#}_{Taylor}(Y)^{(j)} + \sum_{i=0}^{2n} \sum_{\substack{l+j=i \\ n<l\le 2n, n<j\le 2n}} \mathsf{Det}^{\#}_{Taylor}(X)^{(l)} \cdot \mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}\,, \tag{31}$$

where the rightmost big term is a sum of only zeros, by construction of the homogeneous circuits (since $\mathsf{Det}^{\#}_{Taylor}(X)$ and $\mathsf{Det}^{\#}_{Taylor}(Y)$ are specified as already syntactic homogeneous circuits when input to the homogenization algorithm $\mathsf{Det}^{\#}_{Taylor}(X)^{(l)} = \mathsf{Det}^{\#}_{Taylor}(Y)^{(l)} = 0$, for all

$l > n$). We are thus left with the leftmost big sum in (31). We proceed with

$$\sum\nolimits_{i=0}^{2n} \sum_{\substack{l+j=i \\ 0\le l\le n, 0\le j\le n}} \mathsf{Det}^{\#}_{Taylor}(X)^{(l)} \cdot \mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}$$

$$= \sum\nolimits_{i=0}^{n} \mathsf{Det}^{\#}_{Taylor}(X)^{(i)} \cdot \sum\nolimits_{i=0}^{n} \mathsf{Det}^{\#}_{Taylor}(Y)^{(i)}$$
$$= \mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y),$$

where the first equation is by rearrangement and the second equation is by (28) and (29). By summing (27) for all $i = 0, \dots, 2n$ and using similar reasoning for $\mathsf{Det}^{\#}_{Taylor}(XY)$, we conclude that there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y) = \mathsf{Det}^{\#}_{Taylor}(XY).$$

The fact that in the above $\mathbb{P}_c(\mathbb{Z})$-proof every proof-line is a sum of syntactic homogeneous circuits in which every node appears with its syntactic-degree upper bound d_{ub} stems from the construction. □

10 Balancing Algebraic Circuits and Proofs in the Theory

We start by providing some background and overview of the $\mathbf{NC}^2$-algorithm for constructing a balanced circuit, given as an input an upper bound (in unary) on the syntactic-degree of the input circuit.

The input to the algorithm is C, d where C is a syntactic-homogeneous circuit and d is an upper bound on the syntactic-degree of C in unary.

The output of the algorithm is a balanced circuit denoted $[C]$ computing the polynomial $\widehat{C}$. That is, if C has size s, then the depth of $[C]$ is $O(\log s \log d + \log^2 d)$ and the size of $[C]$ is poly(s, d).

The algorithm itself follows the general scheme of Valiant *et al.* [VSBR83] that proceeds by induction on the logarithm of the degree of the polynomial computed by the circuit, however there are differences that help us fit the algorithm in $\mathbf{FNC}^2$ (for a very clear exposition of the [VSBR83] algorithm we refer the reader to [RY08] (cf. [HT15]), though our treatment is self contained).

Specifically, we use a preprocessing step to record in advance for every pair of nodes w and v if w is in the scope of the circuit rooted by v. Furthermore, in the first stage of the algorithm (corresponding to the base case of the Valiant et al. algorithm) we need to compute the coefficients of certain linear forms computed by possibly non-balanced circuits. We show that both the preprocessing step and the first stage of the algorithm are $\mathbf{AC}^0$-reducible to matrix powering; where matrix powering is known to be computable in $\mathbf{VNC}^2$ (cf. [CF12]). Another difference is that while Valiant et al. [VSBR83] use the notion of degree of a node, and Hrubeš and Tzameret [HT15] use the syntactic-degree of a node, we are going to use the relaxed notion of syntactic-degree upper bound $d_{\mathrm{ub}}(v)$ of a node v introduced in Section 9 and its variant $d^{+}_{\mathrm{ub}}(v)$ (defined below).

Notation: Recall that we now only work with division free circuits. We use the following notation throughout this section: F, C are circuits and $\widehat{F}, \widehat{C}$ are the corresponding polynomials they compute. For convenience we denote by f the polynomial $\widehat{F}$. For a node v in F we write F_v to denote the subcircuit rooted at v and f_v denotes the polynomial $\widehat{F_v}$. We write $u \in F$ to mean that u is a node in the circuit F.

We will need to construct with an $\mathbf{FNC}^2$ algorithm some linear polynomials computed by F_v, whenever $v \in F$ and $d_{\text{ub}}(v) \leq 1$ as well as the linear polynomials $\partial w f_v$ whenever $0 \leq d_{\text{ub}}(v) - d_{\text{ub}}(w) \leq 1$ (see below). However, we cannot directly compute the integer coefficients in these linear polynomials because their (sub-)circuits are not balanced (and we apparently cannot evaluate circuits of high-depth in $\mathbf{VNC}^2$).

We show how to compute the linear polynomials we need in in Lemma 10.3 and Lemma 10.5. To facilitate these lemmas we need to treat scalar nodes $c \in \mathbb{Z}$ occurring in the circuit as if they are variables (and hence even circuits with only scalars get balanced throughout the balancing algorithm). Formally, this means defining their syntactic-degree as 1 instead of 0, as follows (so that both variables and scalars are now treated as syntactic-degree 1 circuits).

Denote by $d^+_{\text{ub}}(\cdot)$ the syntactic-degree upper bound defined similar to $d_{\text{ub}}(\cdot)$, except that scalar nodes are associated with syntactic-degree upper bound 1 (instead of 0) in the algorithm for homogenizing circuits shown in Section 9. Note that any circuit F_v rooted by the node v such that $d^+_{\text{ub}}(v) \leq 1$ cannot contain product nodes (as this would make $d^+_{\text{ub}}(v) > 1$ by definition). In Lemma 10.3 we show how to compute in $\mathbf{FNC}^2$ a circuit with no product nodes.

Also note that it may happen that a node v in a circuit has polynomially bounded $d_{\text{ub}}(u)$ but exponential large $d^+_{\text{ub}}(u)$, for example in case we have a linear chain of n products computing $(((2)^2)^2 \dots {}^2) = 2^{2^n}$. We deal with this problem in Lemma 10.7.

Definition 10.1 (Partial derivative polynomial $\partial w f_v$)**.** *Let w, v be two nodes in F. We define the* partial derivative of F with respect to w, *denoted $\partial w f_v$, as the following* polynomial*:*

$$\partial w f_v := \begin{cases} 0, & \text{if } w \notin F_v, \\ 1, & \text{if } w = v, \text{ and otherwise:} \\ \partial w f_{v_1} + \partial w f_{v_2}, & v = v_1 + v_2; \\ (\partial w f_{v_1}) \cdot f_{v_2}, & \text{if either } v = v_1 \cdot v_2 \text{ and } d^+_{\text{ub}}(v_1) \geq d^+_{\text{ub}}(v_2) \\ & \qquad \text{or } v = v_2 \cdot v_1 \text{ and } d^+_{\text{ub}}(v_1) > d^+_{\text{ub}}(v_2). \end{cases} \tag{32}$$

The idea behind this definition is the following: let w, v be two nodes in F and assume that $d^+_{\text{ub}}(w) > \frac{d^+_{\text{ub}}(v)}{2}$ (we will use $\partial w f_v$ only under this assumption). Then for any product node $v_1 \cdot v_2 \in F_v$, w can be a node in at most one of F_{v_1}, F_{v_2}, namely the one with the higher syntactic-degree. If we replace the node w in F_v by a new variable z that does not occur in F, then F_v computes a polynomial $g(z, x_1, \dots, x_n)$ which is *linear* in z, that is $g(z, x_1, \dots, x_n) = h_0 \cdot z + h_1$ for some polynomials h_0, h_1 in the $x_1, \dots, x_n$ variables, and $\partial w f_v = h_0$. Namely, $\partial w f_v$ is the standard partial derivative $\partial z g$.

Proposition 10.2. *Let w, v be two nodes in a syntactically homogeneous circuit F such that $d^+_{\text{ub}}(w) > \frac{d^+_{\text{ub}}(v)}{2}$. Then the polynomial $\partial w f_v$ has degree at most $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w)$.*

Proof. By induction on the size of F_v.
Base case: F_v is a single node. If $F_v = w$ then $\partial w f_v = 1$, and so $d^+_{\text{ub}}(\partial w f_v) = 0 = d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w)$ and the claim holds. If $F_v \neq w$ then $\partial w f_v = 0$ and the claim holds similarly.
Induction step:
Case 1: $v = v_1 + v_2$. Then $\partial w f_v = \partial w f_{v_1} + \partial w f_{v_2}$, and by induction hypothesis the degrees of $\partial w f_{v_1}$ and $\partial w f_{v_2}$ are at most $d^+_{\text{ub}}(v_1) - d^+_{\text{ub}}(w)$ and $d^+_{\text{ub}}(v_2) - d^+_{\text{ub}}(w)$, respectively. Since F

is syntactically homogeneous $d^+_{\mathrm{ub}}(v) = d^+_{\mathrm{ub}}(v_1) = d^+_{\mathrm{ub}}(v_2)$ and so the degree of $\partial w f_v$ is at most $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w)$.
Case 2: $v = v_1 \cdot v_2$. Assume that $d^+_{\mathrm{ub}}(v_1) \geq d^+_{\mathrm{ub}}(v_2)$. Then $\partial w f_v = (\partial w f_{v_1}) \cdot f_{v_2}$ and by induction hypothesis the degree of $\partial w f_{v_1}$ is at most $d^+_{\mathrm{ub}}(v_1) - d^+_{\mathrm{ub}}(w)$. Thus, the degree of $\partial w f_v$ is at most $d^+_{\mathrm{ub}}(v_1) + d^+_{\mathrm{ub}}(v_2) - d^+_{\mathrm{ub}}(w) = d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w)$. The case where $d^+_{\mathrm{ub}}(v_1) < d^+_{\mathrm{ub}}(v_2)$ is similar. □

Comment: We have defined $\partial w f_v$ as a *polynomial*. Below we shall construct (polynomial-size and balanced) *circuits* $[\partial w f_v]$ that compute the polynomial $\partial w f_v$. We will make sure that the construction of $[\partial w f_v]$ is correct in the sense that it computes $\partial w f_v$ and also that it has a syntactic-degree at most $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w)$. The correctness of the construction follows from [VSBR83] (see also [RY08, HT15]) where in our construction the notion of a syntactic-degree is used, instead of the notion of degree.

Overview of the balancing algorithm: Let F be a syntactic-homogeneous arithmetic circuit of syntactic-degree d. For every node $v \in F$ we introduce the corresponding *node* $[F_v]$ in $[F]$ (intended to compute the polynomial $\widehat{f_v}$); and for every pair of nodes $v, w \in F$ such that $d^+_{\mathrm{ub}}(w) > \frac{d^+_{\mathrm{ub}}(v)}{2}$, we introduce the *node* $[\partial w f_v]$ in $[F]$ (intended to compute the polynomial $\partial w f_v$). Note that given a syntactic-homogeneous circuit F, we can assume that every node comes with a number that denotes its syntactic-degree—this stems from the $\mathbf{FNC}^2$ algorithm for homogenization in Section 9; but note that according to this algorithm circuits that compute zero may be assigned *higher* syntactic-degrees than they actually possess. Since we are given an upper bound on the the syntactic-degree of the circuit in advance this will not interfere with the algorithm.

The algorithm starts with a preprocessing step that determines some properties of the circuit graph. Then it proceeds in steps $i = 0, \ldots, \lceil \log d \rceil$. In each step i we construct:

1. Circuits computing f_v, for all nodes v in F with $2^{i-1} < d^+_{\mathrm{ub}}(v) \leq 2^i$;
2. Circuits computing $\partial w f_v$, for all pairs of nodes w, v in F with $2^{i-1} < d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 2^i$ and $d^+_{\mathrm{ub}}(w) > \frac{d^+_{\mathrm{ub}}(v)}{2}$.

Each step adds depth $O(\log s)$ to the new circuit, which at the end amounts to a depth $O(\log^2 d + \log d \cdot \log s)$ circuit. Furthermore, each node v in F adds $O(s)$ nodes in the new circuit and each pair of nodes v, w in F adds $O(s)$ nodes in the new circuit. This amounts finally to a circuit of size $O(s^3)$.

The preprocessing step and step $i = 0$ are done in $\mathbf{FNC}^2$ as they both use matrix powering (in fact the class DET, which is the $\mathbf{AC}^0$-closure of matrix powering, suffices here). Each other stage constructs a group of nodes (namely, a part of the circuit having depth $O(\log s)$). Steps $i = 1$ to $i = \lceil \log d \rceil$ are done in $\mathbf{FAC}^0$ by constructing the nodes and wiring simultaneously. Thus overall the balancing algorithm is in $\mathbf{FNC}^2$.

10.1 Preliminaries for the Balancing Algorithm

Lemma 10.3 (in $\mathbf{VNC}^2$). *Given a (division free) algebraic circuit F of size s with no product gates, there exists a depth $O(\log n)$ circuit computing $\widehat{F}$ of size* $\mathrm{poly}(s)$*, for n the number of variables.*[9]

[9] Notice that if the input algebraic circuit F was a formula instead of a circuit, it would be trivial to output the balanced formula computing $\widehat{F}$: simply build a balanced binary tree whose leaves are all the variables occurring in F (variables that occur more than once should also occur more than once in the resulting formula). Also, notice that although there are no scalars in C, a monomial can occur with a coefficient in $\widehat{C}$ different from 1.

Proof. By assumption, the circuit F computes a big sum of variables, where a variable can occur with an integer coefficient. We will now represent circuits with unbounded fan-in as adjacency matrices. We first construct an upper triangular matrix $A = \{A_{ij}\}_{i,j\in[s]}$ that represents F: for every $j > i \in [s]$, A_{ij} *is labeled with the number of edges from node i to node j in the circuit*. In the initial stage, A is a 0-1 matrix because every node i can have at most one directed edge to node j. This construction is done already in $\mathbf{V}^0$.

Given such a matrix A representing F, the algorithm simply computes A^s. The matrix A^s has c on its (i, r)th entry iff the number of different paths from node i to the output gate r is c. Thus, we can consider the matrix A^s as corresponding to a depth 1 circuit: each leaf i in this circuit represents the input variable x_i or a scalar $k \in \mathbb{Z}$, and is connected to the root r of the original circuit with a single edge labelled with some integer c; this integer c is the total number of different paths in the original circuit leading from the input node x_i or a scalar $k \in \mathbb{Z}$ to the root. Thus cx_i or ck is the contribution of the input node x_i or the scalar node k to the linear polynomial $\widehat{F}$. It is thus immediate to construct a circuit (of depth $O(\log n)$) that computes the linear polynomial $\widehat{F}$: simply construct a big sum of the cx_i's and ck's.

The fact that matrix powering is definable in $\mathbf{VNC}^2$ is shown in Cook and Fontes [CF12].

□

We will also need the following two lemmas:

Lemma 10.4. *There is a Σ_1^B-definable function in $\mathbf{VNC}^2$ for deciding, given a circuit C and two nodes w, v in C, if w is in F_v.*

Proof. This is similar to Lemma 10.3 above. We first construct the adjacency matrix A_C of the circuit C as a directed graph: the dimension of A_C is $s \times s$ with s being the number of nodes in C, each entry in A_C is of number sort, and $A_C(w, u)$ is 1 iff w has a directed edge towards u or $w = u$, and 0 otherwise.

Then, w has a directed path to v iff $A_C^s(w, v) \neq 0$, where matrix powering is definable in $\mathbf{VNC}^2$ as mentioned above. □

Lemma 10.5. *There is a Σ_1^B-definable function in $\mathbf{VNC}^2$ whose input is a (division free) circuit F with n variables and a pair of nodes w, v in F where w is in F_v and $0 \le d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \le 1$, and whose output is an $O(\log(n))$-depth circuit computing $\partial w f_v$.*

Proof. In case $v = w$ we output the circuit 1. Otherwise, first note that since $0 \le d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \le 1$, either $d^+_{\text{ub}}(v) \le 1$ or $d^+_{\text{ub}}(v) \ge 2$ and $d^+_{\text{ub}}(w) > d^+_{\text{ub}}(v)/2$. Hence, by Proposition 10.2, the polynomial $\partial w f_v$ is a linear polynomial $a_1x_1 + \cdots + a_nx_n + b$. Therefore, it remains to show how to construct the circuit that computes this linear polynomial in $\mathbf{VNC}^2$.

Fact 1: by definition of d^+_{ub}, for every node r in F we have $d^+_{\text{ub}}(r) \ge 1$. Hence, for every product gate $u = t \cdot s$ we have $d^+_{\text{ub}}(u) = d^+_{\text{ub}}(t) + d^+_{\text{ub}}(s) \ge 2$.

Fact 2: there cannot be a product gate u in F_v such that w has *two different paths* directed from w to u (recall that edges are directed from leaves to root).

This is because otherwise $d^+_{\text{ub}}(v) \ge d^+_{\text{ub}}(u) \ge 2d^+_{\text{ub}}(w) \ge 2$ (the last inequality is by the fact above), and hence $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \ge d^+_{\text{ub}}(w) \ge 2$ in contrast to the assumption.

Fact 3: Let ρ be a path from v to w (including v and excluding w) in F_v. Then there exist at most one product gate in ρ.

The reason is as follows: assume there are more than one product gates in ρ (occurring "above" w). By Fact 1 every such product gate in ρ increases the syntactic-degree upper bound d^+_{ub} along ρ by at least 1. Hence, $d^+_{\mathrm{ub}}(v) \geq d^+_{\mathrm{ub}}(w) + 2$ in contrast to the assumption that $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 1$.

We thus conclude that every product gate $u \neq w$ in F_v, either does not have w in its scope, or is the only product gate on the path from w to v along u. Let $u = t \cdot s$ be a product gate in F_v that has w in its scope, and assume without loss of generality that F_t has w in its scope and F_s does not (by Fact 2 it cannot be that both have w in their scope). We argue that F_s *has no product gates*. Otherwise, by Fact 1 $d^+_{\mathrm{ub}}(s) \geq 2$ and so $d^+_{\mathrm{ub}}(v) \geq d^+_{\mathrm{ub}}(u) \geq d^+_{\mathrm{ub}}(w) + d^+_{\mathrm{ub}}(s) \geq d^+_{\mathrm{ub}}(w) + 2$ in contrast to the assumption $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 1$.

Let U be the set of all product gates $u = t_u \cdot s_u$ in F_v such that F_{t_u} has (without loss of generality) in its scope w. The above arguments imply that the polynomial $\partial w f_v = \sum_{u \in U} \widehat{F}_{s_u}$ and that there are no product gates in the F_{s_u}'s. But the set U is easily $\mathbf{\Sigma}^B_0$-defined in $\mathbf{V}^0$. And by Lemma 10.3 we can thus construct a $O(\log n)$ depth circuit, for n the number of variables, computing the sum $\sum_{u \in U} \widehat{F}_{s_u}$. □

10.1.1 Taking Care of Nodes with High d^+_{ub} Measure

In this technical section we make sure that in the circuits we consider nodes have polynomially bounded d^+_{ub} measure. For this purpose we show that the identities proved can be assumed to be of polynomial syntactic-degree (irrespective of the other identities between circuits appearing throughout the $\mathbb{P}_c(\mathbb{Z})$-proofs), and then apply high syntactic-degree eliminations in proofs (Theorem 9.2; now with d^+_{ub} replaced for d_{ub}).

First note the following:

Fact 10.6. *All the statements about proof-construction and transformations that we presented up to this point for d_{ub} also holds true for d^+_{ub}.*

Fact 10.6 holds because of the following: let C be a circuit and let C' be the sum of its syntactic homogeneous components with all nodes in C' appearing with their syntactic-degree upper bound. Let C_Y be C in which we substitution every scalar leaf $c \in \mathbb{Z}$ to a new variable y_c. Then the output of the homogenization algorithm on the input C_Y results in a sum of syntactic homogeneous components of C_Y such that every node appear with its syntactic-degree upper bound d^+_{ub}. If we now substitution back the scalars $c \in \mathbb{Z}$ for the variables y_c we get the circuit C' in which every node appears with its syntactic-degree upper bound d^+_{ub} (instead of d_{ub}).

Lemma 10.7 (in $\mathbf{VNC}^2$). *Given a positive natural number n there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of the determinant identities (5) and (6), where the determinant in (5) and (6) is written as the division free circuit denoted $\mathrm{Det}^\star_{Taylor}(A)$, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z. Moreover, in this proof every circuit is a sum of syntactic homogeneous circuits in which every node u appears with its syntactic-degree upper bound d^+_{ub}, and $d^+_{\mathrm{ub}}(u) = O(n)$.*

Proof. Let π be the $\mathbb{P}_c(\mathbb{Z})$-proof in Corollary 9.6. Every node u in π appears with its syntactic-degree upper bound $d_{\mathrm{ub}}(u)$. By Fact 10.6 we could have assumed that every node appears with the syntactic-degree upper bound d^+_{ub} only that we need to make sure that d^+_{ub} for all nodes in π are polynomially bounded. We do this as follows.

By inspection of $\mathrm{Det}^\#_{Taylor}(X)$, we will show that we can construct in $\mathbf{VNC}^2$ a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Det}^\star_{Taylor}(X) = \mathrm{Det}^\#_{Taylor}(X)$, where $\mathrm{Det}^\star_{Taylor}(X)$ is a division free circuit with all nodes u

having $d^+_{\mathrm{ub}}(u) = O(n)$. We then homogenize this $\mathbb{P}_c(\mathbb{Z})$-proof using Theorem 9.2 to get rid of all nodes with high d^+_{ub} measure. Combining this proof with the $\mathbb{P}_c(\mathbb{Z})$-proof in Corollary 8.10 we obtain the desired proof.

We start by identifying a property that will help us to determine the d^+_{ub} measure of nodes.

Claim (Not necessarily in $\mathbf{VNC}^2$)**.** *Let F be a circuit (possibly* with *division) over the variables $x_1, x_2, \ldots$. Assume that F has the following property:*

> Property ✠*: either F is a scalar, or every scalar leaf α in F is a child of a plus gate u where $u := \alpha + h$ and h contains at least one x_i variable.*

Then, $d^+_{\mathrm{ub}}(F) = d_{\mathrm{ub}}(F)$.

Proof of claim: Since we are not proving this claim in the theory we can proceed by induction on the size of F. If $F := x_i$, then we are done. If $F := \alpha + v$, for some scalar α, since v contains some x_i variables, $d^+_{\mathrm{ub}}(F) = d^+_{\mathrm{ub}}(v)$. We also have $d_{\mathrm{ub}}(v) = d^+_{\mathrm{ub}}(v)$ by induction hypothesis, and $d_{\mathrm{ub}}(v) = d_{\mathrm{ub}}(F)$ since v contains some x_i variables, and we are done. If $F := v + w$ with both v, w different from a scalar, then the claim follows by induction hypothesis. Similarly, if $F := v \cdot w$ then the claim follows by induction hypothesis. $\blacksquare_{\text{Claim}}$

We proceed to prove Lemma 10.7. We show that $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ has property ✠, and thus $d^+_{\mathrm{ub}}(\mathsf{Det}_{circ^{-1}}(I_n + zX)) = d_{\mathrm{ub}}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$.

Consider the circuit $\mathsf{Det}_{circ^{-1}}(X)$. The only scalars in $\mathsf{Det}_{circ^{-1}}(X)$ are the 0-1 constants that occur in the identity matrix I_{n-1} in (9). In (9) the scalars 0-1 all appear in a subcircuit of the form $0 + h$ or $1 + h$ for some h with x_i variables as required by ✠. Now consider $\mathsf{Det}_{circ^{-1}}(I_n + zX)$, which results by replacing the variables x_{ij} in $\mathsf{Det}_{circ^{-1}}(X)$ by the term $0 + zx_{ij}$ or by $1 + zx_{ij}$ in case $i = j$. But this substitution preserves the property ✠.

We now argue that $\mathbf{VNC}^2$ can construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathsf{Det}^{\star}_{Taylor}(X) = \mathsf{Det}^{\#}_{Taylor}(X)$. Recall definition (16) of $\mathsf{Det}^{\#}_{Taylor}(X)$ in Section 6.3.

The only scalar in $\mathsf{Det}^{\#}_{Taylor}(X)$ is 1. In order to deal with nodes that have high d^+_{ub} values in $\mathsf{Det}^{\#}_{Taylor}(X) \restriction \gamma$ it suffices to deal with nodes in $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$, since other parts in (16) do not increase d^+_{ub} more than by a polynomial factor. For this purpose we construct using the $\mathbf{\Sigma}^B_0$-**COMP** axiom a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof that eliminates 1 from products with 1 in $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$, using the axiom $1 \cdot 1 = 1$. This is done by pointing specifically to where the 1's are in $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$; these are $\mathbf{\Sigma}^B_0$-definable number functions in $\mathbf{V}^0$. We omit the details. □

10.2 Formal Description of the Balancing Algorithm

For a syntactically homogeneous circuit G and a natural number m let

$$\mathcal{B}_m(G) := \big\{t \in G \ : \ t = t_1 \cdot t_2, \text{ where } \ d^+_{\mathrm{ub}}(t) > m \text{ and } d^+_{\mathrm{ub}}(t_1), d^+_{\mathrm{ub}}(t_2) \le m\big\}. \tag{33}$$

Notice that $\mathcal{B}_m(G)$ is a $\mathbf{\Sigma}^B_0$-definable relation in $\mathbf{V}^0$.

Note: In the construction of the balanced circuit of F, given nodes $v, w \in F$, the notation $[F_v]$ and $[\partial w f_v]$ stand for ***nodes*** (and not circuits). When we write $[F_v] := C$ for a circuit C we mean that the node $[F_v]$ is defined to be the root of the circuit C, where C possibly contains other (previously constructed) nodes like $[F_u]$, for some $u \in F$. In other words, the algorithm simply connects the node $[F_v]$ to a circuit for which some of its leaves are already constructed nodes.

$\mathbf{FNC}^2$-Algorithm for Balancing a Circuit F (Construction of $[F]$)

Input: F where F is a sum of one or more syntactic-homogeneous circuits over the variables $x_1, \dots, x_n$, in which every node u appears with its syntactic-degree upper bound $d^+_{\text{ub}}(u)$.

Output: A circuit $[F]$ computing the polynomial $\widehat{F}$. That is, if s is the size of F then $\text{depth}([F]) = O(\log s \log d + \log^2 d)$ and the size of $[F]$ is $\text{poly}(s, d)$, where $d = d^+_{\text{ub}}(F)$.

Preprocessing step: For every pair of nodes w, v we prepare a list that determines whether w is in F_v. This is done by running in parallel for all pairs w, v in F the $\mathbf{NC^2}$-algorithm in Lemma 10.4 for checking if w is in F_v described

Step $i = 0$:

Part (a):[10] We construct the node $[F_v]$, for all nodes $v \in F$ such that $d^+_{\text{ub}}(v) \le 1 = 2^i$.

Let $v \in F$ be such that $d^+_{\text{ub}}(v) \le 1$.

Claim 10.8. *$\widehat{F_v} = a_1x_1 + \dots + a_nx_n + \sum_{c \in J} b_c c$, for $a_1, \dots, a_n, b_c \in \mathbb{Z}$ and $J \subset \mathbb{Z}$. Furthermore, there exists an $\mathbf{FNC}^2$-construction that given F constructs the depth $O(\log n)$ circuit $a_1x_1 + \dots + a_nx_n + \sum_{c \in J} b_c c$.*

Proof of claim: Since $d^+_{\text{ub}}(v) \le 1$, there are no product gates in F_v. Thus, F_v is a circuit with only plus gates, which means $\widehat{F_v}$ is as stated in the claim. By Lemma 10.3 (and Theorem 3.5) we can construct in $\mathbf{FNC}^2$ the circuit $a_1x_1 + \dots + a_nx_n + \sum_{c \in J} b_c c$ (we do not evaluate the circuit). ■Claim

We define

$$[F_v] := a_1x_1 + \dots + a_nx_n + \sum_{c \in J} b_c c.$$

Part (b): Let w, v be a pair of nodes in F with $2d^+_{\text{ub}}(w) > d^+_{\text{ub}}(v)$:

Case 1: Assume w is not a node in F_v (this can be checked using the list from the preprocessing step). Define

$$[\partial w f_v] := 0.$$

Case 2: Assume that w is in F_v and $0 \le d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \le 1$. Again, this is checked by the list from the preprocessing step, and since we the input circuit F is assumed to contain the value of d^+_{ub} for each node.

Thus, by Proposition 10.2, the polynomial $\partial w f_v$ is a linear polynomial $a_1x_1 + \dots + a_nx_n + b$. Using Lemma 10.5 and similar notation and reasoning as Claim 10.8 define

$$[\partial w f_v] := a_1x_1 + \dots + a_nx_n + \sum_{c \in J} b_c c\,.$$

Step $i + 1$:

[10] This base case uses an $\mathbf{FNC}^2$ algorithm, but since it is done only in the base case, the whole algorithm still is in $\mathbf{FNC}^2$.

The construction in this step is done in $\mathbf{V}^0$, assuming we have the list from the preprocessing step above.

Part (a): Assume that for some $0 \le i \le \lceil \log(d) \rceil$:

$$2^i < d^+_{\rm ub}(v) \le 2^{i+1}.$$

Put $m = 2^i$, and define (recall that here $[\partial w f_v]$, $[F_{t_1}]$ and $[F_{t_2}]$ are *nodes*)

$$[F_v] := \sum_{\substack{t \in \mathcal{B}_m(F_v) \\ t = t_1 \cdot t_2}} [\partial t f_v] \cdot [F_{t_1}] \cdot [F_{t_2}] \,.$$

Part (b): Let w, v be a pair of nodes in F with $2d^+_{\rm ub}(w) > d^+_{\rm ub}(v)$:

Assume that w is in F_v and that for some $0 \le i \le \lceil \log(d) \rceil$:

$$2^i < d^+_{\rm ub}(v) - d^+_{\rm ub}(w) \le 2^{i+1}.$$

Put $m = 2^i + d^+_{\rm ub}(w)$. Define:

$$[\partial w f_v] := \sum_{t \in \mathcal{B}_m(F_v)} [\partial t f_v] \cdot [\partial w f_{t_1}] \cdot [F_{t_2}] \,,$$

where here for every given $t \in \mathcal{B}_m(F_v)$, t_1, t_2 are nodes such that $t = t_1 \cdot t_2$ and $d^+_{\rm ub}(t_1) \ge d^+_{\rm ub}(t_2)$, or $t = t_2 \cdot t_1$ and $d^+_{\rm ub}(t_2) < d^+_{\rm ub}(t_1)$.

Finally, define $[F]$ as the circuit with output node $[F_u]$, where u is the output node of F.

By construction, the algorithm computes the correct output: the fact that $[F]$ has the correct depth stems from the construction as explained in the overview of the balancing algorithm above (see also [VSBR83, RY08, HT15]). The fact that $[F]$ has the correct size stems from the fact that the algorithm is Σ^B_0-definable in $\mathbf{VNC}^2$. The fact that $[F]$ computes $\widehat{F}$ is shown below by constructing in $\mathbf{VNC}^2$ a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = [F]$ for a syntactic homogeneous circuit F (this stems from Lemma 10.11; see again [VSBR83, RY08, HT15]).

Note that given an algebraic circuit F with $d^+_{\rm ub}(u)$, for all nodes u in F, polynomially bounded, our balancing algorithm provides a way to balance F already in $\mathbf{FNC}^2$. As mentioned in Section 2, by first balancing an input circuit and then evaluating it (assuming e.g. it is over the integers, as in the next section) this gives rise to an $\mathbf{NC^2}$ evaluation procedure for algebraic circuits of any depth (given as input an upper bound on their syntactic-degree in unary and assuming the syntactic degree $d^+_{\rm ub}$ of the circuit is polynomial) that is different from the previously known algorithm by Miller *et al.* [MRK88] (their algorithm does not require the syntactic-degree as input) and that of Allender *et al.* [AJMV98] (which is implicit in that work but can be extracted from the text [All18]).

10.3 Balancing Proofs in $\mathbf{VNC}^2$

For balancing $\mathbb{P}_c(\mathbb{Z})$*-proofs* we need to show the proof-theoretic counterpart of the balancing algorithm. This is similar to the proof-theoretic counterpart of the homogenization theorem shown in Section 9. We start by showing some properties of the constructions of the base cases of the balancing algorithm described in Lemmas 10.3 and 10.5 that $\mathbf{VNC}^2$ can prove.

Lemma 10.9 (in $\mathbf{VNC}^2$). *(i) Let F be a circuit with no product gates and no scalars (and no division gates). Assume that $v = v_1 + v_2$ is a node in F such that $d^+_{\text{ub}}(v) \leq 1$. Then, there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $[F_v] = [F_{v_1}] + [F_{v_2}]$.*

(ii) Let F be a circuit with no scalars and syntactic-degree d, and a pair of nodes w, v in F, such that w is in F_v and $0 \leq d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$. Then, there is a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$[\partial w F_v] = [\partial w F_{v_1}] + [\partial w F_{v_2}], \qquad \textit{in case } v = v_1 + v_2; \tag{34}$$

$$[\partial w F_v] = [\partial w F_{v_1}] \cdot [F_{v_2}], \qquad \textit{in case } v = v_1 \cdot v_2 \textit{ and } d^+_{\text{ub}}(v_1) \geq d^+_{\text{ub}}(v_2) \textit{ or } v = v_2 \cdot v_1 \textit{ and } d^+_{\text{ub}}(v_1) > d^+_{\text{ub}}(v_2). \tag{35}$$

Proof. Part (i). Consider Lemma 10.3. The circuit $[F_v]$ is constructed according to this lemma by first computing the integer coefficients of each of the input variables in the linear form computed by F_v. It thus suffices to prove (in $\mathbf{VNC}^2$) that for every input variable x_i in F_v, the coefficient of x_i in F_v equals the sum of the coefficients of x_i in F_{v_1}, F_{v_2}. Assuming we can prove this, we can directly construct the $\mathbb{P}_c(\mathbb{Z})$-proof of $[F_v] = [F_{v_1}] + [F_{v_2}]$.

We use a result from Cook and Fontes [CF12], stating that the theory $V\#L$ which is contained in $\mathbf{VNC}^2$, $\mathbf{\Sigma}^B_1$-defines the string function $\text{PowSeq}_{\mathbb{Z}}(n, s, A)$. This string function receives an $n \times n$ integer matrix A and outputs a string coding the sequence of powers of A: $(A, A^2, \ldots, A^s)$.

Let $F'_{v_1} := F_{v_1} \cup \{(v_1, r)\}$ and $F'_{v_2} := F_{v_2} \cup \{(v_2, r)\}$. That is, F'_{v_1} is the (non-legit) circuit F_{v_1} to which we add the directed edge from v_1 to the output node r in F_v, and similarly F'_{v_2}, so that $F_v = F'_{v_1} \cup F'_{v_2}$. Assume that A_v, A_{v_1}, A_{v_2} are the 0-1 adjacency matrices of the circuits F_v, F'_{v_1}, F'_{v_2}, respectively, where the dimensions of all the matrices all equal s, the number of nodes in F_v and the (u, w)th entry in all three matrices corresponds to a directed edge from node u to node w. Using $\mathbf{\Sigma}^B_0$-induction on the power $i = 1, \ldots, s$, and using the strings $(A_v, A^2_v, \ldots, A^s_v), (A_{v_1}, A^2_{v_1}, \ldots, A^s_{v_1}), (A_{v_2}, A^2_{v_2}, \ldots, A^s_{v_2})$, we argue that for every input node u in F_v

$$A^i_v[u, r] = A^i_{v_1}[u, r] + A^i_{v_2}[u, r]\,,$$

where $A[u, r]$ denotes the (u, r)th entry of the matrix A, and as before r is the output node of F_v.

Part (ii). Here we use the construction in Lemma 10.5.

Case 1: $v = v_1 + v_2$. According to Lemma 10.5, $[\partial w F_v]$ is defined as the sum $\sum_{u \in U} \widehat{F}_{s_u}$, where U is the set of all product gates $u = t_u \cdot s_u$ in F_v such that F_{t_u} has (without loss of generality) in its scope w, and where we construct each F_{s_u} in the sum using Lemma 10.3, similar to part (i). Similar to part (i) we proceed by the number $\mathbf{\Sigma}^B_0$-induction on $i = 1, \ldots, s$, where s is the size of F_v to prove

$$A^i_v[u, r] = \sum_{u \in U} A^i_{s_u}[u, r]\,.$$

Case 2: $v = v_1 \cdot v_2$. This is similar to case 1. According to Lemma 10.5 and using the terminology of case 1 above, $[\partial w F_v]$ is defined as the sum $\sum_{u \in U} \widehat{F}_{s_u}$. Only that by assumption, the only product gate that has w is in its scope must be v itself (because there can be no two nested product gates with w in their scope by assumption $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$). Assume without loss of generality that v_1 has w in its scope. Then, v_2 does not have w in its scope (by assumption on degree, as explained in the proof of Lemma 10.5). Thus, $[\partial w F_v] = \sum_{u \in U} \widehat{F}_{s_u} = \widehat{F}_{v_2} = 1 \cdot [F_{v_2}] = [\partial w F_{v_1}] \cdot [F_{v_2}]$. □

Recall that the length number function $\lceil \log_2(n) \rceil$ is $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ (see [CN10]). This is the main theorem of this section:

Theorem 10.10 (in $\mathbf{VNC}^2$). *1. If F is a sum of one or more syntactic homogeneous circuits, of size s and depth t, such that $d^+_{\text{ub}}(F) = d$, then $F = [F]$ has a $\mathbb{P}_c$-proof of size* $\text{poly}(s, d)$ *and depth $O(t + \log s \cdot \log d + \log^2 d)$.*

2. Let π be a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ of syntactic-degree at most d and size s, and where every circuit is a sum of syntactic homogeneous circuits with every node appearing with its d^+_{ub} value. Then, $[F] = [G]$ has a $\mathbb{P}_c$-proof of size $\text{poly}(s, d)$ *and depth $O(\log s \cdot \log d + \log^2 d)$.*

Theorem 10.10 will be proved analogously to Theorem 9.2: the proof is similar to the proof of Theorem 9.2, only that instead of using Lemma 9.3 we use the analogous Lemma 10.11 below that demonstrates some essential properties of $[F]$ that have short $\mathbb{P}_c(\mathbb{Z})$-proofs.

Lemma 10.11 (in $\mathbf{VNC}^2$). *Let F_1, F_2 be syntactic homogeneous circuits of syntactic degree at most d and size at most s. Then, there exist $\mathbb{P}_c(\mathbb{Z})$-proofs of:*

$$[F_1 \oplus F_2] = [F_1] + [F_2] \,, \tag{36}$$

$$[F_1 \otimes F_2] = [F_1] \cdot [F_2] \,, \tag{37}$$

such that the proofs have size $\text{poly}(s, d)$ *and depth $O(\log d \cdot \log s + \log^2 d)$. Furthermore, $[z] = z$ has a constant-size proof whenever z is a variable or an integer.*

The proof of Theorem 10.10 is deferred to Section 10.3.2.

10.3.1 Proof of Lemma 10.11

We now prove Lemma 10.11. The proof is similar to Lemma 4.4 in [HT15], except that we use $d^+_{\text{ub}}(\cdot)$ instead of syntactic-degrees $d(\cdot)$ and that we construct the $\mathbb{P}_c(\mathbb{Z})$-proof in $\mathbf{FNC}^2$ instead of by induction on the structure of F (which would have necessitate using $\mathbf{\Sigma}_1^B$-induction).

The statement concerning $[z] = z$ is clear: if z is an integer, $[z]$ and z are the same circuit. If z is a variable, $[z]$ is the circuit $1 \cdot z$.

We need to construct proofs of equations (36) and (37).

Let $m(s, d)$ and $r(s, d)$ be functions such that for any circuit F with $d^+_{\text{ub}}(F) = d$ and size s, $[F]$ has depth at most $r(s, d)$ and size at most $m(s, d)$. Since the balancing algorithm shown above is $\mathbf{\Sigma}_0^B$-definable in $\mathbf{VNC}^2$ we can choose

$$m(s, d) = \text{poly}(s, d) \ \text{ and } \ r(s, d) = O(\log^2 d + \log d \cdot \log s).$$

Notation: In the following, $[F_v]$ and $[\partial w F_v]$ will denote *circuits*: $[F_v]$ and $[\partial w F_v]$ are the subcircuits of $[F]$ with output nodes $[F_v]$ and $[\partial w F_v]$, respectively; the defining relations between the nodes of $[F]$ (see the definition of $[F]$ above) translate to equalities between the corresponding circuits. For example, if v and m are as in part (a) Case 2, of the definition of $[F]$, then, using just the axioms C1 and C2, we can prove

$$[F_v] = \sum_{t \in \mathcal{B}_m(F_v)} [\partial t F_v] \cdot [F_{t_1}] \cdot [F_{t_2}] \,. \tag{38}$$

Here, the left hand side is understood as the circuit $[F_v]$ in which $[\partial t F_v], [F_{t_1}], [F_{t_2}]$ appear as *subcircuits*, and so can share common nodes, while on the right hand side the circuits have *disjoint nodes*.

Also, note that if F has size s and degree d, the proof of (38) has size $O(s^2 m(s,d))$ and has depth $O(r(s,d))$. We shall use these kind of identities in the current proof.

Let $\lambda(s,i)$ be a function such that

$$\lambda(s,0) = O(s^4) \quad \text{and} \quad \lambda(s,i) \leq O(s^4 \cdot m(s,d)) + \lambda(s,i-1). \tag{39}$$

Recurrence (39) implies $\lambda(s,d) = \text{poly}(s,d)$.

The following proposition (which is similar to Proposition 4.10 in [HT15]) suffices to conclude the lemma (it is enough to take F in the statement as either $F_1 \oplus F_2$ or $F_1 \otimes F_2$, and v as the root of F).

Proposition 10.12 (in $\mathbf{VNC}^2$)**.** *Let F be a syntactically homogenous circuit of syntactic degree at most d and size s. For every $i = 0, \ldots, \lceil \log d \rceil$ there exists a $\mathbb{P}_c$ proof-sequence Ψ_i of size at most $\lambda(s,i)$ and depth at most $O(r(s,d))$, such that the following hold:*

Part (a): *For every node $v \in F$ with*

$$d^+_{\text{ub}}(v) \leq 2^i, \tag{40}$$

Ψ_i contains the following equations:

$$[F_v] = [F_{v_1}] + [F_{v_2}]\,, \qquad \text{in case } v = v_1 + v_2, \quad \text{and} \tag{41}$$

$$[F_v] = [F_{v_1}] \cdot [F_{v_2}]\,, \qquad \text{in case } v = v_1 \cdot v_2. \tag{42}$$

Part (b): *For every pair of nodes $w \neq v \in F$, where $w \in F_v$, and with*

$$d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 2^i \quad \text{and} \tag{43}$$

$$2d^+_{\text{ub}}(w) > d^+_{\text{ub}}(v), \tag{44}$$

Ψ_i contains the following equations:

$$[\partial w F_v] = [\partial w F_{v_1}] + [\partial w F_{v_2}], \qquad \text{in case } v = v_1 + v_2; \tag{45}$$

$$[\partial w F_v] = [\partial w F_{v_1}] \cdot [F_{v_2}], \qquad \text{in case } v = v_1 \cdot v_2 \text{ and } d^+_{\text{ub}}(v_1) \geq d^+_{\text{ub}}(v_2) \text{ or } v = v_2 \cdot v_1 \text{ and } d^+_{\text{ub}}(v_1) > d^+_{\text{ub}}(v_2). \tag{46}$$

Proof. Similar to previous constructions the idea is to construct all parts of the $\mathbb{P}_c(\mathbb{Z})$-proof *simultaneously* in $\mathbf{VNC}^2$. This is done in an analogue manner to the balancing algorithm above.

Step $i = 0$. We need to devise the proof sequence Ψ_0.

Part (a): proof of (41). Let $d^+_{\text{ub}}(v) \leq 2^0$. By definition, $[F_v] = \sum_{i=1}^n a_i x_i + b$, where a_i's are integers and b is a sum of constant integers. Further, by construction $[F_v]$ does not contain product gates and thus $v = v_1 + v_2$, and we need to prove only (41). This stems from Lemma 10.9 part (i).

Part (b): proof of (45) and (46). Similarly to part (a) above, this follows from Lemma 10.9 part (ii).

Overall, Ψ_0 will be the union of all the above proofs, so that Ψ_0 contains all equations (41) (for all nodes v satisfying (40)), and all equations (45) and (46) (for all nodes v, w satisfying (43) and (44)). The proof sequence Ψ_0 has size $\lambda(s,0) = O(s^4)$ and has depth $O(\log s)$.

Step $i+1$: We wish to construct the proof-sequence Ψ_{i+1}.

Part (a): proof of (41) and (42). Let v be any node in F such that

$$2^i < d^+_{\mathrm{ub}}(v) \leq 2^{i+1}.$$

Case 1: Assume that $v = v_1 + v_2$. We show how to construct the proof of $[F_v] = [F_{v_1}] + [F_{v_2}]$. Let $m = 2^i$. From the construction of $[\cdot]$ we have:

$$[F_v] = [F_{v_1+v_2}] = \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t(F_{v_1+v_2})] \,. \tag{47}$$

Since $d^+_{\mathrm{ub}}(v_1) = d^+_{\mathrm{ub}}(v_2) = d^+_{\mathrm{ub}}(v)$, we also have

$$[F_{v_e}] = \sum_{t \in \mathcal{B}_m(F_{v_e})} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t(F_{v_e})], \quad \text{for } e \in \{1,2\} \,. \tag{48}$$

If $t \in \mathcal{B}_m(F_v)$ then $d^+_{\mathrm{ub}}(t) > m = 2^i$. Therefore, for any $t \in \mathcal{B}_m(F_v)$, since $d^+_{\mathrm{ub}}(v) \leq 2^{i+1}$, we have $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(t) < 2^i$ and $2d^+_{\mathrm{ub}}(t) > d^+_{\mathrm{ub}}(v)$ and $t \neq v$ (since t is a product gate). Thus, by construction, the proof-sequence Ψ_i contains, for any $t \in \mathcal{B}_m(F_v)$, the equations

$$[\partial t(F_{v_1+v_2})] = [\partial t F_{v_1}] + [\partial t F_{v_2}],$$

and we can compute the positions of these proof-lines in the string encoding of Ψ_i (using some natural encoding). Therefore, pointing to these proof-lines in Ψ_i as premises, we construct a $\mathbb{P}_c(\mathbb{Z})$-proof that (47) equals:

$$\begin{aligned} &\sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot ([\partial t F_{v_1}] + [\partial t F_{v_2}]) \\ &= \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_1}] + \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_2}]. \end{aligned} \tag{49}$$

If $t \in \mathcal{B}_m(F_v)$ and $t \notin F_{v_1}$ then $[\partial t F_{v_1}] = 0$. Similarly, if $t \in \mathcal{B}_m(F_v)$ and $t \notin F_{v_2}$ then $[\partial t F_{v_2}] = 0$. Hence we can prove

$$\sum_{t \in \mathcal{B}_m(F_v)} [\partial t F_{v_e}] = \sum_{t \in \mathcal{B}_m(F_{v_e})} [\partial t F_{v_e}], \quad \text{for } e = 1, 2. \tag{50}$$

Thus, using (48) we have that (49) equals:

$$\begin{aligned} &\sum_{t \in \mathcal{B}_m(F_{v_1})} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_1}] + \sum_{t \in \mathcal{B}_m(F_{v_2})} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_2}] \\ &\qquad = [F_{v_1}] + [F_{v_2}]. \end{aligned} \tag{51}$$

The above proof of (51) from Ψ_i has size $O(s^2 \cdot m(s,d))$ and depth $O(r(s,d))$.

The proof of Case 2 where $v = v_1 \cdot v_2$, and the proofs of Part (b) for equations (45) and (46) are similar to Case 1 above, and are identical to those cases in the proof of Proposition 4.10 in [HT15]; like Case 1, the only difference is that we construct with an $\mathbf{FNC}^2$ construction all the $\mathbb{P}_c(\mathbb{Z})$-proofs Ψ_i together, for every $i = 0, \ldots, \lceil \log d \rceil$, where in Ψ_{i+1} we point to proof-lines that appear in Ψ_i (whose position can be computed using a reasonable encoding scheme for proof-lines). For self containment we put these cases in the appendix Section D. □

This concludes the proof of Proposition 10.12, and hence of Lemma 10.11.

10.3.2 Proof of Theorem 10.10

Proof of Theorem 10.10. **Part (i).** We use the balancing algorithm above and Lemma 10.11 to construct for every node v in F a (part of) the proof of $[F_v] = F_v$, simultaneously. We can use the balancing algorithm because F is a sum of syntactic homogeneous circuits with all nodes appearing together with their associated syntactic-degree upper bound d^+_{ub} .

Case 1: For a leaf u we construct the equation $u = u$, which is correct since $[u] = u$.

Case 2: For $v = v_1 \circ v_2$, where $\circ \in \{+, \cdot\}$, Lemma 10.11 gives $[F_v] = [F_{v_1}] \circ [F_{v_2}]$. We then point to the equations $[F_{v_i}] = F_{v_i}$, $i \in \{1, 2\}$, which gives a proof of $[F_v] = F_{v_1} \circ F_{v_2} = F_v$.

The proof has size poly(s, d). The depth of the proof never exceeds the depth of F and the depth of the proofs of $[F_v] = [F_{v_1}] \circ [F_{v_2}]$.

Part (ii). We assumed that π is a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ of syntactic-degree at most d and size s, in which every circuit is a sum of syntactic homogeneous circuits with every node appearing with its d^+_{ub} value. Similar to previous constructions, we are going to simultaneously construct a (part of a) $\mathbb{P}_c(\mathbb{Z})$-proof of $[F_1] = [F_2]$ using pointers to previous lines (that we can compute in parallel), for every proof-line $F_1 = F_2$ in π. This resembles the proof structure of Theorem 9.2. Like part (i), we can use the balancing algorithm because by assumption each of the circuits F_1, F_2 is given to us as a sum of syntactic homogeneous circuits with all nodes appearing together with their associated syntactic-degree upper bound d^+_{ub}.

Let m_0 and k_0 be such that (36) and (37) have $\mathbb{P}_c(\mathbb{Z})$-proofs of size at most m_0 and depth k_0, whenever $F_1 \oplus F_2$, respectively, $F_1 \otimes F_2$ have size at most s' and syntactic degree at most d.

Case 1: $F = H$ is an axiom of $\mathbb{P}_c(\mathbb{Z})$. Then, $[F] = [H]$ has a $\mathbb{P}_c$-proof of size $c_1 m_0$ and depth $c_2 k_0$, where c_1, c_2 are some constants independent of s', d. The axiom A1 is immediate and the axiom A10 follows from the fact that $[F] = \widehat{F}$, for $F = c$, $c \in \mathbb{Z}$.. The rest of the axioms are an application of Lemma 10.11, as follows. Axioms C1 and C2 are already the statement of Lemma 10.11. For the other axioms, take, for example,

$$F_1 \cdot (G_1 + G_2) = F_1 \cdot G_1 + F_1 \cdot G_2 \,.$$

We are supposed to give a proof of

$$[F_1 \cdot (G_1 + G_2)] = [F_1 \cdot G_1 + F_1 \cdot G_2] \,,$$

with a small depth. By Lemma 10.11 we have a $\mathbb{P}_c(\mathbb{Z})$-proof

$$[F_1 \cdot (G_1 + G_2)] = [F_1] \cdot [G_1 + G_2] = [F_1] \cdot ([G_1] + [G_2]) = [F_1] \cdot [G_1] + [F_1] \cdot [G_2] \,.$$

Lemma 10.11 gives again:

$$[F_1] \cdot [G_1] + [F_1] \cdot [G_2] = [F_1 \cdot G_1] + [F_1 \cdot G_2] = [F_1 \cdot G_1 + F_1 \cdot G_2].$$

Here we applied Lemma 10.11 to circuits of size at most s', and the proof of $[F_1 \cdot (G_1 + G_2)] = [F_1 \cdot G_1 + F \cdot G_2]$ has size at most $c_0 m_0$ and depth at most $c_1 k_0$, for some constants c_0, c_1.

Case 2: An application of rules R1, R2 translates to an application of R1, R2. For the rules R3 and R4, it is sufficient to show the following: if π uses the rule

$$\frac{F_1 = F_2 \qquad G_1 = G_2}{F_1 \circ G_1 = F_2 \circ G_2}, \ \circ \in \{\cdot, +\},$$

then there is a proof of $[F_1 \circ G_1] = [F_2 \circ G_2]$, of size $c_1 m_0$ and depth $c_2 k_0$, from the equations $[F_1] = [G_1]$ and $[F_2] = [G_2]$. This is again an application of Lemma 10.11.

Altogether, we obtain a proof of $[F] = [G]$ of size at most $c_1 s' m_0$ and depth $c_2 k_0$. □

As a corollary of Theorem 10.10 and Corollary 10.7 we finally obtain the balanced PI-proof of the determinant identities in $\mathbf{VNC}^2$. Denote by $\mathsf{Det}_{balanced}$ the circuit obtained by applying the balancing algorithm on $\mathsf{Det}^{\star}_{Taylor}(X)$. That is,

$$\mathsf{Det}_{balanced}(X) := [\mathsf{Det}^{\star}_{Taylor}(X)]. \tag{52}$$

Corollary 10.13 (in $\mathbf{VNC}^2$)**.** *Given a positive natural number n there exists a $O(\log^2 n)$-depth $\mathbb{P}_c(\mathbb{Z})$-proof of the determinant identities* (5) *and* (6)*, where the determinant in* (5) *and* (6) *is written as the division free circuit $\mathsf{Det}_{balanced}(A)$, for A, the $n \times n$ symbolic matrix X or Y, or their product XY, or a symbolic triangular matrix Z.*

11 Applying the Reflection Principle and Wrapping Up

Here we conclude the proofs of the determinant identities in the theory by proving and applying the reflection principle for $\mathbb{P}_c(\mathbb{Z})$-proofs in $\mathbf{VNC}^2$ (Theorem 4.1).

The Determinant Function DET **in** $\mathbf{VNC}^2$**.** As presented in the introduction Section 4, given an $n \times n$ integer matrix A, the determinant function DET(A) in $\mathbf{VNC}^2$ is defined to first construct an $O(\log^2 n)$-depth algebraic circuit for the determinant polynomial of a symbolic $n \times n$ matrix, and then evaluate the circuit under A, using the fact that the evaluation of $O(\log^2 n)$-depth algebraic circuits over the integers is definable in the theory as shown below.

Formally, the balanced circuit for the determinant constructed by DET is the circuit $\mathsf{Det}_{balanced}(X)$ from (52). This construction was shown above (when constructing the PI-proofs).

11.1 Algebraic NC^2-Circuit Value Problem

We show that there is an $\mathbf{FNC}^2$ algorithm that receives an *algebraic* circuit over $\mathbb{Z}$ with n input variables, size polynomial in n and depth $O(\log^2 n)$, together with an assignment of integers to the variables written as binary strings, and outputs the value of the circuit under the assignment.

The algorithm proceeds as follows: **i**) convert the input balanced algebraic circuit into a balanced Boolean circuit computing the same polynomial, where integers are written as binary strings; **ii**) layer the circuit; **iii**) convert the layered circuit into a monotone circuit; **iv**) evaluate the balanced monotone Boolean circuit using the evaluation function for such circuits which is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$.

Step (i): from balanced algebraic circuits to balanced Boolean circuits. We show how to transform a polynomial-size $O(\log^2 n)$-depth algebraic circuit into a polynomial-size $O(\log^2 n)$-depth Boolean circuit with an $\mathbf{FNC}^2$ algorithm. We use the following two facts:

Fact 11.1 (By Vinay [Vin91])**.** *Given an algebraic circuit of* poly(n)*-size and* poly(n)*-degree, our algorithm (and the original [VSBR83] algorithm) that balances the circuit into $O(\log^2 n)$-depth, in fact balances (with straightforward modifications) the circuit into $O(\log n)$-depth* in which the plus gates have unbounded fan-in *(and product remains a binary operation).*

Fact 11.2. *The Boolean (multi-valued) function* StringAdd *computing the addition of two integers written in binary is in* $\mathbf{FO}$*-uniform* $\mathbf{FAC}^0$ *(see [CN10, p. 85]). The Boolean (multi-valued) function* StringMult *computing the product of two integers written in binary is in* $\mathbf{FO}$*-uniform* $\mathbf{FTC}^0$ *(see [CN10, IX.3.6]).*

From Fact 11.2 we conclude that in $\mathbf{VNC}^2$ we can construct a constant depth fan-in two circuit for addition of two binary integers. Since a plus gate of unbounded fan-in can be simulated by a polynomial-size and $O(\log n)$-depth circuit of plus gates only, we get that in $\mathbf{VNC}^2$ we can construct a polynomial-size fan-in two Boolean circuit of $O(\log n)$-depth for computing iterated addition of binary integers.

Using Fact 11.1 above, given an $O(\log^2 n)$-depth algebraic circuit we have the following $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{VNC}^2$ for constructing the corresponding polynomial-size $O(\log^2 n)$-depth (fan-in two) Boolean circuit:

1. Every unbounded fan-in plus gate is replaced by a polynomial-size fan-in two and depth $O(\log n)$ circuit computing the corresponding iterated sum of integers;

2. Every fan-in two product gate is replaced by a polynomial-size and depth $O(\log n)$ circuit computing the corresponding product of two integers.

The resulting Boolean circuit is thus an $O(\log^2 n)$-depth circuit (with a fan-in two) and polynomial-size in n. This Boolean circuit is encoded in the same way as algebraic circuits are encoded; namely, via the encoding scheme in Section 5.1.1 (with the obvious modifications: instead of designating $+, \cdot$ we designate $\wedge, \vee, \neg$).

Step (ii): layering Boolean circuits. For the evaluation of Boolean circuits in the theory we need to have circuits that are layered, namely in which every node belongs to a single layer i, and nodes in layer i may only go to nodes in layer $i+1$. We can convert within $\mathbf{FNC}^2$ any $O(\log^2 n)$-depth Boolean circuit from Step (i) above into a layered Boolean circuit, as follows.

$\mathbf{FNC}^2$-algorithm for layering balanced Boolean circuits

Input: A Boolean circuit F of depth $c \log^2 n$, for some constant c (encoded as in Section 5.1.1).

Output: A layered Boolean circuit F' computing the same function as F.

Algorithm

1. Let A be the 0-1 adjacency matrices of F where the dimensions of A equal s, the number of nodes in F and the (u, w)th in A, denoted $A[u, w]$ is 1 iff there is a directed edge from node u to node w in F. Using the $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ string function $\mathrm{PowSeq}_{\mathbb{Z}}(n, s, A)$, that receives an $n \times n$ integer matrix A and outputs a string coding the sequence $(A, A^2, \dots, A^s)$ of powers of A, we find the shortest length of a directed path from a leaf in F to each of the internal nodes in F: the shortest directed length of a path from a leaf u to a node v is the minimal i such that $A^i[u, v] \neq 0$.

2. Let F' be the circuit F in which for every node $u \in V$, for V the set of nodes of F, change u to (u, ℓ), where ℓ is the shortest directed length of a path from a leaf in F to u. Hence, ℓ will serve as *the layer* of (u, ℓ) in F'.

3. We now add dummy edges and nodes "$1 \cdot u$" to F', to force every node u to have edges directed only to subsequent layers. Specifically, we scan the nodes of F' from layer 0 to the top layer $c \log^2 n$, and for each node (u, ℓ) that is connected with a directed edge e to node (v, j), for $j > \ell + 1$, we discard e and add two new nodes and three new edges as follows. Assuming that $(v, j) = (u, \ell) \circ w$, for $\circ \in \{+, \cdot\}$, let $(v, j) = ((u, \ell) \cdot 1) \circ w$, where the new node 1 is on layer ℓ, the new node $\cdot$ is on layer $\ell + 1$ and two new edges are added from (u, ℓ) to $\cdot$ and from 1 to $\cdot$, and a third edge is added from $\cdot$ to (v, j). After this the node (u, ℓ) has a directed edge only to nodes in layer $\ell + 1$. Doing this sequentially for all $c \log^2 n$ layers we end up with a layered circuit F'.

Step (iii): convert layered circuits into a monotone circuits. Here we need to apply sequentially De Morgan rules, from top layer to bottom layer, until all negation in the circuit are in the input level. There is no need to add new layers, since the De Morgan rules preserve the number of layers: $\neg(A \wedge B) \to \neg A \vee \neg B$, $\neg\neg A \to \mathsf{true} \wedge (\mathsf{true} \wedge A)$, $\neg(A \vee B) \to \neg A \wedge \neg B$.

For balanced circuits this is done in $\mathbf{FNC}^2$ precisely the same way as Part 3 in the algorithm in Step (ii) above, only that we start from the top layer to layer 0.

Step (iv): evaluation of balanced monotone Boolean circuits. We define the function $\mathsf{Eval}_{al}(F, A)$ that receives the string variable F encoding an algebraic circuit over the integers and an assignment of integers to the variables of F written as a two-dimensional array A, and outputs the binary string representing the value of the algebraic circuit encoded by F under A.

Let us denote by Eval_{lmb} the $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ function that evaluates a layered and monotone Boolean circuit of depth $O(\log^2 n)$ as shown in (3) (Section 3.3). By Steps (i) to (iii) and using Eval_{lmb} we conclude that Eval_{al} is $\mathbf{\Sigma}_1^B$-definable string function in $\mathbf{VNC}^2$. Note that the input Boolean variables are both the binary strings representing the integers input A and the negation of these binary strings (we need their negation because this is the input to the *monotone* circuit.)

11.2 Proving the Reflection Principle for $\mathbb{P}_c(\mathbb{Z})$

We shall prove the following reflection principle for $\mathbb{P}_c(\mathbb{Z})$:

Theorem 11.3. *(Theorem 4.1 restated; In $\mathbf{VNC}^2$) Let π be an $O(\log^2 n)$-depth $\mathbb{P}_c(\mathbb{Z})$-proof of the circuit equation $F = G$. Then $F = G$ is true in $\mathbb{Z}$, in the sense that $\forall A \in \mathbb{Z}^n\, (\mathsf{Eval}_{al}(F, A) = \mathsf{Eval}_{al}(G, A))$.*

Proof. The proof proceeds by the *number* induction (see Proposition 3.2) on the number of proof lines in π, using Lemma 11.4 below.

Since the evaluation function Eval_{al} is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ we can use this function in the number induction axiom (see Section A.1). Speciffically, consider the $\mathbf{\Sigma}_0^B$-formula $Q(n) := \forall i \leq n\, (\mathsf{Eval}_{al}(left(\pi^{[i]}), A) = \mathsf{Eval}_{al}(right(\pi^{[i]}), A))$, where $left(\pi^{[i]})$ and $right(\pi^{[i]})$ are the left (resp. right) hand side circuit in the ith proof-line in π. Then the induction states that assuming the first line is true under an assignment A, namely, $Q(0)$, and if $Q(n) \to Q(n+1)$ is true, namely if all proof-lines $\leq n$ are true under an assignment A, then also the $(n+1)$th line is true under A—then we finish the argument since we end up with $\forall n \leq length(\pi)(Q(n))$.

It thus remains to prove each of the following cases: 1) Axioms of $\mathbb{P}_c(\mathbb{Z})$. We show that the evaluation of $\mathbb{P}_c(\mathbb{Z})$ axioms under integer assignments is universally true: $\forall \alpha \in \mathbb{Z}^n (F(\alpha) = G(\alpha))$, when $F = G$ is an axiom. For example, $F + 0 = F$ holds for every integer assignment to F; 2) The rules of $\mathbb{P}_c(\mathbb{Z})$ are sound under integer assignments. These two cases are proved in Lemma 11.4. □

Lemma 11.4. *(in* $\mathbf{VNC}^2$*) (i) Let* F_1, F_2, G_1, G_2 *be* $O(\log^2 n)$*-depth circuits, and* A *an assignment of integers to their input variables. If* $\mathsf{Eval}_{al}(F_1, A) = \mathsf{Eval}_{al}(G_1, A)$ *and* $\mathsf{Eval}_{al}(F_2, A) = \mathsf{Eval}_{al}(G_2, A)$ *then* $\mathsf{Eval}_{al}(F_1 \circ F_2, A) = \mathsf{Eval}_{al}(G_1 \circ G_2, A)$*, for* $\circ \in \{+, \times\}$*. (ii) Let* F, G *be* $O(\log^2 n)$*-depth circuits,* $F = G$ *an axiom of* $\mathbb{P}_c(\mathbb{Z})$ *and* A *an assignment of integers to the input variables of* F, G*. Then,* $\mathsf{Eval}_{al}(F, A) = \mathsf{Eval}_{al}(G, A)$*.*

Proof. Part (i). Let us consider the rule $F_1 = G_1$ and $F_2 = G_2$ derives $F_1 + F_2 = G_1 + G_2$. We need to prove that $\mathsf{Eval}_{al}(F_1 + F_2, A) = \mathsf{Eval}_{al}(G_1 + G_2, A)$.

Denote by $\mathsf{StringAdd}_{fla}(X, Y)$ the Σ_0^B*-formula* from Fact 11.2 for adding two binary integers (we reserve the symbol StringAdd to denote the corresponding constant depth Boolean *circuit*). Specifically, we have (see [CN10]):

Definition 11.5 ($\mathsf{StringAdd}_{fla}$)**.** *The* Σ_0^B*-formula for computing carries in a carry-save adder is:*

$$Carry(i, X, Y) \leftrightarrow \exists k < i\big(X(k) \wedge Y(k) \wedge \forall j < i(k < j \rightarrow (X(j) \vee Y(j)))\big).$$

And the Σ_0^B*-defining axiom for* $\mathsf{StringAdd}_{fla}$ *is (where* $\oplus$ *is exclusive or):*

$$R_+(X, Y, Z) \leftrightarrow \big(|Z| \leq |X| + |Y| \wedge \forall i < |X| + |Y|(Z(i) \leftrightarrow X(i) \oplus Y(i) \oplus Carry(i, X, Y))\big).$$

The definition of the Boolean circuit StringAdd is similar to the **FO** formula in Definition 11.5: $Carry(i, X, Y)$ is defined as above except that $\exists k < i$ turns into $\bigvee_{i=0}^{k-1}$ and $\forall j < i$ turns into $\bigwedge_{j=0}^{i-1}$ and $X(i), Y(i)$ are interpreted as the ith bits of the input X, Y, that is the Boolean variable x_i, y_i, respectively. Thus, $\mathsf{StringAdd} := X(i) \oplus Y(i) \oplus Carry(i, X, Y)$ (where $\oplus$ here is built from $\wedge, \vee, \neg$).

Since $\mathsf{Eval}_{al}(F_1, A) = \mathsf{Eval}_{al}(G_1, A)$ and $\mathsf{Eval}_{al}(F_2, A) = \mathsf{Eval}_{al}(G_2, A)$,

$$\mathsf{StringAdd}_{fla}(\mathsf{Eval}_{al}(F_1, A), \mathsf{Eval}_{al}(F_2, A)) = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{al}(G_1, A), \mathsf{Eval}_{al}(G_2, A)). \tag{53}$$

Recall that by construction, $\mathsf{Eval}_{al}(F_1 + F_2, A)$ first converts the algebraic circuit $F_1 + F_2$ into a Boolean circuit of the form $\mathsf{StringAdd}(F_1', F_2')$, where F_1', F_2' are the monotone, layered and Boolean versions of F_1, F_2, respectively, as described in the algorithm above (Steps (i) to (iii)), and then evaluates it using $\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(F_1', F_2'), A)$ (and similarly for G_1, G_2) (for simplicity we shall ignore here the fact that also StringAdd is turned into a monotone circuit). Therefore, we can prove:

$$\mathsf{Eval}_{al}(F_1 + F_2, A) = \mathsf{Eval}_{lmb}(\mathsf{StringAdd}(F_1', F_2'), A), \tag{54}$$

$$\mathsf{Eval}_{al}(G_1 + G_2, A) = \mathsf{Eval}_{lmb}(\mathsf{StringAdd}(G_1', G_2'), A). \tag{55}$$

By (53), (54) and (55) it suffices to prove

$$\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(F_1', F_2'), A) = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{al}(F_1, A), \mathsf{Eval}_{al}(F_2, A)), \tag{56}$$

$$\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(G_1', G_2'), A) = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{al}(G_1, A), \mathsf{Eval}_{al}(G_2, A)). \tag{57}$$

Let us prove (56) (as (57) is similar). By the discussion above $\mathsf{StringAdd}_{fla}(\mathsf{Eval}_{al}(F_1, A), \mathsf{Eval}_{al}(F_2, A)) = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A))$. Hence, it remains to prove

$$\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(F_1', F_2'), A) = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A)). \tag{58}$$

First note that the evaluation function Eval_{lmb} ((3) in Section 3.3) works the same for *multi-output* circuits. Second, recall that the function Eval_{lmb} is defined so that given a circuit F it produces an evaluation string (as defined in (3)) for the *whole* circuit F, and then outputs the evaluation string only of the *top layer* (namely, the output nodes).

The idea of the proof of (58) is the following: consider the left hand side of (58). The evaluation string of the circuit $\mathsf{StringAdd}(F_1', F_2')$ produced by Eval_{lmb} given A is the same as the combination of the separate evaluation strings of F_1' and F_2', excluding the top layers which belongs to the evaluation of (the constant many layers of the circuit) $\mathsf{StringAdd}$. Therefore, we can prove that

$$\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(F_1', F_2'), A) = \mathsf{Eval}_{lmb}(\mathsf{StringAdd}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A)), A) \tag{59}$$

(note that in (59) the rightmost input A on the right hand side does not have any effect, since the circuit $\mathsf{StringAdd}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A))$ has no variables).

Therefore, to conclude (58) it remains to show

$$\begin{aligned}&\mathsf{Eval}_{lmb}(\mathsf{StringAdd}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A)), A)\\ &\qquad = \mathsf{StringAdd}_{fla}(\mathsf{Eval}_{lmb}(F_1', A), \mathsf{Eval}_{lmb}(F_2', A)).\end{aligned} \tag{60}$$

This is done by the number induction on the Σ_0^B-formula

$$\Psi(i, X, Y) := |X| = |Y| = i \wedge \mathsf{Eval}_{lmb}(\mathsf{StringAdd}(X, Y), A) = \mathsf{StringAdd}_{fla}(X, Y)$$

(note again that A on the left hand side does not have any effect when X, Y are interpreted as constant binary strings (devoid of variables)). This is done using the evaluation string produced for the Boolean circuit $\mathsf{StringAdd}$ by Eval_{lmb}. The idea is that the **FO** formula and the Boolean circuit for $\mathsf{StringAdd}$ are almost identical. More generally, we have the following claim that is proved by construction; and formally, by number induction on the depth of the circuit (equivalently, the number of layers in C'):

Claim 11.6. *Let C be an $\mathbf{FNC}^2$ function written in the language $\mathcal{L}_A^2$ augmented with function symbols for $\mathbf{FNC}^2$ string and number functions. Further, let C' be the corresponding monotone layered Boolean circuit for C. Then, $\mathbf{VNC}^2$ can prove that $C(A) = \mathsf{Eval}_{lmb}(C', A)$.*

The same reasoning is applied to $\mathsf{StringMult}$, for dealing with the rule $F_1 = G_1$ and $F_2 = G_2$ derives $F_1 \cdot F_2 = G_1 \cdot G_2$.

Part (ii) is similar to part (i) and we omit the details. □

11.3 Wrapping Up

Using the definition of DET, Theorem 4.1 and Corollary 10.13 we are finally in a position to conclude the main theorem.

Theorem 11.7 (Main theorem)**.** *The following determinant identities are provable in* $\mathbf{VNC}^2$*:*

$$\forall n \forall A(\mathrm{Mat}_{\mathbb{Z}}(A,n) \wedge \mathrm{Mat}_{\mathbb{Z}}(B,n) \to \mathsf{DET}(A) \cdot \mathsf{DET}(B) = \mathsf{DET}(AB)) \,, \tag{61}$$

$$\forall n \forall A(\mathrm{triangMat}_{\mathbb{Z}}(A,n) \to \mathsf{DET}(A) = A[1,1] \cdots A[n,n]) \,. \tag{62}$$

Where in (61) $\mathrm{Mat}_{\mathbb{Z}}(A,n)$ means that A is an $n \times n$ integer matrix, with integer entries are encoded by strings as usual, and $\mathrm{triangMat}_{\mathbb{Z}}(A,n)$ means that A is a lower or upper $n \times n$ triangular matrix, and $A[i,j]$ is the (i,j)th integer entry in A .

Using the translation between bounded arithmetic theories and propositional proofs as shown in [CN10] we can also extend the result in [HT15] to work over the integers:

Theorem 11.8. *There are polynomial-size propositional* $\mathbf{NC^2}$*-Frege proofs of the determinant identities over the integers.*

In Theorem 11.8, $\mathbf{NC^2}$-Frege is defined as in [HT15], namely, these are families of standard propositional (Frege) proofs with size $\mathrm{poly}(n)$ in which every proof-line is a circuit of depth $O(\log^2 n)$, and where we augment the system with rules for manipulating circuits similar to the rules C1, C2 in $\mathbb{P}_c$ (it is possible to characterize these proofs as restricted Extended Frege proofs). Moreover, integers in the $\mathbf{NC^2}$-Frege proofs are encoded by fixed length binary strings, that is sequences of propositional variables. Note that for every fixed length of binary strings encoding integers, we will have a different propositional proof.

12 Corollaries

Here we show some further theorems of linear algebra that can be proved in $\mathbf{VNC}^2$, using similar arguments as before. Specifically, we show that the Cayley-Hamilton theorem and the co-factor expansion of the determinant are provable in $\mathbf{VNC}^2$, as well as the *hard matrix identities* identified by Soltys and Cook in [SC04].

The Cayley-Hamilton (C-H) theorem states that for the (univariate) *characteristic polynomial* of a matrix A in the variable z, defined as

$$p_A(z) := \det(zI - A),$$

it holds that $p_A(A) = 0$, where $p_A(A)$ is a univariate polynomial in the matrix A, product is interpreted as matrix product, and scalar multiplication of a matrix is interpreted as usual, and where the right hand side 0 stands for the all zero matrix.

The characteristic polynomial of a matrix is defined in the theory as follows: we introduce a Σ_1^B-definable string function $p(A,n)$ that receives an $n \times n$ integer matrix A and outputs a division free $O(\log^2 n)$-depth algebraic circuit with n^2 input variables, where the coefficient of z^i, for $i = 0, \ldots, n$, in the circuit is computed (as a sub-circuit) by

$$\left[\mathrm{coeff}_{z^i}\left(\mathsf{Det}^{\#}_{Taylor}(zI_n - A)\right)\right],$$

namely, the balanced circuit that extracts the (constant) coefficient of the determinant polynomial of $zI_n - A$; recall that $\mathsf{Det}^{\#}_{Taylor}$ is a division free circuit with a polynomial syntactic-degree. Thus, overall the string function $p(A,n)$ outputs the following circuit, written as an $O(\log^2 n)$-depth circuit, for the characteristic polynomial of A:

$$p(A,n) := \sum_{i=0}^{n} \left[\text{coeff}_{z^i}\left(\mathsf{Det}^{\#}_{Taylor}(zI_n - A)\right)\right] \cdot z^i. \tag{63}$$

The C-A theorem is expressed in the theory as follows:

$$\forall n \forall A (\mathrm{Mat}_{\mathbb{Z}}(A,n) \to \text{eval}\left(p(A,n), A\right) = \mathbf{0}_n), \tag{64}$$

where $\mathbf{0}_n$ is the all zero $n \times n$ integer matrix, eval (X, A) is the string function that evaluates the circuit C under the integer assignment A, $\mathrm{Mat}_{\mathbb{Z}}(A,n)$ is the relation that holds iff A is an $n \times n$ integer matrix, and $p(\cdot)$ is $\mathbf{\Sigma}^B_1$-definable function that receives a matrix and outputs a circuit (in fact a formula) that computes its characteristic polynomial (with a single input variable z).

Corollary 12.1. *The Cayley-Hamilton theorem, expressed as in (64), is provable in* $\mathbf{VNC}^2$.

Proof. This follows the same line of arguments demonstrated for the $\mathbf{VNC}^2$-proofs of the determinant identities. We first construct using $\mathbf{\Sigma}^B_0$-**COMP** the $\mathbb{P}_c(\mathbb{Z})$-proof of the C-H theorem shown in Proposition 9.4 in [HT15] and then use the reflection principle as in Section 11. The only difference is that we need to use part (3) in Lemma 6.3 (we did not use this part before), and for this we need to supply the witnesses for the syntactic-degrees of the nodes in (63). This needs more work, and is shown in the appendix in Lemma C.1. □

Other basic results in linear algebra that are provable in $\mathbf{VNC}^2$ are the cofactor expansion of the determinant and the inversion principle, as follows.

The *inversion principle* is the following formula in $\mathbf{VNC}^2$:

$$\forall n \forall A, B(\mathrm{Mat}_{\mathbb{Z}}(A,n) \wedge \mathrm{Mat}_{\mathbb{Z}}(B,n) \to (AB = I \to BA = I)).$$

Soltys and Cook [SC04] showed that the inversion principle is equivalent in the theory LA (that can be interpreted in $\mathbf{VNC}^2$ by Cook and Fontes [CF12][11]) to the following principles they called collectively (including the inversion principle itself) *the hard matrix identities*:

$$\begin{aligned} &AB = I \wedge AC = I \to B = C \\ &AB = I \to AC \neq 0 \vee C = 0 \\ &AB = I \to A^t B^t = I\,. \end{aligned}$$

Corollary 12.2. *The inversion principle is provable in* $\mathbf{VNC}^2$.

For an $n \times n$ matrix X let $X[i|j]$ be the $(n-1) \times (n-1)$ minor obtained by removing the ith row and jth column from X (recall that a sum of integer numbers represented in binary is definable in $\mathbf{VNC}^2$ (cf. [CN10])).

Corollary 12.3. *The following* cofactor expansion *of the determinant is provable in* $\mathbf{VNC}^2$*:*

$$\forall n \forall A \left(\mathrm{Mat}_{\mathbb{Z}}(A,n) \to \left(\mathrm{DET}(A) = \sum_{j=1}^{n} (-1)^{i+j} A(i,j) \mathrm{DET}(A[i|j])\right)\right).$$

[11] Though here we have to be careful, because the encoding of matrices and polynomials and the determinant we introduce is different from the encoding of [SC04, CF12].

The proofs of Corollaries 12.2 and 12.3 are similar to the proof of Corollary 12.1. It uses the adjoin of a matrix $\text{Adj}(X)$ which is defined to be the $n \times n$ matrix whose (i, j)th entry is $(-1)^{i+j}\text{DET}(X[i|j])$, where DET is the determinant function ($\mathbf{\Sigma}_1^B$-defined in $\mathbf{VNC}^2$). Then we proceed as in Corollary 12.1 following Proposition 9.1 and 9.2 from [HT15].

13 Conclusions and Open Problems

We established a proof of the basic determinant identities and other basic statements of linear algebra such as the Cayley-Hamilton theorem in the weakest logical theory known to date. This answers an open question of, e.g., Cook and Nguyen [CN10]. We achieved this by formalizing in the theory $\mathbf{VNC}^2$ the construction of the PI-proof demonstrated in Hrubeš-Tzameret [HT15], and using a reflection principle for PI-proofs in the theory. Due to the central role of linear algebra and the determinant function, these results are expected to be relevant to further basic work in bounded arithmetic.

As mentioned in Section 2.2 the complexity classes $\#\mathbf{SAC}^1 \subseteq \mathbf{TC}^1$ that are above DET but below $\mathbf{NC^2}$, can compute the required depth reduction and the evaluation of algebraic circuits, and we believe that our construction can be carried out more or less the same in theories corresponding to these classes (though theories for these classes have not been investigated yet).

It will be very interesting to establish the same identities in a theory that corresponds to the complexity class DET whose complete (under $\mathbf{AC}^0$-reductions) problems are the integer determinant itself and matrix powering; such a theory denoted $V\#L$ was introduced in [CF12]. This would necessitate a completely new argument different from ours (possibly following Berkowitz' [Ber84] algorithm for the determinant) and may also contribute to the simplification of the proofs. The reason is that our argument utilizes crucially the evaluation of Boolean $\mathbf{NC^2}$-circuits in the theory, while it is not expected that such evaluation is doable in the class DET.

Acknowledgements

We thank Pavel Hrubeš for useful discussions while working on [HT15], Eric Allender for very helpful correspondence regarding [AJMV98] and Emil Jeřabek for clearing up things about $\mathbf{AC}^1$ and $\mathbf{TC}^1$. An extended abstract of this work appeared initially at LICS 2017.

Appendix

A Definability in Bounded Arithmetic

Here we give more details on the theories $\mathbf{V}^0$ and $\mathbf{VNC}^2$. Specifically, we wish explain now how to extend the language $\mathbf{V}^0$ and $\mathbf{VNC}^2$ with new function symbols (in a conservative way; see below).

We write $\exists!y\varphi$ to denote $\exists x(\varphi(x) \wedge \forall y(\varphi(y/x) \to x = y))$, where y is a variable not appearing in φ:

Definition A.1 (Two-sorted definability)**.** *Let $\mathcal{T}$ be a theory over the language $\mathcal{L} \supseteq \mathcal{L}_A^2$ and let Φ be a set of formulas in the language $\mathcal{L}$. A* number *function f is Φ-**definable in a theory** $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, y, \vec{X})$ in Φ such that $\mathcal{T}$ proves*

$$\forall \vec{x} \forall \vec{X} \exists ! y \varphi(\vec{x}, y, \vec{X})$$

and it holds that[12]

$$y = f(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, y, \vec{X}). \tag{65}$$

A string *function F is Φ-**definable in a theory** $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that $\mathcal{T}$ proves*

$$\forall \vec{x} \forall \vec{X} \exists ! Y \varphi(\vec{x}, \vec{X}, Y)$$

and it holds that

$$Y = F(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}, Y). \tag{66}$$

Finally, a relation $R(\vec{x}, \vec{X})$ is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that it holds that

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{67}$$

The formulas (65)*,* (66)*, and* (68) *are the* defining axioms *for f, F, and R, respectively.*

Definition A.2 (Conservative extension of a theory)**.** *Let $\mathcal{T}$ be a theory in the language $\mathcal{L}$. We say that a theory $\mathcal{T}' \supseteq \mathcal{T}$ in the language $\mathcal{L}' \supseteq \mathcal{L}$ is* conservative over $\mathcal{T}$ *if every $\mathcal{L}$ formula provable in $\mathcal{T}'$ is also provable in $\mathcal{T}$.*

We can expand the language $\mathcal{L}$ and a theory $\mathcal{T}$ over the language $\mathcal{L}$ by adding symbols for arbitrary functions f (or relations R) to $\mathcal{L}$ *and their defining axioms* A_f (or A_R) to the theory $\mathcal{T}$. If the appropriate functions are definable in $\mathcal{T}$ (according to Definition A.1) then the theory $\mathcal{T} + A_f$ $(+A_R)$ is conservative over $\mathcal{T}$. This enables us to add new function and relation symbols to the language while proving statement inside a theory; as long as these function and relation symbols are definable in the theory, every statement in the original language proved in the extended theory (with the additional defining-axioms for the functions and relations) is provable in the original theory over the original language.

However, extending the language and the theory in such a way ***does not guarantee*** that one can use the new function symbols in the ***comprehension*** (and induction) axiom schemes. In other words, using the comprehension (and induction) axioms over the expanded language may lead to a theory that is not a conservative extension. Therefore, definability will not be enough for our purposes. We will show below precisely how to make sure that a function is *both* definable in the theories we work with and also can be used in the corresponding comprehension and induction axiom schemes (while preserving conservativity).

When extending the language with new function symbols we can assume that in *bounded formulas* the bounding terms possibly use function symbols from the expanded language (because any definable function in a bounded theory can be bounded by a term in the original language $\mathcal{L}_A^2$ (cf. [CN10])).

[12] Meaning, it holds semantically in the standard two-sorted model $\mathbb{N}_2$.

A.1 Introducing New Definable Functions in $\mathbf{V}^0$ and $\mathbf{VNC}^2$

Here we describe a process (presented in Section V.4. in [CN10]) by which we can extend the language $\mathcal{L}_A^2$ of $\mathbf{V}^0$ with new function symbols, obtaining a conservative extension of $\mathbf{V}^0$ *that can also prove the comprehension and induction axiom schemes in the extended language, and similarly for* $\mathbf{VNC}^2$.

First note that every relation or function symbol has an intended or standard interpretation over the standard model $\mathbb{N}_2$ (for instance, the standard interpretation of the binary function "$+$" is that of the addition of two natural numbers). If not explicitly defined otherwise, we will always assume that a defining axiom of a symbol in the language defines a symbol in a way that its interpretation in $\mathbb{N}_2$ is the standard one. Note also that we shall use the same symbol $F(\vec{x}, \vec{X})$ to denote both the function and the *function symbol* in the (extended) language in the theory.

Definition A.3 (Relation representable in a language)**.** *Let Φ be a set of formulas in a language $\mathcal{L}$ that extends $\mathcal{L}_A^2$. We say a relation $R(\vec{x}, \vec{X})$ (over the standard model) is* ***representable*** *by a formula from Φ iff there is a formula $\varphi(\vec{x}, \vec{X})$ in Φ such that in the standard two-sorted model $\mathbb{N}_2$ (and when all relation and function symbols in $\mathcal{L}$ get their intended interpretation), it holds that:*

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{68}$$

We say that a number function $f(\vec{x}, \vec{X})$ is *polynomially-bounded* if $f(\vec{x}, \vec{X}) \leq \text{poly}(\vec{x}, |\vec{X}|)$. We say that a string function $F(\vec{x}, \vec{X})$ is *polynomially-bounded* if $|F(\vec{x}, \vec{X})| \leq \text{poly}(\vec{x}, |\vec{X}|)$.

Definition A.4 (Bit-graph)**.** *Let $F(\vec{x}, \vec{X})$ be a polynomially-bounded string function. We define the* ***bit-graph*** *of F to be the relation $R(i, \vec{x}, \vec{X})$, where i is a number variable, such that*

$$F(\vec{x}, \vec{X})(i) \leftrightarrow i < t(\vec{x}, \vec{X}) \wedge R(i, \vec{x}, \vec{X}) \tag{69}$$

holds in the standard two-sorted model, for some number term $t(\vec{x}, \vec{X})$.

Definition A.5 ($\mathbf{\Sigma}_0^B$-definability from a language; Definition V.4.12. in [CN10])**.** *We say that a* number function $f(\vec{x}, \vec{X})$ is $\mathbf{\Sigma}_0^B$-definable from a language $\mathcal{L} \supseteq \mathcal{L}_A^2$*, if f is polynomially-bounded and its graph*[13] *is represented by a $\mathbf{\Sigma}_0^B(\mathcal{L})$-formula φ. We call the formula φ the* defining axiom of f*. We say that a* string function F is $\mathbf{\Sigma}_0^B$-definable from a language $\mathcal{L} \supseteq \mathcal{L}_A^2$*, if F is polynomially-bounded and its* <u>bit-graph</u> *(as in (69)) is representable by a $\mathbf{\Sigma}_0^B(\mathcal{L})$-formula φ. We call the formula φ the* defining axiom of F *or, equivalently, the* bit-defining axiom of F*.*

Note: We used the term *defining axiom of a function f* in both the case where f is defined *from a language* (Definition A.5) and in case f is definable *in the theory* (Definition A.1). In general it is important not to confuse these two notions. Nevertheless, we will show in the sequel that *for our purposes* these two notions coincide: when we define a function from a language the function will be definable also in the relevant theory, and so the defining axiom of f from the language will be the defining axiom of f in the theory (when the theory is possibly conservatively extended to include new function symbols).

The following is a definition of $\mathbf{AC}^0$ functions. This definition coincides with the definition of $\mathbf{FAC}^0$ as $\mathbf{FO}$-uniform multi-output Boolean circuit families of polynomial-size and constant depth [CN10].

[13] I.e., the relation $R(\vec{x}, \vec{X}, y)$, such that $f(\vec{x}, \vec{X}) = y$ iff $R(\vec{x}, \vec{X}, y)$ holds in the standard model.

Definition A.6 ($\mathbf{FAC}^0$)**.** *A string (number) function is in* $\mathbf{FAC}^0$ *if it is polynomially-bounded and its bit-graph (graph, respectively) is definable by a* Σ_0^B*-formula in the language* $\mathcal{L}_A^2$*.*

Definition A.7 ($\mathbf{AC}^0$-reduction)**.** *A number function* f *is* $\mathbf{AC}^0$-reducible to $\mathcal{L} \supseteq \mathcal{L}_A^2$ *iff there is a possibly empty sequence of functions* $F_1, \dots, F_k$ *such that* F_i *is* Σ_0^B*-definable from* $\mathcal{L} \cup \{F_1, \dots, F_{i-1}\}$*, for any* $i = 1, \dots, k$*, and* f *is* Σ_0^B*-definable from* $\mathcal{L} \cup \{F_1, \dots, F_k\}$*.*

We are now finally ready to describe the standard process enabling one to extend a theory $\mathcal{T} \supseteq \mathbf{V}^0$ over the language $\mathcal{L}_A^2$ (and specifically, the theories $\mathbf{V}^0$ and $\mathbf{VNC}^2$) with new function symbols, obtaining a conservative extension of $\mathcal{T}$ such that the new function symbols can be used in comprehension and induction axiom schemes in the theory (see Section V.4. in [CN10] for the proofs):

(i) If the number function f is Σ_0^B-definable from $\mathcal{L}_A^2$, then $\mathcal{T}$ over the language $\mathcal{L}_A^2 \cup \{f\}$, augmented with the defining axiom of f, is a conservative extension of $\mathcal{T}$ and we can also prove the comprehension and induction axioms for $\Sigma_0^B(f)$-formulas.

(ii) If the string function F is Σ_0^B-definable from $\mathcal{L}_A^2$, then $\mathcal{T}$ over the language $\mathcal{L}_A^2 \cup \{F\}$, augmented with the *bit-defining* axiom of F, is a conservative extension of $\mathcal{T}$ and we can also prove the comprehension and induction axioms for $\Sigma_0^B(F)$-formulas.

(iii) We can now iterate the above process of extending the language $\mathcal{L}_A^2(f)$ (or equivalently, $\mathcal{L}_A^2(F)$) to conservatively add more functions $f_2, f_3, \dots$ to the language, which can also be used in comprehension and induction axioms.

By the aforementioned and by Definition A.7, we can extend the language of a theory with a new function symbol f, *whenever* f *is* $\mathbf{AC}^0$*-reducible to* $\mathcal{L}_A^2$. This results in an extended theory (in an extended language) which is conservative, and can prove the comprehension and induction axioms for formulas in the extended language. When defining a new function in $\mathbf{V}^0$ or $\mathbf{VNC}^2$ we may simply say that it is Σ_0^B*-definable* or *bit-definable* in the theory and give its Σ_0^B-defining or bit-defining axiom (this axiom can use also previously Σ_0^B-defined (or bit defined) function symbols).

Extending the language of $\mathbf{V}^0$ and $\mathbf{VNC}^2$ with new *relation* symbols is simple: every relation $R(\vec{x}, \vec{X})$ which is representable by a $\Delta_1^1(\mathcal{L})$ formula ([CN10, Section V.4.1]), where $\mathcal{L}$ is an extension of the language with new function symbols obtained as shown above, can be added itself to the language. This results in a conservative extension of $\mathbf{V}^0$ ($\mathbf{VNC}^2$, resp.) that also proves the Σ_0^B-induction and comprehension axioms in the extended language.

A.2 Some Basic Formalizations in $\mathbf{V}^0$

In this section we show how to formalize basic objects in $\mathbf{V}^0$. Most formalizations here are routine (cf. [CN10, MT14]).

Natural number sequences of constant length For two numbers x, y let $\langle x, y\rangle := (x+y)(x+y+1)+2y$ be the *pairing function*, and let *left*(z), *right*(z) be the (Σ_0^B-definable in $\mathbf{V}^0$) projection functions of the first and second element in the pair z, respectively. We also Σ_0^B-define inductively $\langle v_1, \dots, v_k\rangle := \langle\langle v_1, \dots, v_{k-1}\rangle, v_k\rangle$, for any constant $k > 2$. Then $\mathbf{V}^0$ proves the injectivity of the pairing function and enables us handling such pairs in a standard way.

Notation: Given a number x, coding a sequence of natural numbers of length k, we write $\langle x\rangle_i^k$, for $i = 1, \dots, k$, to denote the number in the ith position in x. This is a $\mathbf{\Sigma}_0^B$-definable function in $\mathbf{V}^0$ (defined via $\mathit{left}(x)$, $\mathit{right}(x)$ functions).

Natural and integer number sequences If we wish to talk about sequences of numbers (whether natural, integers or rationals) where the *length of a sequence is non-constant*, we have to use string variables instead of number variables. Using the number-tupling function we can encode sequences as sets of numbers (recall that a string is identified with the finite set of numbers encoding it): a sequence is encoded as a string Z such that, the xth number in the sequence is y if the number $\langle x, y\rangle$ is in Z. Formally, we have the following $\mathbf{\Sigma}_0^B$-defining formula for the number function $\mathrm{seq}(x, Z)$ returning the xth element in the sequence Z:

$$\begin{aligned} y = \mathrm{seq}(x, Z) \leftrightarrow (y < |Z| \wedge Z(\langle x, y\rangle) \wedge \forall z < y \neg Z(\langle x, z\rangle)) \\ \vee (\forall z < |Z| \neg Z(\langle x, z\rangle) \wedge y = |Z|). \end{aligned} \tag{70}$$

Formula (70) states that the xth element in the sequence coded by Z is y iff $\langle x, y\rangle$ is in Z *and* no other number smaller than y also "occupies the xth position in the sequence", and that if no number occupies position x then the function returns the length of the string variable Z.

We define the number function $\mathit{length}(Z)$ to be the length of the sequence Z, as follows:

$$\ell = \mathit{length}(Z) \leftrightarrow \mathit{SEQ}(\ell, Z) \wedge \exists w < |Z| \exists j < |Z| (Z(w) \wedge w = \langle \ell - 1, j\rangle). \tag{71}$$

The defining axiom of $\mathit{length}(Z)$ states that Z encodes a sequence and is the lexicographically smallest string that encodes this sequence and that $\ell - 1$ is the largest position in the sequence which is occupied (by definition there will be no pair $\langle a, b\rangle \in Z$ with $a > \ell - 1$).

Array of strings We wish to encode a *sequence* of strings as an array. We use the function $\mathrm{RowArray}(x, Z)$ to denote the xth string in Z as follows (we follow the treatment in [CN10, Definition V.4.26, page 114]).

Definition A.8 (Array of strings)**.** *The string function* $\mathrm{RowArray}(x, Z)$*, abbreviated* $Z^{[x]}$*, is* $\mathbf{\Sigma}_0^B$*-definable in* $\mathbf{V}^0$ *using the following bit-definition:*

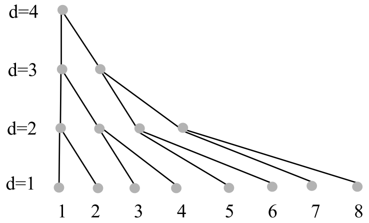
$$\mathrm{RowArray}(x, Z)(i) \ \leftrightarrow \ (i < |Z| \wedge Z(\langle x, i\rangle)).$$

Matrices An $n \times n$ integer matrix is coded as an array of n strings, where each of the n strings is itself an array that represents a row in the matrix, that is an array of n integer numbers.

A.3 Binary Tree Construction in $\mathbf{V}^0$

Here we demonstrate a $\mathbf{\Sigma}_0^B$-construction in $\mathbf{V}^0$ of a binary tree encoding. This can be used to construct a formula that computes for example the inner product of two vectors as in Section 5.1.3. Specifically, we show that the string function $F(n)$ that receives a number n, which we assume is a power of 2 for simplicity, and outputs a string that describes the edges of a binary tree with n leaves is $\mathbf{\Sigma}_0^B$-definable in $\mathbf{V}^0$.

Consider the tree shown in the picture below. Each node in the tree belongs to a single layer $d = 1, \dots, \log(n) + 1$, and in each layer d the nodes are labeled from 1 to $2^{\log(n)+1-d}$. The wires of the tree T are encoded by a three-dimensional array, namely a string E such that $E(d,u,v)$ holds iff the output of gate u on layer d is connected to the input of gate v on layer $d+1$.



To show that the string function $F(n)$ Σ_0^B-definable in $\mathbf{V}^0$ (equivalently, Σ_1^B-definable in $\mathbf{V}^0$), according to Section A.1 we need to demonstrate a Σ_0^B-formula that bit-defines the tree encoding as follows. Let

$$\Phi(d,u,v) \equiv n \le du \ \wedge\ \exists x \le n(u = 2x \to 2v = u) \wedge \\ \exists x \le n(u = 2x+1 \to 2v = u+1). \quad (72)$$

Then $\Phi(d,u,v)$ is true iff u is a node that occur in the dth layer ($n \le du$) and that if u is even then u connects to node $n/2$ in the $(d+1)$th layer, and otherwise it connects to node $\frac{u+1}{2}$ in the $(d+1)$th layer. For the bit-definition of $F(n)$ we introduce the following Σ_0^B-formula:

$$\varphi(n,i) \equiv i = (r,u,v) \wedge r < n \ \wedge\ u \le n \ \wedge v \le n \ \wedge \Phi(r,u,v)\,.$$

B Algorithm for coeff

The following is similar to the homogenization algorithm from Section 9.

Algorithm for Constructing $\text{coeff}_{z^k}(\cdot)$ in Uniform $\mathbf{FAC}^0$

Input: an arithmetic circuit C of size s and a natural number k.
Output: an arithmetic circuit computing $\text{coeff}_{z^k}(C)$.

Algorithm: Every node v in C is duplicated $k+1$ times into the nodes $[v,0],\dots,[v,k]$, such that $[v,i]$ is (the root of) a circuit computing the (polynomial) coefficient of z^i in $\widehat{C_v}$. The algorithm is doable in $\mathbf{FAC}^0$ because every new node $[v,i]$ depends only on the copies of the two nodes u,w that goes into v, and these nodes are already known from the input circuit, namely, they are $[u,i],[w,i]$, for $i=0,\dots,k+1$, where $v = u+w$ or $v = u\cdot w$ in C. Hence, the wiring of the new circuit is done in parallel for each of the new nodes as follows:
Case 0: v is a leaf in C. If $v \neq z$ then define $[v,0] = v$, and $[v,i] = 0$ for all $i = 1,\dots,k$. Otherwise, $v = z$ and we define $[v,1] = 1$, and $[v,i] = 0$ for all $1 \neq i \in \{0,\dots,k\}$.
Case 1: $v = u + w$ in C. Define $[v,i] := [u,i] + [w,i]$ for every $i = 0,\dots,k$.
Case 2: $v = u \times w$ in C. Define $[v,i] := \sum_{\substack{j+r=i \\ j,r=0,\dots,k}} [u,j] \times [w,r]$.

C Witnessing Syntactic-Degrees

Witnesses for syntactic-degrees For most part our work we do not need to witness precise syntactic-degree of nodes, since syntactic-degree upper bounds are enough. However for the

Cayley-Hamilton theorem we need to have witnesses for precise syntactic-degrees of nodes. We sketch here how to obtain such witnesses.

Note that computing the syntactic-degree of a node in a circuit is doable in $\mathbf{NC^2}$. This was noted for example by Allender *et al.* [AJMV98] (replace every scalar gate by 0, every variable gate by 1, every product gate by $+$ and every plus gate by $\max$, and then evaluate the circuit within $\mathbf{NC^2}$; e.g., using the algorithm implicit in [AJMV98], or the algorithm in [MRK88]). However, to actually use this algorithm in the theory we would need also to prove its correctness; this is likely doable (as we essentially show for the [VSBR83] circuit balancing algorithm in Section 10), but we will opt for a shorter solution: we simply witness the syntactic-degrees of all the specific circuits (and their nodes) we need.

The witness for the syntactic-degrees of nodes in a circuit is a string that stores pairs of numbers (v, d), with v the node label and d its syntactic-degree. We can store each syntactic-degree as a natural number since we will need to witness only circuits with polynomial syntactic-degrees.

It is easy to formulate a $\mathbf{\Sigma}_0^B$-formula $\phi(C, W)$ with C a circuit and W the string that contains all the syntactic-degrees of the nodes in C, such that $\phi(C, W)$ holds iff W is correct: for every addition gate $t = v_1 + v_2$ it checks that $d(t) = \max\{d(v_1), d(v_2)\}$, and for every product gate $t = v_1 \cdot v_2$ it checks that $d(t) = d(v_1) + d(v_2)$, and for leaves it checks $d(x_i) = 1$ and $d(c) = 0$ for $c \in \mathbb{Z}$.

Lemma C.1 (in $\mathbf{V}^0$)**.** *There exists a witness for the syntactic-degree of all nodes in* $\mathsf{Det}^{\#}_{Taylor}(X)$.

Proof. We show how to witness in the theory $\mathbf{V}^0$ the syntactic-degrees of the nodes in $\mathsf{Det}^{\#}_{Taylor}(X)$. Recall the definition of $\mathsf{Det}^{\#}_{Taylor}(X)$ in (16). In order to compute the syntactic-degree of nodes we do the following.

First we show that there is a $\mathbf{\Sigma}_0^B$-definable number function in $\mathbf{V}^0$ that computes the syntactic-degree of a node v, given n and the node v as inputs, where v is a node in $\mathsf{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$, assuming the syntactic-degree of v is at most n. The case for $\mathsf{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ is similar. From this, using (16), we can conclude that there is a $\mathbf{\Sigma}_0^B$-definable number function in $\mathbf{V}^0$ that computes the syntactic-degree of a node v, given n, in $\mathsf{Det}^{\#}_{Taylor}(X)$, for all nodes of syntactic-degree at most n.

Recall the encoding scheme for circuits $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ described in 5.1.3, and let d denote the "inductive level" in the definition of $\mathsf{Det}_{circ^{-1}}$ in (10). To compute the syntactic-degrees of nodes that are at most n in $\mathsf{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ we wish to compute the *pair* of numbers corresponding to the syntactic-degrees of $(\mathsf{Num}(v), \mathsf{Den}(v))$ for each node v in $\mathsf{Det}_{circ^{-1}}(I_n + zX)$.

Observe that every inductive level d in the circuit $\mathsf{Det}_{circ^{-1}}(X)$ has a "base" syntactic-degree (as a function of d), on top of which we add a number that depends on the gate we consider. For example, consider the circuit $F_1 := X_1^{-1}(I_{n-1} + \delta(X)^{-1}v_1^t v_2 X_1^{-1})$ from (9). If we know the syntactic-degree of the output nodes in level $n-1$, namely the output nodes of X_1^{-1}, then we can easily compute the syntactic-degrees of other nodes in F_1. Note however that this cannot be computed inductively in such a way within $\mathbf{V}^0$, rather we need to show the explicit number functions. Also, notice the the syntactic-degree of some nodes in F_1 is exponential because the repeated multiplication of X_1^{-1} by itself, hence we shall need to consider only those nodes whose syntactic-degree is polynomial in n.

It is enough to show that there is a $\mathbf{\Sigma}_0^B$-formula that determines the (polynomial-bounded) syntactic-degree pair $(\mathsf{Num}(v), \mathsf{Den}(v))$ of a node v, based on the inductive level d in which the node occurs and the type or position of the gate in that level. For example, some gates in F_1, for every level d, are leaves—for instance, the entries of I_{n-1} correspond to scalar leaves that have syntactic-degree pair $(0,0)$, and some others are variable leaves—for instance, $v_1^t v_2$ corresponds to an inner product with leaves variables from v_1, v_2, having syntactic-degree pair $(1,0)$, for every level d.

We demonstrate this idea on $\delta(X)^{-1}$ which is the (n,n) entry of X^{-1} of dimension $n \times n$. Similar reasoning works for the rest of the entries of X^{-1} as well as $\text{Det}_{circ^{-1}}(I_n + zX)$.

For instance, if $n = 2$ (note that $\delta(X)$ is defined for X of dimension $n \times n$, only for $n > 1$), then $\delta(X)^{-1} = (x_{22} - x_{21} \cdot x_{11}^{-1} \cdot x_{12})^{-1}$. Thus, $\text{Num}(\delta(X)^{-1}) = \text{Den}(x_{22}) \cdot \text{Den}(x_{21} \cdot x_{11}^{-1} \cdot x_{12}) = \text{Den}(x_{22}) \cdot \text{Den}(x_{21}) \cdot \text{Den}(x_{11}^{-1}) \cdot \text{Den}(x_{12}) = 1 \cdot 1 \cdot x_{11} \cdot 1$. Hence, $d(\text{Num}(\delta(X)^{-1})) = 1$.

□

Using witnesses for syntactic-degrees we can prove Lemma 6.3 part 3, which was used in the proof of Theorem 12.1.

Lemma C.2 (in $\mathbf{V}^0$). *Given a division free circuit F of syntactic-degree d and a witness for the syntactic-degrees of all nodes in F, there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = \sum_{k=0}^{d} F^{(k)}$. Moreover, $F = \sum_{i=0}^{d} \text{coeff}_{z^i}(F) \cdot z^i$ has a $\mathbb{P}_c(\mathbb{Z})$-proof.*

Proof. We shall prove the first statement (the second is similar). Note that a big sum is an abbreviation of a sum written as a logarithmic depth tree of plus gates with the summands at the leaves (we also need to use obvious steps such as applying the associativity and commutativity of addition axioms in $\mathbb{P}_c(\mathbb{Z})$-proofs of big sums).

For every node v in F we construct *simultaneously* a (partial) $\mathbb{P}_c(\mathbb{Z})$-proof sequence terminating with

$$F_v = \sum_{k=0}^{d(v)} F_v^{(k)} \tag{73}$$

as follows:
Case 1: v is a variable x_i. Then we construct a proof of $F_v := x_i = \sum_{k=0}^{d(v)} {F_v}^{(k)}$, which is immediate by construction. Similarly for a constant node.
Case 2: $v = u \oplus w$ and let $d = d(v)$. Then we use Lemma 9.3 to construct the following (partial) $\mathbb{P}_c(\mathbb{Z})$ proof-sequence. In the witnesses for this proof-sequence we add pointers to proof-lines that are constructed in parallel (for nodes that appear closer to the leaf in the tree). We can compute the line numbers to be pointed to just by looking at the current node (hence we can carry out the construction in $\mathbf{V}^0$). The pointers are constructed as number-functions by using the nodes (e.g., we can label line numbers with the nodes in F they correspond to, adding a secondary index to the index of the line).

$$\begin{aligned}
\sum\nolimits_{i=0}^{d} F_v^{(i)} &= \sum\nolimits_{i=0}^{d} (F_u \oplus F_w)^{(i)} && \text{by assumption} \\
&= \sum\nolimits_{i=0}^{d} ({F_u}^{(i)} + {F_w}^{(i)}) && \text{by Lemma 9.3} \\
&= \sum\nolimits_{i=0}^{d} {F_u}^{(i)} + \sum\nolimits_{i=0}^{d} {F_w}^{(i)} && \text{rearrangement} \\
&= F_u + F_w && \text{by ``previous'' lines} \\
& && \text{(add explicit pointers to the appropriate proof-lines)} \\
&= F_u \oplus F_w = F_v && \text{by axiom C1.}
\end{aligned}$$

Case 3: $v = u \otimes w$ and let $d_1 = d(u), d_2 = d(w)$ and $d = d(v) = d_1 + d_2$. This is similar to the Case 2 only that it is crucial here to use the specified syntactic-degrees of nodes along paths from leaves to the root.

$$
\begin{aligned}
\sum\nolimits_{i=0}^{d} F_v^{(i)} &= \sum\nolimits_{i=0}^{d} (F_u \otimes F_w)^{(i)} && \text{by assumption} \\
&= \sum\nolimits_{i=0}^{d} \sum_{\substack{l+j=i \\ 0\le l\le d_1, 0\le j\le d_2}} F_1^{(l)} \cdot F_2^{(j)} \,. && \text{by Lemma 9.3 part (3)} \\
&= \sum\nolimits_{i=0}^{d_1} F_u{}^{(i)} \cdot \sum\nolimits_{i=0}^{d_2} F_w{}^{(i)} && \text{rearrangement} \\
&= F_u \cdot F_w && \text{by "previous" lines} \\
& && \text{(add explicit pointers to the appropriate proof-lines)} \\
&= F_u \otimes F_w = F_v && \text{by axiom C1.}
\end{aligned}
$$

□

D Remaining Proof of Proposition 10.12

Proof of Proposition 10.12 continued. This is taken almost verbatim from [HT15], except that we use d^+_{ub} in instead of the true syntactic-degrees of nodes, and noticing that all predicates we use (like $t \in \mathcal{B}_m(F_v)$) are definable in $\mathbf{VNC}^2$.

Case 2: Assume that $v = v_1 \cdot v_2$. We wish to prove $[F_v] = [F_{v_1}] \cdot [F_{v_2}]$. Let $m = 2^i$. We assume without loss of generality that $d^+_{\text{ub}}(v_1) \ge d^+_{\text{ub}}(v_2)$. By the definition of $[\cdot]$, we have:

$$[F_v] = [F_{v_1 \cdot v_2}] = \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_v].$$

If $v \in \mathcal{B}_m(F_v)$, then $\mathcal{B}_m = \{v\}$ and we have $[F_v] = [F_{v_1}] \cdot [F_{v_2}] \cdot [\partial_v F_v]$. Since $[\partial_v F_v] = 1$, this gives $[F_v] = [F_{v_1}] \cdot [F_{v_2}]$, and we are done.

Otherwise, assume $v \notin \mathcal{B}_m(F_v)$. Then $m = 2^i < d^+_{\text{ub}}(v_1)$ (since, if $d^+_{\text{ub}}(v_1) \le m$, then also $d^+_{\text{ub}}(v_2) \le m$ and so by definition $v \in \mathcal{B}_m(F_v)$). Further, because $d^+_{\text{ub}}(v_1) \le 2^{i+1}$, we have

$$[F_{v_1}] = \sum_{t \in \mathcal{B}_m(F_{v_1})} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_1}] \,. \tag{74}$$

Since $d^+_{\text{ub}}(v) \le 2^{i+1}$ and $d^+_{\text{ub}}(t) > m = 2^i$, for any $t \in \mathcal{B}_m(F_v)$, we have

$$d^+_{\text{ub}}(v) - d^+_{\text{ub}}(t) \le 2^i \quad \text{and} \quad 2d^+_{\text{ub}}(t) > d^+_{\text{ub}}(v).$$

Since $v \ne t$, Ψ_i contains, for any $t \in \mathcal{B}_m(F_v)$, the equation:

$$[\partial t(F_{v_1 \cdot v_2})] = [\partial t F_{v_1}] \cdot [F_{v_2}]. \tag{75}$$

Using (75) for all $t \in \mathcal{B}_m(F_v)$, we can prove the following with a $\mathbb{P}_c(\mathbb{F})$ proof of size $O(s^2 \cdot m(s, d))$

and depth $O(r(s,d))$:

$$\begin{aligned}\sum_{t\in\mathcal{B}_m(F_v)}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial t F_v] &= \sum_{t\in\mathcal{B}_m(F_v)}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial t(F_{v_1\cdot v_2})]\\ &= \sum_{t\in\mathcal{B}_m(F_v)}[F_{t_1}]\cdot[F_{t_2}]\cdot([\partial t F_{v_1}]\cdot[F_{v_2}])\\ &= [F_{v_2}]\cdot\sum_{t\in\mathcal{B}_m(F_v)}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial t F_{v_1}]. \end{aligned} \tag{76}$$

Since $\mathcal{B}_m(F_{v_1})\subseteq\mathcal{B}_m(F_v)$, we can conclude as in (50) that

$$\sum_{t\in\mathcal{B}_m(F_v)}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial t F_{v_1}] = \sum_{t\in\mathcal{B}_m(F_{v_1})}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial t F_{v_1}]\,.$$

Using (74), (76) equals $[F_{v_2}]\cdot[F_{v_1}]$. The above proof-sequence (using Ψ_i as a premise) has size $O(s^2\cdot m(s,d))$ and depth $O(r(s,d))$.

We now append Ψ_i with all proof-sequences of $[F_v]=[F_{v_1}]+[F_{v_2}]$ for every v from Case 1, and all proof-sequences of $[F_v]=[F_{v_1}]\cdot[F_{v_2}]$ for every v from Case 2. We obtain a proof-sequence Ψ'_{i+1} of size

$$\lambda(s,i+1)\le O(s^3\cdot m(s,d))+\lambda(s,i),$$

and depth $O(r(s,d))$.

In Part (b), we extend Ψ'_{i+1} with more proof-sequences to obtain the final Ψ_{i+1}.

Part (b): proof of (45) and (46). Let $v\neq w$ be a pair of nodes in F such that $w\in F_v$ and assume that

$$2^i<d^+_{\text{ub}}(v)-d^+_{\text{ub}}(w)\le 2^{i+1}\;\;\text{and}\;\;2d^+_{\text{ub}}(w)>d^+_{\text{ub}}(v).$$

Let

$$m=2^i+d^+_{\text{ub}}(w).$$

Case 1: Suppose that $v=v_1+v_2$. We need to prove

$$[\partial w F_v]=[\partial w F_{v_1}]+[\partial w F_{v_2}] \tag{77}$$

based on Ψ_i as a premise. By construction of $[\partial w F_v]$,

$$\begin{aligned}[\partial w F_v] &= \sum_{t\in\mathcal{B}_m(F_v)}[\partial t F_v]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}]\\ &= \sum_{t\in\mathcal{B}_m(F_v)}[\partial t(F_{v_1+v_2})]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}].\end{aligned} \tag{78}$$

Since $d^+_{\text{ub}}(v_1)=d^+_{\text{ub}}(v_2)=d^+_{\text{ub}}(v)$, we also have

$$[\partial w F_{v_e}]=\sum_{t\in\mathcal{B}_m(F_{v_e})}[\partial t F_{v_e}]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}],\qquad\text{for } e=1,2\,. \tag{79}$$

Since $m = 2^i + d^+_{\text{ub}}(w)$, we have $d^+_{\text{ub}}(t) > 2^i + d^+_{\text{ub}}(w)$, for any $t \in \mathcal{B}_m(F_v)$. Thus, by $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \le 2^{i+1}$, we get that for any $t \in \mathcal{B}_m(F_v)$:

$$d^+_{\text{ub}}(v) - d^+_{\text{ub}}(t) \le 2^i \quad \text{and} \quad 2d^+_{\text{ub}}(t) > d^+_{\text{ub}}(v), \quad \text{and}$$
$$t \ne v \text{ (since } t \text{ is a product gate).}$$

Therefore, for any $t \in \mathcal{B}_m(F_v)$, Ψ_i contains the equation

$$[\partial t(F_{v_1+v_2})] = [\partial t F_{v_1}] + [\partial t F_{v_2}].$$

Thus, based on Ψ_i, we can prove that (78) equals:

$$\begin{aligned}&\sum_{t\in\mathcal{B}_m(F_v)} ([\partial t F_{v_1}] + [\partial t F_{v_2}]) \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] \\ &= \sum_{t\in\mathcal{B}_m(F_v)} [\partial t F_{v_1}] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] + \sum_{t\in\mathcal{B}_m(F_v)} [\partial t F_{v_2}] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}]. \end{aligned} \tag{80}$$

As in (50), using (79) we can derive the following from (80):

$$\begin{aligned}\sum_{t\in\mathcal{B}_m(F_{v_1})} [\partial t F_{v_1}] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] + \sum_{t\in\mathcal{B}_m(F_{v_2})} [\partial t F_{v_2}] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] \\ = [\partial w F_{v_1}] + [\partial w F_{v_2}].\end{aligned}$$

The proof of (77) from Ψ_i shown above has size $O(s^2 \cdot m(s,d))$ and depth $O(r(s,d))$.

Case 2: Suppose that $v = v_1 \cdot v_2$. We assume without loss of generality that $d^+_{\text{ub}}(v_1) \ge d^+_{\text{ub}}(v_2)$ and show how to prove

$$[\partial w F_v] = [\partial w F_{v_1}] \cdot [F_{v_2}]. \tag{81}$$

By construction of $[\partial w F_v]$:

$$\begin{aligned}[\partial w F_v] &= \sum_{t\in\mathcal{B}_m(F_v)} [\partial t F_v] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] \\ &= \sum_{t\in\mathcal{B}_m(F_v)} [\partial t(F_{v_1\cdot v_2})] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}].\end{aligned} \tag{82}$$

Similar to the previous case, for any $t \in \mathcal{B}_m(F_v)$ we have

$$d^+_{\text{ub}}(v) - d^+_{\text{ub}}(t) < 2^i \quad \text{and} \quad 2d^+_{\text{ub}}(t) > d^+_{\text{ub}}(v).$$

If $v \in \mathcal{B}_m(F_v)$ then $\mathcal{B}_m(F_v) = \{v\}$ and so (82) is simply $[\partial v F_v] \cdot [\partial w F_{v_1}] \cdot [F_{v_2}] = [\partial w F_{v_1}] \cdot [F_{v_2}]$ as required. Otherwise, assume that $v \notin \mathcal{B}_m(F_v)$. Then Ψ_i contains the following equation, for any $t \in \mathcal{B}_m(F_v)$:

$$[\partial t(F_{v_1\cdot v_2})] = [\partial t F_{v_1}] \cdot [F_{v_2}].$$

Using premises from Ψ_i, we can then prove that (82) equals:

$$\sum_{t\in\mathcal{B}_m(F_v)} ([\partial t F_{v_1}] \cdot [F_{v_2}]) \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] = \left(\sum_{t\in\mathcal{B}_m(F_v)} [\partial t F_{v_1}] \cdot [\partial w F_{t_1}] \cdot [F_{t_2}] \right) \cdot [F_{v_2}]. \tag{83}$$

As in (50), we have $\sum_{t\in\mathcal{B}_m(F_v)}[\partial t F_{v_1}]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}] = \sum_{t\in\mathcal{B}_m(F_{v_1})}[\partial t F_{v_1}]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}]$. Also, since $v_1\cdot v_2 = v \notin \mathcal{B}_m(F_v)$, we have $d^+_{\rm ub}(v_1) > m = 2^i + d^+_{\rm ub}(w)$, and so

$$[\partial w F_{v_1}] = \sum_{t\in\mathcal{B}_m(F_{v_1})} [\partial t F_{v_1}]\cdot[\partial w F_{t_1}]\cdot[F_{t_2}]\,. \tag{84}$$

Hence by (84), (83) equals $[\partial_w F_{v_1}]\cdot[F_{v_2}]$.

The above proof of (81) from Ψ_i has size $O(s^2\cdot m(s,d))$ and depth $O(r(s,d))$.

We now append Ψ'_i from Part (a) (which also contains Ψ_i) with all proof-sequences of $[\partial w F_v] = [\partial w F_{v_1}] + [\partial w F_{v_2}]$ in Case 1 and all proof sequences $[\partial w F_v] = [\partial w F_{v_1}]\cdot[F_{v_2}]$ in Case 2, above. We obtain the proof-sequence Ψ_{i+1} of size

$$\lambda(s,i+1) \le O(s^4\cdot m(s,d)) + \lambda(s,i),$$

and depth $O(r(s,d))$, as required. □

References

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998. 2.1, 2, 10.2, 13, C

[All18] Eric Allender. Personal communication, 2018. 2.1, 2, 10.2

[BBP95] Maria Luisa Bonet, Samuel R. Buss, and Toniann Pitassi. Are there hard examples for Frege systems? In *Feasible mathematics, II (Ithaca, NY, 1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 30–56. Birkhäuser Boston, Boston, MA, 1995. 1

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18:147–150, 1984. 13

[BP98] Paul Beame and Toniann Pitassi. Propositional proof complexity: past, present, and future. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (65):66–89, 1998. 1

[Bus86] Samuel R. Buss. *Bounded Arithmetic*, volume 3 of *Studies in Proof Theory*. Bibliopolis, 1986. 3

[CF12] Stephen A Cook and Lila A Fontes. Formal Theories for Linear Algebra. *Logical Methods in Computer Science*, Volume 8, Issue 1, March 2012. 1, 2, 2.2, 10, 10.1, 10.3, 12, 11, 13

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. ASL Perspectives in Logic. Cambridge University Press, 2010. 1, 1, 2, 2, 3, 3.2, 3.2, 3.3, 3.3, 3.5, 5.1.3, 10.3, 11.2, 11.2, 11.3, 12, 13, A, A.1, A.5, A.1, A.1, A.2, A.2

[Coo85] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–21, 1985. 1

[HP93] P. Hájek and P. Pudlák. *Metamathematics of First-order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1993. 3

[HT09] Pavel Hrubeš and Iddo Tzameret. The proof complexity of polynomial identities. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 41–51, 2009. 1, 3, 3.5, 3.7

[HT15] Pavel Hrubeš and Iddo Tzameret. Short proofs for the determinant identities. *SIAM J. Comput.*, 44(2):340–383, 2015. (A preliminary version appeared in Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC'12)). (document), 1, 1, 2, 2, 2, 2.1, 3, 3.5, 3.7, 3.6, 4, 4, 5.1.2, 5.1.2, 5.2.1, 6.2, 6.2, 6.2, 8.1, 8.1, 8.1, 10, 10, 10.2, 10.3.1, 10.3.1, 10.3.1, 11.3, 11.3, 12, 12, 13, 13, D

[Jeř05] Emil Jeřábek. *Weak pigeonhole principle, and randomized computation*. PhD thesis, PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2005. 1

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1995. 3

[MRK88] Gary L. Miller, Vijaya Ramachandran, and Erich Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM J. Comput.*, 17(4):687–695, 1988. 2.1, 4, 10.2, C

[MT14] Sebastian Müller and Iddo Tzameret. Short propositional refutations for dense random 3CNF formulas. *Annals of Pure and Applied Logic*, 165:1864–1918, 2014. Extended abstract in Proceedings of the 27th Annual ACM-IEEE Symposium on Logic In Computer Science (LICS), 2012. A.2

[PT16] Tonnian Pitassi and Iddo Tzameret. Algebraic proof complexity: Progress, frontiers and challenges. *ACM SIGLOG News*, 3(3), 2016. 3

[RY08] Ran Raz and Amir Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008. 10, 10, 10.2

[SC04] Michael Soltys and Stephen Cook. The proof complexity of linear algebra. *Ann. Pure Appl. Logic*, 130(1-3):277–323, 2004. 1, 2, 12, 12, 11

[Sol01] Michael Soltys. *The complexity of derivations of matrix identities*. PhD thesis, University of Toronto, Toronto, Canada, 2001. 1

[Str73] Volker Strassen. Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973. (in German). 1, 2.1, 4, 6.1, 9

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. 3.4

[Vin91] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proc. 6th IEEE Structure in Complexity Theory Conference*, pages 270–284, 1991. 2.1, 11.1

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983. 2.1, 4, 10, 10, 10.2, 11.1, C

— Page left blank for ECCC stamp —

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il