Uniform, Integral and Feasible Proofs for the Determinant Identities*

Iddo Tzameret† Stephen A. Cook‡

November 13, 2020

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 propositional proofs operating with $\mathbf{NC^2}$-circuits of the basic determinant identities over the integers (previous propositional proofs hold only over the two element field).

*A version of this paper is to appear in the *Journal of the ACM* (JACM).

†Department of Computer Science, Royal Holloway, University of London. Iddo.Tzameret@gmail.com

‡Department of Computer Science, University of Toronto. sacook@cs.toronto.edu

Contents

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 class of all languages that can be decided by uniform families of $O(\log^2 n)$-depth and polynomial-size circuits. This 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 class 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).

In this work we are interested not in the complexity of computing the determinant per se, but in the complexity of the concepts we need to use in order to *prove* the basic properties of the determinant, and more generally to prove and develop basic linear algebra.

The field that studies the computational complexity of concepts needed to prove different statements is called *bounded arithmetic*, and constitutes the proof-theoretic approach to computational complexity. Bounded arithmetic is in fact a general name for a family of weak formal theories of arithmetic (that is, natural numbers). These theories are characterized by their axioms, usually starting from a basic set of axioms providing the most basic properties of numbers each bounded arithmetic theory possesses different additional axioms postulating the existence of different sets of numbers, or different kinds of induction principles. Based on its specific axioms each theory of bounded arithmetic proves the totality of functions from different complexity classes (e.g., polynomial-time functions, $\mathbf{NC^2}$ functions, etc.). We can typically consider such theories as working over a logical language that contains the function symbols of that prescribed complexity class. In this sense proofs in the theory use concepts from a specific complexity class, and we can say that the theory captures 'reasoning in this class' (e.g., 'polynomial-time reasoning').

While the first theory for polynomial-time reasoning was the equational theory $\mathbf{PV}$ considered by Cook [Coo75], bounded arithmetic goes back to the work of Parikh [Par71] and Paris-Wilkie [PW85]. In a seminal work Buss [Bus86] introduced other theories of bounded arithmetic and laid much of the foundation for future work in the field.

Theories of bounded arithmetic correspond not only to complexity classes but also to propositional proofs by way of propositional translations (going back to [Coo75], and the later independent work of [PW85]): if a statement of a given form is provable in a given bounded arithmetic theory then the same statement is suitably translated to a family of propositional formulas with short (polynomial-size) proofs in a corresponding propositional proof system (cf. [CN10, Chapter VII] for a systematic treatment of this).

One goal of bounded arithmetic is to serve as a framework in which the 'bounded reverse mathematics' program is developed (in an analogy to Friedman and Simpson reverse mathematics program [Sim99]). In this program one seeks to find the weakest theory capable of proving a given theorem. Special theorems of interest are those of computer science and computational complexity theory. The motivation here is to shed light on the role of complexity classes in proofs, in the hope to delineate for example those concepts that are needed to progress on major problems in computational complexity from those that are not. For instance, it has been identified that apparently most results in contemporary computational complexity can be proved using polynomial-time concepts (e.g., in $\mathbf{PV}$) (cf. [Pic15]), and it is important to understand whether stronger theories and concepts are needed to prove certain results.

Some examples of results in bounded reverse mathematics and meta-mathematics of complexity are Jordan Curve theorem in $\mathbf{AC}^0$-reasoning [NC07] (where $\mathbf{AC}^0$ is the class of languages computable with families of constant depth Boolean circuits), Barrington's Theorem in $\mathbf{NC}^1$-reasoning [Ngu08] (cf. [CN10, Sec. IX.5.5]), prime factorization theorem in polynomial-time reasoning [CN10, Exercise VI.4.4], sorting network in an extension of $\mathbf{NC}^1$-reasoning [Jeř11], expander graph construction in $\mathbf{NC}^1$-reasoning [BKKK20], Toda's Theorem in bounded arithmetic [BKZ15] and many parts of complexity theory (including the PCP theorem) in polynomial-time reasoning [Pic15].

Due to its basic nature, linear algebra, which naturally underlies many theorems in computer science and computational complexity, has been identified by many works as important in bounded arithmetic and proof complexity. In particular, it has been conjectured that since the integer determinant is computable in $\mathbf{NC}^2$ the multiplicativity of the determinant function $\mathrm{DET}(A) \cdot \mathrm{DET}(B) = \mathrm{DET}(AB)$, for two matrices A, B, or the related Cayley-Hamilton theorem can be proved in $\mathbf{NC}^2$-reasoning. Cook and Nguyen presented this specific conjecture in their monograph [CN10, Table 2, page 8, and Open Problems section IX 7.1]. This conjecture was first considered essentially in [SC04] (see also [CF12, Sol01]), and before that in the propositional setting by Cook and Rackoff and specifically Bonet *et al.* [BBP95] (see also [BP98]). That the determinant properties can be proved within a theory that captures $\mathbf{NC}^2$-reasoning is aligned with the intuition that basic properties of many constructions and functions of a given complexity class are provable without the need to use concepts beyond that class.

The weakest theory known to date to prove the multiplicativity of the determinant is the theory $\mathbf{PV}$ for polynomial-time reasoning; this was shown essentially by Soltys and Cook [SC04] (cf. [CF12, Jeř05]). Their work introduced three formal theories of increasing strength for reasoning about linear algebra. The weaker of them is the quantifier free theory LA that allows the basic ring properties of matrices to be formulated and proved. The intermediate theory LAP adds the matrix powering operator to LA. Berkowitz's algorithm [Ber84] reduces the determinant function

to matrix powering (over any field), and the determinant function in LAP is thus defined using this algorithm. Accordingly, LAP can be considered as a formal theory for reasoning with concepts in the complexity class DET for which the determinant is complete [Coo85]. LAP can prove that the co-factor expansion of the determinant, the multiplicativity of the determinant and the Cayley-Hamilton theorem are all provable from each other. However, it cannot apparently prove any of these statements by themselves. For this purpose [SC04] extended the theory LAP to ∀LAP which includes induction over formulas with bounded universal matrix quantifiers. ∀LAP can be considered a feasible theory tailored for reasoning about matrices that incorporate polynomial-time computable concepts (close to Buss's S^1_2 [Bus86] or $\mathbf{PV}$): [SC04] showed an interpretation of ∀LAP (when the underlying field is finite or the rationals) into Buss's S^1_2 theory. In this relatively strong theory they were able to prove the multiplicativity of the determinant and hence also the Cayley-Hamilton theorem and the co-factor expansion of the determinant. Subsequent work of Thapen and Soltys [TS05] showed that some linear algebra, namely Gaussian elimination, can be developed in a weaker theory than ∀LAP: using the theory LA plus existential induction on matrices, denoted ∃LA, they were able to prove the commutativity of matrix inverse in the theory. However, ∃LA is also quite strong, as it was shown [TS05] to interpret the second-order version of polynomial-time reasoning $\mathbf{V}^1$.

The induction used in ∀LAP to prove the determinant identities is beyond the strength of $\mathbf{NC^2}$-reasoning and it remained open until now whether the determinant identities, and the basic statements of linear algebra like Cayley-Hamilton theorem and the co-factor expansion of the determinant can be proved using concepts not going beyond the computational complexity class of linear algebra itself, namely $\mathbf{NC^2}$-reasoning.

The main goal of this work is to provide a positive answer to this question, over the integers, namely providing proofs of the multiplicativity of the determinant, the Cayley-Hamilton theorem and the co-factor expansion of the determinant using $\mathbf{NC^2}$-reasoning. Our proof in the theory for $\mathbf{NC^2}$-reasoning is completely different from [SC04] and depends instead on the work of Hrubeš and Tzameret [HT15] who constructed $\mathbf{NC^2}$-Frege *propositional* proofs of the determinant identities over the two-element field.

We define the determinant in the theory based on an evaluation of an algebraic circuit with division simulating in essence Gaussian elimination (using Schur complement). We then build a witness for the determinant identities in the form of a line-by-line algebraic equational proof of the identities as constructed in [HT15]. This algebraic equational proof was originally constructed in [HT15] by induction on the dimension n of the matrices, and used concepts computable by polynomial-size algebraic circuits with division. We show that this equational proof can be constructed already in $\mathbf{NC^2}$-reasoning. Though the witness is constructible in the theory, before it can be used it needs to be converted *in the theory* into something that $\mathbf{NC^2}$-reasoning can prove its correctness, namely (a sequence of) $O(\log^2 n)$ depth Boolean circuits, the evaluation of which is a complete problem for $\mathbf{NC^2}$ and constitutes the basis of the theory for $\mathbf{NC^2}$-reasoning denoted $\mathbf{VNC}^2$ in [CN10]. These constructions and conversions are highly non-trivial and depend on many results in structural algebraic circuit complexity.

Note that although [HT15] showed that in the *propositional* case the multiplicativity of the determinant over $GF(2)$ can be proved with polynomial-size propositional proofs operating with $\mathbf{NC^2}$ Boolean circuits, 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-$\mathbf{NC^2}$. For instance, [HT15] crucially used elimination of division gates from algebraic circuits which we do not know how to do using uniform weak computational models like uniform-$\mathbf{NC^2}$ (for division elimination one needs to use a non-constructive existential statement by Strassen [Str73] about field assignments that do not nullify a given polynomial; this statement is based on the Schwartz-Zippel lemma [Sch80, Zip79]). Further a priori existential constructions in [HT15] are the Valiant *el al.* [VSBR83] balancing of algebraic circuits, homogenization and division normalization of algebraic circuits [Str73]. We explain our construction in the theory and some of the differences between our construction to [HT15] in the overview section below.

1.1 Overview

Our goal is to prove the multiplicativity of the determinant over the integers with $\mathbf{NC^2}$-reasoning. For $\mathbf{NC^2}$-reasoning we take the theory $\mathbf{VNC}^2$ defined in [CN10] as a two-sorted formal theory of natural numbers, in which the first sort is for natural numbers and the second sort is for finite sets of natural numbers intended to encode bit-strings (see Section 2). It is usual to consider the first sort of natural numbers as "first-order objects" and the second sort of sets of natural numbers as "second-order objects". Integers in the theory are represented as binary strings (hence, second-order objects), and matrices are encoded as a two-dimensional array of integers (cf. [CF12]). We will use the term *numbers* in the theory to refer to the first-sort of natural numbers and *integers* to refer to their bit-strings representation as a second-sort.

We wish to define the determinant function $\mathrm{DET}(\cdot)$ over the integers such that $\mathbf{VNC}^2$ proves that for every n and m and a pair of $n \times n$ matrices A, B with integer entries of bit-length m

$$\mathrm{DET}(A) \cdot \mathrm{DET}(B) = \mathrm{DET}(AB)\,, \tag{1}$$

and for every (lower or upper) triangular $n \times n$ matrix with integer entries of bit-length m with $c_{11}, \ldots, c_{nn}$ on the diagonal

$$\mathrm{DET}(C) = c_{11} \cdots c_{nn}. \tag{2}$$

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 triangular matrices (this in turn follows from the fact that every square matrix is equal to the product PLU with P a permutation matrix and L, U lower and upper triangular matrices, respectively; as well as the fact that every permutation matrix can be shown to be a product of triangular matrices corresponding to elementary matrix transformations).

It is known that we can prove elementary facts about matrices, such as the definability of matrix products AB, the statement expressing associativity and commutativity of matrix products $A(BC) = (AB)C$ and $A + B = B + A$, resp., and so forth, in the theory $\mathbf{VNC}^1$, the theory for $\mathbf{NC^1}$-reasoning [Sol01, SC04, CF12]. However, these identities are computationally seemingly simpler than the multiplicativity of the determinant for example, since additions and multiplication of matrices have lower known computational complexity (polynomial-size threshold circuits of constant depth would suffice for these operations over the integers), while the determinant is only known to be computable in $\mathbf{NC^2}$.

We note that in this work the circuit class $\mathbf{NC^2}$ is assumed to be a uniform circuit class. 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 in the theory.

Defining the determinant function in the theory. Given an $n \times n$ integer matrix, the provably total (formally in our case $\mathbf{\Sigma}^B_1$-definable; see Section 2.2) string function (recall that we encode integers as strings) in $\mathbf{VNC}^2$ for the determinant is defined by evaluating an algebraic circuit simulating Gaussian elimination.

In particular, the determinant function first constructs an algebraic circuit (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 a formal polynomial over n^2 distinct variables). This algebraic circuit captures the recursive formula as in Schur complement. Then, *eliminate the division gates* in the algebraic circuit using, among other conversions, substitutions of power series in the circuit (cf. Strassen [Str73]). Then, *homogenize* the circuit getting rid of high degrees [Str73], *balance the circuit* to achieve the squared logarithmic depth $O(\log^2 n)$ (following [VSBR83]), and more precisely $O(\log n)$-depth circuit with unbounded fan-in plus gates and fan-in two product gates (following [Vin91]). We now evaluate the algebraic circuit over the input integer matrix. This consists of several steps: convert the algebraic circuit into an $O(\log^2 n)$-depth Boolean circuit computing the same polynomial over the integers (coded as bit-strings). This is done by simulating additions and products by carry-save adders and binary integer products. Then layer the Boolean circuit so that each node connects only to the subsequent layer. And finally evaluate the Boolean circuit using the fact that the $\mathbf{NC^2}$ circuit evaluation problem is in $\mathbf{NC^2}$.

Note that since we show that the determinant function as defined above is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$, by [CN10] it means that this function, including all the constructions in it, are 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 line-by-line algebraic proof over the integers (as in [HT15]; see Section 2.8) of these identities. Thus, by soundness of these proofs which we show is provable in $\mathbf{VNC}^2$, these identities must be true.

More precisely, an algebraic proof of a polynomial identity, in symbols a $\mathbb{P}_c(\mathbb{Z})$-proof (introduced in [HT09]), is a sequence of equations between algebraic circuits over $\mathbb{Z}$, each of which is either an instance of the polynomial-ring axioms (such as commutativity, distributivity, etc.) or was derived by addition or multiplication of previous equations.

We demonstrate a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{VNC}^2$ that given an input n in unary, outputs $\mathbb{P}_c(\mathbb{Z})$-proofs of the determinant identities as in (1) and (2) (for $n \times n$ matrices). 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 finish the argument, we transform this algebraic proof into a corresponding line-by-line Boolean proof and then use evaluation of $O(\log^2 n)$-depth Boolean circuits to conclude the soundness of the proofs: using (number) induction on proof-length we argue that for every assignment of integers, the determinant identities must hold. Since the determinant function in the theory is defined by itself as the evaluation of the circuit computing the determinant, the argument is concluded.

Overall, in our argument the main "non-syntactic" property we need the theory to express is the evaluation of $O(\log^2 n)$-depth Boolean circuits. The axioms of $\mathbf{VNC}^2$ are tailored for this purpose, as they include the existence of a string evaluating any given monotone Boolean circuit of $O(\log^2 n)$-depth and polynomial-size (see Definition 2.6), which we show is sufficient to evaluate general polynomial-size algebraic circuits of polynomial degree over the integers (see definition of Eval_{alg} in Sec. 11.1 Step (iii)). An additional ability of $\mathbf{VNC}^2$ is to express and reason about matrix powering and we use this when balancing circuits.

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 new technical obstacles that we face are *uniformity* and *parallelism* as we explain in what follows.

Uniformity here means that we need the whole proof to be constructible in *uniform*-$\mathbf{NC^2}$.

In order to simulate Gaussian elimination we start with algebraic proofs and circuits with division gates, denoted $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs. To get the division-free $\mathbb{P}_c(\mathbb{Z})$-proofs which then can turn into Boolean proofs, as mentioned above, we need to eliminate division gates from certain algebraic circuits and proofs. To eliminate division gates like u/v (for two circuits u, v), one needs to find an assignment to the variables in which the polynomial computed by v is nonzero. In general we do *not* know how to do this in the theory, since this requires an existential statement based on Schwartz-Zippel lemma [Zip79, Sch80]. We solve this problem by showing that if we assign the variables for matrix entries with 0-1 values based on the identity matrix we do not nullify the division gates in our circuit and proofs.

Moreover, the initial $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are of high syntactic-degree and unbounded depth. But in order to show that the resulting $\mathbb{P}_c(\mathbb{Z})$-proofs are correct (sound) we need to start from correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs. For $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs however the correctness is *semantic*: the division rule is a semantic rule since it requires that the circuit in division does not compute the zero polynomial, a property which we do not know how to compute in $\mathbf{NC^2}$.

To solve this problem we introduce the technical concept of *a provably good division gate*: a division gate u^{-1} is provably good whenever there exists a witness that u is nonzero. For us, this witness will be a certain kind of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$, where ρ is an assignment to the variables of matrix entries in u that corresponds to the identity matrix.

Accordingly, to express that a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is correct we express its syntactic-correctness together with explicit witnesses demonstrating that each division gate u^{-1} is provably good. (In particular, we strengthen the results of [HT15] that demonstrated the cases in which the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of the determinant identities are definable, into provable definability: we show that not only the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs do not contain zero division, but that the theory can prove the existence of witnesses for this.)

Parallelism here means that the construction of the algebraic $\mathbb{P}_c(\mathbb{Z})$-proofs from [HT15] *must be done by itself in* $\mathbf{NC^2}$. To make the construction parallel we need to devise several $\mathbf{AC}^0$- and $\mathbf{NC^2}$-algorithms. 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 a theory for $\mathbf{AC}^0$-reasoning denoted $\mathbf{V}^0$ [CN10]. Among the algorithms we devise are the following:

(**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 follows Strassen's algorithm [Str73]. (**ii**) Converting algebraic circuits C into a 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 *et 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 $\mathbf{FNC^2}$ *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]. This algorithm 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 not enough. We further need to show that the correctness of these algorithms can be formalized efficiently with $\mathbb{P}_c(\mathbb{Z})$- and $\mathbb{P}_c^{-1}(\mathbb{Z})$-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 $\mathbb{P}_c(\mathbb{Z})$-proofs of the determinant identities.

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 $GF(2)$. Note also that unlike [HT15] we do not need to simulate small fields in big ones since we work over $\mathbb{Z}$.

1.1.1 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

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

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 $\mathbf{NC^2}$ (namely, $\mathsf{DET} \subseteq \#\mathbf{SAC}^1 \subseteq \mathbf{TC}^1 \subseteq \mathbf{NC^2}$), 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 $\mathbf{VNC}^2$.

1.1.2 Organization

The preliminaries for this work are somewhat long and are given in Section 2. They consist 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 equational proof systems operating with algebraic circuits [HT09, HT15]. Section 3 explains in some detail the encoding scheme for algebraic circuits and proofs in the theory. Sections 4 to 10 are dedicated to the construction of the $\mathbb{P}_c(\mathbb{Z})$- and $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in the theory. Section 11 establishes the reflection principle for $\mathbb{P}_c(\mathbb{Z})$-proofs, and Section 12 demonstrates $\mathbf{VNC}^2$ proofs of further basic statement in linear algebra. We finish with conclusions and open problems in Section 13.

2 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 (though formally the language of bounded arithmetic theories is sometimes different from that of Peano Arithmetic, the theories can be interpreted in Peano Arithmetic nevertheless). 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 2.3 below for the definitions of $\mathbf{NC^2}$ and $\mathbf{FNC}^2$, and Definition 2.11 for $\mathbf{AC}^0$ and $\mathbf{FAC}^0$.

Definition 2.1 (Language of two-sorted arithmetic $\mathcal{L}_A^2$)**.** *The language of two-sorted arithmetic, denoted $\mathcal{L}_A^2$, consists of the following relation, function and constant symbols:*

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

[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].

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 (and this is abbreviated as $N(n)$).

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 terms and formulas in the usual way. For formulas, we use 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 $\forall x(x \leq t \rightarrow \ldots)$ 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 Π_0^B if it uses *no string quantifiers* and all number quantifiers are bounded. A formula is in Σ_{i+1}^B or Π_{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 $\psi \in \Pi_i^B$ and $\psi \in \Sigma_i^B$, respectively, and t_i does not contain X_i, for all $i = 1, \ldots, m$. We write $T(t)$ to abbreviate $t \in T$, for a number term t and a string term T. For a formula ψ we write $\psi(a/x)$ to denote the *substitution instance* of ψ in which every free occurrence of the variable x is replaces by the term a.

As mentioned above, 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 with S_N.

2.1 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**. $y + 1 = |X| \rightarrow X(y)$

$$\textbf{SE. } (|X| = |Y| \wedge \forall i \leq |X|\,(X(i) \leftrightarrow Y(i))) \rightarrow X = Y$$
$$\boldsymbol{\Sigma}^B_0\textbf{-COMP. } \exists X \leq y \forall z < y\,(z \in X \leftrightarrow \varphi(z))\,, \text{ for all } \varphi \in \boldsymbol{\Sigma}^B_0$$
$$\text{where } X \text{ does not occur free in } \varphi\,.$$

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, $\boldsymbol{\Sigma}^B_0$**-COMP** is the comprehension axiom *scheme* for $\boldsymbol{\Sigma}^B_0$-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 $\boldsymbol{\Sigma}^B_0$ property.

Proposition 2.2 (Corollary V.1.8. [CN10])**.** *The theory* $\mathbf{V}^0$ *proves the* number *induction axiom scheme for* $\boldsymbol{\Sigma}^B_0$*-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.

2.2 Definability in Bounded Arithmetic

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

Definition 2.3 (Two-sorted definability)**.** *Let* $\mathcal{T}$ *be a theory over the language* $\mathcal{L} \supseteq \mathcal{L}^2_A$ *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[3]

$$y = f(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, y, \vec{X}). \tag{3}$$

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

$$Y = F(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}, Y). \tag{4}$$

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*

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{5}$$

The formulas (3)*,* (4)*, and* (5) *are the* defining axioms *for* f*,* F*, and* R*, respectively.*

Definition 2.4 (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* $\mathcal{T}$*.*

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

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 2.3) 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])).

We shall seek to define the determinant function in a theory via a Σ_1^B-formula. Following Theorem 2.7 below the Σ_1^B-definable functions of $\mathbf{V}^0$ (equivalently, the Σ_0^B-definable functions of $\mathbf{V}^0$) are precisely the $\mathbf{FAC}^0$ functions, and that the Σ_1^B-definable functions of $\mathbf{VNC}^2$ are precisely the $\mathbf{FNC}^2$ functions.

2.3 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 $g(n)$. A computation tree is said to have *time* t if every path from the root to a leaf in the tree is at most t. An alternating Turing machine is said to *work in time* $g(n)$ if for every input x of length n the time of every computation tree of x is at most $g(n)$.

Definition 2.5 (Uniform $\mathbf{NC^2}$)**.** *The uniform complexity class* $\mathbf{NC^2}$ *is 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 (unary) number and string variables, respectively, such that the relation of the function is defined (resp. bit-defined; see Definition 2.10) in $\mathbf{NC^2}$ (a relation R is *defined in* $\mathbf{NC^2}$ if the language containing the set of tuples 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 two $\vee, \wedge$ gates and fan-in one $\neg$ 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 2.5). 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 2.5.

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

2.4 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 $0, \ldots, d$, 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]).

For a natural number d we write as usual $[d]$ to denote $\{1, \ldots, d\}$. The layered monotone Boolean circuit C with $d+1$ layers is encoded as follows: I (for "inputs") is a string variable, where $|I| \leq n$, that defines the (Boolean) input gates to the circuit. Further, G (for "gates") is a string variable such that $G(x, y)$ holds for $x \in [d]$ 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 (for "edges") 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 $0 \leq z \leq d$, row $Y^{[z]}$ contains the gates on layer z that output 1.

$$\delta_{LMCV}(n,d,E,G,I,Y) \equiv \forall x < n \forall z < d \Big((Y(0,x) \leftrightarrow I(x)) \wedge \big(Y(z+1,x) \leftrightarrow \big(\big(G(z+1,x) \wedge \forall u < n, E(z,u,x) \rightarrow Y(z,u)\big) \vee (\neg G(z+1,x) \wedge \exists u < n, E(z,u,x) \wedge Y(z,u))\big)\big)\Big). \tag{6}$$

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

$$Fanin2(n,d,E) \equiv \forall z < d \forall x < n \exists u_1 < n \exists u_2 < n \forall v < n (E(z,v,x) \rightarrow (v = u_1 \vee v = u_2)\big). \tag{7}$$

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

Definition 2.6 ($\mathbf{VNC}^2$)**.** *The theory* $\mathbf{VNC}^2$ *has vocabulary* $\mathcal{L}_A^2$ *and is axiomatized by the axioms of* $\mathbf{V}^0$ *and the axiom:*

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

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]).

The following is the main theorem for $\mathbf{V}^0$ and $\mathbf{VNC}^2$:

Theorem 2.7. ([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.

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

Here we give more details on the theories $\mathbf{V}^0$ and $\mathbf{VNC}^2$. Specifically, we explain how to conclude that a function is definable in the theory, and how to extend the language $\mathbf{V}^0$ and $\mathbf{VNC}^2$ with new function symbols (in a conservative way; see below). We will describe a process (see [CN10, Section V.4]) 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 2.8 (Relation representable in a language)**.** *Let Φ be a set of formulas in a language $\mathcal{L}$ that extends $\mathcal{L}^2_A$. 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:*

$$R(\vec{x},\vec{X}) \leftrightarrow \varphi(\vec{x},\vec{X}). \tag{8}$$

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

Definition 2.9 (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*

$$F(\vec{x},\vec{X})(i) \leftrightarrow \forall i R(i,\vec{x},\vec{X}) \tag{9}$$

holds in the standard two-sorted model.

Definition 2.10 (Σ^B_0-definability from a language; Definition V.4.12. in [CN10])**.** *We say that a* number function $f(\vec{x},\vec{X})$ is Σ^B_0-definable from a language $\mathcal{L} \supseteq \mathcal{L}^2_A$*, if f is polynomially-bounded and its graph*[4] *is represented by a $\Sigma^B_0(\mathcal{L})$-formula φ (where $\Sigma^B_0(\mathcal{L})$ is the class of Σ^B_0-formulas in the language $\mathcal{L}$). We call the formula φ the* defining axiom of f*. We say that a* string function F is Σ^B_0-definable from a language $\mathcal{L} \supseteq \mathcal{L}^2_A$*, if F is polynomially-bounded and its* bit-graph *(that is, the relation $R(i,\vec{x},\vec{X})$ as in (9)) is representable by a $\Sigma^B_0(\mathcal{L})$-formula φ, in which case we say that φ is the* bit-defining formula of F*. 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 2.10) and in case f is definable *in the theory* (Definition 2.3). 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].

Definition 2.11 ($\mathbf{FAC}^0$)**.** *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^B_0-formula in the language $\mathcal{L}^2_A$.*

Definition 2.12 ($\mathbf{AC}^0$-reduction)**.** *A string function F (resp. a number function f) is* $\mathbf{AC}^0$-reducible to $\mathcal{L} \supseteq \mathcal{L}^2_A$ *iff there is a possibly empty sequence of (either string or number) functions $F_1,\ldots,F_k$ such that F_i is Σ^B_0-definable from $\mathcal{L} \cup \{F_1,\ldots,F_{i-1}\}$, for any $i = 1,\ldots,k$, and F (resp. f) is Σ^B_0-definable from $\mathcal{L} \cup \{F_1,\ldots,F_k\}$.*

[4] 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.

We are now 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 (see Section V.4. in [CN10] for the proofs). Recall once more that this will enable us to go from a semantic definition in a language to the proof-theoretic notion of *definability in the theory*:

Fact 2.13.

(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 $f_2, f_3, \ldots$ to the language, which can also be used in comprehension and induction axioms.

By the aforementioned and by Definition 2.12, 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.

2.6 Some Basic Formalizations in $\mathbf{V}^0$

In this section we fix some basic notation for primitive objects coded in $\mathbf{V}^0$. Most formalizations here are routine and can be found in [CN10] (cf. [MT14]). Recall that when speaking about *numbers* in the theory we mean natural numbers (that is, first-sort objects), and speaking about *integers* or *strings* we refer to the second-sort objects.

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*. We also Σ_0^B-define inductively $\langle v_1, \ldots, v_k\rangle := \langle\langle v_1, \ldots, v_{k-1}\rangle, v_k\rangle$, for any constant $k > 2$. Then $\mathbf{V}^0$ proves the injectivity of the pairing function and enables us to handle such pairs in a standard way.

Sequences of numbers of variable length. If we wish to talk about sequences of natural numbers 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: 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 Σ_0^B-defining formula for the number function $\mathrm{seq}(x, Z)$ returning the xth element in the sequence Z:

$$\begin{aligned} y = \mathrm{seq}(x, Z) \leftrightarrow (y < |Z| \wedge Z(\langle x, y\rangle) \wedge \forall z < y \neg Z(\langle x, z\rangle)) \\ \vee (\forall z < |Z| \neg Z(\langle x, z\rangle) \wedge y = |Z|). \end{aligned} \tag{10}$$

Formula (10) 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.

Array of strings. We wish to encode a *sequence* of strings as an array, namely, a two-dimensional array. We use the function $\mathrm{Row}(x, Z)$ to denote the xth string in Z as follows (we follow the treatment in [CN10, Definition V.4.26, page 114]). The string function $\mathrm{Row}(x, Z)$, abbreviated $Z^{[x]}$, is Σ_1^B-definable in $\mathbf{V}^0$ (and hence can be used in induction axiom) using the following bit-definition:

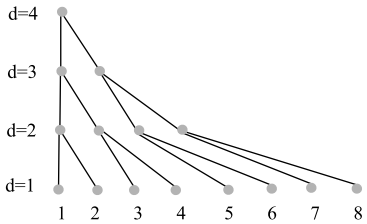
$$\mathrm{Row}(x, Z)(i) \leftrightarrow (i < |Z| \wedge Z(\langle x, i\rangle)).$$

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, where each integer is coded with m bit-vectors, for some chosen natural number m.

2.6.1 Example: Binary Tree Construction in $\mathbf{V}^0$

Here we provide a simple example of a Σ_1^B-definable string function in $\mathbf{V}^0$ that encodes a simple syntactic object. 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 Σ_1^B-definable in $\mathbf{V}^0$. This can be used to construct a formula that computes for example the inner product of two vectors as in Section 3.1.3.

The tree is encoded in the scheme described in Section 3.1.1, with V as a string (finite set of numbers) consisting of the nodes, with each node $u \in V$ is a number and a string E encoding edges, with each number in E interpreted as a pair of numbers (u, v) such that $u, v \in V$. Every node $u \in V$ is a pair $u = (d, w)$ with $d \in \{1, \ldots, \log(n) + 1\}$ being the layer of the tree and w is the index of the node in the layer d that runs from 1 to n/d. For two nodes $u, v \in E$ with $u = (d_1, w_1), v = (d_2, w_2)$, u has a directed edge to v iff $d_2 = d_1 + 1$ and if w_1 is even then $w_2 = w_1/2$ and otherwise $w_2 = \frac{w_1+1}{2}$. This is shown in the figure below.



Formally, to show that the string function $F(n)$ is Σ_1^B-definable in $\mathbf{V}^0$ (equivalently, Σ_0^B-definable in $\mathbf{V}^0$), according to Section 2.5 we need to demonstrate a Σ_0^B-formula that *bit-defines* the tree encoding, as follows. Let

$$\begin{aligned} \Phi(d, w_1, w_2) := d \cdot w_1 \leq n \ \wedge \ \exists x \leq n(w_1 = 2x \rightarrow 2w_2 = w_1) \wedge \\ \exists x \leq n(w_1 = 2x + 1 \rightarrow 2w_2 = w_1 + 1). \end{aligned} \tag{11}$$

The following is the $\mathbf{\Sigma}_0^B$-formula that bit-defines the string function that returns the string E encoding the edges of the tree, given the number of leaves n:

$$\varphi(n,i) := \exists d \leq n \exists w_1 \leq n \exists w_2 \leq n \left(i = ((d,w_1),(d+1,w_2)) \wedge \Phi(d,w_1,w_2)\right).$$

Bit-defining the nodes V function is similar.

2.7 Polynomials and Algebraic Circuits

For a very good treatise 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 $X := \{x_1, x_2, \dots\}$. 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 ring. 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 ring element in $\mathbb{G}$. All the other nodes have *fan-in* (that is, in-degree) *two* and are labeled by either an addition gate $+$ or a product gate $\times$. Every node in an algebraic circuit C *computes* a polynomial as follows: an input node computes the variable or scalar that labels it. A $+$ (or $\times$) gate is said to compute the addition (product, resp.) of the (commutative) polynomials computed by its incoming nodes. The polynomial computed by a node u in an algebraic circuit C is denoted $\widehat{u}$. Given a circuit C, we denote by $\widehat{C}$ the polynomial computed by C, that is, the polynomial computed by the output node of C. An algebraic circuit is called a *formula*, if its 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. For an algebraic circuit C we write $C(a/x)$ to denote the *substitution instance* of C in which every occurrence of the node x is replaced by the sub-circuit a; in case $C(x)$ is written with its displayed variable(s) x we can write $C(x)(a/x)$ for this substitution instance. We say that a polynomial is *homogeneous* whenever every monomial in it has the same (total) degree.

Definition 2.14 (Syntactic-degree $d(\cdot)$)**.** *Let C be a circuit (without division) 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 $d(v) := d(u) + d(w)$.*

(For syntactic-degree of circuits with division see Definition 6.2.)

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

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

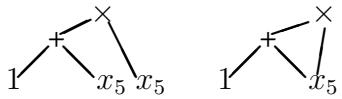
denote *any* circuit H whose output node is $u+v$ or $u\times v$, respectively, where H_u is identical to the circuit F and H_v is identical to the circuit G. In other words, $F\oplus G$ denotes a circuit with output node $+$ with the two incoming subcircuits F and G, where F and G *may not be disjoint*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

To clarify, $F\oplus G$ and $F\otimes G$ are different from, e.g., the *similarity* relation in [Jeř04, Definition 2.1] between two circuits $Q \sim Q'$ in which Q' can be the "minimal DAG" representation of Q. The circuit F in $F \oplus G$ *is identical* to F while in a circuit that is *similar* to $F + G$ we can have the subcircuit F replaced by a minimal DAG representation of F (and the same with G, and with $\otimes$). Hence, in $F \oplus G$ disjoint copies of nodes computing the same circuit that occur in *both* F and G can be contracted into a single node in $F \oplus G$. While disjoint copies of nodes computing the same circuit in F alone *cannot* be contracted into a single node (and the same with G, and with $\otimes$). Therefore, F cannot be contracted to a minimal DAG and neither can G.

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:



2.8 Equational Proofs of Polynomial Identities

In this section we give the necessary background on the 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].

Polynomial identities proofs as originally introduced in [HT09], denoted $\mathbb{P}_c$ (and $\mathbb{P}_c(\mathbb{G})$ when we wish to be explicit about the underlying 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 $\mathbb{P}_c(\mathbb{G})$-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 $\mathbb{P}_c(\mathbb{Z})$-proofs are treated as purely syntactic objects (similar to the way a propositional formulas in propositional proofs are syntactic objects). Thus, simple computations such as multiplying out brackets, are done explicitly, step by step.

Definition 2.15 ($\mathbb{P}_c(\mathbb{G})$, [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):*

$$\text{R1}\quad \frac{F = G}{G = F} \qquad \text{R2}\quad \frac{F = G \qquad G = H}{F = H}$$

$$\text{R3}\quad \frac{F_1 = G_1 \quad F_2 = G_2}{F_1 + F_2 = G_1 + G_2} \qquad \text{R4}\quad \frac{F_1 = G_1 \qquad F_2 = G_2}{F_1 \cdot F_2 = G_1 \cdot G_2}.$$

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 $a = b + c\,, \quad a' = b' \cdot c'$ *(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, each called a **proof-line**, $F_1 = G_1,\ F_2 = G_2, \ldots, F_k = G_k$, 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 $\mathbb{P}_c(\mathbb{Z})$-proof can be easily verified for correctness in deterministic polynomial-time by syntactically checking that each proof line is derived from previous lines by one of the inference rules or is an axiom (assuming A10 can be checked in polynomial-time, which is true for natural encoding of standard rings like rationals, integers, etc.). The predicate for correctness of $\mathbb{P}_c(\mathbb{Z})$-proofs is expressible with a $\mathbf{\Sigma}^B_0$-formula in $\mathbf{V}^0$ (see Section 3.2).

2.9 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 $\mathbb{Z}$.

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 $GF(2)$).

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 denoted **Div**:

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

Note that if F^{-1} is well-defined then both F is well-defined (assuming for example F itself contains division gates) and $F \neq 0$.

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 proof there may occur division gates u^{-1} for which $\widehat{u} = 0$, that is, in the axioms A1 to A10 and C1, C2, the circuits may contain division gates that compute the zero polynomial and in the axiom Div the circuit F^{-1} is *not* necessarily well-defined. We do not know how to check in (uniform) $\mathbf{NC^2}$ that a circuit is well-defined: there is no known $\mathbf{NC^2}$ algorithm to determine that an input circuit computes the zero polynomial. Hence, there is no $\mathbf{VNC}^2$ predicate for the correctness of $\mathbb{P}_c^{-1}$-proofs, only for the syntactic correctness of $\mathbb{P}_c^{-1}$-proofs. Since we will need to express the correctness of $\mathbb{P}_c^{-1}$-proof in $\mathbf{VNC}^2$ in some way our solution will be as follows: for every division gate u^{-1} in a given $\mathbb{P}_c^{-1}$-proof we are going to make sure $\mathbf{VNC}^2$ proves the existence of a $\mathbb{P}_c^{-1}$-proof of $u \restriction \rho = 1$ for some integer assignment ρ to the variables in u. In other words, every $\mathbb{P}_c^{-1}$-proof will be equipped with witnesses showing that its division gates are nonzero (see Section 4.2 about provably good nodes). In this case we will say that the $\mathbf{VNC}^2$ proves that the $\mathbb{P}_c^{-1}$-proof is ***correct*** (and not merely syntactically correct).

Note that if a $\mathbb{P}_c(\mathbb{Z})$-proof is syntactically correct then it is (fully) correct because $\mathbb{P}_c(\mathbb{Z})$-proofs do not contain the Div rule application.

3 Encoding Circuits and Proofs in the Theory

Here we explain how to encode algebraic circuits and $\mathbb{P}_c^{-1}(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-proofs in the theory. Specifically, in Section 3.1.2 we describe the circuit $\mathsf{Det}_{circ^{-1}}$, namely a circuit with division for the determinant. In Section 3.1.3 we explain how to construct $\mathsf{Det}_{circ^{-1}}$ in $\mathbf{V}^0$.

3.1 Encoding Circuits

In order to talk about algebraic circuits, Boolean circuits and $\mathbb{P}_c(\mathbb{Z})$- and $\mathbb{P}_c^{-1}(\mathbb{Z})$-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 $\mathbb{P}_c^{-1}(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-proofs in the theory follows similar lines, and we will not always define all the encoding details explicitly. We remark that all our circuits will have gates of fan-in at most two.

3.1.1 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 nodes are identified by natural numbers; this is convenient for our encoding scheme); (**ii**) a string of gates G, where each gate is a natural number interpreted as a pair of natural numbers (v, t) (using the $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ 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; the first two connectives are binary and the third is unary) or is the ith input; (**iii**) a two-dimensional string of *input gates* I, where, if the least significant bit of the ith string is 0, then the ith string of I excluding the least significant bit encodes a *variable* x_j, where for convenience the index j of an input variable x_j is represented using the binary representation of j; if the least significant bit is 1 the ith string of I excluding the least significant bit represents an integer scalar encoded in binary; and finally (**iv**) a string E encoding a set of numbers, each number is a pair that

represents a directed edge between two nodes, where $(u, v) \in E$ means that there is an incoming edge to $v \in V$ emanating from $u \in V$.

3.1.2 Circuit with Division for the Determinant

First we need to define the determinant circuit *with division* denoted $\mathsf{Det}_{circ^{-1}}$. This is done using the recursive Schur complement (similar to [HT15]), and can be viewed as performing 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 division 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:
$$X = \begin{pmatrix} X_1 & v_1^t \\ v_2 & x_{nn} \end{pmatrix}, \tag{12}$$
where $X_1 = \{x_{ij}\}_{i,j\in[n-1]}$, $v_1 = (x_{1n}, \ldots, x_{(n-1)n})$ and $v_2 = (x_{n1}, \ldots, x_{n(n-1)})$. Assuming we have constructed X_1^{-1}, let the *Schur complement* be defined as
$$\delta(X) := x_{nn} - v_2 X_1^{-1} v_1^t . \tag{13}$$
Since $\delta(X)$ computes a single *non-zero* rational function, $\delta(X)^{-1}$ is well-defined. Finally, let
$$X^{-1} := \begin{pmatrix} X_1^{-1}\left(I_{n-1} + \delta(X)^{-1} v_1^t v_2 X_1^{-1}\right) & -\delta(X)^{-1} X_1^{-1} v_1^t \\ -\delta(X)^{-1} v_2 X_1^{-1} & \delta(X)^{-1} \end{pmatrix}. \tag{14}$$

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 (12) and let $\delta(X)$ be as in (13). Let
$$\mathsf{Det}_{circ^{-1}}(X) := \mathsf{Det}_{circ^{-1}}(X_1) \cdot \delta(X) = \mathsf{Det}_{circ^{-1}}(X_1) \cdot (x_{nn} - v_2 X_1^{-1} v_1^t) . \tag{15}$$

The definition in (14) should be understood as a circuit with n^2 output gates that 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 Definitions 2.14 and 6.2). The fact that $\mathsf{Det}_{circ^{-1}}(X)$ indeed computes the determinant (as a rational function) stems, e.g., from the fact that $\mathbb{P}_c^{-1}(\mathbb{Z})$ can prove the two identities that characterize the determinant (Proposition 4.5).

Let M be a matrix in which each entry is a circuit (possibly with division) written either as a single circuit with n^2 output gates (such as X^{-1} above) or as n^2 separate circuits. Then we denote by M^{-1} the matrix X^{-1} of the symbolic matrix X in which each input variable x_{ij} is substituted by the (i, j)th entry of M.

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

Let $(M_n)_{n=1}^{\infty}$ be a family of $n \times n$ matrices in which each entry is a circuit (again, written either as a single circuit with n^2 output gates or n^2 separate circuits). We say that M_n (or M, when the subscript is not important to state explicitly) is a ***Σ_1^B-definable matrix*** in $\mathbf{V}^0$ if there is a Σ_1^B-definable string function in $\mathbf{V}^0$ that given a natural number n (in unary) outputs the circuit(s) encoding of M_n. In this section we show that the $n \times n$ inverse matrix X^{-1} from (14) above is Σ_1^B-definable in $\mathbf{V}^0$. We denote by $\mathsf{write}_{X^{-1}}(n)$ the string function that outputs the multi-output circuit X^{-1} given as input a unary integer n.

Note that the definition in (14) is implicitly a construction that uses Σ_1^B-induction (that is, the number induction axiom as in Proposition 2.2 in which we use Σ_1^B instead of Σ_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 neither $\mathbf{V}^0$ nor $\mathbf{VNC}^2$ has the number induction axiom for Σ_1^B-formulas we will construct the circuit by utilizing a natural encoding scheme, and formally by showing that the bit-graph of the string function $\mathsf{write}_{X^{-1}}(n)$ is definable with a Σ_0^B-formula (see Definition 2.9 for bit-definability). This idea and similar encoding is then used in the sequel to construct all of the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in the theory. (On the other hand, to proceed to the final division free $\mathbb{P}_c(\mathbb{Z})$-proofs using balancing we need to consider different arguments, including the axioms of $\mathbf{VNC}^2$, e.g., to be able to compute matrix powering (see Section 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 Σ_0^B-definable in $\mathbf{V}^0$ tupling number function). For each *inductive level* $d = 1, \ldots, n$ in the inductive definition of X^{-1} in (14), 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 nodes of the circuit* 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 the "inductive" level 1 in the definition of X^{-1} (when $n = 1$, $X^{-1} = x_{11}^{-1}$), the second coordinate is $(1,1)$, meaning the $(1,1)$-entry of the circuit computing the inverse of x_{11}, and the last coordinate is 0, meaning this is the output node of the circuit that computes x_{11}^{-1}. 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 (the latter information is expressed also in G). This allows us to bit-define in $\mathbf{V}^0$ a function that constructs the corresponding E and G as shown below.

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 with a Σ_0^B-formula the string function $\mathsf{write}_{X^{-1}}$ which by Fact 2.13 is enough to conclude that the function is Σ_1^B-definable in $\mathbf{V}^0$. We only need to construct, given some level d and the pair (i,j), the sub-circuits whose nodes will be $(d,(i,j),\ell)$, for some ℓ, according to the definition in (14). We will use the following notation and functions in the theory for this purpose.

Notations and basic functions for constructing sub-circuits. Let F be one of the four minors in (14) used to define the matrix inverse X^{-1}, for example $\delta(X)^{-1}$ (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 inputs are $\overline{I},\overline{O}$ serving as the input and output nodes, respectively, to the circuit F, d is the index "level" (used to record the induction-level of the inductive circuit constructions as in (14)) 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 (14), (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 (14). We want to construct a Σ_0^B-formula that bit-defines a function that given n outputs F_1. Note that F_1 is implicitly 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} in (14) (for the other simpler three minors). We define similar write_{F_i} functions for these F_i's.

To show that $\mathsf{write}_{X^{-1}}(n)$ is a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{V}^0$ we need to demonstrate how to bit-define this function using a $\mathbf{\Sigma}^B_0$-formula, and for this we need to show how to bit-define its sub-circuits. In our case we need to show how to bit-define for example $\mathsf{write}_{v\cdot u}$ using a $\mathbf{\Sigma}^B_0$-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 as demonstrated in Section 2.6.1. 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 $\mathbf{\Sigma}^B_0$-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 $\mathbf{\Sigma}^B_0$ bit-definition of write_{A+B} in $\mathbf{V}^0$.

Now that we have set up the notation and the functions for constructing sub-circuits, we can bit-define with a $\mathbf{\Sigma}^B_0$-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 all have $\mathbf{\Sigma}^B_0$-formulas that bit-define them.

Let $\mathsf{write}_{\mathsf{level}(X^{-1})}(n,d,\ell,\overline{I},\overline{O})$ be the string function that outputs (E,V,G) encoding the sub-circuit for the dth inductive level of X^{-1}, and let $\mathsf{write}_{x_{11}^{-1}}\big(n,1,0,((0,(1,1),0)),((1,(1,1),0))\big)$ be the string function that outputs the encoding of the circuit "x_{11}^{-1}" (given the overall dimension n, inductive-level 1, running index 0, input entry $(1,1)$ in X represented by $(0,(1,1),0)$ and output represented by $(1,(1,1),0)$). Assuming that $\mathsf{write}_{\mathsf{level}(X^{-1})}$ and $\mathsf{write}_{x_{11}^{-1}}$ have both $\mathbf{\Sigma}^B_0$-formulas that bit-define them, then the bit-definition of $\mathsf{write}_{X^{-1}}$ is given by the following $\mathbf{\Sigma}^B_0$-formula $\varphi(n,i)$, for n, i number sorts, and using $\mathsf{write}_{\mathsf{level}(X^{-1})}$ and $\mathsf{write}_{x_{11}^{-1}}$ as function symbols:

$$\exists 2 \le d \le n \Big(\mathsf{write}_{\mathsf{level}(X^{-1})}\big(n,d,1,\mathrm{Inp}_{F_1}(d),\mathrm{Out}_{F_1}(d)\big)(i) \vee \cdots \vee$$
$$\mathsf{write}_{\mathsf{level}(X^{-1})}\big(n,d,1,\mathrm{Inp}_{F_4}(d),\mathrm{Out}_{F_4}(d)\big)(i)\big) \vee \mathsf{write}_{x_{11}^{-1}}\big(n,1,0,((0,(1,1),0)),((1,(1,1),0))\big)(i)\Big),$$

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

3.2 Encoding and Witnessing Polynomial Identity Proofs

Recall the proof-systems $\mathbb{P}_c(\mathbb{Z})$ and $\mathbb{P}^{-1}_c(\mathbb{Z})$ which are proof systems that establish equalities between algebraic circuits without and with division, respectively, over the integers, and recall also the concept of correct and syntactically correct proofs (Section 2.9).

$\mathbb{P}^{-1}_c(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-proofs are encoded as two dimensional arrays 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 circuit encoding is done as described in Section 3.1.3). Furthermore, the encoding of $\mathbb{P}^{-1}_c(\mathbb{Z})$- and $\mathbb{P}_c(\mathbb{Z})$-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 it is not an axiom we specify 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 Div 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 need a special treatment. Consider $F_1 \oplus F_2 = F_1 + F_2$, and let V be the set of nodes (numbers) belonging to $F_1 \oplus F_2$. Every node $u \in V$, excluding the plus at the root, occurs as at most 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[5]. C2 is treated similarly.

When we talk about $\mathbb{P}_c^{-1}(\mathbb{Z})$- or $\mathbb{P}_c(\mathbb{Z})$-proofs 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 correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-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).

Definition 3.1 ($\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$, $\mathbb{P}_c(\mathbb{Z})$-proofs)**.** *Let $(\pi_n)_{n\in\mathbb{N}}$ be a sequence of $\mathbb{P}_c^{-1}(\mathbb{Z})$ or $\mathbb{P}_c(\mathbb{Z})$-proofs. We say that π_n* **is a $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}$(or $\mathbb{P}_c(\mathbb{Z})$, resp.) proof in a theory $\mathcal{T}$** *if the string function $f(n)$ that on the number input n outputs π_n is $\mathbf{\Sigma}_1^B$-definable in $\mathcal{T}$. In case π_n is a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (resp. $\mathbb{P}_c(\mathbb{Z})$-proof), we also assume that $\mathcal{T}$ proves the syntactic correctness (resp. correctness) of π_n. (When the parameter n is clear from the context we suppress it and may say that a $\mathbb{P}_c(\mathbb{Z})$- or $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F = G$ is $\mathbf{\Sigma}_1^B$-definable in $\mathcal{T}$, meaning that $F = G$ is parameterized by n).*

By Section 2.5, to show that $f(n)$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ (and can be used in the induction and comprehension schema of $\mathbf{V}^0$) it is enough to show the existence of a $\mathbf{\Sigma}_0^B$-formula $\varphi(n, i)$ with two natural numbers parameters n, i, that bit-defines the function $f(n)$.

We shall use the following simple statement that allows to use substitutions in $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs:

Proposition 3.2 (Substitution in $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs)**.** *Assume that in a theory $\mathcal{T} \supseteq \mathbf{V}^0$ the function that receives $n = |\vec{x}|$ and outputs a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F(\vec{x}) = G(\vec{x})$ is $\mathbf{\Sigma}_1^B$-definable. Let $\overline{H}$ be a sequence of circuits $\mathbf{\Sigma}_1^B$-definable as a string function from natural numbers $n = |\vec{x}|$ to $\overline{H}$. Then, the string function from n to a (provably in $\mathbf{V}^0$) syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F(\vec{x}/\overline{H}) = G(\vec{x}/\overline{H})$, where the variables in $\vec{x}$ are replaced by the circuits in $\overline{H}$, is $\mathbf{\Sigma}_1^B$-definable in $\mathcal{T}$.*

[5] 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.

Proof. Since the function that outputs a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F(\vec{x}) = G(\vec{x})$ is $\mathbf{\Sigma}_1^B$-definable in $\mathcal{T}$, and since by assumption $\mathcal{T}$ proves the syntactic correctness of this proof, $\mathcal{T}$ can $\mathbf{\Sigma}_1^B$-define the function that outputs each circuit in the proof (and prove it is syntactically correct). In each such circuit we substitute the input variables $\vec{x}$ by $\overline{H}$: since circuits are encoded as graphs substitution is formalized by substituting input nodes by graphs which is easily shown to be a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{V}^0$ (note that the function that given a circuit outputs its input nodes is by itself a $\mathbf{\Sigma}_1^B$-definable number function in $\mathbf{V}^0$). □

4 Existence of Proofs with Division for the Determinant Identities

4.1 Overview

We denote by X, Y the symbolic $n \times n$ matrices where the (i, j)th entries of X and Y are the variables x_{ij} and y_{ij}, respectively. Accordingly, in our $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs the X, Y matrices are encoded by their entries x_{ij}, y_{ij} to which we refer as *the* X, Y *variables* (namely, X, Y are not formal variables by themselves).

In this section we show (Proposition 4.5) a $\mathbf{\Sigma}_1^B$-definable string function in $\mathbf{V}^0$ that given a natural number n outputs a correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the following equations:

$$\mathsf{Det}_{circ^{-1}}(X) \cdot \mathsf{Det}_{circ^{-1}}(Y) = \mathsf{Det}_{circ^{-1}}(XY) \tag{16}$$

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

where U is a lower (equivalently, upper) triangular matrix in the variables x_{ij} (where each entry in U is a circuit possibly with division).

Recall that $\mathbb{P}_c^{-1}$-proofs consist of sequences of equations between algebraic circuits over $\mathbb{Z}$. In the proofs we construct in this section circuits have exponential syntactic-degrees (though the theory cannot express this fact), they are not necessarily homogeneous, and they have division gates. Recall also from Section 3.1.3 that $\mathsf{Det}_{circ^{-1}}(X)$ computes the determinant as a rational function and not as a polynomial (namely, it contains division gates).

To express the fact that a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is correct in the theory we use the notion of syntactic correctness together with witnesses that witness that the division gates used throughout the proof are nonzero (as required by the axiom Div of $\mathbb{P}_c^{-1}(\mathbb{Z})$); hence all rules and axioms in the proof are applied correctly. More precisely, we introduce the notion of *provably good nodes* which are division nodes for which there are (specific kind of) $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs witnessing they evaluate to 1 when the matrices X, Y are the identity matrices. In fact, we will specify precisely for which matrices that substitute X, Y and U, respectively, such witnesses for division nodes in the proof can be constructed.

We are going to construct a $\mathbf{\Sigma}_0^B$-formula that bit-defines a string function that given the number n outputs the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs. Recall that this would mean that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ according to Definition 3.1.

4.2 Provably Good Nodes

We begin by providing background definitions and statements that will help us eliminate division gates from circuits in the theory in future sections. Since division elimination in general [Str73] is not a uniform process as it builds on the mere *existence* of an assignment of field elements that allow for division elimination, we are going to show that a specific assignment, namely the identity matrix assignment, is sufficient for our purposes. The idea is to show that the theory can prove that every division gate in a circuit in the proof does not lead to division by zero when the matrices are substituted for the identity matrices. For this we introduce the following definitions.

A $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is said to be ***division axiom free*** if it does not use the axiom Div of division $F \cdot F^{-1} = 1$. Given a circuit F and a substitution ρ of variables in F by other circuits, $F \restriction \rho$ stands for a substitution instance of F in which substitution are performed as in Section 3.2.

Definition 4.1 (Identity matrices assignment)**.** *Given a natural number n, the* **identity matrices assignment** *ρ is defined to be the assignment of 0 and 1 elements to the* variables *x_{ij}, y_{ij} such that $\rho(x_{ij}) = \rho(y_{ij}) = 1$ if $i = j \in [n]$ and $\rho(x_{ij}) = \rho(y_{ij}) = 0$, if $i \neq j \in [n]$. In other words, $X \restriction \rho = Y \restriction \rho = I_n$, for I_n the $n \times n$ identity matrix and X, Y two $n \times n$ symbolic matrices.*

Definition 4.2 (Provably good node, circuit and proof)**.** *Let n be a natural number and ρ be the identity matrices assignment and let u^{-1} be a division gate in a circuit that uses the variables x_{ij}, y_{ij} (for $i, j \in [n]$). We say that the division gate u^{-1} is* **provably good** *whenever $u \restriction \rho = 1$ has a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π in which all division gates in π already appear in $u \restriction \rho$ (that is, we do not introduce new division gates in the proof). In this case we also say that ρ* is provably good for *u^{-1}. Accordingly, if all the division gates in a circuit C (a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, resp.) are provably good we say that ρ* is provably good for *C* (for the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, resp.) *and that C* (the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, resp.) is provably good*. We say that $\mathbf{V}^0$* **proves that** *u^{-1}* **is provably good** *whenever $\mathbf{V}^0$ proves that $u \restriction \rho = 1$ has a* syntactically correct *division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π in which all division gates in π already appear in $u \restriction \rho$. Accordingly, if $\mathbf{V}^0$ proves that all the division gates in a circuit C (a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, resp.) are provably good we say that $\mathbf{V}^0$* proves that *C* (the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, resp.) is provably good*.*

Using the concept of provably correct proofs we can now express in the theory the correctness of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs (and not merely syntactic correctness): let π be a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof. We say that $\mathbf{V}^0$ ***proves that*** π ***is correct*** whenever $\mathbf{V}^0$ proves that π is *both* syntactically correct and provably good. In Sections 8 and 9 that we will show that indeed such a correctness property suffices for our purposes in the sense that the theory will be able to prove the correctness of a (division-free) $\mathbb{P}_c(\mathbb{Z})$-proof of the determinant identities based on this formulation of correctness for $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof.

We will use the following definition:

Definition 4.3 (Provably invertible matrix)**.** *Let M be a matrix in which each entry is a circuit in the variables x_{ij}, y_{ij}. Then, M is said to be a* provably invertible matrix *if every division gate in M_n^{-1} is provably good.*

Note that if M_n is a $\mathbf{\Sigma}_1^B$-definable family of matrices in $\mathbf{V}^0$ (parameterised by n) then (using Proposition 3.2) M_n^{-1} is also a $\mathbf{\Sigma}_1^B$-definable matrix-family $\mathbf{V}^0$, since the latter is a substitution instance of X^{-1} of $n \times n$ in which we substitute entries of M_n. What we will need to show in some cases is not only that M_n^{-1} is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$, but that $\mathbf{V}^0$ *can also prove that M is provably invertible*, namely that every division gate in M_n^{-1} is provably good. Formally this would mean showing that there is a string function that on the natural number input n outputs the syntactically

correct division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs that witness the fact that all the division gates in M^{-1} are good.

In particular, we will strengthen the results of [HT15] that demonstrated the cases in which the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of the determinant identities are definable, into *provable* definability: we show that not only the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs do not contain zero division, but that $\mathbf{V}^0$ can prove the existence of syntactically correct division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs that witness this fact. More formally, the statement that expresses the fact that a matrix A is provably invertible is a formula that states that if A is a matrix and u is a division gate in A^{-1}, then u^{-1} is provably good.

For an $n \times n$ matrix A and a natural number $k \leq n$ we denote by $A[k]$ the $k \times k$ matrix restricted to rows 1 to k and columns 1 to k, namely the matrix $A(i,j)_{i,j\in[k]}$.

We have the following lemma:

Proposition 4.4 (Some facts about provably invertible matrices)**.** *Let n be a natural number and assume that A is a matrix in which each entry is a circuit in the variables x_{ij}, y_{ij}, for $i,j \in [n]$, which is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. Then the following hold:*

1. *$\mathbf{V}^0$ proves that the symbolic matrix X is provably invertible.*
2. *If $\mathbf{V}^0$ proves that A is provably invertible then $\mathbf{V}^0$ proves that $\mathsf{Det}_{circ^{-1}}(A)$ is provably good.*
3. *If $\mathbf{V}^0$ proves that A is a triangular matrix with $a_{11},\ldots,a_{nn}$ on the diagonal such that $a_{11}^{-1},\ldots,a_{nn}^{-1}$ are provably good then $\mathbf{V}^0$ proves that A is provably invertible.*
4. *$\mathbf{V}^0$ proves that $A[1],\ldots,A[n-1]$ are provably invertible and $\delta(A)^{-1}$ is provably good iff $\mathbf{V}^0$ proves that the matrix A is provably invertible.*

Proof. (1) We know that X^{-1} is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ by Section 3.1.2. We need to show that $\mathbf{V}^0$ proves that every division gate u^{-1} in X^{-1} is provably good, namely has a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$ with all division gates in this proof already appearing in $u \restriction \rho$. Observe using (14) that every division gate u^{-1} in X^{-1} is either x_{11}^{-1}, or is $\delta(X[k])^{-1}$ for some $k = 1,\ldots,n$, where $\delta(X[k]) = x_{kk} - v_{2k}(X[k])^{-1}v_{1k}^t$, such that v_{2k} is the kth row of $X[k]$ excluding the kth column entry, and v_{1k}^t is the kth column of $X[k]$ excluding the kth row entry (similar to the notation in (12)). The division gate x_{11}^{-1} is immediately proved in $\mathbf{V}^0$ to be provably good. The gates $(x_{kk} - v_{2k}(X[k])^{-1}v_{1k}^t)^{-1}$ are also proved easily in $\mathbf{V}^0$ to be provably good, because under the identity matrices assignment $v_{2k} = \mathbf{0}$ and $v_{1k} = \mathbf{0}$ are zero vectors, and $x_{kk} = 1$.

(2) By assumption $\mathbf{V}^0$ proves that every division gate in A^{-1} is provably good. By the definition of $\mathsf{Det}_{circ^{-1}}(A)$ in (15) every division gate in $\mathsf{Det}_{circ^{-1}}(A)$ already appears in A^{-1}, and so $\mathbf{V}^0$ can prove it is provably good.

(3) Assume that A is lower triangular (the case for upper triangular matrices is similar). We know that $\mathbf{V}^0$ proves that division by each diagonal entry a_{ii} in A is provably good. Since A is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ the inverse matrix A^{-1} is also $\mathbf{\Sigma}_1^B$-definable and we only need to make sure that $\mathbf{V}^0$ proves that all the division gates in A^{-1} are provably good. This follows by inspection of the inverse matrix as in Definition 14. Specifically, we consider all the division gates in A^{-1}, as follows.

In each inductive level n in (14), for n the dimension of the matrix, the *new* division gates that are introduced are those from $\delta(A)^{-1} = (a_{nn} - v_2(A[n-1])^{-1}\mathbf{0})^{-1}$, where $\mathbf{0}$ is the $(n-1)$-length (transposed) zero vector (corresponding to the vector $(A(1,n),\ldots,A(n-1,n))$, which is zero since A is upper triangular) and v_2 is the bottom vector of A excluding the entry $A(n,n)$, namely

$(A(n,1),\ldots,A(n,n-1))$. All the other division gates in A^{-1} come from previous inductive levels smaller than n and are placed inside $(A[n-1])^{-1}$. $\mathbf{V}^0$ proves that the equation $a_{nn} - v_2(A[n-1])^{-1}\mathbf{0} = a_{nn}$ has a syntactically correct division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof, and without adding new division gates that do not already occur in the equation, simply by using zero product axioms A8: $F \cdot 0 = 0$ of $\mathbb{P}_c^{-1}(\mathbb{Z})$. Hence, by substitution in $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs, $\mathbf{V}^0$ proves that $\delta(A)^{-1} = (a_{nn} - v_2(A[n-1])^{-1}\mathbf{0})^{-1} = a_{nn}^{-1}$ has a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof. Since we assumed that $\mathbf{V}^0$ proves that a_{nn}^{-1} is provably good we conclude that $\mathbf{V}^0$ proves that $\delta(A)^{-1}$ is provably good as well.

Since all division gates in A^{-1} come from some inductive level, each division gate in A^{-1} is of the form $\delta(A[k])^{-1}$ for some $k = 1,\ldots,n$, and thus $\mathbf{V}^0$ can prove it is provably good.

(4) This is similar to part (3) above: in each inductive level n in (14), for n the dimension of the matrix, the *new* division gates that are introduced are those from $\delta(A)^{-1} = (a_{nn} - v_2(A[n-1])^{-1}v_1^t)^{-1}$, where v_1 is the vector $(A(1,n),\ldots,A(n-1,n))$ and v_2 is the bottom vector $(A(n,1),\ldots,A(n,n-1))$. All the other division gates in A^{-1} come from previous inductive levels smaller than n and are placed inside $(A[n-1])^{-1}$. □

4.3 Constructing the $\mathbb{P}_c^{-1}(\mathbb{Z})$-Proofs in the Theory

Proposition 4.5.

1. *Let U be an $n \times n$ (upper or lower) Σ_1^B-definable triangular matrix in $\mathbf{V}^0$ with $u_{11},\ldots,u_{nn}$ on the diagonal, in the variables x_{ij}, y_{ij} for $i,j \in [n]$. Then the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*

$$\mathsf{Det}_{circ^{-1}}(U) = u_{11}\cdots u_{nn} \tag{18}$$

is Σ_1^B-definable in $\mathbf{V}^0$, and further $\mathbf{V}^0$ proves that if $u_{11}^{-1},\ldots,u_{nn}^{-1}$ are all provably good then the proof is provably good.

2. *Let X and Y be $n \times n$ symbolic matrices. Then the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*

$$\mathsf{Det}_{circ^{-1}}(X \cdot Y) = \mathsf{Det}_{circ^{-1}}(X) \cdot \mathsf{Det}_{circ^{-1}}(Y) \tag{19}$$

is Σ_1^B-definable in $\mathbf{V}^0$. Further, the proof is provably good for $X = A, Y = B$ provided that $\mathbf{V}^0$ proves that $A[k], B[k]$ and $A[k]B[k]$ are provably invertible for every $k \in [n]$, and A, B are Σ_1^B-definable matrices in $\mathbf{V}^0$.

The rest of this subsection is devoted to proving Proposition 4.5.

We shall follow the construction in [HT15, Section 7]: we show that the construction can be carried out in $\mathbf{V}^0$, but in addition we also show that $\mathbf{V}^0$ explicitly proves that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are provably good (that is, no zero division occurs in the proofs) whenever the matrices in the equation proved are provably invertible.

We prove parts 1 and 2 together, and we break the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs into several parts. We need to bit-define the function that constructs the required proofs. More precisely, we show that there exists a Σ_0^B-formula $\varphi(n,i)$ such that for all natural numbers n, i, $\varphi(n,i)$ holds iff $i \leq \mathrm{poly}(n)$ and the ith bit in the string that encodes the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the $n \times n$ determinant identity (18) is 1 (and similarly for (19)). The fact that these $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are syntactically correct provably in $\mathbf{V}^0$ will be straightforward, while for showing they are provably good we need to do some work.

We use in our construction, as well as in the sequel, the following simple statement that allows to encode in $\mathbf{V}^0$ proofs that consist of n parts, in which each part uses the conclusion of the previous

part as an assumption, provided that each part is constructed independently and uniformly from the other parts:

Proposition 4.6 ($\mathbf{V}^0$ construction of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in parts)**.** *Let $(\psi_n)_{n\in\mathbb{N}}$ be a sequence of equations between algebraic circuits with division over $\mathbb{Z}$, and let $F(n)$ be the string function that on input n outputs a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of ψ_n from assumption ψ_{n-1}. Suppose that $\varphi(n,i)$ is a $\mathbf{\Sigma}_0^B$-formula that bit-defines $F(n)$, where n,i are two number variables. Then, the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(\psi_n)_{n\in\mathbb{N}}$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$.*

Proof. The idea is that $F(n)$ depends only on the number n and not the previous proof-lines, hence there is no need to use $\mathbf{\Sigma}_1^B$-induction here. More precisely, we need to demonstrate a $\mathbf{\Sigma}_0^B$-formula that bit-defines the function that on input n outputs the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of ψ_n. We encode the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof so that it is a two-dimensional array S: the jth string in S, denoted $S^{[j]}$, is the (partial) $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof $F(n)$. Since $\varphi(n,i)$ bit-defines $F(n)$, the following $\mathbf{\Sigma}_0^B$-formula bit-defines the two-dimensional array S:

$$\Phi(n,k) := \exists j \le n\,(k = (j,i) \wedge \varphi(j,i))\,.$$

The only thing left to make sure is that $F(n)$ correctly points to the proof-line that holds the assumption ψ_{n-1}. For this we can simply assume that the first proof-line (i.e., equation) in $F(n)$ is ψ_{n-1}, and that it points to the last proof-line in $F(n-1)$ (which is also ψ_{n-1}; hence, repeating the same equation, and assuming that repetition of proof-lines is legitimate in $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs). □

We begin our $\mathbf{V}^0$-construction with the proof of $X \cdot X^{-1} = I_n$. Given $n \times n$ matrices X, Y, A, the expressions $XY = A$ in the context of a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof 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, the expression $X^{-1} = A$ stands for a set of n^2 separate circuits for each of the entries in X^{-1} (this is achieved by taking the same single multi-output circuit for X^{-1} as before, and duplicating each of the n^2 output gates together with their sub-circuits, so the increase in size is by a factor of n^2).

Lemma 4.7. *Let X be an $n \times n$ symbolic matrix. Then, the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $X \cdot X^{-1} = I_n$ and $X^{-1} \cdot X = I_n$ are $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. Moreover, if $X = A$ is a $\mathbf{\Sigma}_1^B$-definable matrix that $\mathbf{V}^0$ proves is provably invertible then $\mathbf{V}^0$ proves that these $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are provably good.*

Proof. This can potentially be constructed by induction on n. However, similar to the construction of $\mathsf{Det}_{circ^{-1}}$ in the theory (Section 3.1.3), 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 is bit-defined by a $\mathbf{\Sigma}_0^B$-formula in $\mathbf{V}^0$ (equivalently, a $\mathbf{\Sigma}_1^B$-definable function in $\mathbf{V}^0$). Since we already have the basic encoding scheme for circuits in the proof this is quite easy to achieve. We proceed as follows.

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, and in which all division gates are provably good because $\rho(x_{11}) = 1$. Otherwise, let $n > 1$ and X be as in (12). We want to construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $X \cdot X^{-1} = I_n$ from the assumption $X_1 \cdot X_1^{-1} = I_{n-1}$.

Abbreviate $a := \delta(X)$ and $D := I_{n-1} + a^{-1}v_1^t v_2 X_1^{-1} - a^{-1}v_1^t v_2 X_1^{-1}$, and $E := v_2 X_1^{-1} + a^{-1}(v_2 X_1^{-1} v_1^t - x_{nn})v_2 X_1^{-1}$. Using some rearrangements, and the definition of a, we have (see

[HT15, Proposition 7.2]):

$$\begin{aligned} X \cdot X^{-1} &= \begin{pmatrix} X_1 & v_1^t \\ v_2 & x_{nn} \end{pmatrix} \cdot \begin{pmatrix} X_1^{-1}(I_{n-1} + a^{-1}v_1^t v_2 X_1^{-1}) & -a^{-1}X_1^{-1}v_1^t \\ -a^{-1}v_2 X_1^{-1} & a^{-1} \end{pmatrix} \\ &= \begin{pmatrix} D & -a^{-1}v_1^t + a^{-1}v_1^t \\ E & a^{-1}(-v_2 X_1^{-1} v_1^t + x_{nn}) \end{pmatrix} \\ &= \begin{pmatrix} I_{n-1} & \mathbf{0} \\ v_2 X_1^{-1} - a^{-1}a v_2 X_1^{-1} & a^{-1}a \end{pmatrix} \\ &= \begin{pmatrix} I_{n-1} & \mathbf{0} \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{20}$$

To encode this proof we do the following: we use a Σ_0^B-formula denoted $\varphi(n,i)$ to bit-define the string function that given a natural number n outputs the above $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in (20), from the assumption $X_1 \cdot X_1^{-1} = I$, where X_1 has dimension $(n-1)\times(n-1)$. This is done by specifying $X_1 \cdot X_1^{-1} = I$ as a previous proof-line (formally, a collection of $(n-1)^2$ proof-lines) from which we derive our new proof-lines. Such a sequence is bit-defined by a Σ_0^B-formula similar to the encoding in Section 3.1.3: we compose basic constructions of matrix multiplications (which are written as 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 3.1.3). Proposition 4.6 suffices thus to conclude the first part of Lemma 4.7.

For the second part of the lemma: if $X = A$, for an $n \times n$ Σ_1^B-definable and provably invertible matrix A in $\mathbf{V}^0$, then by definition every division gate in A^{-1} is provably good. Hence, every division gate in the top equation in (20) is provably good. By inspection of the proof-lines that come after this top line we see that every division gate is one of the inverse gates that already appeared in A^{-1}. For example, let us inspect the (n,n)th entry in the second line from the top in (20), which when substituting A for X is $v_2 \cdot (-a^{-1}A_1^{-1}v_1^t) + a_{nn}a^{-1}$ (where $A_1 = A[n-1]$), which we turned by simple rearrangement to $a^{-1}(-v_2 A_1^{-1} v_1^t + a_{nn})$. The term a^{-1} appears already in A^{-1}, as well as all the division gates in A_1^{-1}. This then is equal to $a^{-1} \cdot a$ by definition of a, which then leads by the division axiom to 1, concluding that all division gates appearing in this derivation already appear in A^{-1}. □

Lemma 4.8. *The $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the identity $(XY)^{-1} = Y^{-1}X^{-1}$, for two $n \times n$ symbolic matrices X, Y, is Σ_1^B-definable in $\mathbf{V}^0$. Furthermore, if $X = A, Y = B$ where A, B, AB are Σ_1^B-definable and provably invertible in $\mathbf{V}^0$, then $\mathbf{V}^0$ proves that this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good.*

Proof. To keep distinctness of variables, we encode the variables x_{ij} differently from the variables y_{ij} (e.g., the input variables x_{ij} are encoded as before, while the y_{ij} variables are encoded as the pair 0 with the encoding of x_{ij}, which is always different from the encoding of x_{ij} since the pairing function is injective). By Proposition 3.2, since we have a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $XX^{-1} = I_n$, we can substitute in this proof to get a proof of $(XY)^{-1}(XY) = I_n$. We also have $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of basic properties of matrix products like associativity of matrix products and addition, and of $I_n X = X$. Hence, we can construct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $((XY)^{-1}(XY))Y^{-1}X^{-1} = Y^{-1}X^{-1}$. On the other hand, by associativity of matrix product and Proposition 4.7 $((XY)^{-1}(XY))Y^{-1}X^{-1} = (XY)^{-1}(X(YY^{-1})X^{-1}) = (XY)^{-1}$ and so $(XY)^{-1} = Y^{-1}X^{-1}$.

For the second part of the lemma, we observe that since AB is provably invertible, by Proposition 4.7 every division gate in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(AB)^{-1}(BA) = I_n$ is provably good. And since A and B are provably invertible then by Proposition 4.7 the proof of $(AB)^{-1}(A(BB^{-1})A^{-1}) = (AB)^{-1}$ is provably good. □

Let X be as in (12) and similarly $Y = \begin{pmatrix} Y_1 & u_1^t \\ u_2 & y_{nn} \end{pmatrix}$. By definition of X^{-1} in (14) (and similarly Y^{-1}) the entry in the bottom right corner of $(XY)^{-1}$ is $\delta(XY)^{-1}$, and the entry in the bottom right corner of $Y^{-1}X^{-1}$ is $\delta(Y)^{-1}\delta(X)^{-1}((u_2Y_1^{-1})(X_1^{-1}v_1^t)+1)$. By Lemma 4.8 we have a proof of $(XY)^{-1} = Y^{-1}X^{-1}$ and so

$$\delta(XY)^{-1} = \delta(Y)^{-1}\delta(X)^{-1}\left((u_2Y_1^{-1})(X_1^{-1}v_1^t)+1\right).$$

Multiplying both sides by $\delta(XY)\delta(Y)\delta(X)$ we obtain a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of

$$\delta(Y)\delta(X) = \delta(XY)(1 + u_2Y_1^{-1}X_1^{-1}v_1^t), \tag{21}$$

where the proof is provably good if $X = A$ and $Y = B$ for A, B, AB provably invertible and $\mathbf{\Sigma}_1^B$-definable matrices in $\mathbf{V}^0$.

We proceed to construct the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of the identities (18), (19). We first provide the following two lemmas.

Lemma 4.9.

1. *Let A, L, U be $n \times n$ $\mathbf{\Sigma}_1^B$-definable matrices in $\mathbf{V}^0$ with L lower triangular and U upper triangular and assume that $\mathbf{V}^0$ proves that A, L, U are provably invertible. Then the string function that given n in unary outputs the syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*

$$\mathsf{Det}_{circ^{-1}}(LAU) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(A)\mathsf{Det}_{circ^{-1}}(U) \tag{22}$$

is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. Moreover, $\mathbf{V}^0$ proves that LAU is provably invertible and that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (22) is provably good.

2. *Let A be an $n \times n$ $\mathbf{\Sigma}_1^B$-definable matrix in $\mathbf{V}^0$. Then $\mathbf{V}^0$ proves that if A is provably invertible then there exists a provably invertible lower triangular matrix $L(A)$ and a provably invertible upper triangular matrix $U(A)$ such that $A = L(A)\cdot U(A)$ has a syntactically correct and provably good $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof.*

Proof. Part 1. Using $LAU = L(AU)I_n$ it is easy to see that it is enough to separately consider the cases in which $U = I_n$ and $L = I_n$. We prove the former, the latter is similar. We thus construct the proof of $\mathsf{Det}_{circ^{-1}}(LA) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(A)$.

As before, by Proposition 4.6 it suffices to demonstrate a $\mathbf{\Sigma}_0^B$-formula that bit-defines a string function that given a natural number n outputs a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(LA) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(A)$ from the assumption $\mathsf{Det}_{circ^{-1}}(L_1A_1) = \mathsf{Det}_{circ^{-1}}(L_1)\mathsf{Det}_{circ^{-1}}(A_1)$, where L, A have dimension $n \times n$ and L_1, A_1 have both dimension $(n-1)\times(n-1)$. Such a sequence is bit-defined by a $\mathbf{\Sigma}_0^B$-formula similar to the encoding shown in Section 3.1.3, as follows. If $n = 1$, the $\mathbf{\Sigma}_0^B$-formula is clear. If $n > 1$, write

$$L = \begin{pmatrix} L_1 & \mathbf{0} \\ u & \ell_{nn} \end{pmatrix}, \; A = \begin{pmatrix} A_1 & v_1^t \\ v_2 & a_{nn} \end{pmatrix}, \text{ and so } LA = \begin{pmatrix} L_1A_1 & L_1v_1^t \\ uA_1 + \ell_{nn}v_2 & \ell_{nn}a_{nn} + uv_1^t \end{pmatrix}. \tag{23}$$

Assume that $\mathbf{V}^0$ proves that L_1A_1 is provably invertible and that there exists a syntactically correct and provably good $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(L_1A_1) = \mathsf{Det}_{circ^{-1}}(L_1)\mathsf{Det}_{circ^{-1}}(A_1)$. We

want to show that $\mathbf{V}^0$ proves that LA is provably invertible and that there exists a syntactically correct and provably good $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(LA) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(A)$.

By $\delta(A) = a_{nn} - v_2 A_1^{-1} v_1^t$, and using rearrangement and Lemma 4.8 we can construct in $\mathbf{V}^0$ the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof:

$$\begin{aligned}\delta(LA) &= (\ell_{nn}a_{nn} + uv_1^t) - (uA_1 + \ell_{nn}v_2)(L_1A_1)^{-1}L_1v_1^t = \\ &= (\ell_{nn}a_{nn} + uv_1^t) - (uA_1 + \ell_{nn}v_2)A_1^{-1}L_1^{-1}L_1v_1^t = \\ &= \ell_{nn}a_{nn} + uv_1^t - uv_1^t - \ell_{nn}v_2A_1^{-1}v_1^t = \ell_{nn}(a_{nn} - v_2A_1^{-1}v_1^t) = \\ &= \delta(L)\delta(A)\,.\end{aligned}$$

Since $\mathbf{V}^0$ proves that L and A are provably invertible, Proposition 4.4 part 4 implies that $\mathbf{V}^0$ also proves that $\delta(L)^{-1}$ and $\delta(A)^{-1}$ are provably good. Hence, by the above $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof $\mathbf{V}^0$ also proves that $\delta(LA)^{-1}$ is provably good: we first derive $\delta(LA)^{-1} = (\delta(L)\delta(A))^{-1} = \delta(L)^{-1} \cdot \delta(A)^{-1}$ using the above proof—this does not add new division gates that do not already appear in $\delta(LA)^{-1}$; and then we use the fact that $\mathbf{V}^0$ proves that $\delta(L)^{-1}$ and $\delta(A)^{-1}$ are provably good.

By assumption $\mathbf{V}^0$ proves that L_1A_1 is provably invertible, which by (23) means that $(LA)[n-1]$ is provably invertible. From this and the fact that $\mathbf{V}^0$ proves that $\delta(LA)^{-1}$ is provably good, using Proposition 4.4 part 4, we conclude that $\mathbf{V}^0$ proves that LA is provably invertible.

Finally, by definition we have $\mathsf{Det}_{circ^{-1}}(A) = \mathsf{Det}_{circ^{-1}}(A_1) \cdot \delta(A)$ and $\mathsf{Det}_{circ^{-1}}(L) = \mathsf{Det}_{circ^{-1}}(L_1)\delta(L)$ and $\mathsf{Det}_{circ^{-1}}(LA) = \mathsf{Det}_{circ^{-1}}(L_1A_1)\delta(LA)$. We can conclude $\mathsf{Det}_{circ^{-1}}(LA) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(A)$ from the assumption $\mathsf{Det}_{circ^{-1}}(L_1A_1) = \mathsf{Det}_{circ^{-1}}(L_1)\mathsf{Det}_{circ^{-1}}(A_1)$ and the equation $\delta(LA) = \delta(L)\delta(A)$.

For part 2 we proceed once more to use Proposition 4.6. We construct a Σ_0^B-formula to bit-define a string function that given a natural number n outputs $L(A), U(A)$ and a syntactically correct $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $L(A)U(A) = A$ from the assumption $L(A_1)U(A_1) = A_1$, where $L(A_1), U(A_1)$ are lower and upper triangular, respectively, and have dimension $(n-1)\times(n-1)$.

If $n = 1$, let $L(a_{11}) = a_{11}$ and $U(a_{11}) = 1$. If $n > 1$, write A as in (23). By Lemma 4.7 we have a Σ_0^B-formula that bit-defines the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of $L(A_1)L(A_1)^{-1} = 1$ and $U(A_1)^{-1}U(A_1) = 1$. Therefore, we have a Σ_0^B-formula defining the following elementary $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (in which we also define $L(A), U(A)$) using as assumption the proof-line $A_1 = L(A_1)U(A_1)$:

$$\begin{pmatrix} A_1 & v_1^t \\ v_2 & a_{nn} \end{pmatrix} = \begin{pmatrix} L(A_1) & 0 \\ v_2U(A_1)^{-1} & a_{nn} - v_2A_1^{-1}v_1^t \end{pmatrix} \cdot \begin{pmatrix} U(A_1) & L(A_1)^{-1}v_1^t \\ 0 & 1 \end{pmatrix}.$$

The fact that $\mathbf{V}^0$ proves that $L(A), U(A)$ are provably invertible, provided that $\mathbf{V}^0$ proves that A is provably invertible, follows from similar reasoning as before. □

Lemma 4.10. *Let A be an $n \times n$ Σ_1^B-definable matrix in $\mathbf{V}^0$ that $\mathbf{V}^0$ proves to be provably invertible and let v_1, v_2 be $n \times 1$ Σ_1^B-definable vectors in $\mathbf{V}^0$ (with coordinates being algebraic circuits) such that $\mathbf{V}^0$ proves that $A + v_1^t v_2$ is provably invertible. Then, the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*

$$\mathsf{Det}_{circ^{-1}}(A + v_1^t v_2) = \mathsf{Det}_{circ^{-1}}(A)(1 + v_2A^{-1}v_1^t)$$

is Σ_1^B-definable in $\mathbf{V}^0$, and further $\mathbf{V}^0$ proves that the proof is provably good.

Proof. We start with the special case:

$$\mathsf{Det}_{circ^{-1}}(I_n + v_1^t v_2) = 1 + v_2v_1^t\,, \quad \text{where } \mathbf{V}^0 \text{ proves that } I_n + v_1^t v_2 \text{ is provably invertible.} \tag{24}$$

Let $v_1 = (u_1, c_1)$ and $v_2 = (u_2, c_2)$. We show a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (24) from the following two assumptions: (i) the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(I_{n-1} + u_1^t u_2) = (1 + u_2 u_1^t)$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$; and (ii) $\mathbf{V}^0$ proves both that this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good and that $I_{n-1} + u_1^t u_2$ is provably invertible. By Proposition 4.6 the first condition implies that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (24) is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. By the second condition $\mathbf{V}^0$ also proves this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good, because $\mathbf{V}^0$ proves that each part of the proof is.

If $n = 1$, the statement is clear. If $n > 1$, write $I_n + v_1^t v_2$ as follows

$$I_n + v_1^t v_2 = \begin{pmatrix} I_{n-1} + u_1^t u_2 & c_2 u_1^t \\ c_1 u_2 & 1 + c_1 c_2 \end{pmatrix}.$$

We first note the following:

Claim. *The $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(I_{n-1} + u_1^t u_2)^{-1} = I_{n-1} - (1 + u_2 u_1^t)^{-1} u_1^t u_2$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ and $\mathbf{V}^0$ proves it is provably good.*

Proof of claim: Abbreviate $D = u_1^t u_2$ and $\beta = (1 + u_2 u_1^t)$. Then we need to show $(I_{n-1} + D)^{-1} = I_{n-1} - \beta^{-1} D$. Recall that we assume that $I_{n-1} + D$ is provably invertible (assumption (ii) above). Multiplying both sides by $(I_{n-1} + D)$ we get $I_{n-1} = I_{n-1} + D - \beta^{-1}(I_{n-1} + D)D = I_{n-1} + D - \beta^{-1} D - \beta^{-1} D^2$. Multiplying both sides by β we get

$$\beta I_{n-1} = \beta(I_{n-1} + D) - (I_{n-1} + D)D. \tag{25}$$

Hence it is enough to show a $\mathbb{P}_c(\mathbb{Z})$-proof of (25), that is $\mathbf{\Sigma}_1^B$-definable and provably good in $\mathbf{V}^0$. By elementary rearrangements $D^2 = u_1^t(\beta - 1)u_2 = (\beta - 1)D$. And so we can use this identity to prove (25) as follows: $\beta(I_{n-1} + D) - (I_{n-1} + D)D = \beta I_{n-1} + \beta D - D - D^2 = \beta I_{n-1} + \beta D - D - (\beta - 1)D = \beta I_{n-1} + \beta D - D - \beta D + D = \beta I_{n-1}$. Our $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good by construction (stemming from $I_{n-1} + D$ being provably invertible). ■Claim

Let $\alpha := u_2 u_1^t$. By the claim, the definition of $\mathsf{Det}_{circ^{-1}}$ and the assumption $\mathsf{Det}_{circ^{-1}}(I_{n-1} + u_1^t u_2) = 1 + \alpha$, we obtain

$$\begin{aligned}
\mathsf{Det}_{circ^{-1}}(I_n + v_1^t v_2) =& \mathsf{Det}_{circ^{-1}}(I_{n-1} + u_1^t u_2)\left((1 + c_1 c_2) - c_2 u_2 (I_{n-1} + u_1^t u_2)^{-1} c_1 u_1^t\right) = \\
=& (1 + \alpha)\left((1 + c_1 c_2) - c_2 u_2 (I_{n-1} - (1 + \alpha)^{-1} u_1^t u_2) c_1 u_1^t\right) = \\
=& (1 + \alpha)(1 + c_1 c_2) - (1 + \alpha) c_1 c_2 u_2 u_1^t + c_1 c_2 u_2 u_1^t u_2 u_1^t = \\
=& (1 + \alpha)(1 + c_1 c_2) - (1 + \alpha) c_1 c_2 \alpha + c_1 c_2 \alpha^2 = \\
=& 1 + \alpha + c_1 c_2 = 1 + v_2 v_1^t .
\end{aligned}$$

This gives a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (24) which is also provably good, provably in $\mathbf{V}^0$.

Finally, we need to conclude $\mathsf{Det}_{circ^{-1}}(A + v_1^t v_2) = \mathsf{Det}_{circ^{-1}}(A)(1 + v_2 A^{-1} v_1^t)$ from (24), and show that $\mathbf{V}^0$ proves that the proof is provably good. Let $L := L(A)$ and $U := U(A)$ be the matrices from the statement of Lemma 4.9 part 2. This lemma (and the definition of $\mathsf{Det}_{circ^{-1}}$) gives a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of:

$$\begin{aligned}
\mathsf{Det}_{circ^{-1}}(A + v_1^t v_2) =& \mathsf{Det}_{circ^{-1}}(LU + v_1^t v_2) = \mathsf{Det}_{circ^{-1}}(L)\mathsf{Det}_{circ^{-1}}(I_n + L^{-1} v_1^t v_2 U^{-1})\mathsf{Det}_{circ^{-1}}(U) = \\
=& \mathsf{Det}_{circ^{-1}}(LU)(1 + v_2 U^{-1} L^{-1} v_1^t) = \mathsf{Det}_{circ^{-1}}(A)(1 + v_2 A^{-1} v_1^t) .
\end{aligned}$$

The above $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is provably good because by assumption A and $A + v_1^t v_2$ are both provably good. □

We can now conclude the main proof of this section:

Proof of Proposition 4.5. Part 1 follows from the definition of $\mathsf{Det}_{circ^{-1}}$. Specifically, assume that U is lower triangular (the other case is similar), then $\mathsf{Det}_{circ^{-1}}(U) = \mathsf{Det}_{circ^{-1}}(U_1) \cdot (u_{nn} - v_2 U_1^{-1}\mathbf{0})$, where v_2 is the nth row of U excluding u_{nn}, and $U_1 = U[n-1]$. Similar to previous arguments in this section, the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u_{nn} - v_2 U_1^{-1}\mathbf{0} = u_{nn}$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$, and $\mathbf{V}^0$ proves that the proof is provably good: the division gates in this proof come from U_1^{-1}, but we assumed that $\mathbf{V}^0$ proves that $u_{11}^{-1}, \ldots, u_{nn}^{-1}$ are all provably good and so by Proposition 4.4 part 3 $\mathbf{V}^0$ proves that U (and specifically $U[n-1]$) are provably invertible. Hence, using as before Proposition 4.6 we can conclude part 1.

Part 2 is proved once more by using Proposition 4.6. If $n = 1$, it is immediate. Assume that $n > 1$. Let

$$X = \begin{pmatrix} X_1 & v_1^t \\ v_2 & x_{nn} \end{pmatrix}, \quad Y = \begin{pmatrix} Y_1 & u_1^t \\ u_2 & y_{nn} \end{pmatrix}.$$

We want to show that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(XY) = \mathsf{Det}_{circ^{-1}}(X)\mathsf{Det}_{circ^{-1}}(Y)$ from the assumption that $\mathsf{Det}_{circ^{-1}}(X_1Y_1) = \mathsf{Det}_{circ^{-1}}(X_1)\mathsf{Det}_{circ^{-1}}(Y_1)$ is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ and that $\mathbf{V}^0$ proves that it is provably good.

Note that $(XY)[n-1] = X_1Y_1 + v_1^t u_2$. Thus, by the definition of $\mathsf{Det}_{circ^{-1}}$, we have

$$\mathsf{Det}_{circ^{-1}}(X) = \mathsf{Det}_{circ^{-1}}(X_1)\delta(X), \quad \mathsf{Det}_{circ^{-1}}(Y) = \mathsf{Det}_{circ^{-1}}(Y_1)\delta(Y) \quad \text{and}$$
$$\mathsf{Det}_{circ^{-1}}(XY) = \mathsf{Det}_{circ^{-1}}(X_1Y_1 + v_1^t u_2)\delta(XY),$$

and we are supposed to prove:

$$\mathsf{Det}_{circ^{-1}}(X_1Y_1 + v_1^t u_2)\delta(XY) = \mathsf{Det}_{circ^{-1}}(X_1)\delta(X) \cdot \mathsf{Det}_{circ^{-1}}(Y_1)\delta(Y). \tag{26}$$

By Lemma 4.10 we have

$$\mathsf{Det}_{circ^{-1}}(X_1Y_1 + v_1^t u_2) = \mathsf{Det}_{circ^{-1}}(X_1Y_1)(1 + u_2(X_1Y_1)^{-1}v_1^t), \tag{27}$$

where $\mathbf{V}^0$ proves that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of this equation is provably good whenever $\mathbf{V}^0$ proves that $(XY)[n-1] = X_1Y_1 + v_1^t u_2$ is provably invertible. We know that $\mathbf{V}^0$ proves that $X_1Y_1 + v_1^t u_2$ is provably invertible because by assumption $\mathbf{V}^0$ proves that $X[n]Y[n] = XY$ is provably invertible, and $(X_1Y_1 + v_1^t u_2)^{-1} = ((XY)[n-1])^{-1}$ is used in the definition of $(XY)^{-1}$.

We now use the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(X_1Y_1)^{-1} = Y_1^{-1}X_1^{-1}$ from Lemma 4.8. Note that if we assume that $\mathbf{V}^0$ proves that $X = A, Y = B$ and $A[k], B[k], A[k]B[k]$ are provably invertible for all $k \in [n]$, then in particular $\mathbf{V}^0$ proves that $X_1 = A[n-1], Y_1 = B[n-1]$ and $A[n-1]B[n-1]$ are provably invertible, and thus under this assumption Lemma 4.8 states that $\mathbf{V}^0$ proves that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $(X_1Y_1)^{-1} = Y_1^{-1}X_1^{-1}$ is provably good. From this, (27), and the assumption $\mathsf{Det}_{circ^{-1}}(X_1Y_1) = \mathsf{Det}_{circ^{-1}}(X_1)\mathsf{Det}_{circ^{-1}}(Y_1)$, we get

$$\mathsf{Det}_{circ^{-1}}(X_1Y_1 + v_1^t u_2) = \mathsf{Det}_{circ^{-1}}(X_1)\mathsf{Det}_{circ^{-1}}(Y_1)(1 + u_2Y_1^{-1}X_1^{-1}v_1^t).$$

Hence, in order to prove (26), it suffices to prove

$$(1 + u_2Y_1^{-1}X_1^{-1}v_1^t)\delta(XY) = \delta(X)\delta(Y),$$

which follows from (21) (where $\mathbf{V}^0$ proves that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of (21) is provably good if $\mathbf{V}^0$ proves that $X = A, Y = B$ and A, B and AB are provably invertible). □

5 Homogenization in $\mathbf{V}^0$

In the previous section we constructed in $\mathbf{V}^0$ a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof with division of the determinant identities. The theory $\mathbf{V}^0$ proves that this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is correct in the sense that it proves that each proof-line follows from previous lines correctly, or is an axiom, and that all division gates are provably good (meaning specifically that there is no division by zero, and that the Div inference rule is applied correctly). On the other hand, the syntactic-degrees of circuits in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (as defined for circuits with division in Definition 6.2) are not polynomially bounded. In order to be able to balance the circuits and then use the reflection principle for division-free $\mathbb{P}_c(\mathbb{Z})$-proofs in Theorem 11.3 we need to eliminate division and high syntactic-degrees. In this section we start developing an approach to eliminate high syntactic-degrees from $\mathbb{P}_c(\mathbb{Z})$-proofs in $\mathbf{V}^0$.

Recall the concepts of a homogeneous polynomial and a syntactic homogeneous circuit from Section 2.7. Here we demonstrate a $\mathbf{\Sigma}_1^B$-definable $\mathbf{FAC}^0$ algorithm in $\mathbf{V}^0$, for homogenizing algebraic circuits *without* division. This means that the algorithm receives algebraic circuits with no division gates and outputs the corresponding sum of homogenous components of the polynomial computed by the input circuit. We will use this algorithm in the theory in Section 9 in order to eliminate high syntactic-degrees from proofs, but we also use it in Section 6 to write the determinant as a circuit with no division gates.

In order to balance circuits in Section 10 we will need to work with the syntactic-degrees of nodes in homogeneous circuits. Nevertheless, we show that for most part we can get away with syntactic-degree *upper bounds* and not (the precise) syntactic-degrees, and so our homogenization algorithm below outputs a circuit in which every node carries a label specifying its syntactic-degree upper bound. On the other hand, for technical reasons, in the proof of the Cayley-Hamilton theorem in Section 12 we need to know precisely the syntactic-degree of nodes in some circuit (in order to use Lemma 7.3 part (3)). In this case we shall simply *witness* the syntactic-degrees of every node in the specific circuit we need, in which case the algorithm below outputs a homogeneous circuit with syntactic-degrees specified on every node.

Remark 5.1. *We note that we do not know of an $\mathbf{FAC}^0$ algorithm that computes the syntactic-degree of a node in a given circuit. On the other hand, 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 also need to prove its correctness; this is likely doable (as we essentially show for the [VSBR83] circuit balancing algorithm in Section 10), but we have opted for a shorter solution that uses syntactic-degree upper bounds or witness syntactic-degrees.*

Definition 5.2. *A* witness for the syntactic-degree *of all nodes in a circuit is a string that stores pairs of numbers (v, d), with v the node label and d its syntactic-degree, for every node v in the circuit.*

We can store each syntactic-degree as a natural number since we 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 $d(v)$ of the nodes v 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 $c \in \mathbb{Z}$, and x_i a variable.

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

Input: an algebraic circuit C with no division of size s and a natural number d.

Optional input 1: A syntactic-degree witness (Definition 5.2) for all the nodes in C (including the root that has syntactic-degree d).

Optional input 2: A natural number i (the intended degree of the syntactic homogeneous component to compute).

Output:

1. An algebraic circuit C' of size $O(d^2 \cdot s)$ such that C' is a sum of syntactic homogeneous circuits $C' = C^{(0)} + \cdots + C^{(d)}$.

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 $I \subseteq \{0, \ldots, d\}$ then output C, augmented with the nodes $[u, j] = 0$, for all nodes $u \in C$ and all $j \in \{0, \ldots, d\} \setminus I$.

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, only that instead of building the circuits by induction from leaves towards the 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 $[v, 0], \ldots, [v, d]$. For a node $[v, i]$ we call i the ***syntactic-degree upper bound of*** $[v, i]$, denoted as

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

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 computable 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 $i = 0, \ldots, d + 1$, 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 $i = 1, \ldots, d$. Otherwise, v is a variable x_j, and we define $[v, 1] = x_j$, and $[v, i] = 0$ for all $1 \neq i \in \{0, \ldots, d\}$.
Case 1: $v = u + w$ in C. Define $[v, i] := [u, i] + [w, i]$ for every $i = 0, \ldots, d$.
Case 2: $v = u \times w$ in C. Define $[v, i] := \sum_{\substack{j+k=i \\ j,k=0,\ldots,d}} [u, j] \times [w, k]$.

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

[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 $0 \leq j \leq d$ and $j \neq 1$.
Case 1: $v = u + w$ in C. Define $[v,i] := [u,i] + [w,i]$ for every $i = 0,\ldots,d(v)$, and $[v,i] := 0$ for $i = d(v)+1,\ldots,d$.
Case 2: $v = u \times w$ in C. Define $[v,i] := \sum_{\substack{j+k=i \\ j,k=0,\ldots,d}} [u,j] \times [w,k]$, for every $i = 0,\ldots,d(v)$, and $[v,i] := 0$ for $i = d(v)+1,\ldots,d$.[7]

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

Note on syntactic-degree upper bounds. Notice that if we do not have a witness for syntactic-degrees the above algorithm produces a syntactic homogeneous circuit (or a sum of such circuits) 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* (but not a witness for the syntactic-degree; this witness can be checked by following the upper bounds of syntactic-degrees of each of the nodes leading to the node). Assuming we get as input a *correct* syntactic-degree upper bound, that is, $d \geq d(C)$, then in output (1) we get $\widehat{C'} = \widehat{C}$.

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

We are going to construct $\mathbb{P}_c(\mathbb{Z})$-proofs that contain witnesses for the syntactic-degree upper bounds of every node.

Definition 5.3 (syntactic-degree upper bounds in proofs; d_{ub})**.** *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 *if every such node u is a pair of numbers $[u,i]$ for $d_{\mathrm{ub}}(u) = i$, according to the construction in the homogenization algorithm above.*

We have the following main theorem about homogenization of proofs:

Theorem 5.4 (Homogenizing $\mathbb{P}_c(\mathbb{Z})$-proofs homogeneous component-wise)**.** *Let F, G be two $\mathbf{\Sigma}^B_1$-definable algebraic circuits over $\mathbb{Z}$ with syntactic-degree d, and assume that there is a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ which is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$. Then, for every $k = 0,\ldots,d$, the following proof is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$: the $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(k)} = G^{(k)}$ in which every circuit is a sum of syntactic homogeneous circuits inside which every node u appears with its syntactic-degree upper bound, and $d_{\mathrm{ub}}(u) \leq k$.*

Remark 5.5. *Under the conditions stated in Theorem 5.4 we can also conclude easily that the following proof is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$: the $\mathbb{P}_c(\mathbb{Z})$-proof of $\sum_{i=0}^{k} F^{(i)} = \sum_{i=0}^{k} G^{(i)}$ in which every node u in the proof appears with its syntactic-degree upper bound, and $d_{\mathrm{ub}}(u) \leq k$. However, from this we cannot*

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

always conclude in $\mathbf{V}^0$ that $F = G$ is $\mathbb{P}_c(\mathbb{Z})$-provable. This is because without syntactic-degree witnesses for F we cannot necessarily conclude in $\mathbf{V}^0$ that there is a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = \sum_{i=0}^{k} F^{(i)}$. In Section 9 we are going to show that for the purpose of the determinant identities we do not need the syntactic-degree witnesses.

We need the following lemmas before concluding this theorem.

Lemma 5.6. *Let $F_1 \oplus F_2$ and $F_1 \otimes F_2$ be two $\mathbf{\Sigma}_1^B$-definable circuits[8] in $\mathbf{V}^0$ and let k be a natural number. Then, the following equation have $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c(\mathbb{Z})$-proofs in $\mathbf{V}^0$, in which every circuit is a sum of syntactic homogeneous circuits inside which every node u appears with its syntactic-degree upper bound:*

1. $(F_1 \oplus F_2)^{(k)} = F_1^{(k)} + F_2^{(k)}$;
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 5.4. For every $k = 0, \ldots, d$, we devise a $\mathbf{\Sigma}_1^B$-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 $d_{\text{ub}}(u) \leq k$. This is done in a manner resembling the algorithm above for homogenizing circuits and using Proposition 4.6. 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: the 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 $\mathbb{P}_c(\mathbb{Z})$. We construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $S^{(k)} = T^{(k)}$ with syntactic degree $\leq k$.

Lemma 5.6 gives $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ $\mathbb{P}_c(\mathbb{Z})$-proofs of $(F_1 \oplus F_2)^{(k)} = (F_1 + F_2)^{(k)}$ and $(F_1 \otimes F_2)^{(k)} = (F_1 \cdot F_2)^{(k)}$, as required for the axioms C1 and C2.

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

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

$$(F_1 \cdot (F_2 \cdot F_3))^{(k)} = ((F_1 \cdot F_2) \cdot F_3)^{(k)} . \tag{28}$$

By part (ii) of Lemma 5.6 the equations

$$(F_1 \cdot (F_2 \cdot F_3))^{(k)} = \sum_{i=0}^{k} \left(F_1^{(i)} \cdot \sum_{j=0}^{k-i} F_2^{(j)} F_3^{(k-i-j)} \right) \tag{29}$$

$$((F_1 \cdot F_2) \cdot F_3)^{(k)} = \sum_{i=0}^{k} \left(\sum_{j=0}^{i} F_1^{(j)} F_2^{(i-j)} \right) \cdot F_3^{(k-i)} , \tag{30}$$

can be proved in $\mathbb{P}_c(\mathbb{Z})$. In $\mathbb{P}_c(\mathbb{Z})$, the right hand sides of both (29) and (30) can be written as $\sum_{i+j+l=k} F_1^{(i)} F_2^{(j)} F_3^{(l)}$. This gives the proof of (28).

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:

$$\frac{S_1 = T_1 \quad S_2 = T_2}{S_1 \cdot S_2 = T_1 \cdot T_2} . \tag{31}$$

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 $i = 0, \ldots, k$, as follows: by Lemma 5.6 (in fact, direct construction suffices here, because here S_1, S_2 are disjoint circuits and the same with T_1, T_2), we can construct the proofs of $(S_1 \cdot S_2)^{(k)} = \sum_{i=0,\ldots,k} S_1^{(i)} \cdot S_2^{(k-i)}$ and $(T_1 \cdot T_2)^{(k)} = \sum_{i=0,\ldots,k} T_1^{(i)} \cdot T_2^{(k-i)}$. Hence, $(S_1 \cdot S_2)^{(k)} = (T_1 \cdot T_2)^{(k)}$ can be proved from the assumptions $S_1^{(i)} = T_1^{(i)}, S_2^{(i)} = T_2^{(i)}, i = 0, \ldots k$.

Note that, similar to Proposition 4.6, this is done independently and simultaneously for every proof-line. That is, given the rule application (31), 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 $i = 0, \ldots, k$; 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 $\mathbf{\Sigma}_1^B$-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: If the line $S = T$ is one of the rules R1 to R3, then 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 5.6. That every proof-line is written as a sum of syntactic homogeneous circuits follows by inspection of the construction. □

We will need the following claim (we sometimes use the notation "(in $\mathbf{V}^0$)" when the statement that follows (suitably encoded) is proved in the theory $\mathbf{V}^0$ (and similar for $\mathbf{VNC}^2$)):

Claim 5.7 (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 $i = d$. ■Claim

6 Preliminaries for Division Elimination

6.1 Overview

In this section we begin to provide the preliminaries for division elimination that we use in latter sections. 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. [HT15] used Strassen's method to eliminate division gates from proofs, by asserting the existence of an assignment that does not nullify any division gate in the proof. Such a non-constructive result was sufficient in [HT15], but not for our purpose: we do not know how to uniformly find such assignments in (uniform) $\mathbf{NC^2}$, and so we do not know how to eliminate division gates from general algebraic circuits in $\mathbf{VNC}^2$. Our solution is to use the concept of provably good $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs that we introduced for this reason in Section 4.2: we show that as long as the theory proves every division gate in the proof is provably good then we can eliminate division gates from $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs.

In Section 6.2 we show how to use a power series (which can be written as a circuit without division) to simulate in some sense division gates. For example, we can reason in $\mathbf{V}^0$ 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 $x \mapsto 1 - y$ which yields $(1-y)\cdot(1-y)^{-1}$. Substitute $(1-y)^{-1}$ by the term $\mathrm{Inv}_k(1-y)$ defined as:

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

which serves to "approximate" the inverse of $1-y$ up to degree k, in the sense that $(1-y)\cdot\mathrm{Inv}_k(1-y) = 1 - y^{k+1}$. 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 $1-y$ for x, and then substitute the circuit $\mathrm{Inv}_k(1-y)$ for $(1-y)^{-1}$. If $(1-y)^{(0)} = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof, then there exist $\mathbb{P}_c(\mathbb{Z})$-proofs of syntactic-degree at most k for the following equations:*
>
> $$((1-y)\cdot\mathrm{Inv}_k(1-y))^{(0)} = 1, \tag{32}$$
> $$((1-y)\cdot\mathrm{Inv}_k(1-y))^{(i)} = 0, \quad \textit{for } 1 \leq i \leq k." \tag{33}$$

We show in Section 8 (cf. Lemma 8.3) that assuming roughly that the inverse gate x^{-1} is provably good (for instance, in case ρ assigns 1 to x) then $(1-y)^{(0)} = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof.

In Section 6.3 we show how to normalize circuits with division gates in $\mathbf{V}^0$. In particular, 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 themselves division gates we say that C *has a single division gate at the top*. We need our circuits to have, roughly, a single division gate at the top, because later we replace division gates by their corresponding power series, but we do not know how to do it with nested divisions.

6.2 Approximating Inverses by Power Series

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 (namely, the polynomial F under the all-zero assignment to its

variables); we showed that such a syntactic homogeneous circuit can be constructed in $\mathbf{V}^0$ in Section 5. We now define the circuit $\mathrm{Inv}_k(F)$ that will serve as an inverse of F modulo high degree monomials, in the sense that

$$F \cdot \mathrm{Inv}_k(F) = 1 + [\text{monomials of degree greater than } k]\,. \tag{34}$$

Note that because we work over $\mathbb{Z}$ the only way for (34) to hold (for $k > 1$) is when the polynomial computed by $F^{(0)}$, that is $\widehat{F^{(0)}}$, equals 1, since if the constant term in F is not 1, no product of F can compute 1. (In general, the division elimination (as in Strassen [Str73]) needs to work over a field and have an inverse element for $\widehat{F^{(0)}}$.)

Assume that $\widehat{F^{(0)}} = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof 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 abbreviations of $(1-F)\cdots(1-F)$, k times (written as a logarithmic in k depth circuit; since k will always be polynomial in our applications this will be enough for our purposes). Note that $\mathrm{Inv}_k(1-z)$ for a variable z is the truncated Taylor expansion of $\frac{1}{1-z}$ at the point $z = 0$.

The following lemma demonstrates that $\mathrm{Inv}_k(F)$ can *provably* serve as the inverse of F up to degree k (note that neither F nor $\mathrm{Inv}_k(F)$ contain division), under some conditions:

Lemma 6.1. *Assume that F is a Σ_1^B-definable circuit* without division *in $\mathbf{V}^0$ and let η be a Σ_1^B-definable $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(0)} = 1$ and $k \geq 1$ be a natural number. Then, the $\mathbb{P}_c(\mathbb{Z})$-proofs of the following equations are Σ_1^B-definable in $\mathbf{V}^0$, and every node in every circuit in the proofs appears with its syntactic-degree upper bound (see Definition 5.3):*

$$(F \cdot \mathrm{Inv}_k(F))^{(0)} = 1 \tag{35}$$

$$(F \cdot \mathrm{Inv}_k F)^{(i)} = 0, \quad \textit{for } 1 \leq i \leq k\,. \tag{36}$$

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 $\mathbb{P}_c(\mathbb{Z})$:

$$\begin{aligned} F \cdot \mathrm{Inv}_k(F) &= (1-(1-F)) \cdot \left(1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k\right) \\ &= 1 + (1-F) + \cdots + (1-F)^k - (1-F) \cdot \left(1 + (1-F) + (1-F)^2 + \cdots + (1-F)^k\right) \\ &= 1 + (1-F) + \cdots + (1-F)^k - (1-F) - (1-F)^2 - \cdots - (1-F)^{k+1} \\ &= 1 - (1-F)^{k+1}\,. \end{aligned} \tag{37}$$

From (37) and Theorem 5.4 and Lemma 5.6 we construct a $\mathbb{P}_c(\mathbb{Z})$-proof of

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

wherein every node in every circuit in the proof appears with its syntactic-degree upper bound. From Theorem 5.4 we have a $\mathbb{P}_c(\mathbb{Z})$-proof 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 $(1-1)^{k+1} = 0$, hence we conclude (35).

To conclude (36) we proceed as follows. From (37) and Theorem 5.4 we construct $\mathbb{P}_c(\mathbb{Z})$-proofs $(F \cdot \mathrm{Inv}_k(F))^{(i)} = (1-(1-F)^{k+1})^{(i)}$, for all $1 \leq i \leq k$, with all nodes appear with their syntactic-degree upper bounds. From Lemma 5.6 we prove $(1-(1-F)^{k+1})^{(i)} = 1^{(i)} - ((1-F)^{k+1})^{(i)} = 0 - ((1-F)^{k+1})^{(i)}$.

To construct in the theory the $\mathbb{P}_c(\mathbb{Z})$-proof of $((1-F)^{k+1})^{(i)} = 0$, for $1 \leq i \leq k$, we use again Lemma 5.6. 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 $i \leq k$.) □

6.3 Division Normalization

We now show a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{V}^0$ that receives an algebraic circuit F with division and normalizes it by converting it into a circuit with a *single* division gate "at the top", namely a circuit of the form $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1}$ (formally, the output gate is a product gate with one of its children being a division gate). Accordingly we are going to normalize $\mathbb{P}^{-1}_c(\mathbb{Z})$-proofs, so that circuits in the proofs have no division gates that occur in the scope of other division gates (roughly speaking, normalizing every circuit in the proof so that the division gate appears only at the top). We need to normalize circuits in such a way in order to be able to eliminate division gates: as we shall see in Section 8, we replace a division 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 this can be guaranteed only if no division occurs in the scope of other division gates.

To extract the denominator and numerator of circuits with divisions we do the following: 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.

Definition 6.2 (Syntactic-degree for circuit with division)**.** *The* syntactic-degree of a circuit C with division *is*

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

Division normalization is formalized by a $\mathbf{\Sigma}^B_1$-definable string function in $\mathbf{V}^0$ 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 $\mathrm{Num}(v) := \mathrm{Num}(u_1) \cdot \mathrm{Num}(u_2)$ and $\mathrm{Den}(u) := \mathrm{Den}(u_1) \cdot \mathrm{Den}(u_2)$.

We have the following:

Proposition 6.3 (in $\mathbf{V}^0$). *Let F, G be circuits with division. Assume that $F = G$ has a $\mathbb{P}_c^{-1}(\mathbb{F})$-proof. Then $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1} = \mathrm{Num}(G) \cdot \mathrm{Den}(G)^{-1}$ has a syntactically correct[9] $\mathbb{P}_c^{-1}(\mathbb{F})$-proof such that no division gate in the proof occurs in the scope of another division gate.*

Proof. The proof of Proposition 6.3 is similar to the proof of Theorem 5.4 (and Proposition 4.6). We demonstrate only the following case (others are similar): assume that in the original $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F = G$ we have the following rule application: from $S = T$ and $H = K$ derive $S \cdot H = T \cdot K$. We are going to construct a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of:

$$\mathrm{Num}(S \cdot H) \cdot \mathrm{Den}(S \cdot H)^{-1} = \mathrm{Num}(T \cdot K) \cdot \mathrm{Den}(T \cdot K)^{-1}$$

using (pointers to) $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of

$$\mathrm{Num}(S) \cdot \mathrm{Den}(S)^{-1} = \mathrm{Num}(T) \cdot \mathrm{Den}(T)^{-1} \text{ and}$$

$$\mathrm{Num}(H) \cdot \mathrm{Den}(H)^{-1} = \mathrm{Num}(K) \cdot \mathrm{Den}(K)^{-1}.$$

By definition $\mathrm{Num}(S \cdot H) = \mathrm{Num}(S) \cdot \mathrm{Num}(H)$, $\mathrm{Den}(S \cdot H) = \mathrm{Den}(S) \cdot \mathrm{Den}(H)$, $\mathrm{Num}(T \cdot K) = \mathrm{Num}(T) \cdot \mathrm{Num}(K)$ and $\mathrm{Den}(T \cdot K) = \mathrm{Den}(T) \cdot \mathrm{Den}(K)$. Hence, the following is a $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in $\mathbf{V}^0$ (we do not show some trivial rule applications such as associativity, etc.):

$$\begin{aligned}
\mathrm{Num}(S \cdot H) \cdot \mathrm{Den}(S \cdot H)^{-1} &= \mathrm{Num}(S) \cdot \mathrm{Num}(H) \cdot (\mathrm{Den}(S) \cdot \mathrm{Den}(H))^{-1} \\
&= \mathrm{Num}(S) \cdot \mathrm{Num}(H) \cdot \mathrm{Den}(S)^{-1} \cdot \mathrm{Den}(H)^{-1} \\
&= \mathrm{Num}(S) \cdot \mathrm{Den}(S)^{-1} \cdot \mathrm{Num}(H) \cdot \mathrm{Den}(H)^{-1} \\
&= \mathrm{Num}(T) \cdot \mathrm{Den}(T)^{-1} \cdot \mathrm{Num}(K) \cdot \mathrm{Den}(K)^{-1} \\
&= \mathrm{Num}(T) \cdot \mathrm{Num}(K) \cdot \mathrm{Den}(T)^{-1} \cdot \mathrm{Den}(K)^{-1} \\
&= \mathrm{Num}(T \cdot K) \cdot \mathrm{Den}(T \cdot K)^{-1}.
\end{aligned}$$

Note that in the above $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof no division gate appears in the scope of another division gate (because $\mathrm{Num}(\cdot), \mathrm{Den}(\cdot)$ are circuits with no division). □

We now show that provably good nodes are preserved under division normalization, in the sense that for every node in a provably good circuit, $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. Recall that a division gate u^{-1} is provably good (Definition 4.2) whenever there is a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof η of $u \restriction \rho = 1$ where η does not contain new division gates not already in $u \restriction \rho$. Hence, if u has no division gates then η is a $\mathbb{P}_c(\mathbb{Z})$-proof (namely, it does not include division gates by itself). This then implies in particular that if we normalize divisions in a circuit with division F, then in $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1}$, there is only a single division gate and $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(F \restriction \rho) = 1$. This will be used in Section 8 when we completely eliminate division from $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs.

Lemma 6.4 (in $\mathbf{V}^0$). *Let F be a circuit with division and assume that all division gates in F are provably good. Then, for* every *(not necessarily division) node v in F there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.*

[9] Preservation of provably good nodes will be shown in what follows.

Proof. Simultaneously, for *every* node v in F we construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$, from which we conclude that $\mathrm{Den}(F \restriction \rho) = 1$ is provable in $\mathbb{P}_c(\mathbb{Z})$.

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 $\mathrm{Den}(w_2 \restriction \rho) = 1$ 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 (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* Div and in which we do not add new[10] division gates apart from the division gates already in $u \restriction \rho$ then there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) = \mathrm{Den}(u \restriction \rho)$.

Proof of claim: By Proposition 6.3 and by assumption that there is a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$ in which we do not use the axiom Div, we can show that there is a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) \cdot \mathrm{Den}(u \restriction \rho)^{-1} = 1 \cdot 1^{-1}$ that does not use the Div axioms, excluding trivial applications of the Div axioms (these applications stem from the construction of the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in the proof of Proposition 6.3). By multiplying each proof-line by $\mathrm{Den}(u \restriction \rho)$ we get a a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) \cdot \mathrm{Den}(u \restriction \rho)^{-1} \cdot \mathrm{Den}(u \restriction \rho) = \mathrm{Num}(u \restriction \rho) = \mathrm{Den}(u \restriction \rho) \cdot 1 \cdot 1^{-1}$ that does not use the Div axioms, excluding trivial applications of the Div axioms and the last application of Div $\mathrm{Den}(u \restriction \rho)^{-1} \cdot \mathrm{Den}(u \restriction \rho) = 1$. From this it is possible to show that there is a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) = \mathrm{Den}(u \restriction \rho)$ (we omit the details). ■$_{\text{Claim}}$

To finish case 3 we need to construct a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. By definition $\mathrm{Den}(v) = \mathrm{Num}(u)$, and so by substitution (Proposition 3.2) we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = \mathrm{Num}(u \restriction \rho)$. By assumption we have a division axiom free $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $u \restriction \rho = 1$, from which by the claim above we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Num}(u \restriction \rho) = \mathrm{Den}(u \restriction \rho)$. Since we assume that we already have a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(u \restriction \rho) = 1$ (using pointers to this proof) we finally get a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. □

Using Lemma 6.4 we can now strengthen the division normalization of $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs as stated in Proposition 6.3 by showing that normalizing proofs preserves provably good nodes:

Corollary 6.5. *Let F, G be circuits with division in the X, Y variables. Assume that $F = G$ has a $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ $\mathbb{P}_c^{-1}(\mathbb{F})$-proof. Suppose that $\mathbf{V}^0$ proves that for every division gate $v = t^{-1}$ in this proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.*[11] *Then, $\mathrm{Num}(F) \cdot \mathrm{Den}(F)^{-1} = \mathrm{Num}(G) \cdot \mathrm{Den}(G)^{-1}$ has a $\mathbb{P}_c^{-1}(\mathbb{F})$-proof such that no division gate in the proof occurs in the scope of another division gate. Furthermore, this $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$, and $\mathbf{V}^0$ proves that for every division gate $v = u^{-1}$ in this proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $u \restriction \rho = 1$.*

Proof. The statement is identical to Proposition 6.3, except that we require additionally that in the resulting normalized proof for every division gate $v = u^{-1}$ there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $u \restriction \rho = 1$. Inspecting the proof of Proposition 6.3, we observe that *every division gate in the resulting division normalized $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof is of the form $\mathrm{Den}(H)^{-1}$ for some H that appears in the original $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $F = G$*. Therefore, by assumption $\mathbf{V}^0$ proves that $\mathrm{Den}(H \restriction \rho) = 1$ has a $\mathbb{P}_c(\mathbb{Z})$-proof and we are finished. □

[10] We do not care for repeated occurrences of the *same* division gates in the proof.

[11] Note that $\mathrm{Den}(v)$ may be different from t because t can have division gates by itself.

7 From a Rational Function to the Determinant as a Polynomial

7.1 Overview

In order to be able to eliminate both division gates and high syntactic-degrees from the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs in Section 4, 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 final 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-degrees*. This is similar to division gates elimination from circuits following Strassen's work [Str73]: to eliminate division from circuits the circuit should compute a polynomial and not a rational function. (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-degrees; but as long as this only happens in intermediate proof-lines these can be eliminated from the proof in the sequel.)

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 (polynomial) coefficient of z^k in F, where 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 $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ at $z = 0$. Then, $\mathsf{Det}_{Taylor}(X)$ computes the determinant function: since every variable x_{ij} is multiplied by z, the coefficient of z^n is precisely the determinant.

By construction, $\mathsf{Det}_{Taylor}(X)$ will compute the determinant *as a polynomial*, except that it will contain some division gates in a sub-circuit that computes a constant (here 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 prove the correctness of this substitution in some way).

In Section 7.4 we show that there is a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of

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

for an $n \times n$ symbolic matrix X, which is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ and that $\mathbf{V}^0$ proves that it is provably good (Definitions 3.1 and 4.2). Combined with the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof from Section 4, we will get that $\mathbf{V}^0$ proves the existence of a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of the determinant identities in which the determinant circuit in the final identities is replaced by Det_{Taylor}.

In Section 7.5 we reduce the syntactic-degree of the circuit $\mathsf{Det}_{Taylor}(X)$ which 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). 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 no division gates, computes the determinant as a polynomial and has syntactic-degree n.

We start with Section 7.2 in which we show some $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs that express how $\mathsf{Det}_{circ^{-1}}$ behaves under elementary row and column operations.

7.2 Elementary Row and Column Operations

We formalize some elementary Gaussian operations that will be useful in the sequel.

Lemma 7.1. *Let A be an $n \times n$ matrix in the x_{ij}, y_{ij} variables (for $i, j \in [n]$) that is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ and assume that $\mathbf{V}^0$ proves that A is provably invertible. Then the following identities all have $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}^{-1}_c(\mathbb{Z})$-proofs in $\mathbf{V}^0$ and $\mathbf{V}^0$ proves the proofs are provably good under the assignment of identity matrices to the variables (i.e., $x_{ij} = y_{ij} = 0$ and $x_{ii} = y_{ii} = 1$ for $i \neq j \in [n]$):*

1. $\mathsf{Det}_{circ^{-1}}(A) = -\mathsf{Det}_{circ^{-1}}(A')$, *where A' is a matrix obtained from A by interchanging two rows or columns.*

2. $\mathsf{Det}_{circ^{-1}}(A'') = u\mathsf{Det}_{circ^{-1}}(A)$, *where A'' is obtained by multiplying a row in A by u, such that u^{-1} is provably good (and similarly for a column).*

3. $\mathsf{Det}_{circ^{-1}}(A) = \mathsf{Det}_{circ^{-1}}(A''')$, *where A''' is obtained by adding a row to a different row in A (and similarly for columns).*

4. $\mathsf{Det}_{circ^{-1}}(A) = a_{nn}\mathsf{Det}_{circ^{-1}}(A_1 - a^{-1}_{nn}v^t_1 v_2)$, *where A_1, v_1 and v_2 comes from the decomposition (12) (with A instead of X).*

Proof. Part 2 follows from Proposition 4.5: $A'' = TA$ (or $A'' = AT$ for the case of multiplying a column by u) for T a diagonal matrix with either 1 or u on the diagonal. Hence, $\mathbf{V}^0$ proves that $T[k], A[k]$ are provably invertible by Proposition 4.4. Moreover, since for every $k \in [n]$, $T[k]$ is a diagonal matrix with either 1 or u on the diagonal and $\mathbf{V}^0$ proves that $A[k]$ is provably invertible, $\mathbf{V}^0$ clearly also proves that $T[k]A[k]$ is provably invertible. Thus, by Proposition 4.5 (part 2) we have that $\mathsf{Det}_{circ^{-1}}(A'') = u\mathsf{Det}_{circ^{-1}}(A)$ has a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof in $\mathbf{V}^0$ that $\mathbf{V}^0$ proves is provably good.

Part 3 is similar to Part 2: in this case $A''' = T'A$, where T' is a suitable triangular matrix that $\mathbf{V}^0$ can prove is provably invertible.

In part 1 we cannot directly apply Proposition 4.5: we have $A' = TA$ where T is a transposition matrix, but T is not necessarily provably invertible (because not all division gates in T^{-1} are provably good). However, we can write $T = B_1B_2$, where both B_1, B_2 are provably invertible 0-1 triangular matrices and $\mathsf{Det}_{circ^{-1}}(B_1)\mathsf{Det}_{circ^{-1}}(B_2) = -1$. For example, $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$.

More formally, there exists a $\mathbf{\Sigma}^B_1$-definable string function in $\mathbf{V}^0$ that given the natural number n and two natural numbers i, j, outputs two 0-1 matrices B_1, B_2 such that B_1B_2 is the $n \times n$ transposition matrix of rows i and j (similar with columns) and such that $\mathbf{V}^0$ proves that $B_1[k], B_2[k], B_2[k]A[k], B_1[k](B_2A)[k]$ are provably invertible for all $k \in [n]$, and furthermore $\mathbf{V}^0$ proves that $\mathsf{Det}_{circ^{-1}}(B_1)\mathsf{Det}_{circ^{-1}}(B_2) = -1$. By Proposition 4.5 we get that the following equations have $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}^{-1}_c(\mathbb{Z})$-proofs in $\mathbf{V}^0$ that $\mathbf{V}^0$ proves are provably good:

$$\mathsf{Det}_{circ^{-1}}(B_1B_2A) = \mathsf{Det}_{circ^{-1}}(B_1)\mathsf{Det}_{circ^{-1}}(B_2A) = \mathsf{Det}_{circ^{-1}}(B_1)\mathsf{Det}_{circ^{-1}}(B_2)\mathsf{Det}_{circ^{-1}}(A)\,.$$

Part 4 is proved as follows:

$$\begin{aligned}
\mathsf{Det}_{circ^{-1}}(A) &= \mathsf{Det}_{circ^{-1}}(A_1) \cdot \delta(A) \\
&= \mathsf{Det}_{circ^{-1}}(A_1) \cdot (a_{nn} - v_2 A_1^{-1} v_1^t) \\
&= \mathsf{Det}_{circ^{-1}}(A_1) \cdot \big(a_{nn} \cdot \big(1 - a_{nn}^{-1} \cdot v_2 A_1^{-1} v_1^t\big)\big) \\
&= a_{nn} \cdot \big(\mathsf{Det}_{circ^{-1}}(A_1) \cdot \big(1 - a_{nn}^{-1} \cdot v_2 A_1^{-1} v_1^t\big)\big) \\
&= a_{nn} \cdot \big(\mathsf{Det}_{circ^{-1}}(A_1 - a_{nn}^{-1} v_1^t v_2)\big)
\end{aligned}$$

where the last equality follows from Lemma 4.10, the first equality from the definition of $\mathsf{Det}_{circ^{-1}}$ and the rest are elementary rearrangements. □

7.3 Extracting Polynomial Coefficients: Taylor Expansion

In order to compute the determinant as a polynomial, and not as a rational function (with division, that is), we need to write the determinant as the coefficient of z^n in the Taylor expansion (at $z = 0$) of a circuit with division that computes the determinant of $I_n + zX$, for X a symbolic $n \times n$ matrix.

The idea behind this Taylor expansion is the following: if a polynomial $F = F(\overline{x}, z)$ has no occurrence of the variable z in a division sub-circuit u^{-1} then we simply use the same homogenization approach as in Section 5 to extract the polynomial coefficients of each power z^k computed by F. Otherwise, the variable z occurs in a division sub-circuit u^{-1}. In this case we use the method in Section 6.3 to define the numerator and denominator of F simulating the denominator of F as a polynomial "up to the k-th degree", hence eliminating the division gates in F (for technical reasons there will still be some division gates left that we shall deal with later). Then, the polynomial coefficient of z^k is extracted as before (again, as in Section 5). Formally the coefficients of a Taylor expansion are defined as follows:

Definition 7.2 (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 (rational function in $\overline{x}$) 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 $\mathrm{coeff}_{z^k}(F)$, for $k \geq 0$, by induction on the structure of F as follows:*

1. $\mathrm{coeff}_z(z) := 1$ *and* $\mathrm{coeff}_{z^k}(z) := 0$*, if* $k \neq 1$*.*
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 $z = 0$:*

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

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 let*

$$G = \left(1 - F_0^{-1} \cdot \mathrm{Den}(F)\right).$$

We define

$$\mathrm{coeff}_{z^k}(F) := F_0^{-1} \cdot \mathrm{coeff}_{z^k}\left(\mathrm{Num}(F) \cdot \left(1 + G + G^2 + \cdots + G^k\right)\right). \tag{38}$$

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. Also, note that in Case 1 above $\mathrm{coeff}_{z^k}(F)$ *may contain division gates* whenever a division gate that does not contain z occurs in F. Further, in Case 2 the only division gate that can occur in $\mathrm{coeff}_{z^k}(F)$ is the division gate that occurs in the root of F_0^{-1}. In our application, when using $\mathrm{coeff}_{z^k}(F)$ we will need to make sure that $\mathbf{V}^0$ can prove that F_0^{-1} is provably good.

The construction of $\mathrm{coeff}_{z^k}(\cdot)$ in $\mathbf{V}^0$ is similar to the constructions in Section 5, and is shown formally below in Section 7.3.2. The following are the main properties of $\mathrm{coeff}_{z^k}(\cdot)$ (similar to [HT15]) that we can use already in $\mathbf{V}^0$:

Lemma 7.3. *1. If $F_0, \ldots, F_k$ are circuits* with *division not containing the variable z, $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$, and such that $\mathbf{V}^0$ proves that all division gates in the F_i's are provably good, then $\mathrm{coeff}_{z^j}\left(\sum_{i=0}^k F_i z^i\right) = F_j$ has a $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in $\mathbf{V}^0$, that $\mathbf{V}^0$ proves is provably good, for every $j \leq k$.*

2. Assume that F, G are circuits with division, *$\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. Suppose that $F = G$ has a $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof and $\mathbf{V}^0$ proves that the proof is provably good. Then, $\mathrm{coeff}_{z^k}(F) = \mathrm{coeff}_{z^k}(G)$ has a $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in $\mathbf{V}^0$ and $\mathbf{V}^0$ proves it is provably good, for every natural number k.*

3. Let F be a $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ circuit without *division, together with a witness for the syntactic-degree (Definition 5.2) 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 $\mathbf{\Sigma}_1^B$-definable $\mathbb{P}_c(\mathbb{Z})$-proof in $\mathbf{V}^0$.* [12]

Proof. Since the construction of $\mathrm{coeff}_{z^k}(\cdot)$ is very similar to the homogenization algorithm in Section 5, the proofs of parts 1 and 2 are almost identical to the proof of Theorem 5.4 for homogenizing $\mathbb{P}_c(\mathbb{Z})$-proofs, only that here we need in addition to make sure that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are provably good. The fact that the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are provably good can be demonstrated as in the Section 4.3, only that for part 2 we use Proposition 6.3, and the fact that if a circuit with division F is provably good then $\mathrm{Den}(F)^{-1}$ is provably good, which follows from Lemmas 6.4. We omit the details. The proof of part 3 is given in section 7.3.1 below (Lemma 7.5). □

7.3.1 Witnessing Syntactic-Degrees

Lemma 7.4 (in $\mathbf{V}^0$). *There exists a witness for the syntactic-degrees of all nodes in $\mathsf{Det}^{\#}_{Taylor}(X)$.*

Proof. We provide details on 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 (44) (see also (40)), which is a circuit without division of syntactic-degree n that computes as a polynomial the determinant of the symbolic $n \times n$ matrix X.

We explain how to $\mathbf{\Sigma}_1^B$-define a number function in $\mathbf{V}^0$ that computes the syntactic-degree of a node v, given the number n and the node v as inputs, where v is a node in $\mathsf{Det}^{\#}_{Taylor}(X)$. Consider

[12] The only place where we need this part is the proofs of the Cayley-Hamilton theorem in Section 12.

a node v in $\mathrm{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ or in $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$. If we can compute in $\mathbf{V}^0$ the syntactic-degree of such a node v then by inspection of the circuit $\mathsf{Det}^{\#}_{Taylor}(X)$ and and the definition of coeff_k we can conclude that there is a $\mathbf{\Sigma}^B_1$-definable number function in $\mathbf{V}^0$ that computes the syntactic-degree of any given node v in $\mathsf{Det}^{\#}_{Taylor}(X)$ (given n as an input).

Consider a node v in $\mathrm{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ for example. Recall the encoding scheme for circuits $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ described in 3.1.3, and let d denote the "inductive level" in the definition of $\mathsf{Det}_{circ^{-1}}$ in (15). To compute the syntactic-degrees of nodes that are at most n in $\mathrm{Num}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ we wish to compute the pair of syntactic-degrees of the numerator and denominator $(d(\mathrm{Num}(v)), d(\mathrm{Den}(v)))$ that we call *the syntactic-degree pair of* v, for every 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 to compute its syntactic-degree. 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 (14). 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 have explicit ($\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$) number functions. Also, notice the syntactic-degree of some nodes in F_1 is exponential because the repeated multiplication of X_1^{-1} by itself, hence we need to consider only those nodes whose syntactic-degree is polynomial in n.

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 a 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 a syntactic-degree pair $(1,0)$, for every level d.

We demonstrate this idea on $\delta(X)^{-1}$ for $n=2$, 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 $\mathsf{Det}_{circ^{-1}}(I_n + zX)$. We have that $\delta(X)^{-1} = (x_{22} - x_{21} \cdot x_{11}^{-1} \cdot x_{12})^{-1}$. Thus, $\mathrm{Num}(\delta(X)^{-1}) = \mathrm{Den}(x_{22}) \cdot \mathrm{Den}(x_{21} \cdot x_{11}^{-1} \cdot x_{12}) = \mathrm{Den}(x_{22}) \cdot \mathrm{Den}(x_{21}) \cdot \mathrm{Den}(x_{11}^{-1}) \cdot \mathrm{Den}(x_{12}) = 1 \cdot 1 \cdot x_{11} \cdot 1$. Hence, $d(\mathrm{Num}(\delta(X)^{-1})) = 1$. □

Using witnesses for syntactic-degrees we can now prove Lemma 7.3 part 3 (which will be used in the proof of Theorem 12.1).

Lemma 7.5 (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, $F = \sum_{i=0}^{d} \mathrm{coeff}_{z^i}(F) \cdot z^i$ 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

$$F_v = \sum_{k=0}^{d(v)} F_v^{(k)} \tag{39}$$

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 5.6 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).

$$\begin{aligned}
\sum_{i=0}^{d} F_v^{(i)} &= \sum_{i=0}^{d} (F_u \oplus F_w)^{(i)} && \text{by assumption}\\
&= \sum_{i=0}^{d} ({F_u}^{(i)} + {F_w}^{(i)}) && \text{by Lemma 5.6}\\
&= \sum_{i=0}^{d} {F_u}^{(i)} + \sum_{i=0}^{d} {F_w}^{(i)} && \text{rearrangement}\\
&= F_u + F_w && \text{by "previous" lines}\\
& && \text{(add explicit pointers to the appropriate proof-lines)}\\
&= F_u \oplus F_w = F_v && \text{by axiom C1.}
\end{aligned}$$

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.

$$\begin{aligned}
\sum_{i=0}^{d} F_v^{(i)} &= \sum_{i=0}^{d} (F_u \otimes F_w)^{(i)} && \text{by assumption}\\
&= \sum_{i=0}^{d} \sum_{\substack{l+j=i\\ 0\le l\le d_1, 0\le j\le d_2}} F_1^{(l)} \cdot F_2^{(j)}\,. && \text{by Lemma 5.6 part (3)}\\
&= \sum_{i=0}^{d_1} {F_u}^{(i)} \cdot \sum_{i=0}^{d_2} {F_w}^{(i)} && \text{rearrangement}\\
&= F_u \cdot F_w && \text{by "previous" lines}\\
& && \text{(add explicit pointers to the appropriate proof-lines)}\\
&= F_u \otimes F_w = F_v && \text{by axiom C1.}
\end{aligned}$$

□

7.3.2 Algorithm for coeff

The following is similar to the homogenization algorithm from Section 5. It assumes that the input circuit is an algebraic circuit possibly with division, in which z does *not* occur in the scope of any division gate. In case z *does* occur in the scope of a division gate in the circuit we apply Case 2 (using Equation (38)) of Definition 7.2 and then use the algorithm that follows.

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

Input: an algebraic circuit C with z not occurring in the scope of any division gate and a natural number k (given in unary).
Output: an algebraic circuit computing $\mathrm{coeff}_{z^k}(C)$.

Algorithm: Every node v in C is duplicated $k+1$ times into the nodes $[v,0],\ldots,[v,k]$, 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 $i=0,\ldots,k+1$, 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 $i=1,\ldots,k$. Otherwise, $v=z$ and we define $[v,1]=1$, and $[v,i]=0$ for all $1\neq i\in\{0,\ldots,k\}$.
Case 1: $v=u+w$ in C. Define $[v,i]:=[u,i]+[w,i]$ for every $i=0,\ldots,k$.
Case 2: $v=u\times w$ in C. Define $[v,i]:=\sum_{\substack{j+r=i\\ j,r=0,\ldots,k}}[u,j]\times[w,r]$.

7.4 From Determinant as Rational Function to a Polynomial in $\mathbb{P}_c^{-1}(\mathbb{Z})$

As mentioned in the overview for this section, given a symbolic $n\times n$ matrix X the polynomial-size (in n) circuit computing the determinant as a polynomial is defined as follows:

$$\mathsf{Det}_{Taylor}(X):=\mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n+zX)). \tag{40}$$

By the definition of coeff_{z^n}, the circuit $\mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I+zX))$ contains occurrences of the following inverse gate: $(\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n+zX)))^{-1}$ at the point $z=0$ (this is F_0^{-1} in the notation of (38)). $\mathrm{Den}(\mathsf{Det}_{circ^{-1}}(I_n+zX))(0/z)$ has an exponential syntactic-degree which will be dealt with below in Section 7.5.

We can now use Lemma 7.3 to construct the desired $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof:

Lemma 7.6. *Let U be a $\mathbf{\Sigma}_1^B$-definable triangular matrix in $\mathbf{V}^0$ in the variables x_{ij} for $i,j\in[n]$ and with $u_{11},\ldots,u_{nn}$ on the diagonal. Assume that A is either X, XY, or U, where X,Y are $n\times n$ symbolic matrices.*

1. *The $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*
$$\mathsf{Det}_{Taylor}(U)=u_{11}\cdots u_{nn}$$
is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$, and further $\mathbf{V}^0$ proves that if u_{ii}^{-1} are provably good for all $i\in[n]$ then the proof is provably good.

2. *The $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of*
$$\mathsf{Det}_{Taylor}(A)=\mathsf{Det}_{circ^{-1}}(A)$$
is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$. Further, for $A=X,XY$ the theory $\mathbf{V}^0$ proves that the proofs are provably good, and for $A=U$, $\mathbf{V}^0$ proves that if u_{ii}^{-1} are provably good for all $i\in[n]$ then the proof is provably good.

To prove Lemma 7.6 we use Lemma 7.3 parts 1 and 2 (since we do not use part 3 of Lemma 7.3 we do not need to construct a witness for the syntactic-degrees of any circuit in this case). The proof follows [HT15, Proposition 7.9] except that we need to make sure that $\mathbf{V}^0$ proves the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof to be provably good. We start with the following technical lemma:

Lemma 7.7. *Let A be an $n\times n$ matrix in the variables x_{ij},y_{ij},z_{ij} for $i,j\in[n]$ that is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$and that $\mathbf{V}^0$ proves is provably invertible. Then, there exist circuits with divisions $P_0,\ldots,P_{n-1}$ not containing the variable z, such that*

$$\mathsf{Det}_{circ^{-1}}(zI_n+A)=z^n+P_{n-1}z^{n-1}+\cdots+P_0$$

has a Σ_1^B-definable $\mathbb{P}_c^{-1}(\mathbb{F})$-proof in $\mathbf{V}^0$, and for $z = 0$ the theory $\mathbf{V}^0$ proves that the proof is provably good.

Proof. Let F be a circuit in which z does not occur in the scope of any inverse gate. Then, we define the *z-degree of F* as the syntactic-degree of F considered as a circuit computing a univariate polynomial in z (so that all other variables are treated as constants).

We construct a Σ_1^B-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in $\mathbf{V}^0$ of $\mathsf{Det}_{circ^{-1}}(A_1) = \cdots = \mathsf{Det}_{circ^{-1}}(A_n) = z^n + f$, where $A_1, \ldots, A_n$ are matrices to be constructed below with $A_1 = zI_n + A$ and f is a circuit with z-degree smaller than n in which z is not in the scope of any division gate, and such that $\mathbf{V}^0$ proves the proof to be provably good when $z = 0$. Writing f as $\sum_{i=0}^{n-1} P_i z^i$ will conclude the lemma. For this purpose, by Proposition 4.6 it suffices to demonstrate the following for every $k \in [n]$:

1. Assume that A_k is a Σ_1^B-definable in $\mathbf{V}^0$ $(n-k+1) \times (n-k+1)$ matrix of the form

$$\begin{pmatrix} z^k + f & w \\ v^t & zI_{n-k} + Q \end{pmatrix}$$

   where all the entries are circuits with division in which z does not occur in the scope of any division gate, v and w are $1 \times (n-k)$ vectors, and further: f as well as every entry of w have z-degree less than k and both v and Q do not contain the variable z. (Hence, specifically when $k = n$ we get that $A_k = z^k + f$, with f a circuit with z-degree smaller than n in which z is not in the scope of any division gate, as required.)

2. Assume also that $\mathbf{V}^0$ proves that all division gates in A_k are provably good and that A_k is provably invertible, for $z = 0$.

3. There is a Σ_1^B-definable in $\mathbf{V}^0$ $(n-k+1) \times (n-k+1)$ matrix A_{k+1} of the form

$$\begin{pmatrix} z^{k+1} + f & w \\ v^t & zI_{n-k-1} + Q \end{pmatrix}$$

   with the same notation as in (1) above (when $k+1$ is replaced for k; note that the values of w, v, Q, f may be different from the values in (1)). Further, there is a Σ_0^B-formula that bit-defines the string function that given a natural number k outputs a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(A_k) = \mathsf{Det}_{circ^{-1}}(A_{k+1})$, and $\mathbf{V}^0$ proves the proof is provably good when $z = 0$. In particular by 4.4 part 4, $\mathbf{V}^0$ proves that A_{k+1} is provably invertible when $z = 0$.

4. $\mathbf{V}^0$ proves that all division gates in A_{k+1} are provably good when $z = 0$.

We construct the following $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof (as required by part (3)) and encode it by a Σ_0^B-formula similar to the encoding shown in Section 3.1.3. Assume that A_k is written as

$$A_k = \begin{pmatrix} z^k + f_1 & w & f_2 \\ u_1^t & zI_m + Q & u_2^t \\ a_1 & v & z + a_2 \end{pmatrix}$$

where $m = (n-k-1)$ and we allow the possibility that $m = 0$. Suppose that f_1, w and f_2 have z-degree smaller than k, and z does not occur in u_1, u_2, Q, a_1, a_2 and v. By Lemma 7.1 part

1, since A_k is a matrix in the variables x_{ij}, y_{ij}, z_{ij} (for $i,j \in [n-k+1]$) that is Σ_1^B-definable in $\mathbf{V}^0$ and that $\mathbf{V}^0$ proves that A_k is provably invertible when $z=0$, we can switch the first and last column to obtain a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of

$$\mathsf{Det}_{circ^{-1}}(A_k) = -\mathsf{Det}_{circ^{-1}}\begin{pmatrix} f_2 & w & z^k+f_1 \\ u_2^t & zI_m+Q & u_1^t \\ z+a_2 & v & a_1 \end{pmatrix} \tag{41}$$

that is Σ_1^B-definable in $\mathbf{V}^0$, and that $\mathbf{V}^0$ proves is provably good when $z=0$. Furthermore, $\mathbf{V}^0$ clearly proves that the matrix on the right hand side of (41) is provably invertible because it proves that $\mathsf{Det}_{circ^{-1}}(\cdot)$ of this matrix is provably good for $z=0$ (and also using 4.4 part 4).

By Lemma 7.1 part 4, we have the following Σ_1^B-definable in $\mathbf{V}^0$ $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof which $\mathbf{V}^0$ proves is provably good when $z=0$:

$$\mathsf{Det}_{circ^{-1}}(A_k) = -a_1\mathsf{Det}_{circ^{-1}}\begin{pmatrix} f_2 - a_1^{-1}(z^k+f_1)(z+a_2) & w - a_1^{-1}(z^k+f_1)v \\ u_2^t - a_1^{-1}u_1^t(z+a_2) & zI_m+Q-a_1^{-1}u_1^tv \end{pmatrix} = \mathsf{Det}_{circ^{-1}}\begin{pmatrix} (z^k+f_1)(z+a_2)-a_1f_2 & (z^k+f_1)v-a_1w \\ u_2^t - a_1^{-1}u_1^t(z+a_2) & zI_m+Q-a_1^{-1}u_1^tv \end{pmatrix}. \tag{42}$$

We can write $(z^k+f_1)(z+a_2) = z^{k+1} + (f_1z + a_2z^k + f_1a_2)$, where the z-degree of $(f_1z + a_2z^k + f_1a_2)$, as well as of every entry of $(z^k+f_1)v - a_1w$, is at most k. Multiplying the matrix in (42) by $\begin{pmatrix} 1 & 0 \\ -a_1^{-1}u_1^t & I_m \end{pmatrix}$ from the right we obtain A_{k+1} of the form required above in (3). Using Lemma 4.9 we get that (3) is equal to $\mathsf{Det}_{circ^{-1}}(A_{k+1})$ with a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof that is Σ_1^B-definable in $\mathbf{V}^0$ and which $\mathbf{V}^0$ proves is provably good for $z=0$, and further $\mathbf{V}^0$ proves that A_{k+1} is provably invertible (as before, by using using 4.4 part 4 and the fact that $\mathbf{V}^0$ proves that $\mathsf{Det}_{circ^{-1}}(\cdot)$ of this matrix is provably good for $z=0$). □

Proof of Lemma 7.6. Part 1: By Proposition 4.5 we have a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof of $\mathsf{Det}_{circ^{-1}}(I_n+zU) = (1+zu_{11})\cdots(1+zu_{nn})$, that $\mathbf{V}^0$ proves is provably good for $z=0$. By Lemma 7.3 part 2 we thus conclude the proof.

Part 2 follows from Lemma 7.7 as follows. Since X^{-1} (for X a symbolic matrix in x_{ij} variables) is Σ_1^B-definable in $\mathbf{V}^0$ which $\mathbf{V}^0$ proves is provably invertible, there is a Σ_1^B-definable $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in $\mathbf{V}^0$ of the following equation:

$$\mathsf{Det}_{circ^{-1}}(zI_n + X^{-1}) = z^n + Q_{n-1}z^{n-1} + \cdots + Q_0, \tag{43}$$

where the Q_i's are circuits with division that do not contain the variable z and the proof is provably good for $z=0$.

By Proposition 4.5 part 2 the following equation has a Σ_1^B-definable proof in $\mathbf{V}^0$, that $\mathbf{V}^0$ proves to be provably good for $z=0$:

$$\mathsf{Det}_{circ^{-1}}(I_n+zX) = \mathsf{Det}_{circ^{-1}}(zI_n+X^{-1})\cdot\mathsf{Det}_{circ^{-1}}(X).$$

From equation (43) we get a Σ_1^B-definable proof in $\mathbf{V}^0$ of

$$\mathsf{Det}_{circ^{-1}}(I_n+zX) = z^n\mathsf{Det}_{circ^{-1}}(X) + z^{n-1}Q'_{n-1} + \cdots + Q'_0,$$

where $Q'_{n-1},\ldots,Q'_0$ do not contain z. $\mathbf{V}^0$ proves the proof is provably good for $z=0$ and so Lemma 7.3 gives a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof in $\mathbf{V}^0$ of the following equation that $\mathbf{V}^0$ proves is provably good for $z=0$

$$\text{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n+zX)) = \text{coeff}_{z^n}(z^n\mathsf{Det}_{circ^{-1}}(X)+z^{n-1}Q'_{n-1}+\cdots+Q'_0).$$

Since $\mathsf{Det}_{Taylor}(X) = \text{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n+zX))$ and by the definition of $\text{coeff}_{z^n}(\cdot)$, $\text{coeff}_{z^n}(z^n\mathsf{Det}_{circ^{-1}}(X)+z^{n-1}Q'_{n-1}+\cdots+Q'_0) = \mathsf{Det}_{circ^{-1}}(X)$, we are done.

This concludes the proof of Lemma 7.6. □

7.5 Reducing the Syntactic-Degree of the Determinant Polynomial

The (division-free) circuit $\mathsf{Det}_{Taylor}(X) := \text{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n+zX))$ has an exponential syntactic-degree as we explain below. However, in order to eliminate division gates in $\mathbb{P}^{-1}_c(\mathbb{Z})$-proofs (see Corollary 8.2) we need to write the determinant as a division-free circuit with a polynomially bounded syntactic-degree (specifically syntactic-degree n) in the equations we prove (interim equations in the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof can use higher syntactic-degrees).

Potentially, this problem can be remedied by eliminating 0 nodes from circuits, and this is how it was dealt with in [HT15]. However, 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, our solution is different: we are going to explicitly construct in $\mathbf{V}^0$ a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof that witnesses the fact that $\mathsf{Det}_{Taylor}(X)$ is equivalent to a circuit with a polynomially-bounded syntactic-degree.

Let $F(X) = F(x_1,,\ldots,x_n)$ 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 $F(zx_1,\ldots,zx_n)$, which we denote by F'. By Definition 7.2, $\text{coeff}_{z^k}(F') = F_0'^{-1}\cdot\text{coeff}_{z^k}\left(\text{Num}(F')\cdot\left(1+G+G^2+\cdots+G^k\right)\right)$ for $G := (1-F_0'^{-1}\cdot\text{Den}(F'))$, and $F'_0 := \text{Den}(F')(0/z)$. The variable z does not occur in the scope of any division gate in $\text{Num}(F')\cdot\left(1+G+G^2+\cdots+G^k\right)$ and z multiplies each of the x_i variables, and the only division gates in $\text{coeff}_{z^k}(F')$ are of the form $F_0'^{-1} = (\text{Den}(F')(0/z))^{-1}$. Thus, by construction of Den we have that $\text{Den}(\text{coeff}_{z^k}(F'))$ is a power of F'_0 (namely, computes $\widehat{F'^p_0}$, for some p).

Recall that the syntactic-degree of a circuit with division is the sum of the syntactic-degrees of its denominator and its numerator. By inspection of the construction of $\text{coeff}_{z^k}(\cdot)$ (Definition 7.2) we observe that every subcircuit in $\text{coeff}_{z^k}(F')$ contributes to the overall syntactic-degree[13] at most k, except for the occurrences of the subcircuit $F_0'^{-1}$ which contribute to the overall syntactic-degree $d(\text{Den}(F'))$ that may be greater than k; in other words, if we substitute 1 for the occurrences of $F_0'^{-1}$ in $\text{coeff}_{z^k}(F')$ we obtain a circuit of syntactic-degree at most k.

According to the discussion above, $\mathsf{Det}_{Taylor}(X) = \text{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n+zX))$ contains occurrences of $\left(\text{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n+zX\right)\right)(0/z)\right)^{-1}$ that are of syntactic-degree greater than n, while all other nodes contribute at most n to the overall syntactic-degree. To remedy this we define $\mathsf{Det}^{\#}_{Taylor}(X)$ as the circuit $\mathsf{Det}_{Taylor}(X)$ in which we replace the subcircuit $\left(\text{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n+zX\right)\right)(0/z)\right)^{-1}$ by the constant 1. Denote by b the circuit $\left(Den\left(\mathsf{Det}_{circ^{-1}}\left(I_n+zX\right)\right)\right)^{-1}$. Note that indeed $b(0/z)$ computes the polynomial 1. Then we define the determinant circuit with syntactic-degree n as follows:

$$\mathsf{Det}^{\#}_{Taylor}(X) := (\mathsf{Det}_{Taylor}(X))(1/(b(0/z)))\,. \tag{44}$$

[13] The syntactic-degree here is with respect to all variables, not only z.

Lemma 7.8. *Let A be an $n \times n$ symbolic matrix X or the product of two $n \times n$ symbolic matrices XY, or a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ triangular $n \times n$ matrix U in the variables x_{ij} for $i, j \in [n]$, with diagonal entries $u_{11}, \ldots, u_{nn}$ such that $\mathbf{V}^0$ proves that u_{ii}^{-1} is provably good, for all $i \in [n]$. Then, there exists a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of*

$$\mathsf{Det}^{\#}_{Taylor}(A) = \mathsf{Det}_{Taylor}(A).$$

Further, $\mathbf{V}^0$ proves that for every division gate $v = w^{-1}$ in the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.

Remark 7.9. *Note that by Lemma 6.4, if $\mathbf{V}^0$ proves that a division gate v is provably good then $\mathbf{V}^0$ proves that there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. Hence, the former is a stronger property than the latter. We show in Section 8 that the latter property suffices for our purposes.*

Proof. We assume that $A = X$. The other cases are similar. Note that by previous constructions the circuits $\mathsf{Det}^{\#}_{Taylor}(X)$ and $\mathsf{Det}_{Taylor}(X)$ are $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$. We first construct in $\mathbf{V}^0$ a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z) = 1$. For this, we initially construct a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of

$$\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z) = \mathsf{Det}_{circ^{-1}}\left(I_n\right). \tag{45}$$

Since $(I_n + zX)(0/z) = I_n$ has a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof in $\mathbf{V}^0$ (replacing in each step of the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof $0 \cdot u$ by 0 and $0 + u$ by u, for u any subcircuit[14]), $\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z) = \left(\mathsf{Det}_{circ^{-1}}\left(\left(I_n + zX\right)(0/z)\right)\right) = \mathsf{Det}_{circ^{-1}}\left(I_n\right)$ has also such a proof in $\mathbf{V}^0$. Also, $\mathbf{V}^0$ easily proves that the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof is provably good since $\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z)$ and $\mathsf{Det}_{circ^{-1}}\left(I_n\right)$ are provably good circuits.

By Proposition 4.5 the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of

$$\mathsf{Det}_{circ^{-1}}\left(I_n\right) = 1 \tag{46}$$

is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ and $\mathbf{V}^0$ proves the proof is provably good. Using (45), (46) and Corollary 6.5, $\mathbf{V}^0$ proves the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of

$$\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z) = 1 \tag{47}$$

is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ and further, for every division gate $v = t^{-1}$ in this proof $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.

The $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of $\mathsf{Det}^{\#}_{Taylor}(X) = \mathsf{Det}_{Taylor}(X)$ is then constructed as follows. First we derive (47), and then using (47) we substitute each occurrence of the subcircuit $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z)$ in $\mathsf{Det}_{Taylor}(X)$ by the constant 1, to get the desired equality (using also the Div axiom $1 \cdot 1^{-1} = 1$). Thus, in the first part of the proof, for every division gate u^{-1} $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$. In the second part, when we substitute by 1 subcircuits, the proof uses still the same division gates that occur in the first part (because the only division gates in $\mathsf{Det}_{Taylor}(X)$ are $\mathrm{Den}\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z)$). □

From Proposition 4.5, Lemma 7.6, Lemma 7.8 and Remark 7.9 we get a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of the determinant identities where the identities proved have no division gates and are of low syntactic-degree as follows:

[14] 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 $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof that witnesses such a gradual procedure for discarding 0's from the *specific* circuit $\left(\mathsf{Det}_{circ^{-1}}\left(I_n + zX\right)\right)(0/z)$.

Corollary 7.10. *Let X, Y be $n \times n$ symbolic matrices and U be a Σ^B_1-definable in $\mathbf{V}^0$ triangular $n \times n$ matrix in the variables x_{ij} for $i, j \in [n]$, such that $\mathbf{V}^0$ proves that u_{ii}^{-1} are provably good for all $i \in [n]$. Then, the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of the determinant identities*

$$\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y) = \mathsf{Det}^{\#}_{Taylor}(XY) \quad \textit{and} \tag{48}$$

$$\mathsf{Det}^{\#}_{Taylor}(U) = u_{11} \cdots u_{nn} \tag{49}$$

are Σ^B_1-definable in $\mathbf{V}^0$, and $\mathbf{V}^0$ proves that for every division gate $v = t^{-1}$ in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.

8 Eliminating Division Gates

8.1 Overview

In this section we apply our previous results to eliminate division gates from the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs of the determinant identities we constructed in $\mathbf{V}^0$ in Corollary 7.10. Recall the concept of provably good nodes, the identity matrices assignment ρ, the power series Inv_k from Section 6, and division normalization in Section 6.3.

To eliminate division gates we take the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof in Corollary 7.10, normalize division so that every division gate does not appear in the scope of another division gate. We then apply a linear shift on the variables by adding to each variable the value it gets under the identity matrices assignment ρ, so that the all zero assignment would not nullify any division gate (provably in $\mathbf{V}^0$), and then substitute every division gate F^{-1} by a truncated power series $\mathrm{Inv}_{2n}(F)$. Note that $F^{(0)}$ is the value of F under the zero assignment to its variables, and since we shifted the variables by ρ, $F^{(0)}$ evaluates to 1 by our assumption that all division nodes of the original $\mathbb{P}_c^{-1}(\mathbb{Z})$-proofs are provably good. This way we get a division free proof-sequence that is correct, except that the Div axiom $F \cdot F^{-1} = 1$ is replaced by $F \cdot \mathrm{Inv}_{2n}(F) = 1$ which is neither an axiom nor a true identity. We shall deal with this in the next section (Section 9) when eliminating high syntactic-degrees.

8.2 Eliminating Division

In the first phase eliminating division in the proofs will result in proofs that are correct, but only up to some prescribed degree, as defined in what follows.

Definition 8.1 (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 with no division 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* Div*, 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* Div*.*

The witness for syntactic correctness of a correct up to degree k $\mathbb{P}_c(\mathbb{Z})$-proof is similar to that in Section 3.2. 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^B_0-formula $\psi(Z, k)$ that holds iff 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 3.2). The formula $\psi(Z, k)$ only needs to verify that in the division axiom Div 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 $\mathrm{Inv}_k(F)$.

The following will be used in Corollary 9.2:

Corollary 8.2 (Homogenizing correct up to degree d $\mathbb{P}_c(\mathbb{Z})$-proofs)**.** *Let F, G be two $\boldsymbol{\Sigma}^B_1$-definable algebraic circuits over $\mathbb{Z}$ with syntactic-degree d, and assume that there is a correct up to degree d $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ that is $\boldsymbol{\Sigma}^B_1$-definable in $\mathbf{V}^0$. Suppose that for every occurrence of $\mathrm{Inv}_d(H)$ in this $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$, for some circuit H, $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $H^{(0)} = 1$. Then, for every $k = 0, \ldots, d$, the following proof is $\boldsymbol{\Sigma}^B_1$-definable in $\mathbf{V}^0$: the $\mathbb{P}_c(\mathbb{Z})$-proof of $F^{(k)} = G^{(k)}$ in which every circuit is a sum of syntactic homogeneous circuits inside which every node u in the proof appears with its syntactic-degree upper bound, and $d_{\mathrm{ub}}(u) \leq k$.*

Proof. This stems from direct inspection of Theorem 5.4 and its proof. The simulation of all rules and axioms is the same as in Theorem 5.4 except for the simulation of the Div axiom. In the case of Div inspecting of the proof of Theorem 5.4 shows that we do not use the homogeneous components of degree greater than d. Thus, we only need to make sure that $\mathbf{V}^0$ proves there exist $\mathbb{P}_c(\mathbb{Z})$-proofs of $(H \cdot \mathrm{Inv}_d(H))^{(i)} = 0$, for every $1 \leq i \leq d$, and of $(H \cdot \mathrm{Inv}_d(H))^{(0)} = 1$, for every proof-line $H \cdot \mathrm{Inv}_d(H) = 1$ appearing in the $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$, and that these proofs also contain the syntactic-degree upper bound for every node in every circuit. But this stems from Lemma 6.1 and the assumption that $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $H^{(0)} = 1$. □

Let r be one of the variables in X, Y, that is $r \in \{x_{ij}, y_{ij} : i, j \in [n]\}$ and let w_r be a new variable not in X, Y. The mapping

$$\sigma : r \mapsto (\rho(r) - w_r)$$

linearly shifts the variables X, Y by the identity matrices assignment ρ (and also replaces the original variables X, Y by the w_r variables for the sake of clarity).

Recall that substitutions in $\mathbb{P}^{-1}_c(\mathbb{Z})$-proofs are $\boldsymbol{\Sigma}^B_1$-definable in $\mathbf{V}^0$ by Proposition 3.2. Hence, if a $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof π of $F = G$ is $\boldsymbol{\Sigma}^B_1$-definable in $\mathbf{V}^0$ then also the $\mathbb{P}^{-1}_c(\mathbb{Z})$-proof of $F \restriction \sigma = G \restriction \sigma$ is. Recall also that $F \restriction \sigma$ stands for the circuit F represented as a graph with the input variables replaced based on the assignment σ. Further, notice that $\mathrm{Den}(F \restriction \sigma)^{(0)}$ computes precisely the value of $\mathrm{Den}(F)$ under the identity matrices assignment, that is $\widehat{\mathrm{Den}(F \restriction \sigma)^{(0)}} = \widehat{\mathrm{Den}(F \restriction \rho)} = \mathrm{Den}(F \restriction \rho)^{(0)}$. We have the following:

Lemma 8.3. *Let F be a $\boldsymbol{\Sigma}^B_1$-definable circuit in $\mathbf{V}^0$ in the X, Y variables and assume that $\mathbf{V}^0$ proves that there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $F \restriction \rho = 1$, then there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $(F \restriction \sigma)^{(0)} = 1$.*

Proof. By the definitions of σ, ρ and basic rearrangements in $\mathbb{P}_c(\mathbb{Z})$ we have a $\mathbb{P}_c(\mathbb{Z})$-proof of $F \restriction \rho = (F \restriction \sigma) \restriction \overline{0}$ (where $C \restriction \overline{0}$ means substituting 0 for all variables in a given circuit C). By the construction of syntactic-homogeneous circuits (Section 5) the underlying graph of a (division free) circuit $C \restriction \overline{0}$ and the structure of $C^{(0)}$ are *identical*: constant gates stay the same; variables turn 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 new plus or product gates are added to the circuit). Hence, $(F \restriction \sigma) \restriction \overline{0} = (F \restriction \sigma)^{(0)}$ is $\mathbb{P}_c(\mathbb{Z})$-provable. □

We are now ready to construct the (division free) $\mathbb{P}_c(\mathbb{Z})$-proof of the determinant identities, with some restrictions (namely, it is correct only up to a prescribed degree and its variables are linearly shifted).

Lemma 8.4. *Let X, Y be $n \times n$ symbolic matrices and U be a Σ_1^B-definable in $\mathbf{V}^0$ triangular $n \times n$ matrix in the variables x_{ij} for $i, j \in [n]$ with* no *division gates. Then, the correct up to degree $2n$ $\mathbb{P}_c(\mathbb{Z})$-proofs of the shifted determinant identities*

$$(\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y)) \restriction \sigma = \mathsf{Det}^{\#}_{Taylor}(XY) \restriction \sigma \quad \textit{and} \tag{50}$$

$$\mathsf{Det}^{\#}_{Taylor}(U) \restriction \sigma = (u_{11} \cdots u_{nn}) \restriction \sigma \tag{51}$$

are Σ_1^B-definable in $\mathbf{V}^0$. Moreover, for every occurrence of $\mathrm{Inv}_{2n}(H)$ in the proofs, for some circuit H, $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $H^{(0)} = 1$.

Proof. By Corollary 7.10 $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π of the determinant identities where the determinant is written as $\mathsf{Det}^{\#}_{Taylor}$ (intermediate proof-lines may represent the determinant differently, e.g., by Det_{Taylor} or $\mathsf{Det}_{circ^{-1}}$), and further $\mathbf{V}^0$ proves that for every division gate $v = t^{-1}$ in the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathrm{Den}(v \restriction \rho) = 1$.

Applying Corollary 6.5 on π we get that $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof π' of the determinant identities written as $\mathsf{Det}^{\#}_{Taylor}$ in which each division gate H^{-1}, for some circuit H, does not occur in the scope of another division gate, and furthermore $\mathbf{V}^0$ proves that there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $H \restriction \rho = 1$. Therefore, by Lemma 8.3 $\mathbf{V}^0$ proves that for every division gate H^{-1} in the proof there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $(H \restriction \sigma)^{(0)} = 1$.

Now apply the linear shift $\sigma : r \mapsto (\rho(r) - w_r)$ to the variables in π' as described above to get the $\mathbb{P}_c^{-1}(\mathbb{Z})$-proof $\pi' \restriction \sigma$. Since no division gate occurs in the scope of another division gate we can substitute every division $H^{-1} \restriction \sigma$ in in $\pi' \restriction \sigma$ by the truncated power series $\mathrm{Inv}_{2n}(H \restriction \sigma)$ (we use $2n$ because this is the degree of the determinant identities). Let π_0 be the corresponding division-free proof-sequence obtained from $\pi' \restriction \sigma$ by replacing every division gate in such a way. Note that by the above for every occurrence of $\mathrm{Inv}_{2n}(H \restriction \sigma)$ in π_0, $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $(H \restriction \sigma)^{(0)} = 1$.

(To clarify, we note that by definition we have $\widehat{H \restriction \sigma} \cdot \widehat{\mathrm{Inv}_{2n}(H \restriction \sigma)} = 1 + [\text{terms of degree} > 2n]$. Hence, by itself π_0 is not a legal $\mathbb{P}_c(\mathbb{Z})$-proof, rather a correct up to degree $2n$ $\mathbb{P}_c(\mathbb{Z})$-proof, since the axiom of division $F \cdot F^{-1} = 1$ in $\mathbb{P}_c^{-1}(\mathbb{Z})$ does not translate into an axiom in $\mathbb{P}_c(\mathbb{Z})$, rather it translates into $H \restriction \sigma \cdot \mathrm{Inv}_{2n}(H \restriction \sigma) = 1$, which is neither a legal axiom, nor a true identity.) □

9 Eliminating High Degrees From the Proofs

Here we show how to eliminate within $\mathbf{V}^0$ all nodes of syntactic degrees that exceed $2n$ from the correct up to degree $2n$ proofs constructed in Lemma 8.4. This process will also result in all proof-lines appearing as sums of homogeneous components in which every node appears with its syntactic-degree upper bound (which is useful for applying the balancing algorithm in Section 10).

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 2.14) and that $C^{(i)}$ is a syntactic-homogeneous circuit computing the degree i homogeneous part of C (Section 2.7). For a node v in C, C_v denotes the circuit rooted at v.

Claim 9.1. *Let $F(x_1, , \ldots, x_m)$ be a Σ_1^B-definable in $\mathbf{V}^0$ circuit without division, in the displayed input variables. Assume that every input variable is now multiplied by a new variable z to get $F(zx_1, \ldots, zx_m)$, which we denote by F_z. Then, there exists a Σ_1^B-definable in $\mathbf{V}^0$ $\mathbb{P}_c(\mathbb{Z})$-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 Section 7.3.2. 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 $j \in [n]$. ■Claim

Recall the definitions of Det_{Taylor} (40) and $\mathsf{Det}^{\#}_{Taylor}$ (44) and observe that $\mathsf{Det}^{\#}_{Taylor}(X)$ is a division free circuit such that every X variable in it is a product of z. Therefore, by Claim 9.1 *we can assume from now on that* $\mathsf{Det}^{\#}_{Taylor}(X)$ *is written as a syntactic homogeneous circuit of degree* n*, and similar for* $\mathsf{Det}^{\#}_{Taylor}(Y)$. Accordingly, by the homogenization algorithm in Section 5:

$$(\mathsf{Det}^{\#}_{Taylor}(X))^{(i)} = 0, \text{ for all } i < n, \tag{52}$$

$$(\mathsf{Det}^{\#}_{Taylor}(X))^{(n)} = \mathsf{Det}^{\#}_{Taylor}(X)\,. \tag{53}$$

(because when given as input to the homogenization algorithm it will output the input circuit, or a single homogeneous component as in part 3 of the output). And the same holds for $\mathsf{Det}^{\#}_{Taylor}(Y)$.

Corollary 9.2. *Let* X, Y *be* $n \times n$ *symbolic matrices and* $U = \{u_{ij}\}_{i,j\in[n]}$ *be a* $\mathbf{\Sigma}^B_1$*-definable in* $\mathbf{V}^0$ *triangular* $n \times n$ *matrix in the variables* x_{ij} *for* $i, j \in [n]$ *with* no *division gates. Then, the* $\mathbb{P}_c(\mathbb{Z})$*-proofs of the following determinant identities*

$$\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y) = \mathsf{Det}^{\#}_{Taylor}(XY) \quad \textit{and} \tag{54}$$

$$\mathsf{Det}^{\#}_{Taylor}(U) = u_{11} \cdots u_{nn} \tag{55}$$

are $\mathbf{\Sigma}^B_1$*-definable in* $\mathbf{V}^0$*. Moreover, in these proofs every circuit is a sum of syntactic homogeneous circuits in which every node* v *appears with its syntactic-degree upper bound* $d_{\mathrm{ub}}(v) \leq 2n$*.*

Proof. By Lemma 8.4 there exists a correct up to degree $2n$ $\mathbb{P}_c(\mathbb{Z})$ proof-sequence π of the shifted determinant identities (50) and (51), such that for every occurrence of $\mathrm{Inv}_{2n}(H)$ in the proofs, for some circuit H, $\mathbf{V}^0$ proves there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of $H^{(0)} = 1$. Hence, by Corollary 8.2 we have a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$\left(\mathsf{Det}^{\#}_{Taylor}(X) \cdot \mathsf{Det}^{\#}_{Taylor}(Y)\right)^{(k)} \restriction \sigma = (\mathsf{Det}^{\#}_{Taylor}(XY))^{(k)} \restriction \sigma, \tag{56}$$

for every $k = 0, \ldots, 2n$, and similarly for (51), wherein every circuit is a sum of syntactic homogeneous circuits inside which every node appears with its specified syntactic-degree upper bound of at most $2n$.

We now shift back the variables and substitute $\rho(r) - w_r$, for every r in X, Y, by the original variable r. We get $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ $\mathbb{P}_c(\mathbb{Z})$-proofs of

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

$$\mathsf{Det}^{\#}_{Taylor}(U)^{(k)} = (u_{11} \cdots u_{nn})^{(k)}$$

for every $k = 0, \ldots, 2n$, and where in every proof the syntactic-degree upper bound of at most k is specified on all nodes.

Finally, we can conclude the corollary reasoning as follows.
By Lemma 5.6 we have a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$\sum_{i=0}^{2n}\left(\mathsf{Det}^{\#}_{Taylor}(X)\cdot\mathsf{Det}^{\#}_{Taylor}(Y)\right)^{(i)}=\sum_{i=0}^{2n}\sum_{\substack{l+j=i\\0\leq l\leq 2n,0\leq j\leq 2n}}\mathsf{Det}^{\#}_{Taylor}(X)^{(l)}\cdot\mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}\,. \tag{57}$$

This equals

$$\sum_{i=0}^{2n}\sum_{\substack{l+j=i\\0\leq l\leq n,0\leq j\leq n}}\mathsf{Det}^{\#}_{Taylor}(X)^{(l)}\cdot\mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}+\sum_{i=0}^{2n}\sum_{\substack{l+j=i\\n<l\leq 2n,n<j\leq 2n}}\mathsf{Det}^{\#}_{Taylor}(X)^{(l)}\cdot\mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}\,, \tag{58}$$

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 written as 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 (58). We proceed with

$$\begin{aligned}\sum_{i=0}^{2n}\sum_{\substack{l+j=i\\0\leq l\leq n,0\leq j\leq n}}&\mathsf{Det}^{\#}_{Taylor}(X)^{(l)}\cdot\mathsf{Det}^{\#}_{Taylor}(Y)^{(j)}\\&=\sum_{i=0}^{n}\mathsf{Det}^{\#}_{Taylor}(X)^{(i)}\cdot\sum_{i=0}^{n}\mathsf{Det}^{\#}_{Taylor}(Y)^{(i)} &(59)\\&=\mathsf{Det}^{\#}_{Taylor}(X)\cdot\mathsf{Det}^{\#}_{Taylor}(Y), &(60)\end{aligned}$$

where the first equality is by rearrangement and the second equality is by (52) and (53). By summing (56) for all $k=0,\ldots,2n$ and using similar reasoning for $\mathsf{Det}^{\#}_{Taylor}(XY)$, we conclude that there exists a $\mathbb{P}_c(\mathbb{Z})$-proof of

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

Note that by our construction, in the above $\mathbb{P}_c(\mathbb{Z})$-proof every proof-line is a sum of syntactic homogeneous circuits. Observe that this also holds for (59), (60), because $\sum_{i=0}^{n}\mathsf{Det}^{\#}_{Taylor}(X)^{(i)}=\mathsf{Det}^{\#}_{Taylor}(X)^{(n)}=\mathsf{Det}^{\#}_{Taylor}(X)$ is a syntactic homogeneous circuit by (52) and (53) (ignoring zero summands), and similarly for $\sum_{i=0}^{n}\mathsf{Det}^{\#}_{Taylor}(Y)^{(i)}$, and a product of syntactic homogeneous circuits is a syntactic homogeneous circuit in itself. Furthermore, note that every node in the proof appears with its syntactic-degree upper bound d_{ub} by the use of Corollary 8.2. The proof of (55) is similar. □

10 Balancing Algebraic Circuits and Proofs in the Theory

10.1 Overview

In this section we take the $\mathbb{P}_c(\mathbb{Z})$-proofs we constructed in the previous section, and turn them into balanced proofs, namely proofs in which each circuit is of $O(\log^2 n)$-depth. This is the first step in

which we will use the full power of $\mathbf{VNC}^2$ (contrasted with the constructions in $\mathbf{V}^0$ in previous sections).

We first $\mathbf{\Sigma}^B_1$-define in $\mathbf{VNC}^2$ a circuit balancing algorithm: this is a string function that receives a sum of syntactic homogeneous algebraic circuits C (without division) 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 $O(\log s \cdot \log d)$ and size $\text{poly}(s,d)$. Our $\mathbf{FNC}^2$-algorithm provides an $\mathbf{FAC}^0$-implementation of most parts of the classic Valiant *et al.* [VSBR83] algorithm, combining it with some ideas from the Miller *et al.* [MRK88] algorithm, while also using matrix powering, which then entails working in $\mathbf{FNC}^2$.

More generally, we show a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{VNC}^2$ that receives a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ with syntactic-degree d (and in which every proof-line is a sum of syntactic homogeneous circuits), and outputs a $\mathbb{P}_c(\mathbb{Z})$-proof of $[F] = [G]$ in which every circuit is of depth $O(\log s \cdot \log)$ and the size of the proof is $\text{poly}(s,d)$.

Applying this function to the $\mathbb{P}_c(\mathbb{Z})$-proof constructed in the previous section, demonstrates that the depth $O(\log^2 n)$ $\mathbb{P}_c(\mathbb{F})$-proof of the determinant identities where the determinant in the identities proved is written as a balanced division free circuit of syntactic-degree n denoted $\text{Det}_{balanced}$, is $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{VNC}^2$.

10.2 Background Concepts for the Balancing Algorithm

We start by providing some required concepts.

Definition 10.1 (Balanced circuit)**.** *Let D be an algebraic circuit of size s and syntactic-degree d. Then, we say that D is* balanced *if the depth of D is $O(\log s \log d)$.*[15] *Specifically, if d is polynomial in s, then the depth of D is $O(\log^2 s)$.*

Our balancing algorithm 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 original [VSBR83] algorithm we refer the reader to [RY08] (cf. [HT15]), though our treatment is self contained).

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 $\partial w f_v$ (defined below) whenever $0 \leq d_{\text{ub}}(v) - d_{\text{ub}}(w) \leq 1$. However, we cannot directly compute the integer coefficients in these linear polynomials because their (sub-)circuits are not balanced (and the evaluation of circuits of depth $\omega(\log^2 n)$ is apparently beyond $\mathbf{VNC}^2$).

[15] $O(\log s \log d + \log^2 d)$ is the standard depth of "balanced" circuits [VSBR83], when the original circuit C of size s and degree d is not necessarily syntactic homogeneous. The term $\log^2 d$ in the depth is incurred due the conversion of C to a syntactic homogeneous circuit beforehand. Since we will input to our balancing algorithm circuits that are already (sums of) syntactic homogeneous circuits, the depth we get is $O(\log s \log d)$.

We show how to compute the linear polynomials we need in Lemmas 10.4 and 10.6. For the purpose of 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 (hence both variables and scalars are now treated as syntactic-degree 1 circuits).

Denote by $d^+_{\mathrm{ub}}(\cdot)$ the syntactic-degree upper bound defined similar to $d_{\mathrm{ub}}(\cdot)$, except that scalar nodes are associated with syntactic-degree upper bound 1 (instead of 0) in the algorithm for homogenizing circuits shown in Section 5. Note that any circuit F_v rooted by the node v such that $d^+_{\mathrm{ub}}(v) \leq 1$ cannot contain product nodes (as this would make $d^+_{\mathrm{ub}}(v) > 1$ by definition). In Lemma 10.4 we show how to evaluate 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 $d_{\mathrm{ub}}(u)$ but exponential large $d^+_{\mathrm{ub}}(u)$, for example in case we have an iterated squaring such as $(((2)^2)^2 \ldots^2) = 2^{2^n}$. We deal with this problem in Section 10.3.1.

Definition 10.2 (Partial derivative polynomial $\partial_w f_v$)**.** *Let w, v be two nodes in F. We define the* partial derivative of F_v with respect to w*, denoted $\partial_w f_v$, as the following* polynomial*:*

$$\partial_w f_v := \begin{cases} 0, & \text{if } w \notin F_v, \\ 1, & \text{if } w = v, \text{ and otherwise:} \\ \partial_w f_{v_1} + \partial_w f_{v_2}, & v = v_1 + v_2; \\ (\partial_w f_{v_1}) \cdot f_{v_2}, & \text{if either } v = v_1 \cdot v_2 \text{ and } d^+_{\mathrm{ub}}(v_1) \geq d^+_{\mathrm{ub}}(v_2) \\ & \quad \text{or } v = v_2 \cdot v_1 \text{ and } d^+_{\mathrm{ub}}(v_1) > d^+_{\mathrm{ub}}(v_2). \end{cases} \tag{61}$$

The idea behind this definition is the following: let w, v be two nodes in F and assume that $d^+_{\mathrm{ub}}(w) > \frac{d^+_{\mathrm{ub}}(v)}{2}$ (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 $g(z, x_1, \ldots, x_n)$ which is *linear* in z, that is $g(z, x_1, \ldots, x_n) = h_0 \cdot z + h_1$ for some polynomials h_0, h_1 in the $x_1, ..., x_n$ variables, and $\partial_w f_v = h_0$. Namely, $\partial_w f_v$ in our case is the standard partial derivative $\partial_z g$.

Proposition 10.3. *Let w, v be two nodes in a syntactically homogeneous circuit F such that $d^+_{\mathrm{ub}}(w) > \frac{d^+_{\mathrm{ub}}(v)}{2}$. Then the polynomial $\partial_w f_v$ has degree at most $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w)$.*

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 $d^+_{\mathrm{ub}}(\partial_w f_v) = 0 = d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w)$ 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 $d^+_{\mathrm{ub}}(v_1) - d^+_{\mathrm{ub}}(w)$ and $d^+_{\mathrm{ub}}(v_2) - d^+_{\mathrm{ub}}(w)$, 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 con-

struction 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 upper bound (plus), namely, d^+_{ub} 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 our $\mathbf{FAC}^0$ algorithm for homogenization in Section 5; but notice 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 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. Specifically, we use this 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. 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 $O(\log d \cdot \log s)$ 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 as well as 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; matrix powering is known to be $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ and hence is computable in $\mathbf{FNC}^2$ [CF12]). Each of the other stages 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 $\mathbf{FNC}^2$.

Our algorithm is different from that of Valiant *et al.* [VSBR83], since we use a preprocessing step and in the first stage of the algorithm where $i = 0$ (which corresponds 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. Another difference is that while Valiant *et al.* [VSBR83] use the notion of degrees of a node, and Hrubeš and Tzameret [HT15] use syntactic-degrees of nodes, our constructions use the $\mathbf{FAC}^0$-computable relaxed notion of *syntactic-degree upper bound* $d_{\mathrm{ub}}(v)$ *of a node* v introduced in Section 5 and its variant $d^+_{\mathrm{ub}}(v)$.

10.3 Preliminaries for the Balancing Algorithm

Lemma 10.4. *There is a $\mathbf{\Sigma}^B_1$-definable string function in $\mathbf{VNC}^2$ that given a (division free) algebraic circuit F of size s with no product gates outputs a depth $O(\log n)$ algebraic circuit computing $\widehat{F}$ of size* poly(s)*, for n the number of variables.*[16]

[16]Notice that if the input algebraic circuit F was a formula instead of a circuit, it would have been trivial to output the balanced formula computing $\widehat{F}$: simply build a balanced binary tree whose leaves are all the variables occurring in

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 10.5. *There is a $\mathbf{\Sigma}^B_1$-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 10.4 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 $A^s_C(w, v) \neq 0$, where matrix powering is definable in $\mathbf{VNC}^2$ as mentioned above. □

Lemma 10.6. *There is a $\mathbf{\Sigma}^B_1$-definable string 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 $0 \leq d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$, 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 $0 \leq d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$, either $d^+_{\text{ub}}(v) \leq 1$ or $d^+_{\text{ub}}(v) \geq 2$ and $d^+_{\text{ub}}(w) > d^+_{\text{ub}}(v)/2$. Hence, by Proposition 10.3, 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 $d^+_{\text{ub}}(r) \geq 1$. Hence, for every product gate $u = t \cdot s$ we have $d^+_{\text{ub}}(u) = d^+_{\text{ub}}(t) + d^+_{\text{ub}}(s) \geq 2$.

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 $d^+_{\text{ub}}(v) \geq d^+_{\text{ub}}(u) \geq 2d^+_{\text{ub}}(w) \geq 2$ (the last inequality is by the fact above), and hence $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \geq d^+_{\text{ub}}(w) \geq 2$ 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.

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.

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, $d^+_{\text{ub}}(v) \geq d^+_{\text{ub}}(w) + 2$ in contrast to the assumption that $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$.

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 $d^+_{\text{ub}}(s) \geq 2$ and so $d^+_{\text{ub}}(v) \geq d^+_{\text{ub}}(u) \geq d^+_{\text{ub}}(w) + d^+_{\text{ub}}(s) \geq d^+_{\text{ub}}(w) + 2$ in contrast to the assumption $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 1$.

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 10.4 we can thus construct a $O(\log n)$ depth circuit, for n the number of variables, computing the sum $\sum_{u \in U} \widehat{F}_{s_u}$. □

10.3.1 Taking Care of Nodes with High d^+_{ub} Values

To be able to carry out the balancing construction of circuits and proofs in the theory we need to make sure that in all the circuits we consider nodes have polynomially bounded d^+_{ub} values (and not only polynomially bounded d_{ub} values). We first claim that all results about d_{ub} that have been proved up to this point hold also for d^+_{ub}. After which we show that the determinant identities proved can be assumed to be of polynomial (in fact, $2n$) syntactic-degree with respect to d^+_{ub}, and then by the same reasoning as before we can assume that intermediate proof-lines have low d^+_{ub} values as well, using high syntactic-degree elimination in proofs, leading to same statement of Corollary 9.2 in which d^+_{ub} replaces d_{ub}.

Fact 10.7. *All the statements about proof-construction and transformations that we presented up to this point for d_{ub} also hold true for d^+_{ub}.*

Fact 10.7 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 10.8 (Existence of division free circuit for the determinant with low d^+_{ub} values)**.** *There exists a circuit denoted $\mathsf{Det}^\star_{Taylor}(X)$ computing the determinant of X with all nodes having syntactic-degree d^+_{ub} at most n, such that the $\mathbb{P}_c(\mathbb{Z})$-proof of $\mathsf{Det}^\#_{Taylor}(X) = \mathsf{Det}^\star_{Taylor}(X)$ is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$.*

Proof. We will witness our desired circuit $\mathsf{Det}^\star_{Taylor}(X)$. We start by observing the way scalars can contribute to syntactic-degrees d^+_{ub} in $\mathsf{Det}^\#_{Taylor}(X)$. Recall 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 (14), and these scalars occur as $1 \cdot h$ or $0 \cdot h$ for some h that contains variables. 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 $i = j$. Hence, the only scalars in $\mathsf{Det}_{circ^{-1}}(I_n + zX)$ are still 0-1, either from the replacement $0+zx_{ij}$ or by $1+zx_{ij}$ or from I_{n-1} in (14) as before. Recall $\mathsf{Det}_{Taylor}(X) = \mathrm{coeff}_{z^n}(\mathsf{Det}_{circ^{-1}}(I_n + zX))$ and $\mathsf{Det}^{\#}_{Taylor}(X) = (\mathsf{Det}_{Taylor}(X))(1/(b(0/z)))$, where $b = \mathrm{Den}\,(\mathsf{Det}_{circ^{-1}}\,(I_n + zX))$ (see definition (44)). Here, the Denand Numconstructions can add 1's to the circuit, and coeff_{z^n} adds 1's and 0's to the leaves of the circuit. We thus have that still the only scalars in $\mathsf{Det}^{\#}_{Taylor}(X)$ are 0-1. Recall that by Section 7.5 every node u in $\mathsf{Det}^{\#}_{Taylor}(X)$ has $d_{\mathrm{ub}}(u) \leq n$. By inspection of the circuit all 1's and 0's that cause d^{+}_{ub} to become more than n are of the form $1 \cdot (\cdots(1 \cdots) \cdots)$ (and similarly for 0). We need to make sure that all nodes u in $\mathsf{Det}^{\#}_{Taylor}(X)$ have $d^{+}_{\mathrm{ub}}(u) \leq n$.

For this purpose we construct a $\mathbb{P}^{-1}_{c}(\mathbb{Z})$-proof that eliminates 1 from products with 1 in $\mathsf{Det}^{\#}_{Taylor}(X)$ using repeatedly the axiom $1 \cdot 1 = 1$ (and similarly $0 \cdot F = 0$). We can do this because the construction of the circuit $\mathsf{Det}^{\#}_{Taylor}(X)$ is a $\mathbf{\Sigma}^{B}_{1}$-definable function in $\mathbf{V}^0$ as we have demonstrated, which means that there is a $\mathbf{\Sigma}^{B}_{0}$-formula (in the language $\mathcal{L}_A$) that defines the construction of $\mathsf{Det}^{\#}_{Taylor}(X)$ (given the parameter n for X an $n \times n$ matrix). We thus can assume that there are pointers to the occurrences of 1's that we want to discard using $1 \cdot 1 = 1$ (and similarly for 0's). □

Using Lemma 10.8, homogenization of $\mathbb{P}_c(\mathbb{Z})$-proofs (Lemma 5.4) and previous constructions in $\mathbf{V}^0$ of the determinant identities leading up to the $\mathbb{P}_c(\mathbb{Z})$-proofs in Corollary 9.2, and Fact 10.7 showing that we can assume that nodes in $\mathbb{P}_c(\mathbb{Z})$-proofs appear with their syntactic-degree upper bound d^{+}_{ub}, we arrive at:

Corollary 10.9. *Let X, Y be $n \times n$ symbolic matrices and U be a $\mathbf{\Sigma}^{B}_{1}$-definable in $\mathbf{V}^0$ triangular $n \times n$ matrix in the variables x_{ij} for $i, j \in [n]$ with no division gates. Then, there exists a division free circuit computing the determinant of X denoted $\mathsf{Det}^{\star}_{Taylor}(X)$ with all nodes u having $d^{+}_{\mathrm{ub}}(u) \leq n$, such that the $\mathbb{P}_c(\mathbb{Z})$-proofs of the following determinant identities*

$$\mathsf{Det}^{\star}_{Taylor}(X) \cdot \mathsf{Det}^{\star}_{Taylor}(Y) = \mathsf{Det}^{\star}_{Taylor}(XY) \quad \textit{and} \tag{62}$$

$$\mathsf{Det}^{\star}_{Taylor}(U) = u_{11} \cdots u_{nn} \tag{63}$$

are $\mathbf{\Sigma}^{B}_{1}$-definable in $\mathbf{V}^0$. Moreover, in these proofs every circuit is a sum of syntactic homogeneous circuits in which every node u appears with its syntactic-degree upper bound d^{+}_{ub}, and $d^{+}_{\mathrm{ub}}(u) \leq 2n$.

10.4 Formal Description of the Balancing Algorithm

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

$$\mathcal{B}_m(G) := \big\{t \in G \;:\; t = t_1 \cdot t_2, \text{ where } d^{+}_{\mathrm{ub}}(t) > m \text{ and } d^{+}_{\mathrm{ub}}(t_1), d^{+}_{\mathrm{ub}}(t_2) \leq m\big\}. \tag{64}$$

Notice that $\mathcal{B}_m(G)$ is a $\mathbf{\Sigma}^{B}_{0}$-definable relation in $\mathbf{V}^0$ assuming $d^{+}_{\mathrm{ub}}(v)$ is provided for every node v in G.

Note: In the construction of the balanced circuit of F, given nodes $v, w \in F$, $[F_v]$ 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: An algebraic circuit F of size s written as a sum of one or more syntactic-homogeneous circuits over the variables $x_1, \ldots, x_n$, in which every node u appears with its syntactic-degree upper bound $d^+_{\mathrm{ub}}(u)$.

Output: An algebraic circuit denoted $[F]$ of fan-in two computing the polynomial $\widehat{F}$, such that $\mathrm{depth}([F]) = O(\log s \cdot \log d)$ and the size of $[F]$ is $\mathrm{poly}(s, d)$, where $d = d^+_{\mathrm{ub}}(F)$.

Algorithm: Assume that $F = \sum_j F_j$, where each F_j is a syntactic-homogeneous circuit of syntactic-degree upper bound $d^+_{\mathrm{ub}}(F_j) = j$. Then the algorithm below is applied to each syntactic-homogeneous circuit F_j *separately*, and the output of the algorithm will be defined as the *sum* $\sum_j [F_j]$ (this adds only a logarithm in d to the depth). For simplicity of writing we assume below in the algorithm that F is a *single* syntactic-homogeneous circuit (that is, a single F_j, for some j).

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 described in Lemma 10.5 for checking if w is in F_v.

Step $i = 0$:

Part (a):[17] We construct the node $[F_v]$, for all nodes $v \in F$ such that $d^+_{\mathrm{ub}}(v) \leq 1 = 2^i$.

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

Claim 10.10. *$\widehat{F_v} = a_1x_1 + \cdots + a_nx_n + \sum_{c\in J} b_c c$, for $a_1, \ldots, a_n, b_c \in \mathbb{Z}$ and $J \subset \mathbb{Z}$. Furthermore, there exists an $\mathbf{FNC}^2$-construction that given F constructs the depth $O(\log n)$ circuit $a_1x_1 + \cdots + a_nx_n + \sum_{c\in J} b_c c$.*

Proof of claim: Since $d^+_{\mathrm{ub}}(v) \leq 1$, 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 10.4 (and Theorem 2.7) we can construct in $\mathbf{FNC}^2$ the circuit $a_1x_1 + \cdots + a_nx_n + \sum_{c\in J} b_c c$ (we do not evaluate the circuit). ■Claim

Define

$$[F_v] := a_1x_1 + \cdots + a_nx_n + \sum\nolimits_{c\in J} b_c c.$$

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

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 $0 \leq d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 1$. Again, this is checked by the list from the preprocessing step, and since the input circuit F is assumed to contain the value of d^+_{ub} for each node.

[17] This base case uses an $\mathbf{FNC}^2$ algorithm, but since it is done only in the base case, and not in the induction step, the whole algorithm still is in $\mathbf{FNC}^2$.

Thus, by Proposition 10.3, the polynomial $\partial w f_v$ is a linear polynomial $a_1 x_1 + \cdots + a_n x_n + b$. Using Lemma 10.6 and similar notation and reasoning as Claim 10.10 define

$$[\partial w f_v] := a_1 x_1 + \cdots + a_n x_n + \sum\nolimits_{c \in J} b_c c \,.$$

Step $i+1$:

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 $0 \leq i \leq \lceil \log(d) \rceil$:

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

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 $2d^+_{\mathrm{ub}}(w) > d^+_{\mathrm{ub}}(v)$:

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

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

Put $m = 2^i + d^+_{\mathrm{ub}}(w)$. 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 $d^+_{\mathrm{ub}}(t_1) \geq d^+_{\mathrm{ub}}(t_2)$, or $t = t_2 \cdot t_1$ and $d^+_{\mathrm{ub}}(t_2) < d^+_{\mathrm{ub}}(t_1)$.

Finally, let $[F]$ be the circuit with output node $[F_u]$, where u is the output node of F. (Recall also that if F is a sum of two or more syntactic-homogeneous circuits F_j then $[F]$ is defined as the *sum* $\sum_{j=0}^{d} [F_j]$, where this sum is written as a depth $O(\log d)$ circuit.)

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 $\mathbf{\Sigma}^B_1$-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 10.13; see again [VSBR83, RY08, HT15]).

As mentioned in Section 1.1, given an algebraic circuit F with $d^+_{\mathrm{ub}}(u)$ polynomially bounded, for all nodes u in F, by first balancing and then evaluating the circuit (assuming, e.g., it is over the integers, as in the next section) we obtain 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^+_{ub} of the circuit is polynomial). The obtained algorithm is different from the previously known algorithm by Miller *et al.* [MRK88] (their algorithm does not require the syntactic-degree as input) and from that of Allender *et al.* [AJMV98] (which is implicit in that work but can be extracted from their text [All18]).

10.5 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 described above for circuits. This is similar to the proof-theoretic counterpart of the homogenization theorem shown in Section 5. We start by showing some properties of the constructions of the base cases of the balancing algorithm described in Lemmas 10.4 and 10.6 that $\mathbf{VNC}^2$ can prove.

Lemma 10.11 (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 $d^+_{\mathrm{ub}}(v) \leq 1$. 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 $0 \leq d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 1$. Then, there is a $\mathbb{P}_c(\mathbb{Z})$-proof of

$$[\partial w F_v] = [\partial w F_{v_1}] + [\partial w F_{v_2}], \qquad \text{in case } v = v_1 + v_2; \tag{65}$$

$$[\partial w F_v] = [\partial w F_{v_1}] \cdot [F_{v_2}], \qquad \text{in case } v = v_1 \cdot v_2 \text{ and } d^+_{\mathrm{ub}}(v_1) \geq d^+_{\mathrm{ub}}(v_2) \text{ or } v = v_2 \cdot v_1 \text{ and } d^+_{\mathrm{ub}}(v_1) > d^+_{\mathrm{ub}}(v_2). \tag{66}$$

Proof. Part (i). Consider Lemma 10.4. 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 $\mathrm{PowSeq}_{\mathbb{Z}}(n, s, A)$. 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 the number $\mathbf{\Sigma}^B_0$-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 10.6.

Case 1: $v = v_1 + v_2$. According to Lemma 10.6, $[\partial w F_v]$ 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 10.4, 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 10.6 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 $d^+_{\mathrm{ub}}(v) - d^+_{\mathrm{ub}}(w) \leq 1$). 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 10.6). 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 a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{V}^0$ [CN10]. The following is the main theorem of this section:

Theorem 10.12 (Balancing $\mathbb{P}_c(\mathbb{Z})$-proofs in $\mathbf{VNC}^2$)**.** *Let F, G be two $\mathbf{\Sigma}^B_1$-definable algebraic circuits over $\mathbb{Z}$ and assume that there is a $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ of size s which is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$, in which every circuit is a sum of syntactic homogeneous circuits with every node u appearing with its syntactic-degree upper bound $d^+_{\mathrm{ub}}(u) \leq d$. Then, the $\mathbb{P}_c(\mathbb{Z})$-proof of $[F] = [G]$ is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ and the depth of every circuit in the proof is $O(\log s \cdot \log d)$.*

Theorem 10.12 will be proved analogously to Theorem 5.4: the proof is similar to the proof of Theorem 5.4, only that instead of using Lemma 5.6 we use the analogous Lemma 10.13 below demonstrating some essential properties of $[F]$ that have short $\mathbb{P}_c(\mathbb{Z})$-proofs.

Lemma 10.13. *Let F_1, F_2 be $\mathbf{\Sigma}^B_1$-definable circuits in $\mathbf{VNC}^2$ each with size at most s, written as a sum of syntactic homogeneous circuits and with every node u appearing with its syntactic-degree upper bound $d^+_{\mathrm{ub}}(u) \leq d$. Then, the following equations have $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ proofs:*

$$[F_1 \oplus F_2] = [F_1] + [F_2]\,, \tag{67}$$

$$[F_1 \otimes F_2] = [F_1] \cdot [F_2]\,, \tag{68}$$

and $\mathbf{VNC}^2$ proves that the size of proofs is $\mathrm{poly}(s, d)$ and the depth of every circuit in the proof is $O(\log d \cdot \log s)$. Furthermore, $[z] = z$ has a constant-size proof whenever z is a variable or an integer.

We first prove Lemma 10.13 and then Theorem 10.12.

Proof of Lemma 10.13. The proof is similar to Lemma 4.4 in [HT15], except that we use $d^+_{\mathrm{ub}}(\cdot)$ 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 $\mathbf{\Sigma}^B_1$-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 (67) and (68).

Let $m(s, d)$ and $r(s, d)$ be functions such that for any circuit F with $d^+_{\mathrm{ub}}(F) = d$ and size s, $[F]$ has depth at most $r(s, d)$ and size at most $m(s, d)$. By construction of the balancing algorithm and the remarks that follow it, we can choose

$$m(s, d) = \mathrm{poly}(s, d) \;\text{ and }\; r(s, d) = O(\log d \cdot \log s).$$

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

$$[F_v] = \sum_{t \in \mathcal{B}_m(F_v)} [\partial t F_v] \cdot [F_{t_1}] \cdot [F_{t_2}] \,. \tag{69}$$

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 *sub-circuits*, 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 (69) 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

$$\lambda(s,0) = O(s^4) \quad \text{and} \quad \lambda(s,i) \leq O(s^4 \cdot m(s,d)) + \lambda(s,i-1). \tag{70}$$

Recurrence (70) implies $\lambda(s,d) = \text{poly}(s,d)$.

The following proposition (which is a constructive version of Proposition 4.10 in [HT15]) shows how to construct the desired $\mathbb{P}_c(\mathbb{Z})$-proofs of (67) and (68) when F is either $F_1 \oplus F_2$ or $F_1 \otimes F_2$ (and v is the root of F), and where $F_1 \oplus F_2$ and $F_1 \otimes F_2$ are each a *single* syntactic homogeneous circuit, and not a sum of syntactic homogeneous circuits. To see that this suffices for the general case of (67) and (68), note first that in fact $F_1 \otimes F_2$ cannot be a sum of more than one syntactic homogeneous circuits because the gate at its root is a product gate. Second, if $F_1 \oplus F_2$ is a sum of two or more syntactic homogeneous circuits F_j written as $\sum_j F_j$, then we can assume without loss of generality that F_1 is a syntactic homogeneous circuit and $F_2 = \sum_j F_j$ is a sum of one or more syntactic homogeneous circuits. Hence, $F_1 \oplus F_2 = F_1 + \sum_j F_j$ and by the definition of the balancing algorithm $[F_1 \oplus F_2] = [F_1] + \sum_j [F_j] = [F_1] + [F_2]$ and we are done.

Proposition 10.14. *Let F be a syntactically homogenous circuit with all nodes v having their syntactic degree upper bound $d^+_{\text{ub}}(v) \leq d$ given, and assume that F is of size s and is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$. Then, for every $i = 0, \ldots, \lceil \log d \rceil$ there exists a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ $\mathbb{P}_c(\mathbb{Z})$ proof-sequence Ψ_i of size at most $\lambda(s,i)$ and depth at most $O(r(s,d))$, such that:*

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

$$d^+_{\text{ub}}(v) \leq 2^i, \tag{71}$$

Ψ_i contains the following equations:

$$[F_v] = [F_{v_1}] + [F_{v_2}] \,, \quad \text{in case } v = v_1 + v_2, \quad \text{and} \tag{72}$$

$$[F_v] = [F_{v_1}] \cdot [F_{v_2}] \,, \quad \text{in case } v = v_1 \cdot v_2. \tag{73}$$

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

$$d^+_{\text{ub}}(v) - d^+_{\text{ub}}(w) \leq 2^i \quad \text{and} \tag{74}$$

$$2 d^+_{\text{ub}}(w) > d^+_{\text{ub}}(v), \tag{75}$$

Ψ_i contains the following equations:

$$[\partial w F_v] = [\partial w F_{v_1}] + [\partial w F_{v_2}], \qquad \text{in case } v = v_1 + v_2; \tag{76}$$

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

$$\text{or } v = v_2 \cdot v_1 \text{ and } d^+_{\text{ub}}(v_1) > d^+_{\text{ub}}(v_2). \tag{77}$$

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 analogous manner to the balancing algorithm above.

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

Part (a): proof of (72). 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 (72). This stems from Lemma 10.11 part (i).

Part (b): proof of (76) and (77). Similarly to part (a) above, this follows from Lemma 10.11 part (ii).

Overall, Ψ_0 will be the union of all the above proofs, so that Ψ_0 contains all equations (72) (for all nodes v satisfying (71)), and all equations (76) and (77) (for all nodes v, w satisfying (74) and (75)). 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 (72) and (73). Let v be any node in F such that

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

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:

$$[F_v] = [F_{v_1+v_2}] = \sum_{t \in \mathcal{B}_m(F_v)} [F_t] \cdot [F_t] \cdot [\partial t(F_{v_1+v_2})]\,. \tag{78}$$

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

$$[F_{v_e}] = \sum_{t \in \mathcal{B}_m(F_{v_e})} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t(F_{v_e})], \quad \text{for } e \in \{1, 2\}\,. \tag{79}$$

If $t \in \mathcal{B}_m(F_v)$ then $d^+_{\text{ub}}(t) > m = 2^i$. Therefore, for any $t \in \mathcal{B}_m(F_v)$, since $d^+_{\text{ub}}(v) \leq 2^{i+1}$, we have $d^+_{\text{ub}}(v) - d^+_{\text{ub}}(t) < 2^i$ and $2d^+_{\text{ub}}(t) > d^+_{\text{ub}}(v)$ 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 (78) equals:

$$\begin{aligned} &\sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot ([\partial t F_{v_1}] + [\partial t F_{v_2}]) \\ &= \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_1}] + \sum_{t \in \mathcal{B}_m(F_v)} [F_{t_1}] \cdot [F_{t_2}] \cdot [\partial t F_{v_2}]. \end{aligned} \tag{80}$$

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

$$\sum_{t \in \mathcal{B}_m(F_v)} [\partial t F_{v_e}] = \sum_{t \in \mathcal{B}_m(F_{v_e})} [\partial t F_{v_e}], \quad \text{for } e = 1, 2. \tag{81}$$

Thus, using (79) we have that (80) equals:

$$\sum_{t\in\mathcal{B}_m(F_{v_1})}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial tF_{v_1}]+\sum_{t\in\mathcal{B}_m(F_{v_2})}[F_{t_1}]\cdot[F_{t_2}]\cdot[\partial tF_{v_2}] \\ =[F_{v_1}]+[F_{v_2}]. \tag{82}$$

The above proof of (82) 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 (76) and (77) are similar to Case 1 above, and are identical to those cases in [HT15, proof of Proposition 4.10]; like Case 1, the difference from [HT15] is that we construct with an $\mathbf{FNC}^2$ procedure 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 natural encoding scheme for proof-lines).

This concludes the proof of Proposition 10.14. □

Hence we also concluded Lemma 10.13. □

Proof of Theorem 10.12. We assumed that π is a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ $\mathbb{P}_c(\mathbb{Z})$-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. Simultaneously for each proof-line $F_1=F_2$ in π we are going to construct a (part of a) $\mathbb{P}_c(\mathbb{Z})$-proof of $[F_1]=[F_2]$ using pointers to previous lines (the pointers are $\mathbf{\Sigma}^B_1$-definable number functions in $\mathbf{VNC}^2$). This resembles the proof structure of Theorem 5.4. We can use the balancing algorithm on F_1,F_2 because by assumption these circuits are given as a sum of syntactic homogeneous circuits with all nodes appearing together with their associated syntactic-degree upper bound d^+_{ub}.

Case 1: $F=H$ is an axiom of $\mathbb{P}_c(\mathbb{Z})$. Then, $[F]=[H]$ has a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}_c(\mathbb{Z})$-proof in $\mathbf{VNC}^2$ as follows. 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 10.13, as follows. Axioms C1 and C2 are already the statement of Lemma 10.13. For the other axioms, consider for example

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

We need to show that the following has a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}_c(\mathbb{Z})$-proof:

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

Since we assume that all proof-lines are written as sums of syntactic homogeneous circuits with all nodes having their syntactic-degree upper bounds specified, $F_1\cdot(G_1+G_2)$ and $F_1\cdot G_1+F_1\cdot G_2$ are written as such circuits and hence F_1,G_1,G_2 and G_1+G_2 must also be sums of one or more syntactic homogeneous circuits. Therefore, by Lemma 10.13 we have a $\mathbf{\Sigma}^B_1$-definable $\mathbb{P}_c(\mathbb{Z})$-proof of:

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

Lemma 10.13 gives again:

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

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 by Lemma 10.13 and the assumption that F_1, F_2, G_1, G_2 and $F_1 \circ F_2, G_1 \circ G_2$ are all written as sums of syntactic homogeneous circuits, from the equations $[F_1] = [G_1]$ and $[F_2] = [G_2]$ there is a proof of $[F_1 \circ G_1] = [F_2 \circ G_2]$.

Altogether, we obtain a $\mathbf{\Sigma}^B_1$-definable proof of $[F] = [G]$. □

We can now finally obtain the balanced $\mathbb{P}_c(\mathbb{Z})$-proofs 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,

$$\mathsf{Det}_{balanced}(X) := [\mathsf{Det}^\star_{Taylor}(X)]. \tag{83}$$

Corollary 10.15. *Let X, Y be $n \times n$ symbolic matrices and $U = \{u_{ij}\}_{i,j\in[n]}$ be a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$ triangular $n \times n$ matrix in the variables x_{ij} for $i, j \in [n]$ with no division gates. Then, the $\mathbb{P}_c(\mathbb{Z})$-proofs of the following determinant identities*

$$\mathsf{Det}_{balanced}(X) \cdot \mathsf{Det}_{balanced}(Y) = \mathsf{Det}_{balanced}(XY) \quad \textit{and} \tag{84}$$

$$\mathsf{Det}_{balanced}(U) = u_{11} \cdots u_{nn} \tag{85}$$

are $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$. Moreover, in these proofs every circuit has depth $O(\log^2 n)$.

Proof. By Corollary 10.9 the proofs of $\mathsf{Det}^\star_{Taylor}(X) \cdot \mathsf{Det}^\star_{Taylor}(Y) = \mathsf{Det}^\star_{Taylor}(XY)$ and $\mathsf{Det}^\star_{Taylor}(U) = u_{11} \cdots u_{nn}$ are $\mathbf{\Sigma}^B_1$-definable in $\mathbf{V}^0$, and every circuit in these proofs is a sum of syntactic homogeneous circuits in which every node u appears with its syntactic-degree upper bound $d^+_{\mathrm{ub}}(u) \leq 2n$. Applying Theorem 10.12 on these $\mathbb{P}_c(\mathbb{Z})$-proofs we obtain a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ $\mathbb{P}_c(\mathbb{Z})$-proofs of $[\mathsf{Det}^\star_{Taylor}(X) \cdot \mathsf{Det}^\star_{Taylor}(Y)] = [\mathsf{Det}^\star_{Taylor}(XY)]$ and $[\mathsf{Det}^\star_{Taylor}(U)] = [u_{11} \cdots u_{nn}]$. Using Lemma 10.13 and the definition of $\mathsf{Det}_{balanced}(X)$ we obtain a proof of $[\mathsf{Det}^\star_{Taylor}(X) \cdot \mathsf{Det}^\star_{Taylor}(Y)] = [\mathsf{Det}^\star_{Taylor}(X)] \cdot [\mathsf{Det}^\star_{Taylor}(Y)] = \mathsf{Det}_{balanced}(X) \cdot \mathsf{Det}_{balanced}(Y) = \mathsf{Det}_{balanced}(XY)$. □

11 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 $\mathbf{VNC}^2$.

11.1 Algebraic $\mathbf{NC}^2$-Circuit Value Problem

We show that there is a $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ algorithm that receives an *algebraic* circuit over $\mathbb{Z}$ with n input variables that is balanced according to the balancing algorithm in Section 10 together with an assignment of integers to the variables written as an array of binary strings, and outputs the value of the circuit under the assignment. We use the fact that the balancing algorithm we provided in Section 10 results in fact in $O(\log n)$ depth algebraic circuits in which plus gates can be considered to be *unbounded fan-in* and product gates have fan-in two (see Fact 11.1).

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 and the Boolean circuit is *multi-output*, that is, has more than one output gate, one for each bit of the computed integer; **ii**) layer the circuit; **iii**) evaluate the balanced layered Boolean circuit using the evaluation function for such circuits.

Step (i): From balanced algebraic circuits to balanced Boolean circuits. We show how to transform a balanced polynomial-size algebraic circuit as was constructed in Section 10.4 into a polynomial-size $O(\log^2 n)$-depth Boolean circuit with a Σ^B_1-definable $\mathbf{VNC}^2$ algorithm. We use the following two facts:

Fact 11.1 (Observed by Vinay [Vin91])**.** *Given an algebraic circuit of* poly(n)*-size and* poly(n)*-degree, our algorithm in Section 10.4 (and the original [VSBR83] algorithm) that balances the circuit into $O(\log^2 n)$-depth, in fact balances the circuit into $O(\log n)$-depth circuit except that* the plus gates have unbounded fan-in *(and product remains of fan-in two).*

We stress that all our circuits whether algebraic or Boolean, formally have *fan-in two* gates. However, the outputs of the balancing algorithm in Section 10.4 can be *considered* as having $O(\log n)$ depth with unbounded fan-in plus gates and fan-in two product gates. In other words, our balanced algebraic circuits are fan-in two circuits that result from $O(\log n)$-depth circuits with fan-in two product gates and unbounded fan-in plus gates by turning each iterated plus gates $u_1 + \cdots + u_m$ into a tree of logarithmic in m depth with fan-in two plus gate and $u_1, \ldots, u_m$ on the leaves. We say that such fan-in two balanced algebraic circuits are ***implicitly*** an $O(\log n)$-depth circuit with unbounded fan-in plus gates and fan-in two product gates. We can assume further that the balancing algorithm in Section 10.4 outputs circuits with all the the iterated plus gates also specified as such, hence we will be able to convert the circuit into a balanced Boolean circuit in the algorithm that follows.

Fact 11.2. *The Boolean (multi-valued) function denoted* ItAdd_{func} *computing the iterated addition of (two or more) integers written in binary is in* $\mathbf{FO}$*-uniform* $\mathbf{FTC}^0$ *(see [CN10, IX.3.6.1]). 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]). Both of these are* Σ^B_1*-definable in* $\boldsymbol{VTC}^0$ *(and hence also in* $\mathbf{VNC}^2$*), the theory that corresponds to* $\boldsymbol{TC}^0$*, as shown in [CN10, IX.3.6].*

By Fact 11.2 both plus gates of unbounded fan-in (equivalently, iterated addition) and product gates of fan-in two are computable in $\mathbf{FTC}^0$ ($\subseteq \mathbf{FNC}^1$), and hence both have $O(\log n)$-depth fan-in two Boolean circuits of polynomial-size with $\wedge, \vee, \neg$ gates only (where n is the *input size*, namely total size of all numbers written in binary). Denote by $\mathsf{ItAdd}_{\text{circ}}$ the corresponding Boolean circuit of iterated addition $\mathsf{ItAdd}_{\text{func}}$ (see Section 11.2).

By Fact 11.1 an implicit $O(\log n)$-depth algebraic circuit of poly(n) size with unbounded fan-in plus gates and fan-in two product gates can be simulated by fan-in two $O(\log^2(nm))$-depth Boolean circuits of poly(n, m) size, assuming the inputs to the circuit are written in binary, each with m number of bits. Accordingly, we have the following:

Σ^B_1-definable function in $\mathbf{VNC}^2$ for transforming balanced algebraic circuits into Boolean circuits

---

Input: A fan-in two algebraic circuit C of size s, syntactic-degree d and depth $O(\log s \log d)$ which is implicitly an $O(\log d)$-depth circuit with unbounded fan-in plus gates and fan-in two product gates, together with a number m (given in unary) for the bit-length of integers.

Output: A multi-output fan-in two Boolean circuit of depth $O(\log d \cdot \log(sm))$ and size poly(s, d, m) that computes the same polynomial as the input circuit.

Algorithm

1. If $S = D_1 + \cdots + D_r$ (for some $r \leq s$) is an iterated sum in C written as a fan-in two circuit then replace it with a polynomial-size fan-in two and depth $O(\log(rm))$ Boolean circuit computing the corresponding iterated sum of integers.

   Note that we can assume that each iterated sum in C is already marked as such (namely, as part of the specified iterated sum), during the algorithm for balancing in Section 10.4. Moreover, notice that in the balancing algorithm an iterated sum like S is written as a *tree* with leaves D_i's, namely no $+$ gate in S is re-used in the circuit.

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

The algorithm is a straightforward node-by-node transformation of the algebraic circuit and is doable in $\mathbf{FNC}^2$ and hence $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$.

This resulting Boolean circuit is encoded in the same way that algebraic circuits are encoded; namely, via the encoding scheme in Section 3.1.1 (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 $i+1$. While in Section 2.4 the circuits were monotone, here they are not necessarily monotone. The encoding of layered Boolean circuits is done similar to Section 2.4: a layered Boolean circuit with $d+1$ layers $0, \ldots, d$ is encoded with a string variable I, with $|I| \leq n$, which defines the (Boolean) input gates to the circuit. We have a three-dimensional string variable G such that for $0 \leq x \leq d$, $G(x, y, 0)$ holds if the yth gate in layer x is $\wedge$, and $G(x, y, 1)$ holds if the gate is $\vee$ and $G(x, y, 2)$ holds if the gate is $\neg$; this is the only difference between the encoding of non-monotone circuits and monotone circuits (where the latter had G as a two-dimensional array). Accordingly, the wires of C are encoded by a three-dimensional array 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 $z+1$.

We can convert with a $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ algorithm any $O(\log^2 n)$-depth Boolean circuit (with multi-output gates) from Step (**i**) above into a layered Boolean circuit, as follows.

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

---

Input: A multi-output Boolean circuit F of depth $d = c \log^2 n$, for some constant c (encoded as in Section 3.1.1).

Output: A layered multi-output Boolean circuit F' computing the same function as F with d layers.

Algorithm

1. Let A be the $s \times s$ 0-1 adjacency matrix of F where s is the number of nodes in F, and the (u, w)th entry in A, denoted $A[u, w]$, is 1 iff there is a directed edge from node u to node w in F. Note that A is a $\mathbf{\Sigma}_1^B$-definable in $\mathbf{V}^0$ function of F. Using the $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$ string function $\mathrm{PowSeq}_{\mathbb{Z}}(n, s, A)$ (as in Section 10) that receives as input an $n \times n$ integer matrix A and outputs a string coding the sequence $(A, A^2, \ldots, A^s)$ of powers of A,

we find the maximal length of a directed path from each of the internal gates in F to each of the output gates in F: note that $A^i[u,v]$ is the number of distinct directed paths of length precisely i from u to v. Hence, the longest path from any given output gate r to u is the maximal i with $A^i[u,r] \neq 0$. (Observe that $A^i[u,r]$ only includes paths of length precisely i and not paths of length smaller than i.)

2. Let F' be the circuit F in which for every node $u \in V$, for V the set of nodes of F, we change u into the pair $(u, d-\ell)$, where ℓ is the maximal length of a directed path from u to an output node in F (the maximum over all output nodes). The number $d-\ell$ will serve as *the layer* of $(u, d-\ell)$ in F'.

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

Step (iii): Evaluating Layered Boolean circuits. As described in Section 2.4, by definition the theory $\mathbf{VNC}^2$ proves the existence of the evaluation string of $O(\log^2 n)$depth monotone Boolean circuits under an assignment to its inputs (as in (6)). In order to prove the reflection principle in Section 11.2 below it is more convenient to work with evaluation functions for circuits that are not-necessarily monotone. It is possible to show that $\mathbf{VNC}^2$ proves the existence of evaluation strings not only for balanced monotone circuits but also for balanced non-monotone circuits. However, for our purpose it suffices to simply use the fact that the evaluation of (non-monotone) Boolean circuits of depth $O(\log^2 n)$ is in $\mathbf{FNC}^2$ and hence by Theorem 2.7 the following function is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$: denote by $\mathsf{Eval}_{bool}(F,A)$ the string function that receives a layered Boolean circuit F possibly with more than one output gate, and an assignment A to its variables, and returns the string consisting of the output bits of f under A.

We can now define the function $\mathsf{Eval}_{alg}(F,A)$ that receives the string variable F encoding an algebraic circuit over the integers of depth $O(\log n)$, fan-in two product gates and unbounded fan-in plus gates, together with 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 F under the assignment A.

We define $\mathsf{Eval}_{alg}(F,A)$ in the theory so that it first constructs the corresponding layered Boolean circuit of depth $O(\log^2 n)$ denoted $B(F)$ and then evaluates it under A using $\mathsf{Eval}_{bool}(F,A)$ (remember that A is a two-dimensional array of bit-strings representing a sequence of integers hence the input to the former and latter function is the same). By Steps (i), (ii) and the above discussion Eval_{alg} is a $\mathbf{\Sigma}^B_1$-definable string function in $\mathbf{VNC}^2$.

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

Now that we know how to $\mathbf{\Sigma}^B_1$-define in $\mathbf{VNC}^2$ the evaluation of balanced algebraic circuits, we show the following reflection principle for $\mathbb{P}_c(\mathbb{Z})$ in which circuits have logarithmic depth, fan-in

two product gates and essentially unbounded fan-in plus gates:

Theorem 11.3 (Balanced $\mathbb{P}_c(\mathbb{Z})$ reflection principle in $\mathbf{VNC}^2$)**.** *Assume that the $\mathbb{P}_c(\mathbb{Z})$-proof of the circuit equation $F = G$ is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$, and that F, G has n variables. Further, suppose that every circuit in the $\mathbb{P}_c(\mathbb{Z})$-proof is a fan-in two balanced algebraic circuit C which is implicitly a $O(\log n)$-depth circuit with unbounded fan-in plus gates and fan-in two product gates. Then $\mathbf{VNC}^2$ proves that $\forall n \forall m \forall A \left(\mathsf{Eval}_{alg}(F, A) = \mathsf{Eval}_{alg}(G, A)\right)$, where the assignment A is a two-dimensional array of n integers with m bit-length each.*

Before proving this theorem we provide a more general setting under which reflection principles for *unbounded depth* $\mathbb{P}_c(\mathbb{Z})$-proofs hold. This general setting may find other applications in bounded arithmetic. Note that while in Theorem 11.3 we start with a balanced $\mathbb{P}_c(\mathbb{Z})$-proof and thus can evaluate every proof-line directly with the evaluation function $\mathsf{Eval}_{alg}(F, A)$, in the corollary below we start with an *unbalanced* $\mathbb{P}_c(\mathbb{Z})$-proof.

Corollary 11.4 (Generalized $\mathbb{P}_c(\mathbb{Z})$ reflection principle in $\mathbf{VNC}^2$)**.** ***(i)*** *Assume that the $\mathbb{P}_c(\mathbb{Z})$-proof of $\{F_n = G_n\}_{n=1}^{\infty}$ is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$, and suppose that $d^+_{\text{ub}}(F_n) = \text{poly}(n)$ and $d^+_{\text{ub}}(G_n) = \text{poly}(n)$. Then, $\mathbf{VNC}^2$ proves that $\forall n \forall m \forall d \forall A \left(\mathsf{Eval}_{alg}\left(\sum_{i=0}^{d} {F_n}^{(i)}, A\right) = \mathsf{Eval}_{alg}\left(\sum_{i=0}^{d} {G_n}^{(i)}, A\right)\right)$.* ***(ii)*** *Suppose further that every node in the $\mathbb{P}_c(\mathbb{Z})$-proofs of $\{F_n = G_n\}_{n=1}^{\infty}$ appears with a syntactic-degree d^+_{ub} witness[18], where the syntactic-degree d^+_{ub} of F_n, G_n is at most $\text{poly}(n)$. Then, $\mathbf{VNC}^2$ proves that $\forall n \forall m \forall A \left(\mathsf{Eval}_{alg}(F_n, A) = \mathsf{Eval}_{alg}(G_n, A)\right)$.*

Proof. Part (i): first homogenize the $\mathbb{P}_c(\mathbb{Z})$-proof using Theorem 5.4, from which we get $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ $\mathbb{P}_c(\mathbb{Z})$-proofs of ${F_n}^{(i)} = {G_n}^{(i)}$, for all natural $n \geq 1$. This leads immediately to $\mathbb{P}_c(\mathbb{Z})$-proofs of $\sum_{i=0}^{d} {F_n}^{(i)} = \sum_{i=0}^{d} {G_n}^{(i)}$, for all natural $n \geq 1$ and all natural $d \geq 0$, where every circuit in the $\mathbb{P}_c(\mathbb{Z})$-proofs appear as a sum of homogeneous components, and every node appears with its syntactic-degree upper bound d^+_{ub} at most $\text{poly}(n)$. Using Theorem 10.12 we balance these proofs so that every proof-line is of depth $O(\log n)$ with unbounded fan-in plus gates and fan-in two product gates. Using Theorem 11.3 above we are done.

Part (ii): This follows the same proof as part (i), only that due to the precise syntactic-degrees d^+_{ub} for each node we can conclude also that $F_n = \sum_{i=0}^{d} F^{(i)}$ and $G_n = \sum_{i=0}^{d} G^{(i)}$, for some $d = \text{poly}(n)$ (see Remark 5.5 on the need to use syntactic-degrees for this purpose.) □

Proof of Theorem 11.3. The proof proceeds by *number* induction (Proposition 2.2) on the number of proof-lines in π, using Lemma 11.5 below.

Since the evaluation function Eval_{alg} is $\mathbf{\Sigma}^B_1$-definable in $\mathbf{VNC}^2$ we can use this function in the number induction axiom (see Section 2.5). Specifically, let π be the $\mathbb{P}_c(\mathbb{Z})$-proof of $F = G$ and consider the $\mathbf{\Sigma}^B_0$-formula (using the function symbol Eval_{alg}) $Q(n) := \forall i \leq n \left(\mathsf{Eval}_{alg}\left(\text{left}\left(\pi^{[i]}\right), A\right) = \mathsf{Eval}_{alg}\left(\text{right}\left(\pi^{[i]}\right), A\right)\right)$, where $\text{left}(\pi^{[i]})$ and $\text{right}(\pi^{[i]})$ are the left (resp. right) hand side circuits in the ith proof-line in π. Then, the induction states that assuming the first line $\pi^{[0]}$ is true under an assignment A, namely, $Q(0)$, and if $Q(n) \rightarrow Q(n+1)$ 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 (because it is either an axiom or was derived from previous lines). This concludes the argument since we end up with the last proof-line $F = G$ being true under A.

[18] Here we mean syntactic-degrees of nodes when considering also scalars as contributing to the syntactic-degree of products, as was defined for d^+_{ub} (only that in d^+_{ub} we seek an upper bound and here we need a syntactic-degree).

It 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: $\forall n \forall m \leq t(n) \forall A \leq c\,(\mathsf{Eval}_{alg}(F, A) = \mathsf{Eval}_{alg}(G, A))$ when $F = G$ is an axiom of $\mathbb{P}_c(\mathbb{Z})$. For example, $F + 0 = F$ holds for every integer assignment to the variables of F. (2) The rules of $\mathbb{P}_c(\mathbb{Z})$ are sound under integer assignments. This is proved in Lemma 11.5 below. □

To prove Lemma 11.5 we will use the following notation and facts. For two binary strings A, B denote by $A +_b B$ the binary addition of A and B (which is a $\mathbf{\Sigma}^B_1$-definable function in $\mathbf{V}^0$ using the usual carry-save addition). For an algebraic circuit F recall that $B(F)$ denotes the layered Boolean circuit that is constructed by the evaluation function $\mathsf{Eval}_{alg}(F, A)$ as described above in steps (i) and (ii) ($B(F)$ is independent of the assignment A). By definition $\mathsf{Eval}_{alg}(F, A)$ first constructs $B(F)$ and then uses $\mathsf{Eval}_{bool}(B(F), A)$ to evaluate $B(F)$ under A.

Given n binary strings $D_1, \ldots, D_n$ of length m each, encoded as a two-dimensional array $\overline{D}$, $\mathsf{ItAdd}_{\text{func}}(\overline{D}, n, m)$ is the $\mathbf{\Sigma}^B_1$-definable string function in $\mathbf{VNC}^2$ that computes the iterated addition of the D_i's, and $\mathsf{ItAdd}_{\text{circ}}(\overline{D}, n, m)$ is the corresponding layered $O(\log n)$-depth multi-output Boolean circuit that computes this function.

We shall assume that $\mathsf{ItAdd}_{\text{func}}$ is defined in $\mathbf{VNC}^2$ using the evaluation of the Boolean circuit for iterated addition $\mathsf{ItAdd}_{\text{circ}}$ as follows. Let $\overline{Z}$ be a string variable, then $\mathsf{ItAdd}_{\text{func}}(\overline{D}, n, m)$ is defined by:

$$\mathsf{ItAdd}_{\text{func}}(\overline{D}, n, m) := \mathsf{Eval}_{bool}(\mathsf{ItAdd}_{\text{circ}}(\overline{Z}, n, m), \overline{D}). \tag{86}$$

By the same argument as in [CN10, Section IX.3.6.2 (cf. equation (251))], $\boldsymbol{VTC}^0$ (and hence also $\mathbf{VNC}^2$) proves that

$$\mathsf{ItAdd}_{\text{func}}(\overline{D}, 0, m) = \emptyset \quad \text{and} \tag{87}$$

$$\mathsf{ItAdd}_{\text{func}}(\overline{D}, n, m) = \mathsf{ItAdd}_{\text{func}}(\overline{D}, n-1, m) +_b D_n\,, \tag{88}$$

with D_n denoting the nth number in $\overline{D}$ (that is, $D^{[n]}$), and $\emptyset$ the empty string.

Lemma 11.5 (Soundness of $\mathbb{P}_c(\mathbb{Z})$ rules and axioms)**.** *Let F_1, F_2, G_1, G_2, F, G be $O(\log n)$-depth circuits with unbounded fan-in plus gates and fan-in two product gates, and let A be an assignment of integers to their input variables. (i) If $\mathsf{Eval}_{alg}(F_1, A) = \mathsf{Eval}_{alg}(G_1, A)$ and $\mathsf{Eval}_{alg}(F_2, A) = \mathsf{Eval}_{alg}(G_2, A)$ then $\mathsf{Eval}_{alg}(F_1 \circ F_2, A) = \mathsf{Eval}_{alg}(G_1 \circ G_2, A)$, for $\circ \in \{+, \times\}$. (ii) If $F = G$ is an axiom of $\mathbb{P}_c(\mathbb{Z})$, then $\mathsf{Eval}_{alg}(F, A) = \mathsf{Eval}_{alg}(G, A)$.*

Proof. Part (i). Consider the rule in which $F_1 + F_2 = G_1 + G_2$ is derived from $F_1 = G_1$ and $F_2 = G_2$. We need to prove that $\mathsf{Eval}_{alg}(F_1 + F_2, A) = \mathsf{Eval}_{alg}(G_1 + G_2, A)$, assuming that $\mathsf{Eval}_{alg}(F_1, A) = \mathsf{Eval}_{alg}(G_1, A)$ and $\mathsf{Eval}_{alg}(F_2, A) = \mathsf{Eval}_{alg}(G_2, A)$. (The rule for $\times$ is easier and we omit the details.)

It is enough to show the following (and similarly for G_1, G_2):

Claim 11.6. $\mathsf{Eval}_{alg}(F_1, A) +_b \mathsf{Eval}_{alg}(F_2, A) = \mathsf{Eval}_{alg}(F_1 + F_2, A)$.

Having this lemma we are done, because from assumption $\mathsf{Eval}_{alg}(F_1, A) +_b \mathsf{Eval}_{alg}(F_2, A) = \mathsf{Eval}_{alg}(G_1, A) +_b \mathsf{Eval}_{alg}(G_2, A)$ (directly by equality axioms).

Proof of Claim 11.6. Consider the algebraic circuits F_1, F_2, G_1, G_2. Because algebraic circuits possess plus gates of fan-in two while in our translation to Boolean circuits we combine iterated additions into a Boolean sub-circuit that computes this iterated addition of possibly more than two numbers, we need to consider different cases based on the output gate of F_1, F_2, G_1, G_2.

Case 1: The output gates of both F_1, F_2 are not $+$. Thus, the layered Boolean circuit constructed by $\mathsf{Eval}_{alg}(F_1 + F_2, A)$ is $B(F_1 + F_2) = \mathsf{ItAdd}_{\text{circ}}(\overline{B(F_1), B(F_2)}, 2, m)$, assuming the number of bits is m and where $\overline{B(F_1), B(F_2)}$ denotes the two-dimensional array with first string $B(F_1)$ and second string $B(F_2)$. Note that because F_1, F_2 have no plus gates at their output, the $\mathsf{ItAdd}_{\text{circ}}$ at the output (sub-circuit) in $B(F_1+F_2)$ has only the *two* binary strings $B(F_1), B(F_2)$ added together (this is simpler than Case 2 below). By definition $\mathsf{Eval}_{alg}(F_1 + F_2, A)$ first constructs the layered Boolean circuit $B(F_1 + F_2)$ and then evaluates it under A using $\mathsf{Eval}_{bool}(B(F_1 + F_2), A)$. Hence, $\mathbf{VNC}^2$ proves that

$$\begin{aligned}
\mathsf{Eval}_{alg}(F_1 + F_2, A) &= \mathsf{Eval}_{bool}\left(\mathsf{ItAdd}_{\text{circ}}\left(\overline{B(F_1), B(F_2)}, 2, m\right), A\right) \\
&= \mathsf{ItAdd}_{\text{func}}(\overline{\mathsf{Eval}_{bool}(B(F_1), A), \mathsf{Eval}_{bool}(B(F_2), A)}, 2, m) \quad \text{by (86)} \\
&= \mathsf{Eval}_{bool}(B(F_1), A) +_b \mathsf{Eval}_{bool}(B(F_2), A) \quad \text{by (88)} \\
&= \mathsf{Eval}_{alg}(F_1, A) +_b \mathsf{Eval}_{alg}(F_2, A) \quad \text{by definition of } \mathsf{Eval}_{alg}.
\end{aligned}$$

Case 2: Both F_1, F_2 have $+$ output gates. That is, $F_1 = H_1 + \cdots + H_r$ and $F_2 = K_1 + \cdots + K_l$, where $r, l \geq 2$. Let the number of bits be again m, then $B(F_1) = \mathsf{ItAdd}_{\text{circ}}(\overline{H_1, \ldots, H_r}, r, m)$, $B(F_2) = \mathsf{ItAdd}_{\text{circ}}(\overline{K_1, \ldots, K_l}, l, m)$, and $B(F_1 + F_2) = \mathsf{ItAdd}_{\text{circ}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, r + l, m)$, where as before, for k strings $D_1, \ldots, D_k$ we denote by $\overline{D_1, \ldots, D_k}$ the two-dimensional array in which the ith string is D_i. This is similar to Case 1 above, only that we need to show in $\mathbf{VNC}^2$ that

$$\begin{aligned}
&\mathsf{ItAdd}_{\text{func}}(\overline{H_1, \ldots, H_r}, r, m) +_b \mathsf{ItAdd}_{\text{func}}(\overline{K_1, \ldots, K_l}, l, m) \\
&\qquad = \mathsf{ItAdd}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, r + l, m).
\end{aligned}$$

Let Z be a string variable standing for a two-dimensional array of strings $Z_1, Z_2, \ldots$, let i, j, n, m be number variables, and define $\mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, i, j, m)$ to be the Σ^B_1-definable function in $\mathbf{VNC}^2$ that sums the binary numbers $Z_i + \cdots + Z_j$, each of bit-length m (that is, $\mathsf{ItAdd}^{\bullet}_{\text{func}}$ is similar to $\mathsf{ItAdd}_{\text{func}}$ only that we start summing from Z_i and not Z_1).

Note that $\mathbf{VNC}^2$ proves $\mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, n, m) = \mathsf{ItAdd}_{\text{func}}(Z, n, m)$ and $\mathsf{ItAdd}^{\bullet}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, 1, r, m) = \mathsf{ItAdd}_{\text{func}}(\overline{H_1, \ldots, H_r}, r, m)$ and $\mathsf{ItAdd}^{\bullet}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, r + 1, l + r, m) = \mathsf{ItAdd}_{\text{func}}(\overline{K_1, \ldots, K_l}, l, m)$. Hence, it remains to prove:

$$\begin{aligned}
&\mathsf{ItAdd}^{\bullet}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, 1, r, m) +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, r+1, l+r, m) \\
&\qquad = \mathsf{ItAdd}_{\text{func}}(\overline{H_1, \ldots, H_r, K_1, \ldots, K_l}, r + l, m). \quad (89)
\end{aligned}$$

Define the following Σ^B_0-formula (in the language $\mathcal{L}_A \cup \{\mathsf{ItAdd}_{\text{func}}, +_b\}$):

$$\varphi(i, Z, n, m) := \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, i, m) +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, i + 1, n, m) = \mathsf{ItAdd}_{\text{func}}(Z, n, m)\,.$$

To prove (89) it suffices to prove in $\mathbf{VNC}^2$ $\forall i \varphi(i)$. We proceed by a number induction over i using the formula $\varphi(i, Z, n, m)$. The base case $\varphi(0, Z, n, m)$ is $\emptyset +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, n, m) = \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, n, m) = \mathsf{ItAdd}_{\text{func}}(Z, n, m)$, and we are done. The induction step is proved using (88) (which holds also for $\mathsf{ItAdd}^{\bullet}_{\text{func}}$):

$$\begin{aligned}
&\mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, i, m) +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, i + 1, n, m) \\
&\quad = \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, i, m) +_b (Z_{i+1} +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, i + 2, n, m)) \\
&\quad = \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, 1, i + 1, m) +_b \mathsf{ItAdd}^{\bullet}_{\text{func}}(Z, i + 2, n, m)\,.
\end{aligned}$$

Case 3: F_1 has a plus output gate, while F_2 does not, or vice versa. This is similar to the previous cases. □

Part (ii) is similar to part (i) and we omit the details. This concludes Lemma 11.5. □

11.3 Wrapping Up

The Determinant Function DET in $\mathbf{VNC}^2$. Given an $n \times n$ integer matrix A, the determinant function $\mathrm{DET}(A)$ 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, as was shown above. Recall that $\mathsf{Det}_{balanced}(X)$ is the circuit for the determinant defined in (83).

Definition 11.7 (Determinant function DET in $\mathbf{VNC}^2$). *Given an integer matrix A, the determinant in $\mathbf{VNC}^2$ is defined as* $\mathrm{DET}(A) := \mathsf{Eval}_{alg}(\mathsf{Det}_{balanced}(X), A)$.

The fact that $\mathrm{DET}(A)$ computes the determinant function follows, for instance, from Theorem 11.8 below. Since both the string functions $\mathsf{Det}_{balanced}(X)$ and $\mathsf{Eval}_{alg}(F, A)$ are $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$, DET is $\mathbf{\Sigma}_1^B$-definable in $\mathbf{VNC}^2$.

Using the definition of DET, Theorem 11.3 and Corollary 10.15 we are finally in a position to conclude the main theorem. Let $\mathrm{Mat}_{\mathbb{Z}}(A, n, m)$ denote the predicate stating that A is an $n \times n$ integer matrix with integer entries encoded in binary with m bits, and $\mathrm{triangMat}_{\mathbb{Z}}(A, n, m)$ means that A is a lower or upper $n \times n$ triangular matrix with integers encoded in binary with m bits, and $A[i, j]$ is the (i, j)th integer entry of A.

Theorem 11.8 (Main theorem). *The following determinant identities are provable in $\mathbf{VNC}^2$:*

$$\forall n \forall m \forall A \forall B \left(\mathrm{Mat}_{\mathbb{Z}}(A, n, m) \wedge \mathrm{Mat}_{\mathbb{Z}}(B, n, m) \rightarrow \mathrm{DET}(A) \cdot \mathrm{DET}(B) = \mathrm{DET}(AB)\right) ,$$
$$\forall n \forall m \forall A \left(\mathrm{triangMat}_{\mathbb{Z}}(A, n, m) \rightarrow \mathrm{DET}(A) = A[1, 1] \cdots A[n, n]\right) .$$

Using the translation between bounded arithmetic theories and propositional proofs as shown in [CN10] we can extend the result in [HT15] to work over the integers, and not only over $GF(2)$:

Theorem 11.9. *There are polynomial-size propositional $\mathbf{NC^2}$-Frege proofs of the determinant identities over the integers.*

In Theorem 11.9, $\mathbf{NC^2}$-Frege proofs are defined as in [HT15], namely, as 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 rules C1, C2 in $\mathbb{P}_c$ (it is also possible to characterize these proofs as restricted Extended Frege proofs). Integers in these proofs are encoded by fixed length binary strings, that is sequences of propositional variables (thus, for each different bit-length we 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 above. 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 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 $p_A(z)$ of a matrix is defined in the theory as follows: we introduce a $\mathbf{\Sigma}^B_1$-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 and n^2 output variables for each entry of the output $n \times n$ matrix, where the coefficient of z^i, for $i = 0, \ldots, n$, is computed (as a sub-circuit) by

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

namely, the balanced circuit that extracts the (constant) coefficient of z^i in the determinant polynomial of $zI_n - A$. Thus, overall the string function $p(A, n)$ outputs the following $O(\log^2 n)$-depth algebraic circuit for the characteristic polynomial of A:

$$p(A, n) := \sum_{i=0}^{n} \left[\mathrm{coeff}_{z^i}\left(\mathsf{Det}^\star_{Taylor}(zI_n - A)\right)\right] \cdot [X^i], \tag{90}$$

where $[X^i]$ stands for the $O(\log^2 n)$-depth multi-output circuit of matrix powering $[\overbrace{X \cdots X}^{i \text{ times}}]$, and the product $\cdot$ is the usual scalar product of a matrix, that is, for each i, the circuit $\left[\mathrm{coeff}_{z^i}\left(\mathsf{Det}^\star_{Taylor}(zI_n - A)\right)\right]$ multiplies each of the n^2 output gates of $[X^i]$. Denote by $p(A, n, i, j)$ the sub-circuit of $p(A, n)$ that computes the (i, j) entry of the matrix computed by $p(A, n)$.

The Cayley-Hamilton theorem is expressed in the theory as follows:

$$\forall n \forall m \forall A(\mathrm{Mat}_{\mathbb{Z}}(A, n, m) \to \forall i \leq n \forall j \leq n \mathsf{Eval}_{alg}(p(A, n, i, j), A) = \mathbf{0}). \tag{91}$$

Corollary 12.1. *The Cayley-Hamilton theorem, expressed as in (91), is provable in* $\mathbf{VNC}^2$.

Proof of Corollary 12.1. This follows the same line of arguments demonstrated for the $\mathbf{VNC}^2$-proofs of the determinant identities. We first construct the simple $\mathbb{P}_c(\mathbb{Z})$-proof of the Cayley-Hamilton theorem shown in Proposition 9.4 in [HT15] which is based on the $\mathbb{P}_c(\mathbb{Z})$-proof of the multiplication of the determinant and then use the reflection principle as in Section 11. The only difference is that we need to use part (3) in Lemma 7.3 (we did not use this part before), and for this we need to supply the witnesses for the syntactic-degrees of the nodes in (90). This is shown in Lemma 7.4. □

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

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

$$\forall n, \forall m \forall A \forall B(\mathrm{Mat}_{\mathbb{Z}}(A, n, m) \wedge \mathrm{Mat}_{\mathbb{Z}}(B, n, m) \to (AB = I_n \to BA = I_n)), \tag{92}$$

where I_n stands for the $n \times n$ identity matrix. Soltys and Cook [SC04] introduced the quantifier free theory LA that allows the basic ring properties, for example associativity of matrix additions, to be formulated and proved. LA can be interpreted in $\mathbf{VNC}^2$ by Cook and Fontes [CF12][19]. The inversion principle was showed [SC04] to be equivalent in LA to the following principles collectively called (including the inversion principle itself) *the hard matrix identities*:

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

where these identities are quantified as in (92).

The above identities are said to be "hard" in the sense that they seem to require computing inverses in their derivations, meaning that they necessitate using concepts from $\mathbf{NC^2}$ (and therefore appear not to be provable in the relatively weak theory LA). The theory LAP adds the matrix powering operator to LA. Moreover, the theory LAP (over any field) proves that the Cayley-Hamilton theorem implies the hard matrix identities [SC04]. By the results of [SC04] we know that the hard matrix identities, like the Cayley-Hamilton theorem, are provable in $\mathbf{VP}$ a theory for polynomial-time reasoning.

Our work shows the following:

Corollary 12.2. *The hard matrix identities are 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* co-factor expansion *of the determinant is provable in* $\mathbf{VNC}^2$*:*

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

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 $\mathrm{Adj}(X)$ which is defined to be the $n \times n$ matrix whose (i,j)th entry is $(-1)^{i+j}\mathrm{DET}(X[i|j])$, where DET is the determinant function ($\mathbf{\Sigma}_1^B$-defined in $\mathbf{VNC}^2$). Then we proceed as in Corollary 12.1 following [HT15, Proposition 9.1 and 9.2] that builds on the proof of the multiplication of the determinant. We omit the details.

13 Conclusions and Open Problems

We established a proof of the multiplicativity of the determinant and other basic statements of linear algebra such as the Cayley-Hamilton theorem and the co-factor expansion of the determinant in the weakest logical theory known to date (and essentially conjectured to be the weakest possible in the theories corresponding to the $\mathbf{NC}$ hierarchy). This answers an open question of Cook and Nguyen [CN10, IX.7.1. Proving Cayley-Hamilton in $\mathbf{VNC}^2$] and Soltys and Cook [SC04, p. 322,

[19] 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].

Conclusion and open problems, top paragraph] and Cook and Fontes [CF12]. We achieved this by formalizing in the theory $\mathbf{VNC}^2$ the construction demonstrated in Hrubeš-Tzameret [HT15], and using a reflection principle 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.

We showed how to carry out many constructions coming from algebraic circuit complexity in bounded arithmetic. For instance we demonstrated under which conditions one can prove in $\mathbf{NC^2}$-reasoning the soundness of line-by-line proofs of polynomial identities over the integers ($\mathbb{P}_c(\mathbb{Z})$-proofs), even when the proofs are not balanced (Corollary 11.4). This should be helpful to establish further algebraic-based identities in bounded arithmetic. More generally, since bounded arithmetic is a useful framework for the development of the meta-mathematics of computational complexity in which one seeks to understand the complexity of concepts needed to prove results in complexity theory, we may hope that our results and techniques provide new insight into the meta-mathematics of the well developed theory of *algebraic* complexity (in contrast to the traditional emphasis on Boolean complexity).

As mentioned in Section 1.1.1 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 $\mathbf{TC}^1$. We also wish to thank the anonymous reviewers of this work for greatly contributing to the improvement of the exposition. An extended abstract of this work appeared initially at LICS 2017.

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. 1.1, 2, 5.1, 10.4, 13

[All18] Eric Allender. Personal communication, 2018. 1.1, 2, 10.4

[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. 1, 13

[BKKK20] Sam Buss, Valentine Kabanets, Antonina Kolokolova, and Michal Koucky. Expander construction in vnc^1. *Annals of Pure and Applied Logic (to appear)*, 2020. Extended abstract in Proceedings, 8th Conference on Innovations in Computer Science (ITCS), 2017. 1

[BKZ15] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Konrad Zdanowski. Collapsing modular counting in bounded arithmetic and constant depth propositional proofs. *Transactions of the AMS*, (367):7517–7563, 2015. 1

[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. 1, 2

[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, 1.1, 1.1, 1.1.1, 10.2, 10.3, 10.5, 12, 19, 13

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. ASL Perspectives in Logic. Cambridge University Press, 2010. 1, 1.1, 1.1, 1.1, 1.1, 1.1.2, 2, 2.2, 2.2, 2.3, 2.4, 2.4, 2.7, 2.5, 2.10, 2.5, 2.5, 2.5, 2.6, 2.6, 3.1.3, 10.5, 11.2, 11.2, 11.3, 12, 13

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *STOC*, pages 83–97, 1975. 1

[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. 2

[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. 1.1, 1.1.2, 2, 2.8, 2.15

[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, 1.1, 1.1, 1.1.2, 2, 2.8, 2.15, 2.9, 3.1.2, 4.2, 4.3, 4.3, 6.1, 7.1, 7.3, 7.4, 7.5, 10.2, 10.2, 10.2, 10.4, 10.5, 10.5, 10.5, 11.3, 11.3, 12, 12, 13, 13

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Logic*, 129(1-3):1–37, 2004. 2.7

[Jeř05] Emil Jeřábek. *Weak pigeonhole principle, and randomized computation*. PhD thesis, PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2005. 1

[Jeř11] Emil Jeřábek. A sorting network in bounded arithmetic. *Annals of Pure and Applied Logic*, 162(4):341–355, 2011. 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. 2

[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. 1.1, 5.1, 10.1, 10.4

[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. 2.6

[NC07] Phuong Nguyen and Stephen Cook. The complexity of proving discrete jordan curve theorem. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer*, pages 245–254, 2007. 1

[Ngu08] Phuong Nguyen. *Bounded Reverse Mathematics*. PhD thesis, University of Toronto, 2008. 1

[Par71] Rohit Parikh. Existence and feasibility in arithmetic. *The Journal of Symbolic Logic*, 36:494–508, 1971. 1

[Pic15] Ján Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. *Logical Methods in Computer Science*, 11(2), 2015. 1

[PT16] Tonnian Pitassi and Iddo Tzameret. Algebraic proof complexity: Progress, frontiers and challenges. *ACM SIGLOG News*, 3(3), 2016. 2

[PW85] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in mathematical logic (Caracas, 1983)*, volume 1130 of *Lecture Notes in Math.*, pages 317–340. Springer, Berlin, 1985. 1

[RY08] Ran Raz and Amir Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008. 10.2, 10.2, 10.4

[SC04] Michael Soltys and Stephen Cook. The proof complexity of linear algebra. *Ann. Pure Appl. Logic*, 130(1-3):277–323, 2004. 1, 1.1, 12, 12, 19, 13

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. Preliminary version in the *International Symposium on Symbolic and Algebraic Computation (EUROSAM 1979)*. 1, 1.1

[Sim99] Stephen Simpson. *Subsystems of Second Order Arithmetic*. Springer, 1999. 1

[Sol01] Michael Soltys. *The complexity of derivations of matrix identities*. PhD thesis, University of Toronto, Toronto, Canada, 2001. 1, 1.1

[Str73] Volker Strassen. Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973. (in German). 1, 1.1, 1.1, 4.2, 5, 6.1, 6.2, 7.1

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. 2.7

[TS05] Neil Thapen and Michael Soltys. Weak theories of linear algebra. *Arch. Math. Log.*, 44(2):195–208, 2005. 1

[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. 1.1, 1.1, 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. 1, 1.1, 1.1, 5.1, 10.1, 10.2, 15, 10.2, 10.2, 10.4, 11.1

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226. Springer-Verlag, 1979. 1, 1.1

— Page left blank for ECCC stamp —