Formal Theories for Linear Algebra

Stephen Cook and Lila Fontes

Department of Computer Science, University of Toronto
{sacook, fontes}@cs.toronto.edu

Abstract. We introduce two-sorted theories in the style of [CN10] for the complexity classes $\oplus L$ and DET, whose complete problems include determinants over $\mathbb{Z}_2$ and $\mathbb{Z}$, respectively. We then describe interpretations of Soltys' linear algebra theory LAP over arbitrary integral domains, into each of our new theories. The result shows equivalences of standard theorems of linear algebra over $\mathbb{Z}_2$ and $\mathbb{Z}$ can be proved in the corresponding theory, but leaves open the interesting question of whether the theorems themselves can be proved.

1 Introduction

This paper introduces formal theories for the complexity classes $\oplus L$ (also called $ParityL$) and DET for reasoning about linear algebra over the rings $\mathbb{Z}_2$ and $\mathbb{Z}$, respectively. Complete problems for these classes include standard computational problems of linear algebra over their respective rings, such as computing determinants, matrix powers, and coefficients of the characteristic polynomial of a matrix [BDHM92]. (Recently [BKR09] proved that for each $k \geq 1$, computing the permanent mod 2^k of an integer matrix is in $\oplus L$, and hence complete.) Each theory allows induction over any relation in the associated complexity class $\oplus L$ or DET, and the functions definable in each theory are the functions in the class. Thus determinants and characteristic polynomials can be defined in the theories, but it is not clear that their standard properties can be proved without defining concepts outside the associated complexity classes. This remains an interesting open question [SK01,SC04].

The simplest way of defining the classes $\oplus L$ and DET is using uniform AC^0 reductions (see below). Thus $\oplus L = AC^0(det_2)$ and $DET = AC^0(det)$, where det_2 and det are the determinant functions for matrices over $\mathbb{Z}_2$ and $\mathbb{Z}$ respectively, and $AC^0(*)$ is the set of problems AC^0-reducible to $*$.

The usual definitions of these classes involve counting the number of accepting computations of nondeterministic log space Turing machines. Thus $\#L$ is the class of functions f such that for some nondeterministic log space Turing machine M, $f(x)$ is the number of accepting computations of M on input x. Then the sets in $\oplus L$ are those of the form $\{x \mid f(x) \bmod 2 = 1\}$ for some f in $\#L$. It turns out that $AC^0(det) = AC^0(\#L)$, and $AC^0(det_2) = AC^0(\oplus L) = \oplus L$. To get an idea of why det can be reduced to $\#L$, note that Berkowitz's algorithm reduces det to integer matrix powering. It is easy to see that powering $0-1$ matrices reduces to $\#L$ because the entry ij in the kth power of the adjacency

matrix for the configuration graph of a Turing machine is the number of computations of length k between configurations i and j. With a little creative thought this reduction can be generalized to binary integer matrices (see [Fon09,AO96]). Also $DET = \#LH = \bigcup_i \#LH_i$ (the $\#L$ hierarchy), where $\#LH_1 = \#L$ and $\#LH_{i+1} = \#L^{\#LH_i}$ (see [AO96]). (The exponent $\#LH_i$ indicates that a function from this class is allowed to be an oracle for the log space machine whose accepting computations are being counted.)

We should clarify that our definition of DET here differs from that in [Coo85], where DET is defined to be $NC^1(det)$, the closure of $\{det\}$ under the more general NC^1 reductions. Allender proved (see the Appendix to [All04]) that if $AC^0(det) = NC^1(det)$ then the $\#L$ hierarchy collapses to some finite level $\#LH_i$, something that is not known to be true. The present authors (now) believe that $AC^0(det) = \#LH$ is the more natural definition of DET, and makes the corresponding logical theory much easier to formulate.

The complexity classes satisfy the inclusions $L \subseteq \oplus L \subseteq DET \subseteq NC^2 \subseteq P$ and $L \subseteq NL \subseteq DET$ (ignoring the distinction between function and language classes) where L and NL are the problems accepted in deterministic and nondeterministic log space, respectively. It is not known whether $\oplus L$ and NL are comparable. (Of course we cannot disprove the unlikely possibility that all of the above classes could coincide.)

To construct formal theories for the classes $\oplus L$ and DET we follow the framework laid out in Chapter 9 of the monograph [CN10] of Cook and Nguyen for defining theories for complexity classes between AC^0 and P. All of these theories share a common two-sorted (number and string) vocabulary $\mathcal{L}_A^2$. The intention is that the number sort ranges over $\mathbb{N}$ and the string sort ranges over bit strings (more precisely, finite subsets of $\mathbb{N}$). The strings are intended to be inputs to the machine or circuit defining a member of the complexity class, and the numbers are used to index bits in the strings. Each theory VC for a class C extends the finitely-axiomatized base theory V^0 for AC^0 by addition of a single axiom stating the existence of a solution to a complete problem for C. General techniques are presented for defining a universally-axiomatized conservative extension $\overline{VC}$ of VC which has function symbols and defining axioms for each function in FC, and $\overline{VC}$ admits induction on open formulas in this enriched vocabulary. It follows from the Herbrand Theorem that the provably-total functions in $\overline{VC}$ (and hence in VC) are precisely the functions in FC.

Chapter 9 (with earlier chapters) of [CN10] explicitly defines theories for the following classes:

$$AC^0 \subset AC^0(2) \subset TC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq NC \subseteq P \tag{1}$$

These classes are defined briefly as follows. A problem in AC^0 is solved by a uniform family of polynomial size constant depth Boolean circuits with unbounded fanin AND and OR gates. $AC^0(2)$ properly extends AC^0 by also allowing unbounded fanin parity gates (determining whether the inputs have an odd number of 1's) in its circuits. TC^0 allows majority gates rather than parity gates in its circuits (and has binary integer multiplication as a complete problem). NC^1 circuits restrict all Boolean gates to fanin two, but the circuits are allowed to have

logarithmic depth. Problems in L and NL are solved respectively by deterministic and nondeterministic log space Turing machines. NC is defined like NC^1, but the circuits can have polylogarithmic depth (and polynomial size). Problems in P are solved by polynomial time Turing machines.

Our new theories $V{\oplus}L$ and $V\#L$ for $\oplus L$ and DET extend the base theory V^0 for AC^0 by adding an axiom stating the existence of powers A^k of matrices A over $\mathbb{Z}_2$ and $\mathbb{Z}$, respectively, where k is presented in unary. (Matrix powering is a complete problem for these classes [Fon09].) Here there is a technical difficulty of how to nicely state these axioms, since neither the parity function (needed to define matrix multiplication over $\mathbb{Z}_2$) nor integer product and multiple summation (needed to define matrix multiplication over $\mathbb{Z}$) is definable in the base theory V^0. We solve this by basing these theories on the theories $V^0(2)$ (for $AC^0(2)$) and VTC^0, and by using results from [CN10] to translate these axioms to the language of the base theory V^0. We then show that the resulting theories satisfy the requirements of Chapter 9 (existence of "aggregate functions") that allow the existence of the nice universal conservative extensions $\overline{V{\oplus}L}$ and $\overline{V\#L}$ of $V{\oplus}L$ and $V\#L$.

The new theories mesh nicely with the theories for the complexity classes in (1). In particular, we have

$$V^0 \subset V^0(2) \subset VNC^1 \subseteq VL \subseteq V{\oplus}L \subseteq V\#L \subseteq VNC \subseteq VP \tag{2}$$

Next we study the question of which results from linear algebra can be proved in the theories. For this we take advantage of Soltys's theory LAP [SK01,SC04] for formalizing results from linear algebra over an arbitrary field or integral domain. We present two interpretations of LAP: one into $V{\oplus}L$ and one into $V\#L$. Both interpretations translate theorems of LAP to theorems in the corresponding theory, but the translations can alter the meaning of formulas by giving different interpretations of the ring elements.

LAP has three sorts: One for indices (natural numbers), one for field (or ring) elements, and one for matrices. When interpreting LAP into $V{\oplus}L$ our intention is that the field is $\mathbb{Z}_2$, so we interpret field elements as formulas, where true formulas represent 1 and false formulas represent 0. When interpreting LAP into $V\#L$ our intention is that the ring is $\mathbb{Z}$. In this case we interpret field elements as strings representing binary integers.

LAP defines matrix powering, and uses this and Berkowitz's algorithm [Ber84] to define several functions of matrices, including determinant, adjoint, and characteristic polynomial. The following standard principles of linear algebra are discussed:

(i) The Cayley-Hamilton Theorem (a matrix satisfies its characteristic polynomial).
(ii) The axiomatic definition of determinant (the function $det(A)$ is characterized by the properties that it is multilinear and alternating in the rows and columns of A, and $det(I) = 1$).
(iii) The co-factor expansion of the determinant.

Although it remains open whether LAP can prove any of these, a major result from [SK01,SC04] is that LAP proves their pairwise equivalence. As a result of this and our interpretations we have the following.

Theorem 1. *$V\oplus L$ proves the equivalence of (i), (ii), and (iii) over the ring $\mathbb{Z}_2$, and $V\#L$ proves their equivalence over $\mathbb{Z}$.*

An intriguing possibility (not yet realized) is that either $V\oplus L$ or $V\#L$ could use special properties of $\mathbb{Z}_2$ or $\mathbb{Z}$ to prove its version of the principles, while still leaving open whether LAP can prove them (for all integral domains or fields). For example there is a dynamic programming algorithm involving combinatorial graph properties (see the concluding Section 4) whose correctness for $\mathbb{Z}$ might be provable in $V\#L$ using combinatorial reasoning with concepts from $\#L$ which are not available in LAP.

[SK01,SC04] also present the so-called *hard matrix identities*, each equivalent to the implication

$$AB = I \rightarrow BA = I \tag{3}$$

where A, B are square matrices. Again it is open whether LAP proves these identities, but LAP does prove that they follow from any of the principles mentioned in Theorem 1 above. The next result follows from this and our interpretations.

Theorem 2. *$V\oplus L$ proves that (3) over the ring $\mathbb{Z}_2$ follows from any of the three principles mentioned in Theorem 1. The same is true for $V\#L$ over the ring $\mathbb{Z}$.*

[SK01,SC04] introduce an extension $\forall$LAP of LAP, which includes an induction rule that applies to formulas with bounded universally quantified matrix variables, and shows that the three principles mentioned in Theorem 1 and the four matrix identities are all provable in $\forall$LAP. The key idea in this proof uses induction on a polynomial time relation, and in fact this proof can be carried out in the theory VP for polynomial time defined in [CN10]. Since VP (see formula (2)) extends both $V\oplus L$ and $V\#L$ we have the following result (alluded to at the end of Section 6 in [SC04]).

Theorem 3. *The theory VP proves the three principles (i), (ii), (iii) and the matrix identity (3) for both the rings $\mathbb{Z}_2$ and $\mathbb{Z}$.*

2 Theories $V\oplus L$ and $V\#L$

We start by reviewing the two-sorted logic used here and in [CN10]. We have *number* variables $x, y, z, \ldots$ whose intended values are numbers (in $\mathbb{N}$), and *string* variables $X, Y, Z, \ldots$ whose intended values are finite sets of numbers. We think of the finite sets as binary strings giving the characteristic vectors of the sets. For example the string corresponding to the set $\{0, 3, 4\}$ is 10011.

All our two-sorted theories include the basic vocabulary $\mathcal{L}_A^2 = [0, 1, +, \cdot, |\ |; \in, \leq, =_1, =_2]$ which extends the first-order vocabulary of Peano Arithmetic. The symbols $0, 1, +, \cdot$ are intended to take their usual meanings on $\mathbb{N}$. Here $|\ |$ is a

function from strings to numbers, and the intended meaning of $|X|$ is 1 plus the largest element of X, or 0 if X is empty. (If $X = \{0, 3, 4\}$ then $|X| = 5$.) The binary predicate $\in$ is intended to denote set membership. We often write $X(t)$ for $t \in X$ (think bit number t of the string X is 1). The equality predicates $=_1$ and $=_2$ are for numbers and strings, respectively. We will write $=$ for both, since the missing subscript will be clear from the context.

Number terms (such as $x + ((|X| + 1) \cdot |Y|)$) are built from variables and function symbols as usual. The only string terms based on $\mathcal{L}_A^2$ are string variables $X, Y, Z, \ldots$, but when we extend $\mathcal{L}_A^2$ by adding string-valued functions, other string terms will be built as usual. Formulas are built from atomic formulas (e.g. $t = u, t \leq u, X(t), X = Y$) using $\wedge, \vee, \neg$ and $\exists x, \forall x, \exists X, \forall X$.

Bounded quantifiers are defined as usual, except bounds on string quantifiers refer to the length of the string. For example $\exists X \leq t\varphi$ stands for $\exists X(|X| \leq t \wedge \varphi)$.

We define two important syntactic classes of formulas.

Definition 1. *Σ_0^B is the class of $\mathcal{L}_A^2$ formulas with no string quantifiers, and only bounded number quantifiers. Σ_1^B formulas are those of the form $\exists \boldsymbol{X} \leq \boldsymbol{t}\varphi$, where φ is in Σ_0^B and the prefix of bounded quantifiers may be empty.*

Notice our nonstandard requirement that the string quantifiers in Σ_1^B formulas must be in front.

We also consider two-sorted vocabularies $\mathcal{L} \supseteq \mathcal{L}_A^2$ which extend $\mathcal{L}_A^2$ by possibly adding predicate symbols $P, Q, R, \ldots$ and function symbols $f, g, h, \ldots$ and $F, G, H, \ldots$. Here $f, g, h, \ldots$ are *number functions* and are intended to take values in $\mathbb{N}$, and $F, G, H, \ldots$ are *string functions* and are intended to take string values. Each predicate or function symbol has a specified arity (n, m) indicating that it takes n number arguments and m string arguments. Number arguments are written before string arguments, as in

$$f(x_1, \ldots, x_n, X_1, \ldots, X_m) \qquad F(x_1, \ldots, x_n, X_1, \ldots, X_m) \tag{4}$$

The formula classes $\Sigma_0^B(\mathcal{L})$ and $\Sigma_1^B(\mathcal{L})$ are defined in the same way as Σ_0^B and Σ_1^B, but allow function and relation symbols from $\mathcal{L}$ in addition to $\mathcal{L}_A^2$.

2.1 Two-sorted complexity classes

In standard complexity theory an element of a complexity class is either a set of binary strings or a function $f : \{0,1\}^* \to \{0,1\}^*$. In our two-sorted point of view (Chapter 4 of [CN10]) it is convenient to replace a set of strings by a relation $P(\boldsymbol{x}, \boldsymbol{X})$ of any arity (n, m), and functions are generalized to allow both number functions and string functions as in (4). Each standard complexity class, including those in (1) and $\oplus L$ and $\#L$, is defined either in terms of Turing machines or circuit families. These definitions naturally extend to two-sorted versions by representing strings (as inputs to machines or circuits) in a straightforward way as binary strings, but by representing numbers using unary notation. This interpretation of numbers is a convenience, and is justified by our intention that numbers are 'small' and are used to index strings and measure their length. In particular, we have the following definition of two-sorted $\oplus L$.

Definition 2. *$\oplus L$ is the set of relations $P(\boldsymbol{x}, \boldsymbol{X})$ such that there is a nondeterministic log space Turing machine M such that M with input $\boldsymbol{x}, \boldsymbol{X}$ (represented as above) has an odd number of accepting computations iff $P(\boldsymbol{x}, \boldsymbol{X})$ holds.*

The class AC^0 can be defined in terms of uniform polynomial size constant depth circuit families, but it has a nice characterization as those sets recognized by an alternating Turing machine (ATM) in log time with a constant number of alternations. More useful for us, [Imm99] showed that an element of AC^0 can be described as an element of FO, namely the set of finite models of some first-order formula with a certain vocabulary. From this and the ATM definition of two-sorted AC^0, we have the following important results relating syntax and semantics (Theorems 4.18 and 4.19 of [CN10]).

Proposition 1 (Representation Theorems). *A relation $P(\boldsymbol{x}, \boldsymbol{X})$ is in AC^0 (respectively NP) iff it is represented by some Σ_0^B-formula (respectively Σ_1^B-formula) $\varphi(\boldsymbol{x}, \boldsymbol{X})$.*

For example the relation $PAL(X)$ (X is a palindrome) is an AC^0 relation because the Σ_0^B-formula $\forall x, y < |X|(x + y + 1 = |X| \supset (X(x) \leftrightarrow X(y)))$ represents it.

A number function $f(\boldsymbol{x}, \boldsymbol{X})$ (respectively string function $F(\boldsymbol{x}, \boldsymbol{X})$) is p-bounded if there is a polynomial $g(\boldsymbol{x}, \boldsymbol{y})$ such that $f(\boldsymbol{x}, \boldsymbol{X}) \leq g(\boldsymbol{x}, |\boldsymbol{X}|)$ (respectively $|F(\boldsymbol{x}, \boldsymbol{X})| \leq g(\boldsymbol{x}, |\boldsymbol{X}|)$). The *bit graph* of a string function F is the relation B_F defined by $B_F(i, \boldsymbol{x}, \boldsymbol{X}) \leftrightarrow F(\boldsymbol{x}, \boldsymbol{X})(i)$. (Recall that $Y(i)$ stands for $i \in Y$.)

Definition 3. *If C is a class of (two-sorted) relations then FC denotes the corresponding class of functions, where f (respectively F) is in FC iff it is p-bounded and its graph (respectively bit graph) is in C.*

We will consider two-sorted vocabularies $\mathcal{L}$ which extend $\mathcal{L}_A^2$, and in all cases each function and relation symbol in $\mathcal{L}$ has an intended interpretation in our standard two-sorted model (the two universes being $\mathbb{N}$ and the set of finite subsets of $\mathbb{N}$). Thus we can make sense of both syntactic and semantic statements about $\mathcal{L}$.

If $\mathcal{L}$ is a two-sorted vocabulary, then f (respectively F) is Σ_0^B-definable from $\mathcal{L}$ if it is p-bounded and its graph (respectively bit graph) is represented by a formula in $\Sigma_0^B(\mathcal{L})$.

$AC^0(\mathcal{L})$ (the AC^0 closure of $\mathcal{L}$) denotes the closure of $\mathcal{L}$ under Σ_0^B definability. To show a function is in $AC^0(\mathcal{L})$ requires giving a finite sequence of functions such that each is Σ_0^B-definable from the preceding ones. The relations in $AC^0(\mathcal{L})$ are those whose characteristic functions are in $AC^0(\mathcal{L})$.

2.2 Theories V^0, $V^0(2)$, and VTC^0

The theory V^0 for AC^0 is the basis for every two-sorted theory considered here and in [CN10]. It has the two-sorted vocabulary $\mathcal{L}_A^2$, and is axiomatized by the

set 2-BASIC of axioms consisting of 15 Σ_0^B formulas expressing basic properties of the symbols of $\mathcal{L}_A^2$, together with the Σ_0^B comprehension scheme

$$\Sigma_0^B\textbf{-COMP}: \quad \exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z))$$

where $\varphi(z)$ is any Σ_0^B formula with no free occurrence of X.

V^0 has no explicit induction axiom, but nevertheless the induction scheme

$$\Sigma_0^B\textbf{-IND}: \quad \big(\varphi(0) \wedge \forall x(\varphi(x) \supset \varphi(x+1))\big) \supset \forall z \varphi(z)$$

for Σ_0^B formulas $\varphi(x)$ is provable in V^0, using Σ_0^B-**COMP** and the fact that $|X|$ produces the maximum element of the set X.

Definition 4. *A string function $F(\boldsymbol{x}, \boldsymbol{X})$ is Σ_1^B-definable in a two-sorted theory $\mathcal{T}$ if there is a Σ_1^B formula $\varphi(\boldsymbol{x}, \boldsymbol{X}, Y)$ representing the graph $Y = F(\boldsymbol{x}, \boldsymbol{X})$ of F such that $\mathcal{T} \vdash \forall \boldsymbol{x} \forall \boldsymbol{X} \exists! Y \varphi(\boldsymbol{x}, \boldsymbol{X}, Y)$. Similarly for a number function $f(\boldsymbol{x}, \boldsymbol{X})$.*

It is shown in Chapter 5 of [CN10] that V^0 is finitely axiomatizable, and the Σ_1^B-definable functions in V^0 comprise the class FAC^0 (see Definition 3).

The definition in [CN10] of the theory $V^0(2)$ for the class $AC^0(2)$ is based on V^0 and an axiom showing the definability of the function $Parity(x, Y)$. If $Z = Parity(x, Y)$ then $Z(z)$ holds iff $1 \leq z \leq x$ and there is an odd number of ones in $Y(0)Y(1)\ldots Y(z \dot{-} 1)$. The graph of $Parity$ is defined by the Σ_0^B formula $\delta_{parity}(x, Y, Z)$, which is $\neg Z(0) \wedge \forall z < x(Z(z+1) \leftrightarrow (Z(z) \oplus Y(z)))$.

Definition 5. *[CN10] The theory $V^0(2)$ has vocabulary $\mathcal{L}_A^2$ and axioms those of V^0 and $\exists Z \leq x+1\ \delta_{parity}(x, Y, Z)$.*

The complexity class $FAC^0(2)$ is the AC^0 closure of the function $Parity(x, Y)$, and in fact the Σ_1^B-definable functions of $V^0(2)$ are precisely those in $FAC^0(2)$.

The theory VTC^0 for the counting class TC^0 is defined similarly to $V^0(2)$, but now the function $Parity(x, Y)$ is replaced by the function $numones(y, X)$, whose value is the number of elements (i.e. 'ones') of X that are less than y. The axiom for VTC^0 is based on a Σ_0^B formula $\delta_{NUM}(y, X, Z)$ defining the graph of a string function accumulating the values of $numones(y, X)$ as y increases.

The definition of $\delta_{NUM}(y, X, Z)$ uses the pairing function $\langle x, y\rangle$, which is the $\mathcal{L}_A^2$ term $(x+y)(x+y+1)+2y$.

Definition 6. *[CN10] The theory VTC^0 has vocabulary $\mathcal{L}_A^2$ and axioms those of V^0 and $\exists Z \leq 1 + \langle y, y\rangle \delta_{NUM}(y, X, Z)$.*

The class FTC^0 is the AC^0 closure of the function $numones$, and in fact the Σ_1^B-definable functions of VTC^0 are precisely those in FTC^0.

In Chapters 5 and 9 of [CN10] it is shown that the theories $V^0, V^0(2), VTC^0$ have respective universally axiomatized conservative extensions $\overline{V^0}, \overline{V^0(2)}, \overline{VTC^0}$ obtained by introducing function symbols and their defining axioms for all string functions (and some number functions) in the corresponding complexity class. These have the following properties.

Proposition 2. *Let $(FC, V, \overline{V})$ be any of the triples $(FAC^0, V^0, \overline{V^0})$ or $(FAC^0(2), V^0(2), \overline{V^0(2)})$ or $(FTC^0, VTC^0, \overline{VTC^0})$, and let $\mathcal{L}$ be the vocabulary of $\overline{V}$. Then (i) $\overline{V}$ is a universally axiomatized conservative extension of V, (ii) the Σ^B_1-definable functions of both V and $\overline{V}$ are those in FC, (iii) a string function (respectively number function) is in FC iff it has a function symbol (respectively term) in $\mathcal{L}$, (iv) $\overline{V}$ proves the $\Sigma^B_0(\mathcal{L})$-**IND** and $\Sigma^B_0(\mathcal{L})$-**COMP** schemes, and (v) for every $\Sigma^B_1(\mathcal{L})$ formula φ^+ there is a Σ^B_1 formula φ such that $\overline{V} \vdash \varphi^+ \leftrightarrow \varphi$.*

2.3 New Theory $V \oplus L$

Recall (two-sorted) $\oplus L$ is given in Definition 2. It follows from results in [BDHM92] (see [Fon09]) that $\oplus L$ consists of the relations in the AC^0 closure of matrix powering over $\mathbb{Z}_2$. The theory $V \oplus L$ extends $V^0(2)$ by adding a Σ^B_1 axiom for matrix powering over $\mathbb{Z}_2$.

Recall that the theory $\overline{V^0(2)}$ is a conservative extension of $V^0(2)$, and its vocabulary $\mathcal{L}_{FAC^0(2)}$ has function symbols or terms for every function in $FAC^0(2)$. We describe the Σ^B_1 axiom for matrix powering as a $\Sigma^B_1(\mathcal{L}_{FAC^0(2)})$ formula and refer to part (v) of Proposition 2 to conclude that this formula is provably equivalent to a $\Sigma^B_1(\mathcal{L}^2_A)$ formula.

Hence we will freely use FAC^0 functions and the function $Parity(x, Y)$ (used to define the theory $V^0(2)$) when describing formulas which will help express the Σ^B_1 axiom. In particular we will use the pairing function $\langle x, y\rangle$ and its inverses $left(z)$ and $right(z)$. Since the pairing function is not surjective, we use $Pair(z)$ to abbreviate the formula $z = \langle left(z), right(z)\rangle$ which asserts that z codes a pair.

We use the truth values $\{false, true\}$ to represent the elements $\{0, 1\}$ of $\mathbb{Z}_2$, and we represent a matrix over $\mathbb{Z}_2$ with a string X. Here $X(i, j)$ abbreviates $X(\langle i, j\rangle)$ and refers to the entry ij of the matrix. We number rows and columns starting with 0, so if X is an $n \times n$ matrix then $0 \le i, j < n$.

The function $Row(i, X)$ (written $X^{[i]}$) refers to row i of matrix X, and is defined by its bit graph $X^{[i]}(b) \leftrightarrow b < |X| \wedge X(i, b)$.

The string function $ID(n)$ codes the $n \times n$ identity matrix, and has a bit graph axiom $ID(n)(b) \leftrightarrow left(b) < n \wedge Pair(\mathrm{b}) \wedge left(\mathrm{b}) = right(\mathrm{b})$.

In order to define matrix product we start by defining the AC^0 string function $G(n, i, j, X, Y)$ equal to the string of pairwise bit products of row i of X and column j of Y. The bit graph axiom is $G(n, i, j, X, Y)(b) \leftrightarrow b < n \wedge X(i, b) \wedge Y(b, j)$.

Let $PAR(\mathrm{X})$ stand for the formula $Parity(|X|, X)(|X|)$. Thus $PAR(\mathrm{X})$ holds iff X has an odd number of ones. Let $Prod_2(n, X, Y)$ be the product of X and Y treated as $n \times n$ matrices over $\mathbb{Z}_2$. The bit graph axiom is

$$\begin{aligned} &Prod_2(n, X, Y)(b) \leftrightarrow \\ &Pair(b) \wedge left(b) < n \wedge right(b) < n \wedge PAR(G(n, left(b), right(b), X, Y)) \end{aligned} \tag{5}$$

Now we want to define the function $PowSeq_2(n, k, X)$ whose value is a string Y coding the sequence $I, X, X^2, \ldots, X^k$ of powers of the $n \times n$ matrix X. (Note

that this function is complete for $F\oplus L$.) We do this by giving its graph, which is defined by the following $\Sigma^B_0(\mathcal{L}_{FAC^0(2)})$ formula $\delta_{PowSeq_2}(n,k,X,Y)$:

$$\begin{aligned} &Y^{[0]} = ID(n) \wedge \forall i < k(Y^{[i+1]} = Prod_2(n, X, Y^{[i]}) \\ &\wedge \forall b < |Y|\big(Y(b) \supset (Pair(b) \wedge left(b) < n)\big) \end{aligned}$$

(The second line ensures that Y is uniquely defined.) Note that δ_{PowSeq_2} involves the function $Prod_2$ and hence is not equivalent to a Σ^B_0 formula, but by part (v) of Proposition 2, $\overline{V^0(2)}$ proves it is equivalent to a Σ^B_1 formula δ'_{PowSeq_2}.

Definition 7. *The theory $V\oplus L$ has vocabulary $\mathcal{L}^2_A$ and axioms those of $V^0(2)$ and the Σ^B_1 formula $\exists Y \leq 1 + \langle k, \langle n, n\rangle\rangle \delta'_{PowSeq_2}(n,k,X,Y)$.*

Note that the function $PowSeq_2$ is Σ^B_1 definable in $V\oplus L$. Since $\overline{V^0(2)}$ is a conservative extension of $V^0(2)$, it follows that the theory

$$\mathcal{T} = V\oplus L + \overline{V^0(2)} \tag{6}$$

is a conservative extension of $V\oplus L$, and this allows us to reason in $\mathcal{T}$ to make inferences about the power of $V\oplus L$.

Section 9B in [CN10] presents a general method for defining a universal conservative extension $\overline{VC}$ (satisfying the properties of Proposition 2) of a theory VC over $\mathcal{L}^2_A$, where VC is defined in a manner similar to $V^0(2)$ and VTC^0; namely by adding an axiom to V^0 stating the existence of a complete function for the complexity class C. Although our new theory $V\oplus L$ could be construed to fit this pattern, for the purpose of defining $\overline{V\oplus L}$ it is more naturally construed to be the result of adding the modified axiom of Definition 7 where δ' is replaced by the original formula δ, and the base theory is $\overline{V^0(2)}$ rather than V^0. Note that the resulting theory is the same as the conservative extension $\mathcal{T}$ (6) of $V\oplus L$ mentioned above. It turns out that the development in Section 9B easily generalizes to the case in which the base theory is an extension V of V^0 rather than just V^0, and the construction of $\overline{VC}$ works so that Proposition 2 holds, where now the complexity class C is the AC^0 closure of $\{\mathcal{L}, F\}$, where $\mathcal{L}$ is the vocabulary of V and F is the function whose existence follows from the axiom. In the present case, this allows us to define $\overline{V\oplus L}$ satisfying Proposition 2, where now the complexity class C is the AC^0 closure of $\{Parity, PowSeq_2\}$, which is same as the AC^0 closure of $\{PowSeq_2\}$, namely $\oplus L$.

There is a technical requirement for the new function F introduced in the above development in Section 9B, namely that the so-called aggregate F^* of F must be definable and its properties provable in the new theory VC. The aggregate satisfies $\forall i < b(F^*(X))^{[i]} = F(X^{[i]})$ in the simple case that F has a single argument X. In the case of $PowSeq_2$, the aggregate ${PowSeq_2}^*$ takes a sequence of matrices of various sizes as input and outputs a sequence of sequences of powers of the matrices. See [Fon09] to see how to define ${PowSeq_2}^*$ in $V\oplus L$.

2.4 New Theory $V\#L$

As explained in the introduction, our theory $V\#L$ is associated with the class DET, which is the AC^0 closure of *det* (determinant of integer matrices). It is interesting that integer matrix powering is complete for DET, even when restricted to 0-1 matrices, and yet it is still in DET when integers are represented in binary (see [Fon09]). Here we assume integers are represented in binary, and the axiom for $V\#L$ asserts integer matrix powers exist.

To define integer matrix product we must define integer multiplication and iterated integer summation. Both of these problems are complete for the complexity class TC^0, so we define $V\#L$ as an extension of the theory VTC^0 (Section 9C of [CN10]). We work in the conservative extension $\overline{VTC^0}$ of VTC^0. Functions for multiplication and iterated sum over $\mathbb{N}$ are defined in [CN10], and it is not hard to modify them to work for $\mathbb{Z}$. Integers are represented by strings, and matrices by arrays of strings. The entry ij of matrix X is the string $X^{[i][j]}$ obtained by two applications of the function *Row*. Now integer matrix product is the TC^0 function $Prod_{\mathbb{Z}}(n, X, Y)$ defined analogously to formula (5) for $Prod_2$, and the graph of iterated matrix product is defined by the $\Sigma_0^B(\mathcal{L}_{FTC^0})$ formula $\delta_{PowSeq_{\mathbb{Z}}}(n, k, X, Y)$ analogous to δ_{PowSeq_2}. By Theorem 2 $\overline{VTC^0}$ proves $\delta_{PowSeq_{\mathbb{Z}}}$ is equivalent to a Σ_1^B formula $\delta'_{PowSeq_{\mathbb{Z}}}$, which we use in an axiom for $V\#L$.

Definition 8. *The theory $V\#L$ has vocabulary $\mathcal{L}_A^2$ and axioms those of VTC^0 and the Σ_1^B formula $\exists Y \leq t\, \delta'_{PowSeq_{\mathbb{Z}}}(n, k, X, Y)$ for a suitable term t.*

The methods explained after Definition 7 can be used to construct the universal conservative extension $\overline{V\#L}$ satisfying the conditions of Proposition 2 (see [Fon09]).

The following summarizes properties of our new theories.

Theorem 4. *Proposition 2 holds when $(FC, V, \overline{V})$ is either of the triples $(F\oplus L, V\oplus L, \overline{V\oplus L})$ or $(FDET, V\#L, \overline{V\#L})$.*

3 Interpreting LAP

Soltys' theory LAP [SK01,SC04] (Linear Algebra with Powering) is a three-sorted quantifier-free theory based on Gentzen style sequents. The three sorts are indices i, j, k (intended to range over $\mathbb{N}$), field elements a, b, c (intended to range over some fixed field or integral domain $\mathbb{F}$), and matrices A, B, C (intended to range over matrices with entries in $\mathbb{F}$). The vocabulary of LAP has symbols $0, 1, +, -, *, \mathrm{div}, \mathrm{rem}$ (each with a subscript 'index') for indices, and symbols $0, 1, +, -, *, ^{-1}, \mathrm{r}, \mathrm{c}, \mathrm{e}, \sum$ (each with a subscript 'field') for field elements, and relations $\leq_{\mathrm{index}}, =_{\mathrm{index}}, =_{\mathrm{field}}, =_{\mathrm{matrix}}$, and functions $\mathrm{cond}_{\mathrm{index}}, \mathrm{cond}_{\mathrm{field}}, \mathrm{p}$.

The intended interpretations of $0, 1, +, *, ^{-1}$, and $-_{\mathrm{field}}$ are obvious. The other intended interpretations are as follows. The symbol $-_{\mathrm{index}}$ is cutoff subtraction; $\mathrm{div(i, j)}$ and $\mathrm{rem(i, j)}$ are the quotient and remainder functions; $\mathrm{r}(A)$ and $\mathrm{c}(A)$ return the number of rows and columns in A; $\mathrm{e}(A, i, j)$ is the $(i, j)^{\mathrm{th}}$ entry of matrix A; $\sum(A)$ is the sum of all the entries of A; and for α a formula, $\mathrm{cond}_{\mathrm{index}}(\alpha, i, j)$

is i if α is true and j otherwise (similarly for $\mathrm{cond}_{\mathrm{field}}(\alpha, a, b)$). The function $\mathrm{p}(n, A) = A^n$.

Terms and quantifier-free formulas are mostly constructed in the usual way, respecting types. We use n, m for index terms, t, u for field terms, T, U for matrix terms, and α, β for (quantifier-free) formulas. The four kinds of atomic formulas are $m \leq_{\mathrm{index}} n$, $m =_{\mathrm{index}} n$, $t =_{\mathrm{field}} u$, and $T =_{\mathrm{matrix}} U$. Formulas are built from atomic formulas using $\wedge, \vee, \neg$. There are restrictions on terms beginning with cond: If α is a formula with atomic subformulas all of the form $m \leq_{\mathrm{index}} n$ and $m =_{\mathrm{index}} n$, then $\mathrm{cond}_{\mathrm{index}}(\alpha, m', n')$ is a term of type index and $\mathrm{cond}_{\mathrm{field}}(\alpha, t, u)$ is a term of type field.

Terms of type matrix also include the constructed term $\lambda_{ij}\langle m, n, t\rangle$ (with the restriction that i and j are not free in m and n). It defines an $m \times n$ matrix with $(i, j)^{\mathrm{th}}$ entry given by $t(i, j)$. Many matrix functions such as multiplication, addition, transpose, can be defined using λ terms, avoiding the need for separately defined function symbols. For example $A^t = \lambda_{ij}\langle \mathrm{c}(A), \mathrm{r}(A), e(A, j, i)\rangle$ defines the transpose of A .

Lines in an LAP proof are Gentzen-style sequents $\alpha_1, \ldots, \alpha_k \rightarrow \beta_1, \ldots, \beta_\ell$ with the usual meaning $\bigwedge \alpha_i \supset \bigvee \beta_j$. The logical axioms and rules are those of Gentzen's system LK (minus the quantifier rules). The nonlogical axioms are numbered **A1** through **A36**. There are two nonlogical rules: one for induction (9) and one for matrix equality (10).

3.1 Interpreting LAP into $V\oplus L$

Here we interpret the field $\mathbb{F}$ in the semantics of LAP to be $\mathbb{Z}_2$. We will describe the interpretation so that each formula α of LAP is translated into a formula α^σ of $\overline{V\oplus L}$. Here α^σ is in $\Sigma_0^B(\mathcal{L}_{F\oplus L})$, so by Theorem 4 and part (v) of Proposition 2, α^σ is equivalent to a Σ_1^B formula $(\alpha^\sigma)'$ of $V\oplus L$. The translation preserves provability (sequent theorems are translated to sequent theorems) and it also preserves truth in our intended standard models: $(\mathbb{N}, \mathbb{Z}_2, \text{matrices over } \mathbb{Z}_2)$ for LAP and $(\mathbb{N}, \text{finite subsets of } \mathbb{N})$ for $V\oplus L$.

Each index term m (resp. matrix term T) of LAP is translated to a number term m^σ (resp. string term T^σ) of $\overline{V\oplus L}$. Since we represent elements of $\mathbb{Z}_2$ by Boolean values in $V\oplus L$, each field term t is translated to a $\Sigma_0^B(\mathcal{L}_{F\oplus L})$ formula t^σ. These translations are described in detail in [Fon].

The translation of terms involving matrices is complicated. Every matrix of LAP has three attributes: number of rows, number of columns, and matrix entries (field elements). Each matrix term is translated to a string term which codes all of these. Thus an $a \times b$ matrix A is interpreted as a string A^σ such that $A^\sigma(0, \langle a, b\rangle)$ is true, and for all i, j with $1 \leq i \leq a$ and $1 \leq j \leq b$ and $e(A, i, j) = A_{ij} = 1$, the bit $A^\sigma(i, j)$ is true. All other bits of A^σ are false.

We will use a Σ_0^B formula $\mathit{isMatrix}_2(X)$, which asserts that the string X properly encodes a matrix as above. We allow the number of rows and/or columns to be 0, but any entry out of bounds is 0 (a false bit).

The $\overline{V\oplus L}$ functions $f_r(X)$ and $f_c(X)$ extract the number of rows and columns of the matrix coded by X, and are used to translate the LAP terms $\mathrm{r}(T)$ and

c(T). These have defining equations

$$f_r(X) = z \leftrightarrow (\neg \mathit{isMatrix}_2(X) \wedge z = 0) \vee \big(\mathit{isMatrix}_2(X) \wedge \exists y \leq |X|\, X(0, \langle z, y\rangle)\big)$$
$$f_c(X) = z \leftrightarrow (\neg \mathit{isMatrix}_2(X) \wedge z = 0) \vee \big(\mathit{isMatrix}_2(X) \wedge \exists y \leq |X|\, X(0, \langle y, z\rangle)\big)$$

The matrix term $\lambda_{ij}\langle m, n, t\rangle$ is interpreted by the $\overline{V \oplus L}$ term $F_{t^\sigma}(m^\sigma, n^\sigma)$. Here $F_{t^\sigma}(x, y)$ is a string function, which has additional arguments corresponding to any free variables in t^σ other than the distinguished variables i, j (we interpret $i^\sigma = i$ and $j^\sigma = j$). The bit defining formula for F_{t^σ} is

$$F_{t^\sigma}(x, y)(b) \leftrightarrow b = \langle 0, \langle x, y\rangle\rangle \vee \exists i \leq x \exists j \leq y (i > 0 \wedge j > 0 \wedge b = \langle i, j\rangle \wedge t^\sigma(i, j)) \tag{7}$$

where we have written $t^\sigma(i, j)$ to display the distinguished variables i, j. Then $\mathit{isMatrix}_2(F_{t^\sigma}(m^\sigma, n^\sigma))$ is always true.

A matrix power term p(m, T) is translated $F_p(m^\sigma, T^\sigma)$ where $F_p(i, X)$ is a suitable version of a matrix powering function in $\overline{V \oplus L}$. It is related to the straightforward function $\mathit{Pow}_2(n, i, X) = X^i$ defined from PowSeq_2 in the defining axiom for $V \oplus L$, but complicated by the fact that the string A^σ translating an LAP matrix A codes row and column numbers along with matrix entries. See [Fon] for the definition of $F_p(i, X)$.

Atomic formulas involving $=$ and $\leq$ are translated in the obvious way, except $T =_{\text{matrix}} U$ translates into the formula

$$(\mathrm{r}(T) = \mathrm{r}(U))^\sigma \wedge (\mathrm{c}(T) = \mathrm{c}(U))^\sigma \wedge \forall i, j \leq (|T^\sigma| + |U^\sigma|)\ (\mathrm{e}(i, j, T) = \mathrm{e}(i, j, U))^\sigma \tag{8}$$

which asserts that the row number, column number, and entries of T and U are the same.

For general formulas, the connectives $\wedge, \vee, \neg$ translate to themselves, and sequents translate to formulas in the usual way.

It remains to show that theorems of LAP translate into theorems of $\overline{V \oplus L}$ (and hence further translate into theorems of $V \oplus L$). The underlying logic used by LAP is Gentzen's propositional sequent system, in which all axioms are valid sequents and for each rule, the bottom sequent is a logical consequence of the top sequents. Since formulas of LAP are quantifier free, and the connectives $\wedge, \vee, \neg$ translate to themselves, it follows that logical axioms translate to valid formulas, and the translated rules preserve logical consequence.

In addition to the logical axioms, LAP has 36 (nonlogical) axioms A1,A2,...,A36 and two nonlogical rules. The translation of each axiom is a theorem of $\overline{V \oplus L}$. This is easy to show for every axiom, with the exception of A32, A33, and A36. The first two of these help define $\sum(A)$ to be the sum of all entries of the matrix A (see below), and the third gives the recursive step in the definition of powering: $\mathrm{p}(n + 1, A) = \mathrm{p}(n, A) \times A$. Correctness proofs for the axioms are given in [Fon].

Axiom A33 gives the inductive step in the definition of $\sum(A)$. It is
$1 < \mathrm{r}(A), 1 < \mathrm{c}(A) \rightarrow \sum(A) = \mathrm{e}(A, 1, 1) + \sum(R(A)) + \sum(S(A)) + \sum(M(A))$
where

$$R(A) := \lambda_{ij}\langle 1, \mathrm{c}(A) - 1, \mathrm{e}(A, 1, i+1)\rangle$$
$$S(A) := \lambda_{ij}\langle \mathrm{r}(A) - 1, 1, \mathrm{e}(A, i+1, 1)\rangle$$
$$M(A) := \lambda_{ij}\langle \mathrm{r}(A) - 1, \mathrm{c}(A) - 1, \mathrm{e}(A, i+1, j+1)\rangle$$

e	R
S	M

This breaks the matrix A into four parts as indicated on the right, and sums them separately (a similar recursive step is used in Berkowitz's algorithm for computing the characteristic polynomial of a matrix). Correctness of the translation of this axiom involves proving in $\overline{V \oplus L}$ that the parity of the matrix is the sum (mod 2) of the four parities. This is done by considering a submatrix of A consisting of the first i rows together with the first j entries of row $i+1$, and arguing by induction first on i and then on j. The second induction step uses the lemma stating that if two strings differ by exactly one bit, then they have opposite parities.

The first nonlogical rule is induction:

$$\frac{\Gamma, \alpha(i) \to \alpha(i+1), \Delta}{\Gamma, \alpha(0) \to \alpha(n), \Delta} \tag{9}$$

By Theorem 4 and part (iv) of Proposition 2 it follows that $\overline{V \oplus L}$ proves that the translation of the bottom follows from the translation of the top, as required.

The second rule is the Matrix Equality Rule:

$$\frac{S_1 \quad S_2 \quad S_3}{\Gamma \to \Delta, T =_{\mathrm{matrix}} U} \qquad \text{where} \quad \begin{array}{ll} S_1: & \Gamma \to \Delta, e(T,i,j) = e(U,i,j) \\ S_2: & \Gamma \to \Delta, r(T) = r(U) \\ S_3: & \Gamma \to \Delta, c(T) = c(U) \end{array} \tag{10}$$

It is immediate that the translation of the bottom follows from the translation of the top by the way $T =_{\mathrm{matrix}} U$ is translated (8).

It follows that

Theorem 5. *Theorems of LA*P *translate to theorems of* $V \oplus L$.

3.2 Interpreting LAP into $V\#L$

Now we interpret the 'field' $\mathbb{F}$ in the semantics of LAP to be the ring $\mathbb{Z}$ of integers. Of course $\mathbb{Z}$ is not a field, so we cannot translate the axiom A21 $a \neq 0 \to a * (a^{-1}) = 1$ for field inverses. However according to the footnote on page 283 of [SC04], this axiom is not used except in the proof of Lemma 3.1 and Theorem 4.1. The latter theorem states that LAP proves that the Cayley-Hamilton theorem implies the hard matrix identities (3). However it is not hard to see that Theorem 4.1 does hold for integral domains, because the field inverse axiom can be replaced by the cancellation law $a \neq 0, a * b = a * c \to b = c$. Hence LAP with this axiom replacing A21 does prove that the Cayley-Hamilton theorem implies the hard matrix identities. It follows from our interpretation of LAP into $V\#L$ that $V\#L$ proves that the Cayley-Hamilton Theorem implies the hard matrix identities over $\mathbb{Z}$ (see Theorem 2).

The interpretation of LAP into $V\#L$ is similar to the interpretation into $V \oplus L$. We first translate LAP into $\overline{V\#L}$, and then into $V\#L$, using Theorem

4 and part (v) of Proposition 2. As before, terms of type index are translated into number terms. However elements of type field (integers) are translated into strings representing the integers in binary. Binary (rather than unary) notation for integers is chosen to match the axiom for matrix powering in $V\#L$ (see the end of the first paragraph in Section 2.4). Elements of type matrix are translated into strings representing arrays of binary integers. We define string functions $+_{\mathbb{Z}}$ and $\times_{\mathbb{Z}}$ in $\overline{VTC^0}$ representing integer plus and times, and $F_{\sum}(X)$ for iterated integer sum. See [Fon] for more details.

4 Conclusion

There are two general motivations for associating theories with complexity classes. The first is that of reverse mathematics: determining the complexity of concepts needed to prove various theorems, and in particular whether the correctness of an algorithm can be proved with concepts of complexity comparable to that of the algorithm. The second motivation comes from propositional proof complexity: determining the proof lengths of various tautology families in various proof systems.

Both of these motivations are relevant to the earlier open questions of whether LAP can prove properties of the determinant such as the Cayley-Hamilton theorem, and hence the hard matrix identities (3). Now we can refine these questions and apply them to the two complexity classes $\oplus L$ and DET. It is possible that $V\oplus L$ and $V\#L$ could prove these properties for their associated rings $\mathbb{Z}_2$ and $\mathbb{Z}$ by methods not available to LAP. For example $V\oplus L$ might take advantage of the simplicity of $\mathbb{Z}_2$, or $V\#L$ might be able use the algorithmic strength of integer matrix powering (as opposed to matrix powering over an unspecified field) to prove correctness of the dynamic programming algorithm for the determinant in [MV97]. This algorithm is based on a combinatorial characterization of $det(A)$ using clow (closed walk) sequences in the edge-labeled graph specified by the matrix A.

Over the field $\mathbb{Z}_2$ the hard matrix identities translate naturally to a family of propositional tautologies (and over $\mathbb{Z}$ they translate into another family of tautologies). The original motivation for studying these identities was to give further examples of tautology families (like those in [BBP94]) that might be hard for the class of propositional proof systems known as Frege systems. There is a close connection between the strength of a theory needed to prove these identities (or any Σ_0^B formula) and the strength of the propositional proof system required for their propositional translations to have polynomial size proofs. (Chapter 10 of [CN10] gives propositional proof systems corresponding in this way to five of the theories in (2).)

In particular, the fact that the hard matrix identities are provable in VP shows that their propositional translations have polynomial size proofs in Extended Frege systems. If the identities were provable in VNC^1 then the tautologies would have polynomial size Frege proofs. If the identities turn out to be provable in one of our new theories, then the tautologies would have polynomial

size proofs in proof systems (yet to be defined) of strength intermediate between Frege and Extended Frege systems.

Finally, we point out a lesser open problem. The main axiom for our new theory $V\#L$ asserts that integer matrix powers exist, where integers are represented in binary. As explained at the beginning of Section 2.4, integer matrix powering is complete for the complexity class DET even when restricted to 0-1 matrices, because the binary case is AC^0-reducible to the 0-1 case. It would be interesting to investigate whether the nontrivial reduction (see [Fon09]) can be proved correct in the base theory VTC^0, so that $V\#L$ could equivalently be axiomatized by the axiom for the 0-1 case rather than the binary case.

References

[All04] Eric Allender. Arithmetic Circuits and Counting Complexity Classes. In Jan Krajicek, editor, *Complexity of computations and proofs*, pages 33–72. Quaderni di Matematica, 2004.

[AO96] Eric Allender and Mitsunori Ogihara. Relationships Among PL, $\#L$, and the Determinant. *RAIRO - Theoretical Informatics and Applications*, 30:1–21, 1996.

[BBP94] Maria Luisa Bonet, Samuel R. Buss, and Toniann Pitassi. Are there hard examples for frege systems? In P. Clote and J. B. Remmel, editors, *Feasible Mathematics II*, pages 30–56. Birkhauser, 1994.

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and Importance of Logspace-MOD Class. *Mathematical Systems Theory*, 25:223–237, 1992.

[Ber84] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.

[BKR09] Mark Braverman, Raghav Kulkarni, and Sambuddha Roy. Space-Efficient Counting in Graphs on Surfaces. *Computational Complexity*, 18:601–649, 2009.

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010. Draft available from URL `http://www.cs.toronto.edu/~sacook`.

[Coo85] S. A. Cook. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control*, 64:2–22, 1985.

[Fon] Lila Fontes. Interpreting LAP into $V\oplus L$ and $V\#L$. Draft available at `www.cs.toronto.edu/~fontes`.

[Fon09] Lila Fontes. Formal Theories for Logspace Counting. Master's thesis, University of Toronto, 2009. Available at `http://arxiv.org/abs/1001.1960`.

[Imm99] Neil Immerman. *Descriptive Complexity*. Springer, 1999.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chicago Journal of Theoretical Computer Science*, 5, 1997.

[SC04] Michael Soltys and S. A. Cook. The Proof Complexity of Linear Algebra. *Annals of Pure and Applied Logic*, 130:277–323, 2004.

[SK01] Michael Soltys-Kulinicz. *The Complexity of Derivations of Matrix Identities*. PhD thesis, University of Toronto, 2001.