The Complexity of Iterated Multiplication

Neil Immerman* and Susan Landau†

Computer Science Department
University of Massachusetts
Amherst, MA 01003

appeared in *Information and Computation* 116(1) (1995), 103-116

Abstract

For a monoid G, the iterated multiplication problem is the computation of the product of n elements from G. By refining known completeness arguments, we show that as G varies over a natural series of important groups and monoids, the iterated multiplication problems are complete for most natural, low-level complexity classes. The completeness is with respect to "first-order projections" – low-level reductions that do not obscure the algebraic nature of these problems.

1 Introduction

In recent years, the structure of low-level complexity classes (Figure 1) has been of great interest. For many of these classes there is a natural graph problem which serves as a complete set for the class. In this paper we consider the complexity of iterated multiplication. More specifically we study the complexity of multiplying together n elements of a monoid G. As G ranges over a sequence of well-studied monoids, including the symmetric group on 5 elements S_5, the symmetric group on n elements S_n, the monoid of $n \times n$ boolean matrices under multiplication $M_n(\text{bool})$, and the monoid of $n \times n$ integer matrices $M_n(\mathbf{Z})$, the iterated multiplication problem is complete for a corresponding well studied complexity class. Furthermore, the notion of completeness in question is extremely low-level and algebraic. This point of view results in a very pretty picture, and establishes a framework for the investigation of an important series of questions relating algebraic complexity to (boolean) complexity. Figure

*Supported by NSF grants CCR-8806308 and CCR-9008416.
†Supported by NSF grants DMS-8807202 and CCR-8802835.

$$\mathrm{FO} = \mathrm{AC}^0 \subset \mathrm{ThC}^0 \subseteq \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \begin{array}{c} \subseteq \ \mathrm{AC}^1 \ \subseteq \\ \subseteq \mathrm{DET}^* \subseteq \end{array} \mathrm{ThC}^1 \subseteq \mathrm{NC}^2 \subseteq \mathrm{DSPACE}[\log^2 n]$$

Figure 1: Known relationships among low-level complexity classes

2 summarizes the completeness results. The main contribution of this paper is to show that these previously-known completeness results hold with respect to first-order projections. These extremely low-level reductions are defined in Section 3.

A very surprising fact – reported in [BCH, Rei] – is that iterated integer multiplication is in P-uniform ThC^0 but not known to be in (uniform) ThC^0, or even in L. Usually, uniformity does not pose as a serious issue for complexity classes, at least non-randomized ones. The fact that such a natural and important problem as iterated multiplication seems to require polynomial-time uniformity is important and surprising. The issue of uniformity in this algebraic context requires extensive study. We make a step in this direction in Sections 3 and 5.

2 Iterated Multiplication

We begin with some definitions of low-level complexity classes. As usual, L denotes $\mathrm{DSPACE}[\log n]$ and NL denotes $\mathrm{NSPACE}[\log n]$.

Definition 2.1 Define the complexity classes NC^i, AC^i, ThC^i for $i \geq 0$ as follows: These are the sets of problems acceptable by "uniform" sequences of circuits that are polynomial-size, depth $O[(\log n)^i]$ and consist of binary and/or gates, unbounded fan-in and/or gates, unbounded fan-in threshold gates, respectively. These circuits may also contain negation gates, which we may assume are all pushed down to the bottom level.

The meaning of "uniform" in the above definition is that the map from 1^n to the n^{th} circuit is easily computable. For the classes above NC^1 this may be taken to be logspace computable. Alternatively, we would say that the circuit class is polynomial-time uniform if this map is computable in polynomial-time. For NC^1 and below we assume logtime uniformity or equivalently first-order uniformity [BIS]. We discuss uniformity in detail in §3.

Beame, Cook, and Hoover have shown:

Theorem 2.2 ([BCH]) *Iterated integer multiplication (Problem $\prod(\mathbf{Z})$) is computable by polynomial-time uniform NC^1 circuits.*

Beame, Cook, and Hoover proved Theorem 2.2 by using the Chinese Remainder Theorem. Each of the input integers is reduced modulo each of the first n^2

primes. The product of the inputs modulo each of these primes is computed via discrete logs. Then, the products modulo each of the primes are combined to give the final answer.

All of the above steps can be carried out in NC^1 once we are given a polynomial-size table consisting of information such as the discrete log tables for each of the first n^2 primes. Reif pointed out that the [BCH] proofs goes through when NC^1 is replaced by the (perhaps smaller) class ThC^0.

Theorem 2.3 ([Rei]) *Iterated integer multiplication (Problem $\prod(\mathbf{Z})$) is computable by polynomial-time uniform* ThC^0 *circuits.*

We find it quite surprising that Theorem 2.2 seems to require polynomial-time uniformity. A more careful look shows that the amount of polynomial-time computation needed to compute the n^{th} circuit is no more than the computation of a single instance of iterated multiplication:

Corollary 2.4 ([BCH]) *Let P_{n^2} be the product of the first n^2 primes. There is a logspace map from P_{n^2} to the n^{th}* ThC^0 *circuit for iterated integer multiplication.*

This leads us to wonder about the complexity of iterated multiplication problems on other monoids. In particular, the last five years have seen proofs of completeness results for iterated multiplication of elements in S_5, in S_n, in $n \times n$ Boolean matrices, and in $n \times n$ matrices from $\mathbf{Z}$. In §5 we will present proofs of strong versions of these results (that is, using weak reductions).

First we give a precise definition of the iterated multiplication problems we will consider.

Definition 2.5 Let $\mathcal{M}$ be a monoid and let $\mathrm{cd} : \mathcal{M} \to \{0,1\}^*$ be an encoding of the elements of this monoid as binary strings. Then the *Iterated Multiplication Problem* for $\mathcal{M}$ (in symbols $\prod(\mathcal{M})$) is the problem of multiplying together a sequence of elements from $\mathcal{M}$.

More formally, we want to consider these iterated multiplication problems as decision problems. Thus an input will consist of an n-tuple of binary strings coding n elements of $\mathcal{M}$, together with an integer i. This input will be in the language iff the i^{th} bit of the coding of the product of the inputs is a one. Let $x[[i]]$ denote the i^{th} bit of the binary string x.

$$\prod(\mathcal{M}) \;=\; \left\{ [\langle \mathrm{cd}(a_1), \ldots, \mathrm{cd}(a_n)\rangle, i] \in (\mathrm{cd}(\mathcal{M}))^n \times \mathbf{N} \;\middle|\; \mathrm{cd}(\prod_{j=1}^{n} a_j)[[i]] = 1 \right\}$$

Technically, the encoding, cd, could affect the complexity of the iterated multiplication problems. For example, if we coded integers in unary, then iterated integer multiplication would be in AC^0. As long as we use common sense, the complexity of these iterated multiplication problems are not usually sensitive to

Problem	Complexity
$\prod(\mathbf{Z})$	in P-unif ThC^0; in DET*
$\prod(S_5)$, $\prod(M_3(\mathbf{F}_2))$	complete for NC^1
$\prod(S_n)$	complete for L
$\prod(M_n(\text{bool}))$	complete for NL
$\prod(M_n(\mathbf{Z}))$	DET*

Figure 2: The Complexity of Iterated Multiplication

which particular encoding we use. (However, we will see that the coding can effect whether or not the problem is complete via extremely low-level reductions, cf. Remark 6.1.) The encodings we will be using for each of the problems we study are explicitly given in Figure 3.

Theorem 2.6 ([Bar]) $\prod(S_5)$ *is complete for* NC^1 *via* AC^0 *reductions.*

An alternate version of Barrington's Theorem was shown recently by Ben-Or and Cleve, using iterated multiplication of 3×3 matrices over the field with two elements:

Theorem 2.7 ([BC]) $\prod(M_3(\mathbf{F}_2))$ *is complete for* NC^1 *via projections.*

Theorem 2.8 ([CM]) $\prod(S_n)$ *is complete for* L *via* NC^1 *reductions.*

Theorem 2.9 ([Coo85]) $\prod(M_n(\mathit{bool}))$ *is complete for* NL* *via* NC^1 *reductions.*

(Note, that by [Imm88], NL* is equal to NL.)

Theorem 2.10 ([Coo85, Ber]) $\prod(M_n(\mathbf{Z}))$ *is complete for DET* via* NC^1 *reductions.*

Almost always, when a natural problem has been shown to be in a non-uniform circuit complexity class, it has been straightforward to see that it is in the corresponding uniform complexity class, with the exception of probabilistic complexity classes. It is unknown how to do this with problem $\prod(\mathbf{Z})$, which is in polynomial-time uniform ThC^0, but not even known to be in L. This anomaly caused us to first examine the issue of the complexity of iterated multiplication. We spend the rest of this paper discussing uniformity and reductions.

3 Reductions

In this section we describe first-order interpretations. These are very low-level many-one reductions, given by interpretations – a standard concept from logic for translating one language into another [End, Imm87]. We define a very weak version of first-order interpretations, namely first-order projections. These are first-order interpretations that are at the same time projections in the sense of Valiant [Val]. We will see that the completeness results listed in Figure 2 all hold via first-order projections. The value of this observation is that first-order projections are sufficiently low-level that they retain the full algebraic character of the problems.

(In [Imm87] a reduction called "projection translations" were defined. These are equivalent to what we now call "quantifier-free projections" (qfp's). First-order projections (fop's) allow a bit more flexibility without giving away much in power; and so, we now consider fop's to be the very low-level reduction of choice rather than qfp's.)

Background on Descriptive Complexity

In order to define and use very low-level reductions, we take an approach to complexity theory derived from mathematical logic. Our notation follows the conventions of Descriptive Complexity. See [Imm87, Imm89] for more detail and motivation.

We will code all inputs as finite logical structures. For example, the input for problem $\prod(\mathbf{Z})$ – n n-bit integers – is just a binary string $b_0 \ldots b_{n^2-1}$. We associate this string with a finite *structure*

$$\mathcal{A} = \langle \{0,1,\ldots,n-1\}, R_1 \rangle$$

where R_1 is the binary relation on $|\mathcal{A}|$ defined so that $R_1(x,y)$ holds in $\mathcal{A}$ (in symbols, $\mathcal{A} \models R_1(x,y)$) just if $b_{nx+y} = 1$. As is customary, the notation $|\mathcal{A}|$ will be used to denote the universe $\{0,1,\ldots,n-1\}$ of the structure $\mathcal{A}$. We will write $\|\mathcal{A}\|$ to denote n, the cardinality of $|\mathcal{A}|$.

In general, a *vocabulary*

$$\tau = \langle R_1^{a_1}, \ldots, R_t^{a_t}, c_1, \ldots, c_s \rangle$$

is a tuple of input relation symbols and constant symbols. A *structure*

$$\mathcal{A} = \langle \{0,1,\ldots,n-1\}, R_1^{\mathcal{A}}, \ldots, R_t^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}} \rangle$$

of vocabulary τ is a finite set $|\mathcal{A}| = \{0,1,\ldots,n-1\}$ together with relations $R_i^{\mathcal{A}} \subseteq |\mathcal{A}|^{a_i}$, $i = 1,2,\ldots,t$, and elements $c_j^{\mathcal{A}}$, $j = 1,2,\ldots,s$.

Let STRUC$[\tau]$ denote the set of all finite structures of vocabulary τ. We define a complexity theoretic *problem* to be any subset of STRUC$[\tau]$ for some

Problem	Relation	Meaning	Decision Version
$\prod(\mathbf{Z})$	$R_1(x,y)$	bit y of integer x is 1	$\mathrm{PROD}[[c_1]] = 1$
$\prod(S_5)$	$R_2(x,i,j)$	bit j of $\pi_x(i)$ is 1	$\mathrm{PROD}(c_1)[[c_2]] = 1$
$\prod(S_n)$	$R_3(x,i,j)$	bit j of $\pi_x(i)$ is 1	$\mathrm{PROD}(c_1)[[c_2]] = 1$
$\prod(M_n(\text{bool}))$	$R_4(x,i,j)$	$M_x(i,j) = 1$	$\mathrm{PROD}(c_1,c_2) = 1$
$\prod(M_n(\mathbf{Z}))$	$R_5(x,i,j,r)$	bit r of $M_x(i,j)$ is 1	$\mathrm{PROD}(c_1,c_2)[[c_3]] = 1$

Figure 3: Coding of Problems as First-Order Structures

τ. For simplicity, in this paper we will only consider vocabularies consisting of a single input relation R^i of arity i, and perhaps some constant symbols. For example, when $i = 2$ the input is a single binary relation – a graph, with the elements of the universe ranging over the vertices. When $i = 1$ the input is a binary string, with the elements of the universe ranging over the bit positions of the string. Figure 3 gives the coding of the problems we are considering. For example, in problem $\prod(S_n)$ the input structure codes n elements of S_n. The meaning of $R_3(x,i,j)$ is that the j^{th} bit of $\pi_x(i)$ is one where π_x is the x^{th} permutation. See Remark 6.1 for a further explanation of the choice of some of these codings.

The advantage of this approach is that when we consider our inputs as first-order structures we may write properties of them in variants of first-order logic. It is a pleasant surprise that first-order logic provides elegant characterizations of most natural complexity classes [Imm87].

For any vocabulary τ there is a corresponding first-order language $\mathcal{L}(\tau)$ built up from the symbols of τ and the logical relation symbols and constant symbols: $=, \leq, s, \mathrm{BIT}, 0, \mathbf{m}$,[1] using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, \ldots$, and quantifiers: $\forall, \exists$.

Let FO be the set of first-order definable problems. We define the majority quantifier M so that $(Mx)\varphi(x)$ holds if more than half the elements of the universe satisfy $\varphi(x)$. Let FOM be the set of problems definable in FO extended by uses of the majority quantifier. It has often been observed that a first-order sentence has a natural interpretation as a uniform sequence of AC^0 circuits. A similar fact holds for FOM and ThC^0, cf. Fact 3.1. (For example, consider the sentence $\varphi = (\exists x)(\forall y)(\exists z)(E(x,y) \wedge E(y,z))$. For each n, φ determines a circuit C_n which takes as input a binary string of length n^2: the adjacency matrix of a graph on n vertices. The circuit C_n is an n-ary "or" of an n-ary "and" of an

[1] Here $\leq$ refers to the usual ordering on $\{0, \ldots, n-1\}$, s is the successor relation, BIT is described below, and $0, \mathbf{m}$ refer to $0, n-1$, respectively. Some of these are redundant, but useful for quantifier-free interpretations. For simplicity we will assume throughout that $n > 1$ and thus $0 \neq \mathbf{m}$. Sometimes the logical relations are called "numeric" relations. For example, "$\mathrm{BIT}(i,j)$" and "$i \leq j$" describe the numeric values of i and j and do not refer to any input predicates.

n-ary "or" of a binary "and".)

To capture uniform AC^0, we make use of the logical relation "BIT." $\mathrm{BIT}(x,y)$ means that the x^{th} bit in the binary expansion of y is a one. Remember that the variables range over a finite universe, $\{0,1,\ldots,n-1\}$, for some value of n. Thus they may be thought of as $(\log n)$-bit numbers. In [BIS] it is shown that BIT is definable in FOM(wo BIT), if we assume that the majority quantifier may apply to pairs of individual variables. (By "wo BIT" we mean without the logical relation BIT.) Thus FOM = FOM(wo BIT), whereas FO $\neq$ FO(wo BIT). (To see the latter, note that the parity of the universe is expressible in FO, but not in FO(wo BIT). See also [Lin] for further discussion of BIT.)

In fact, the following are completely syntactic definitions for the uniform classes AC^0 and ThC^0:

Fact 3.1 ([BIS]) $\mathrm{FO} = \mathrm{AC}^0$ *and* $\mathrm{FOM} = \mathrm{ThC}^0$.

First-Order Interpretations and Projections

In [Val], Valiant defined the *projection,* an extremely low-level many-one reduction. Projections are weak enough to preserve the algebraic structure of problems such as iterated multiplications. For this reason we find it particularly interesting that projections suffice for proving completeness properties of various iterated multiplication problems.

Definition 3.2 Let $S,T \subseteq \{0,1\}^\star$. A k-ary *projection* from S to T is a sequence of maps $\{p_n\}$, $n = 1,2,\ldots$, such that for all n and for all binary strings s of length n, $p_n(s)$ is a binary string of length n^k and,

$$s \in S \quad \Leftrightarrow \quad p_n(s) \in T\ .$$

Let $s = s_0 s_1 \ldots s_{n-1}$. Then each map p_n is defined by a sequence of n^k literals: $\langle l_0, l_1, \ldots, l_{n^k-1}\rangle$ where

$$l_i \in \{0,1\} \cup \{s_j, \bar{s}_j \,|\, 0 \le j \le n-1\}\ .$$

Thus as s ranges over strings of length n, each bit of $p_n(s)$ depends on at most one bit of s,

$$p_n(s)[[i]] \quad = \quad l_i(s)$$

Projections were originally defined as a non-uniform sequence of reductions – one for each value of n. We now define first-order projections, which are a uniform version of Valiant's projections. The idea of our definition is that the choice of the literals $\langle l_0, l_1, \ldots, l_{n^k-1}\rangle$ in Definition 3.2 is given by a first-order formula in which no input relation occurs. Thus the formula can only talk about bit positions, and not bit values. The choice of literals depends only on n. In order to make this definition, we must first define first-order interpretations.

These are a standard notion from logic for translating one theory into another, cf. [End], modified so that the transformation is also a many-one reduction, [Imm87]. (For readers familiar with databases, a first-order interpretation is exactly a many-one reduction that is definable as a first-order query.)

Definition 3.3 (First-Order Interpretations) Let σ and τ be two vocabularies, with $\tau = \langle R_1^{a_1}, \dots, R_r^{a_r}, c_1, \dots, c_s \rangle$. Let $S \subseteq \mathrm{STRUC}[\sigma]$, $T \subseteq \mathrm{STRUC}[\tau]$ be two problems. Let k be a positive integer. Suppose we are given an r-tuple of formulas $\varphi_i \in \mathcal{L}(\sigma)$, $i = 1, \dots, r$, where the free variables of φ_i are a subset of $\{x_1, \dots, x_{k \cdot a_i}\}$. Finally, suppose we are given an s-tuple, $t_1, \dots, t_s$, where each t_j is a k-tuple of closed terms[2] from $\mathcal{L}(\sigma)$. Let $I = \lambda_{x_1 \dots x_d} \langle \varphi_1, \dots, \varphi_r, t_1, \dots, t_s \rangle$ be a tuple of these formulas and closed terms. (Here $d = \max_i(ka_i)$.)

Then I induces a mapping $\hat{I}$ from $\mathrm{STRUC}[\sigma]$ to $\mathrm{STRUC}[\tau]$ as follows. Let $\mathcal{A} \in \mathrm{STRUC}[\sigma]$ be any structure of vocabulary σ, and let $n = ||\mathcal{A}||$. Then the structure $\hat{I}(\mathcal{A})$ is defined as follows:

$$\hat{I}(\mathcal{A}) \;=\; \langle \{0, \dots, n^k - 1\}, R_1, \dots, R_r, c_1, \dots, c_s \rangle$$

Here each c_j is given by the corresponding k-tuple of closed terms t_j. The relation R_i is determined by the formula φ_i, for $i = 1, \dots, r$. More precisely, let the function $\langle \cdot, \dots, \cdot \rangle : |\mathcal{A}|^k \to |\hat{I}(\mathcal{A})|$ be given by

$$\langle u_1, u_2, \dots, u_k \rangle \;=\; u_k + u_{k-1} n + \cdots + u_1 n^{k-1}$$

Then,

$$R_i \;=\; \big\{ (\langle u_1, \dots, u_k \rangle, \dots, \langle u_{1+k(a_i-1)}, \dots, u_{ka_i} \rangle) \;\big|\; \mathcal{A} \models \varphi_i(u_1, \dots u_{ka_i}) \big\}$$

If the structure $\mathcal{A}$ interprets some variables $\bar{u}$ then these may appear freely in the the φ_i's and t_j's of I, and the definition of $\hat{I}(\mathcal{A})$ still makes sense. This will be important in Definition 4.2 where we define operators in terms of first-order interpretations.

Suppose that $\hat{I}$ is a many-one reduction from S to T, i.e. for all $\mathcal{A}$ in $\mathrm{STRUC}[\sigma]$,

$$\mathcal{A} \in S \quad \Leftrightarrow \quad \hat{I}(\mathcal{A}) \in T$$

Then we say that I is a k-ary *first-order interpretation* of S to T. Furthermore, if the φ_i's are quantifier-free and do not include BIT then I is a *quantifier-free interpretation.*

[2] A closed term is an expression involving constants and function symbols. This is as opposed to an open term which also has free variables. In this paper, since we do not have function symbols, closed terms are synonymous with constant symbols. Note that a more general way to interpret constants and functions is via a formula φ such that $\vdash (\forall \bar{x})(\exists! y)\varphi(\bar{x}, y)$. However, in this paper the simpler definition using closed terms suffices.

Note that I induces a map which we will also call I from $\mathcal{L}(\tau)$ to $\mathcal{L}(\sigma)$. For $\varphi \in \mathcal{L}(\tau)$, $I(\varphi)$ is the result of replacing all relation and constant symbols in φ by the corresponding formulas and closed terms in I. Note that if I is a k-ary interpretation then each variable in φ is replaced by a k-tuple of variables. Furthermore, the logical relations $s, \leq, =$ are replaced by the corresponding quantifier-free formulas on k-tuples ordered lexicographically. For example, with $k = 2$, an occurrence of the successor relation, $s(x, y)$, would be replaced by

$$I(s(x,y)) \;=\; (x_1 = y_1 \wedge s(x_2, y_2)) \;\vee\; (x_2 = \mathbf{m} \wedge y_2 = 0 \wedge s(x_1, y_1))$$

The logical constants, $0, \mathbf{m}$, are replaced by k-tuples of the same constants.

Note that the logical relation BIT when mapped to k-tuples cannot be easily replaced by a quantifier-free formula. This is the reason we have omitted BIT from the allowable logical formulas in our definition of quantifier-free interpretations and quantifier-free projections. However, BIT on tuples is definable in FO (with BIT), so we retain BIT when talking about full first-order interpretations and first-order projections, cf. [Lin].

It follows immediately from the definitions that:

Proposition 3.4 *Let σ, τ, and I be as in Definition 3.3. Then for all sentences $\varphi \in \mathcal{L}(\tau)$ and all structures $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,*

$$\mathcal{A} \models I(\varphi) \quad \Leftrightarrow \quad \hat{I}(\mathcal{A}) \models \varphi$$

Example 3.5 As an example, define the GAP problem to be the set of directed graphs containing a path from vertex 0 to vertex $\mathbf{m}$. We now present a first-order interpretation of GAP to $\prod(M_n(\mathrm{bool}))$. Since GAP is known to be complete for NSPACE[$\log n$] via first-order interpretations (Fact 4.1) it follows that so is $\prod(M_n(\mathrm{bool}))$. Following the notation of Definition 3.3, the vocabulary of GAP is $\sigma = \langle E^2 \rangle$ consisting of a binary edge relation E. The vocabulary of $\prod(M_n(\mathrm{bool}))$ is $\tau = \langle R_4, c_1, c_2 \rangle$ where $R_4(x, i, j)$ is true just if entry (i, j) of matrix x is 1. (See Figure 3.)

The arity of the interpretation will be $k = 1$. We will reduce an instance G of the GAP problem to the problem of multiplying together n copies of the adjacency matrix. Note that entry $(0, \mathbf{m})$ of this product is a 1 iff there is a path in G from 0 to $\mathbf{m}$, i.e., iff $G \in$ GAP.

Thus the first-order interpretation must give the meaning of $R_4(x, i, j)$ as $E(i, j)$, independently of x. The first-order interpretation is simply $I = \langle E(x_2, x_3), 0, \mathbf{m} \rangle$. In fact, I is a quantifier-free interpretation.

We are now ready to define first-order projections, a syntactic restriction of first-order interpretations.

If each formula in the first-order interpretation I satisfies this syntactic condition then it follows that $\hat{I}$ is also a projection in the sense of Valiant. In this case we call I a first-order projection.

Definition 3.6 (First-Order Projections) Let I be a k-ary first-order interpretation from S to T as in Definition 3.3. Let $I = \langle \varphi_1, \ldots, \varphi_r, t_1, \ldots, t_s \rangle$. Suppose further that the φ_i's all satisfy the following *projection condition:*

$$\varphi_i \equiv \alpha_1 \vee (\alpha_2 \wedge \lambda_2) \vee \cdots \vee (\alpha_s \wedge \lambda_s) \tag{1}$$

where the α_j's are mutually exclusive formulas in which no input relations occur, and each λ_j is a literal, i.e. an atomic formula $P(x_{j_1}, \ldots x_{j_a})$ or its negation.

In this case the predicate $R_i(\langle u_1, \ldots, u_k \rangle, \ldots, \langle \ldots, u_{ka_i} \rangle)$ holds in $\hat{I}(\mathcal{A})$ if $\alpha_1(\bar{u})$ is true, or if $\alpha_j(\bar{u})$ is true for some $1 < j \leq t$ and the corresponding literal $\lambda_j(\bar{u})$ holds in $\mathcal{A}$. Thus each bit in the binary representation of $\hat{I}(\mathcal{A})$ is determined by at most one bit in the binary representation of $\mathcal{A}$. We say that I is a *first-order projection.*

Finally define a *quantifier-free projection* to be a first-order projection that is also a quantifier-free interpretation. Write $S \leq_{\text{fop}} T$, $S \leq_{\text{qfp}} T$ to mean that S is reducible to T via a first-order projection, respectively a quantifier-free projection.

Observe that the first-order interpretation I of Example 3.5 easily fits into the form of Equation 1. Here α_1 = false, α_2 = true, and $\lambda_2 = E(x_2, x_3)$. Thus this I is a first-order projection and in fact a quantifier-free projection. In symbols, GAP$\leq_{\text{qfp}} \prod(M_n(\text{bool}))$.

In the following proposition we show that these new reductions behave like other reductions.

Proposition 3.7 *The relations $\leq_{\text{fop}}$ and $\leq_{\text{qfp}}$ are reflexive and transitive.*

Proof Let $A \subseteq \text{STRUC}[\tau]$ be any problem where $\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle$. Then the identity map from STRUC[τ] to itself is given by the quantifier-free projection I_0 where

$$I_0 \;=\; \langle R_1(x_1, \ldots, x_{a_1}), \ldots, R_r(x_1, \ldots, x_{a_r}), c_1, \ldots, c_s \rangle$$

Thus $\leq_{\text{fop}}$ and $\leq_{\text{qfp}}$ are reflexive.

For transitivity, suppose that I is a k_1-ary first-order projection of $A \subseteq \text{STRUC}[\tau_1]$ to $B \subseteq \text{STRUC}[\tau_2]$ and J is a k_2-ary first-order projection of B to $C \subseteq \text{STRUC}[\tau_3]$. Recall that the interpretation I induces a map from $\mathcal{L}(\tau_2)$ to $\mathcal{L}(\tau_1)$. Thus, $I(J)$ is a k_1k_2-ary interpretation from $\mathcal{L}(\tau_3)$ to $\mathcal{L}(\tau_1)$. It is immediate that $\widehat{I(J)} = \hat{J} \circ \hat{I}$. Thus, $I(J)$ is a first-order interpretation from A to C. Furthermore, if I and J were quantifier-free then so is $I(J)$. It thus suffices to show that $I(J)$ is a first-order projection.

Recall that $I(J)$ is the result of replacing each variable in J by a k_1-tuple of variables, each relation symbol R by $I(R)$, and each constant symbol c by $I(c)$. From Definition 3.6 we have that each formula in I or J is of the form of

Equation 1. If we substitute a formula of this form for one of the λ_i's then we obviously remain in the same form. This is less clear when we have to negate the formula.

However, observe that the negation of the formula φ_i in Equation 1 is still a formula of the same form. Namely,

$$\neg\varphi_i \equiv \neg(\alpha_1 \vee \ldots \vee \alpha_s) \ \vee \ (\alpha_2 \wedge \neg\lambda_2) \vee \cdots \vee (\alpha_s \wedge \neg\lambda_s)$$

Hence, $I(J)$ is still a projection. ∎

As another example of quantifier-free projections, we prove the following:

Proposition 3.8 *The following quantifier-free projections exist. (Refer to Figure 3.)*

$$\prod(S_5) \ \leq_{\mathrm{qfp}} \ \prod(S_n) \ \leq_{\mathrm{qfp}} \ \prod(M_n(\mathit{bool})) \ \leq_{\mathrm{qfp}} \ \prod(M_n(\mathbf{Z}))$$

Proof $(\prod(S_5) \leq_{\mathrm{qfp}} \prod(S_n))$: This is immediate because S_5 is a subgroup of S_n and we have coded the two problems in the same way. Thus the quantifier-free projection is the identity map, $\hat{I_1} = \mathrm{id}$, where $I_1 = \langle R_2(x_1, x_2, x_3), c_1, c_2 \rangle$

$(\prod(S_n) \leq_{\mathrm{qfp}} \prod(M_n(\mathrm{bool})))$: This is similar to the above situation. Every element π of S_n may be thought of as an element $A(\pi)$ of $M_n(\mathrm{bool})$ as follows:

$$(A(\pi))_{i,j} = \begin{cases} 1 & \text{if } \pi(i) = j \\ 0 & \text{if } \pi(i) \neq j \end{cases}$$

However, there is a problem. We chose to code the problem $\prod(S_n)$ in a bitwise fashion, see Figure 3. Thus, each bit of the matrix $A(\pi)$ depends upon $\log n$ bits of the coding of π. In order to be a projection we must depend on at most one bit.

The solution to this problem is to increase the arity of the boolean matrix so that it can figure out the value of $j = \pi(i)$ via a path of length at most n. Think of $A(\pi)$ as a boolean adjacency matrix. Then we simulate the move $\pi : i \mapsto j$ by

$$(i,0,0) \ \rightarrow \ (i, v_1, 1) \ \rightarrow \ (i, v_2, 2) \ \rightarrow \cdots \rightarrow \ (i, v_{\log n}, \log n) \ \rightarrow \ (j, 0, 0)$$

where $0 = v_0$, and $v_{r+1} = v_r + 2^r(\pi(i)[[r]])$, and thus, $j = v_{\log n}$.

The above would be a quantifier-free projection except that BIT is needed to say that $v_{r+1} = v_r + 2^r$. The details of expressing BIT as a qfp of $\prod(M_n(\mathrm{bool}))$ are complicated and so we omit them here. (See [Imm87, Lemma 3.2, Cor. 3.8] where it is shown that BIT is a qfp of GAP.) In any case, it follows from Theorem 5.1 that $\prod(S_n) \leq_{\mathrm{qfp}} \prod(M_n(\mathit{bool}))$.

$(\prod(M_n(\mathrm{bool})) \leq_{\mathrm{qfp}} \prod(M_n(\mathbf{Z})))$: Again every element of $M_n(\mathrm{bool})$ may be thought of as an element of $M_n(\mathbf{Z})$ by mapping the boolean values 0,1 to the integers 0,1 respectively. We may think of the n boolean matrices $B_0, B_1, \ldots, B_{n-1}$

as a sequence of n adjacency matrices for a layered graph in which all edges of B_i go from layer i to layer $i+1$. Thus, there is a path from point s of layer 0 to point t of layer n iff entry (s,t) in $\prod B_i$ is 1. Let $Z_0, \ldots, Z_{n-1}$ be the $n \times n$ integer matrices resulting from the above boolean matrices by mapping the boolean values 0,1 to the integers 0,1. It is well known that entry (s,t) in $\prod Z_i$ is equal to the number of paths from point s, layer 0 to point t, layer n. Thus, a particular entry of the product of the boolean matrices is 0 iff the same entry of the product of the corresponding integer matrices is 0.

We have to work a little here because the answer to $\prod(M_n(\mathbf{Z}))$ does not automatically code whether or not an entry is 0 into a single bit. Suppose we want to compute entry (c_1, c_2) in the product of the boolean matrices. It suffices to modify the problem by adding $2^r - 1$ new paths from c_1 to c_2, where r is chosen so that 2^r is greater than the number of possible original paths. In this way the r^{th} bit of entry (c_1, c_2) of the product of the integer matrices will be 1 iff entry (c_1, c_2) of the product of the original boolean matrices is 1. We will use the value $r = n^2 - 1$ (which certainly suffices).

We now give the explicit coding of a binary quantifier-free projection of $\prod(M_n(\text{bool}))$ to $\prod(M_n(\mathbf{Z}))$. We are projecting a structure $\mathcal{A} = \langle\{0, \ldots, n-1\}, R_4, c_1, c_2\rangle$ consisting of n $n \times n$ boolean matrices onto a structure $\hat{I_3}(\mathcal{A}) = \langle\{0, \ldots, n^2-1\}, R_5, c_1, c_2, c_3\rangle$ consisting of n^2 $n^2 \times n^2$ n^2-bit integer matrices. The upper left $n \times n$ corners of the first n matrices of $\hat{I_3}(\mathcal{A})$ represent the original boolean matrices. The upper left corners of the remaining $n^2 - n$ matrices are the $n \times n$ identity matrix. In addition, entry (c_1, n^2-1) of the 0^{th} integer matrix contains the value $2^{n^2-1} - 1$. The integer matrices all have value 1 in entry (n^2-1, n^2-1). Finally, the last integer matrix contains a 1 in entry (n^2-1, c_2).

The quantifier-free projection I_3 is given by the following hard-to-read formula. We suggest that the reader look at the picture which follows. Then the reader should check that each bit in the picture of $\hat{I_3}(\mathcal{A})$ (which follows) depends on at most one bit from $\mathcal{A}$'s input relation. (The upper left hand corners depend on the corresponding bits from $\mathcal{A}$. None of the other entries depend on the input relation of $\mathcal{A}$.) Finally, the reader should observe that the specification of $\hat{I_3}(\mathcal{A})$ can be accomplished by the following quantifier-free projection.

$$I_3 \quad = \quad \langle \alpha(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8), \langle 0, c_1\rangle, \langle 0, c_2\rangle, \langle \mathbf{m}, \mathbf{m}\rangle\rangle, \qquad \text{where,}$$

$$\begin{aligned}
\alpha \quad &\equiv \quad \alpha_1 \vee (\alpha_2 \wedge R_4(x_1, x_3, x_5)) \\
\alpha_2 \quad &\equiv \quad (x_1 = x_3 = x_5 = x_7 = x_8 = 0) \\
&\qquad \text{[Upper left corner of first } n \text{ matrices copy input]} \\
\alpha_1 \quad &\equiv \quad (x_1 \neq 0 \ \wedge \ x_7 = x_8 = 0 \ \wedge \ x_3 = x_5 \ \wedge \ x_4 = x_6) \\
&\qquad \text{[Rest of the matrices have 1's on the diagonal]} \\
&\qquad \vee \ (x_7 = x_8 = 0 \ \wedge \ x_3 = x_4 = x_5 = x_6 = \mathbf{m})
\end{aligned}$$

[Lower right corner of each matrix is a 1]
$\vee\ (x_1 = x_2 = x_3 = 0\ \wedge\ x_4 = c_1\ \wedge\ x_5 = x_6 = \mathbf{m}\ \wedge\ \neg(x_7 = x_8 = \mathbf{m}))$
[Entry (c_1, n^2-1) of matrix 0 is $2^{n^2-1}-1$]
$\vee\ (x_1 = x_2 = x_3 = x_4 = \mathbf{m}\ \wedge\ x_5 = x_7 = x_8 = 0\ \wedge\ x_6 = c_2)$
[Entry (n^2-1, c_2) of last matrix is 1]

$$\hat{I}_3 : (A^0)(A^1)\cdots(A^{n-1}) \quad \mapsto$$

$$\begin{array}{c} \\ \\ c_1 \\ \\ \\ \\ \\ \\ \end{array}\begin{pmatrix} A^0_{00} & \cdots & A^0_{0m} & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ A^0_{c_1 0} & \ddots & A^0_{c_1 m} & 0 & \cdots & 2^{n^2-1}-1 \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ A^0_{m0} & \cdots & A^0_{mm} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} * \cdots * \begin{pmatrix} A^m_{00} & \cdots & A^m_{0m} & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ A^m_{m0} & \cdots & A^m_{mm} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} *$$

$$\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \cdots & 1 & \cdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} * \cdots * \begin{array}{c} \\ \\ \\ c_2 \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{cccccccc} & & c_2 & & & & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

∎

Remark 3.9 *The above quantifier-free projection from $\prod(M_n(\mathrm{bool}))$ to $\prod(M_n(\mathbf{Z}))$ is wasteful in the sense that it has arity $k=2$. This means that the resulting universe has size $\|\hat{I}(\mathcal{A})\| = \|\mathcal{A}\|^2$, when size $2\|\mathcal{A}\|$ would easily have sufficed. A more general definition of first-order interpretation would allow some of the coordinates of $\hat{I}(\mathcal{A})$ to be restricted, thus allowing greater efficiency.*

In [Coo71], Cook proved that the boolean satisfiability problem (SAT) is NP complete via polynomial-time Turing reductions. Over the years SAT has been shown complete via weaker and weaker reductions, e.g. polynomial-time many-one [Kar], logspace many-one [Jon], one-way logspace many-one [HIM]. We find it astounding that SAT remains NP complete via reductions that are provably much weaker than L.

Fact 3.10 ([Dah]) *SAT is complete for NP via quantifier-free projections.*[3]

4 Operators and Normal Forms

Many natural complete problems for other important complexity classes remain complete via first-order projections and sometimes even quantifier-free projections. The following example involves variants of the GAP problem: 1GAP in which there is at most one edge leaving any vertex, and AGAP, an alternating version of GAP in which the vertices are marked "and" or "or". The notion of reachability for alternating graphs is defined inductively as for alternating Turing machines. Thus the node t is reachable from an "or" node x if t is reachable from one of x's children; t is reachable from an "and" node if it is reachable from all of the "and" node's children.

Fact 4.1 ([Imm87]) *1GAP, GAP, AGAP are complete for* L, NL, P, *respectively via quantifier-free projections.*

We next discuss the proof of Fact 4.1 in [Imm87] because it sheds light on quantifier-free projections and when they exist. Each of the problems 1GAP, GAP, and AGAP have an operator version which is called DTC, TC, and ATC, respectively. For example, $\mathrm{TC}[\varphi, \bar{s}, \bar{t}]$ is a formula in the language $(\mathrm{FO} + \mathrm{TC})$ meaning that there is a "φ-path" from $s_1, \ldots s_k$ to $t_1, \ldots, t_k$ where a "φ-edge" exists from $x_1, \ldots, x_k$ to $x_{k+1}, \ldots, x_{2k}$ iff $\varphi(x_1, \ldots, x_{2k})$ holds.

Fact 4.1 was then proved by demonstrating that the languages $(\mathrm{FO}+\mathrm{DTC})$, $(\mathrm{FO}+\mathrm{TC})$, and, $(\mathrm{FO}+\mathrm{ATC})$ each satisfy the qfp normal form property (Definition 4.4).

We now put Fact 4.1 in a more general setting. First, for any problem whatsoever, Θ, we associate an operator of the same name:

Definition 4.2 (Operator Form of a Problem) Let σ and τ be vocabularies, and let $\Theta \subset \mathrm{STRUC}[\tau]$ be any problem. Let I be any first-order interpretation with $\hat{I} : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$. Then $\Theta[I]$ is a well-formed formula in the language $(\mathrm{FO} + \Theta)$ with the semantics:

$$\mathcal{A} \models \Theta[I] \quad \Leftrightarrow \quad \hat{I}(\mathcal{A}) \in \Theta.$$

[3] Dahlhaus states his result for quantifier-free interpretations, but he actually constructs a quantifier-free projection. We conjecture that for any "well behaved" complexity class, problems complete for quantifier-free interpretations are also complete for quantifier-free projections, cf. Conjecture 7 in §6.

Thus for all the iterated multiplication problems $\prod(M)$ that we have been discussing, we have a corresponding operator, also written $\prod(M)$. Note that these operators are a generalization of the "monoidal quantifiers" of [BIS].

As an example, let us look at the operator form of the GAP problem, cf. Example 3.5. Suppose we have a formula $\varphi(x_1, x_2)$ with free variables x_1, x_2 and we want to express the existence of a φ-path from s to t. We construct the interpretation $I = \lambda_{x_1,x_2}\langle\varphi, s, t\rangle$. Then $\mathrm{GAP}[I]$ is a formula in the language $(\mathrm{FO} + \mathrm{GAP})$ whose meaning is that there is a φ-path from s to t.

Adding the operator Θ to first-order logic corresponds to having full access to the power of the problem Θ. This justifies the following,

Definition 4.3 (First-Order Turing Reductions) Given problems S and T we will say that S is *first-order Turing reducible* to T ($S \leq_t^{\mathrm{fo}} T$) iff $S \in (\mathrm{FO} + T)$.

See Theorem 8.1 of [BIS] where four other conditions are given that are equivalent to $S \leq_t^{\mathrm{fo}} T$, including the condition that S has DLOGTIME uniform circuits that are AC^0 circuits except that they may also include T gates.

Let $\mathcal{C}$ be a complexity class that is closed under first-order Turing reductions. It follows immediately from Definition 4.3 that problem Θ is complete for complexity class $\mathcal{C}$ via first-order Turing reductions if and only if $\mathcal{C} = (FO+\Theta)$, cf. Proposition 4.8.

Now if the problem Θ is complete for $\mathcal{C} = (\mathrm{FO}+\Theta)$ via a lower-level reduction such as fop or qfp this means that we do not need the full power of arbitrary applications of Θ in $(\mathrm{FO} + \Theta)$. In other words, a normal form theorem for $(\mathrm{FO} + \Theta)$ applies:

Definition 4.4 We say that the language $(\mathrm{FO} + \Theta)$ has the *fop Normal Form Property* iff every formula $\varphi \in (\mathrm{FO} + \Theta)$ is equivalent to a formula ψ, where,

$$\psi \quad = \quad \Theta[\alpha]$$

where α is a first-order projection (cf. Equation 1). If α is a quantifier-free projection then we say that $(\mathrm{FO} + \Theta)$ has the *qfp Normal Form Property.*

As an example, we prove the following.

Proposition 4.5 *Let* $\mathbf{F}_2$ *be the field with two elements. The language* $(\mathrm{FO} + \prod(M_n(\mathbf{F}_2)))$ *has the qfp Normal Form property.*

Proof By induction on the structure of $\varphi \in (\mathrm{FO} + \prod(M_n(\mathbf{F}_2)))$ we prove that

$$\varphi \quad = \quad \prod(M_n(\mathbf{F}_2))[\langle\alpha, \bar{0}\rangle] \tag{2}$$

where $\langle\alpha, \bar{0}\rangle$ is a quantifier-free projection.

The most interesting parts of the proof are the following three cases of the induction. In each case we may assume inductively that ψ is in the form of Equation 2.

($\varphi = \neg\psi$): Here we want to add 1 to the $(0,0)$ entry of the product. This can be done by incrementing the dimension n by 1 and carrying along a 1 in the (n,n) entry of the product so that it can be added to other entries as needed. (Of course, in the present formulation, increasing n by 1 can only be done by increasing n all the way to n^2. cf. Remark 3.9.)

($\varphi = (\forall y)\psi$): Here we have n matrix products, and we want to assert that all of them have entry $(0,0)$ equal to 1. Thus, we pre and post multiply each product with the matrix $E_{0,0}$ which has a single non-zero entry in position $(0,0)$ and then we multiply the resulting n products together. (This is the same as zeroing out all but the first row of each first matrix and all but the first column of each last matrix, cf. α', below.) Clearly the $(0,0)^{\text{th}}$ entry of the result is 1 iff φ holds.

($\varphi = \prod(M_n(\mathbf{F}_2))(\psi)$): In this last case, ψ has three free variables: $\psi(x,i,j)$ represents a product whose $(0,0)$ entry is 1 iff entry (i,j) of matrix x of the product to be done is 1. Our inductive assumption is that ψ is of the form,

$$\psi(x,i,j) \quad = \quad \prod(M_n(\mathbf{F}_2))[\langle\alpha(x',i',j';x,i,j),\bar{0}\rangle]$$

where $\langle\alpha,\bar{0}\rangle$ is a quantifier-free projection. Note that the product is over the variables x',i',j', but the additional variables x,i,j also occur freely in α.

We project the input onto a product of $n^3 \times n^3$ matrices A_x, $x = 0,\ldots,n-1$. The upper left $n \times n$ corner of A_x will consist of the matrix given by the formula $\psi(x,i,j)$.

This is achieved as follows. First let E_x be the product of n $n^3 \times n^3$ matrices whose entries consist of n^2 $n \times n$ matrices along the diagonal, computing $\psi(x,i,j)$. In symbols,

$$\begin{aligned} E_x &= \prod_{x'=0}^{n-1} F_{x,x'}, \qquad \text{where} \\ F_{x,x'}(n^2i+nj+i', n^2i+nj+j') &= \alpha(x',i',j';x,i,j) \end{aligned}$$

Thus, entry (n^2i+nj, n^2i+nj) of E_x is 1 iff $\psi(x,i,j)$ holds.

To complete the construction, we get $A_x = L * E_x * R$ where L, R are the matrices that move all the required entries to the upper $n \times n$ corner. Explicitly,

$$\begin{aligned} L(a,b) &= 1 \text{ iff } a < n \;\&\; n|b \;\&\; n^2a \le b < n^2(a+1) \\ R(a,b) &= 1 \text{ iff } b < n \;\&\; n^2|(a-bn) \end{aligned}$$

We have presented the proof for the above three cases by describing the matrices informally rather than writing out the necessary quantifier-free formulas.

Our experience is that writing out these formulas doesn't really help to get the idea across. However, because it might help, we next write an explicit formula for the second case: $\varphi = (\forall y)\psi$. We hope that the reader will write down just enough of the other formulas to convince herself that it can be done in a straight forward way.

We assume that ψ is in the form of Equation 2, namely,

$$\varphi \quad = \quad (\forall y)\prod(M_n(\mathbf{F}_2))[\langle\alpha(x,i,j;y),\bar{0}\rangle]$$

where α is a quantifier-free projection in which the variable 'y' occurs freely. The following is a quantifier-free projection, α', that puts φ into the required form:

$$\begin{aligned}\alpha'(x_1,x_2,i_1,i_2,j_1,j_2) \quad \equiv \quad & [i_2 = j_2 = 0 \wedge \alpha(x_2,i_1,j_1;x_1) \\ & \wedge (x_2 \neq 0 \vee i_1 = 0) \wedge (x_2 \neq \mathbf{m} \vee j_1 = 0)]\end{aligned}$$

Note, that all we are doing is stringing together the n products,

$$\prod(M_n(\mathbf{F}_2))(\alpha(x,i,j;y)), \qquad y = 0,1,\ldots,n-1$$

For each of these products we zero out all but the first row of the first matrix and all but the first column of the last matrix in order to only save the $(0,0)$ entry of the product.

We comment that BIT has not yet been considered in this proof. In fact what we have shown is that everything in $(\mathrm{FO(wo\ BIT)} + M_n(\mathbf{F}_2))$ is a qfp of $\prod(M_n(F_2))$. Thus it remains to show that $\mathrm{BIT} \in (\mathrm{FO(wo\ BIT)} + \prod(M_n(F_2)))$. But this follows because $\mathrm{BIT} \in (\mathrm{FO(wo\ BIT)} + \mathrm{DTC})$ [Imm87]; and,

$$(\mathrm{FO(wo\ BIT)} + \mathrm{DTC}) \quad \subseteq \quad (\mathrm{FO(wo\ BIT)} + \prod(M_n(F_2)))$$

This last inclusion is obvious because DTC is applicable only when the number of paths under consideration is either 1 or 0, and $\prod(M_n(F_2))$ gives the parity of the number of paths. ∎

The following corollary is immediate from Proposition 4.5:

Corollary 4.6 *Let* $\oplus\mathrm{L}$ *be the class of problems, S, of the form: for some* NL *machine, M,*

$$S \;=\; \{x \mid M \text{ has an odd number of accepting paths on input } x\}$$

Then the class $(\mathrm{FO} + \prod(M_n(\mathbf{F}_2)))$ *is equal to* $\oplus L$. *Furthermore,* $\prod(M_n(\mathbf{F}_2))$ *is complete for* $\oplus L$ *via quantifier-free projections.*

The following lemma was proved in [Imm87], (except that there $(\mathrm{FO} + \mathrm{TC})$ was written as $(\mathrm{FO} + \mathrm{pos\ TC})$ because we didn't yet know that this was equal to $(\mathrm{FO} + \mathrm{TC})$). The proof is by induction on the structure of the formula φ, in order to show that it can be massaged into the form of Definition 4.4. Fact 4.1 is then an immediate corollary.

Lemma 4.7 *Each of the languages* (FO + DTC), (FO + TC), *and,* (FO + ATC) *satisfy the qfp normal form property. Furthermore,* (FO + DTC) = L, (FO + TC) = NL, *and,* (FO + ATC) = P.

The following proposition summarizes the above discussion.

Proposition 4.8 *Let Θ be a problem and $\mathcal{C}$ a complexity class that is closed under first-order Turing reductions. Then*

1. *Θ is $\leq_t^{\text{fo}}$-complete for $\mathcal{C}$ if and only if $\mathcal{C} = (\text{FO} + \Theta)$.,*
2. *Θ is $\leq_{\text{fop}}$-complete for $\mathcal{C}$ if and only if $\mathcal{C} = (\text{FO} + \Theta)$ and $(\text{FO} + \Theta)$ has the fop normal form property, cf. Definition 4.4.*
3. *Θ is $\leq_{\text{qfp}}$-complete for $\mathcal{C}$ if and only if $\mathcal{C} = (\text{FO} + \Theta)$ and $(\text{FO} + \Theta)$ has the qfp normal form property, cf. Definition 4.4.*

5 Completeness Proofs

We next show that the iterated multiplication problems that we have been discussing remain complete for their complexity classes via first-order projections. The first theorem is a refinement of the result from [Coo85] that iterated Boolean matrix multiplication is complete for NL* via NC^1 reductions.

Theorem 5.1 *Iterated Boolean matrix multiplication is complete for* NL *via quantifier-free projections.*

Proof First observe that $\prod(M_n(\text{bool})) \in \text{NL}$. This can be seen as follows: The matrices A_i, $i = 1, \ldots, n$ represent a graph with $n+1$ levels. The edges from level $i-1$ to level i are given by the adjacency matrix A_i. Entry (c_1, c_2) of $\prod_i A_i$ is a one iff there is a path from vertex c_1 level 0 to vertex c_2 in level n.

By Fact 4.1 the GAP problem is complete for NL via quantifier-free projections. Example 3.5 gives a quantifier-free projection from GAP to $\prod(M_n(\text{bool}))$. It follows by the transitivity of $\leq_{\text{qfp}}$ (Lemma 3.7) that $\prod(M_n(\text{bool}))$ is complete for NL via quantifier-free projections. ∎

Theorem 5.2 *Iterated multiplication of elements of S_n is complete for* L *via first-order projections.*

Proof In [CM] it is shown that this problem is complete for L via NC^1 reductions. By modifying Cook and McKenzie's argument we will show that $\text{1GAP} \leq_{\text{fop}} \prod(S_n)$. The theorem then follows from Fact 4.1.

Let G be an instance of 1GAP, i.e. a directed graph with at most one edge leaving any vertex, no edges entering 0 and no edges leaving $\mathbf{m}$. *We will* *consider G as an undirected graph.* Note that $G \in$ 1GAP iff 0 and $\mathbf{m}$ are in the same undirected connected component of G. Furthermore, in this case this component is acyclic.

Give each edge of G two labels – one for each adjacent vertex. Consider the following set of permutations on these labels: for each vertex v take a cycle of all edges adjacent to v, $\rho_v = (e_v f_v \cdots)$; for each e, flip the two labels of e, $\varphi_e = (e_x e_y)$. Now let π_G be the product of all the ρ's times all the φ's,

$$\pi_G \;=\; \prod_{v \in V} \rho_v \cdot \prod_{e \in E} \varphi_e \tag{3}$$

Let e_x be an edge label from G. It is not hard to see that the sequence $e_x, \pi_G(e_x), \pi_G^2(e_x), \ldots$ is a depth first traversal of the connected component of G containing x. (This is Proposition 1 in [CM].) It follows that for any graph G, $G \in$ 1GAP iff for some r, π_G^r maps the edge leaving 0 to an edge entering $\mathbf{m}$.

Note that in this form we do not yet have a many-one reduction because we are asking n questions to $\prod(S_n)$ – one for each value of r – instead of a single question. This problem is solvable as follows: We modify G by attaching a tail of length n to $\mathbf{m}$. More precisely, let $G = \langle\{0, \ldots, n-1\}, E\rangle$ and let $G' = \langle\{0, \ldots, 2n-1\}, E'\rangle$, where

$$E' \;=\; \{\langle 2a, 2b\rangle \mid \langle a, b\rangle \in E\} \,\cup\, \{\langle 2n-2, 1\rangle, \langle 1, 3\rangle, \langle 3, 5\rangle, \ldots, \langle 2n-3, 2n-1\rangle\}$$

Let e_0 be a label of the unique edge leaving 0. Observe that a vertex v has low order bit one iff v is on the tail. It follows that $G \in$ 1GAP iff the low order bit of $\pi_G^n(e_0)$ is 1.

To complete the proof we must show that the permutation $\pi_{G'}$ is a first-order projection of G. For an edge $e = (a, b)$, we will code the edge label e_a as the pair (a, b). According to the definition of $\pi_{G'}$ in Equation 3, $\pi_{G'}$ should map (a, b) to (c, a), where (a, c) is the next edge after (a, b). In order to make this a first-order projection we change the definition slightly,

$$\pi'(a, b) \quad = \quad \begin{cases} (b+1, a) & \text{if } E'(a, b+1) \vee E'(b+1, a) \\ (a, b+1) & \text{otherwise} \end{cases}$$

Thus π' maps possible edge labels cyclically around a vertex until it finds an actual edge which it then takes. One effect of this change is that it now takes about n^2 steps to traverse all the edges in G. Finally, as desired, we have that

$$(G \in \text{1GAP}) \quad \Leftrightarrow \quad \Big((\prod_{i=1}^{n^2} \pi')(0, 0)[[0]] = 1 \Big)$$

To complete the proof of Theorem 5.2 there are two details that we must now take care of. Namely, we assumed that the input graph G has outdegree one and then we considered its undirected version. In fact, we can build two qfps I_a and I_b with the following properties for any H,

a. $(H \in 1\text{GAP})$ iff $(\hat{I}_a(H) \in 1\text{GAP})$ iff $(\hat{I}_a(H) \in \text{GAP})$.

b. $(\hat{I}_b(\hat{I}_a(H))$ is an undirected graph and $(\hat{I}_b(\hat{I}_a(H)) \in \text{UGAP})$ iff $(H \in 1\text{GAP})$

Here UGAP is the undirected version of GAP. Thus we can take $G = \hat{I}_b(\hat{I}_a(G_0))$ in the above proof, where G_0 is the input graph.

The constructions of I_a and I_b are easy. In I_a we walk through the whole graph counting the number of edges out of each vertex and essentially die if there are two edges leaving some vertex. In the following, a is a vertex. We cycle through all the vertices b and increment the counter c if an edge (a, b) is discovered. Then, once all these tests are passed we connect to a copy of G_0.

$$
\begin{aligned}
\alpha(abc, a'b'c') \quad \equiv \quad & c \neq \mathbf{m} \wedge \big[b' = b+1 \wedge a' = a \wedge c = c' \neq \mathbf{m} \wedge \neg E(a,b) \\
& \vee b' = b+1 \wedge a' = a \wedge c = 0 \wedge c' = 1 \wedge E(a,b) \\
& \vee b = \mathbf{m} \wedge b' = 0 \wedge a' = a+1 \wedge c' = 0 \wedge (c = 0 \vee \neg E(a,b)) \\
& \vee a' = 0 \wedge b' = c' = \mathbf{m} \wedge a = b = \mathbf{m} \wedge (c = 0 \vee \neg E(a,b))\big] \\
& \vee (c = c' = \mathbf{m} \wedge b = b' = \mathbf{m} \wedge a \neq \mathbf{m} \wedge a' \neq 0 \wedge E(a,a'))
\end{aligned}
$$

For I_b, we make the graph undirected by adding a second coordinate – a counter:

$$
\begin{aligned}
\beta(ab, a'b') \quad \equiv \quad & ((b' = b+1 \wedge E(a,a')) \vee (b = b'+1 \wedge E(a',a)) \\
& \vee (a = a' = \mathbf{m} \wedge (b = b'+1 \vee b' = b+1)))
\end{aligned}
$$

∎

Theorem 5.3 *The problem $\prod(S_5)$, that is, iterated multiplication of elements of S_5, is complete for* NC^1 *via first-order projections. The same is true for* $\prod(M_3(\text{bool}))$ *and* $\prod(M_3(\mathbf{F}_2))$.

Proof This follows from Barrington's Theorem [Bar] together with a few observations about uniformity from [BIS]. In particular, Proposition 6.4 in [BIS] says that $\prod(S_5)$ is complete for NC^1 via DLOGTIME reductions. From [BIS] we also know that DLOGTIME is contained in FO.

An examination of the proof in [BIS] shows that the DLOGTIME reduction is in fact a projection: Each choice of which element of S_5 to take depends on a single bit of the input. Furthermore, determining which bit it depends on is a DLOGTIME and thus first-order computation.

We note that Barrington's construction and thus this proof goes through for any finite monoid such as $\prod(M_3(\text{bool}))$ or $\prod(M_3(\mathbf{F}_2))$ that contains a non-solvable group. See also [BC] where it is shown that any "algebraic NC^1"

circuit over some ring R can be written as an iterated multiplication problem over $M_3(R)$. In particular, since NC^1 = boolean NC^1, and boolean algebra embeds in $\mathbf{F}_2$, the result from [BC] gives an alternate proof that $\prod(M_3(\mathbf{F}_2))$ is projection complete for NC^1. ∎

Define fop(DET) to be the closure of DET under first-order projections. Recall that DET* is the closure of DET under NC^1 reductions. It is obvious that fop(DET) $\subseteq$ DET* but it remains open whether or not these two classes are equal. (See Conjecture 1 in the list of open problems and conjectures in the next section.) We prove the following:

Theorem 5.4 *The problem $\prod(M_n(\mathbf{Z}))$, that is, iterated integer matrix multiplication is complete for* fop(DET) *via first-order projections.*

Proof It suffices to write DET as a first-order projection of $\prod(M_n(\mathbf{Z}))$. Actually Berkowitz [Ber] has already done all of the hard work for us. Let A be an $n \times n$ integer matrix. Berkowitz showed how to write the coefficients, p_i, of the characteristic polynomial,

$$p(\lambda) \;=\; p_0\lambda^n + \cdots + p_{n-1}\lambda + p_n \;=\; \det(A - \lambda I)$$

in the form

$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \prod_{t=1}^{n} C_t \,.$$

Here each C_t is an $(n+2-t) \times (n+1-t)$ matrix which as we will see is a product of n first-order projections of the initial matrix.

More explicitly, the matrix C_t is given as follows:

$$C_t = \begin{pmatrix} -1 & 0 & 0 & \cdots & 0 \\ a_{tt} & -1 & 0 & \ddots & 0 \\ -R_t \cdot S_t & a_{tt} & -1 & \ddots & 0 \\ -R_t \cdot M_t \cdot S_t & -R_t \cdot S_t & a_{tt} & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & -1 \\ -R_t \cdot M_t^{n-t-1} \cdot S_t & -R_t \cdot M_t^{n-t-2} \cdot S_t & \cdots & \cdots & a_{tt} \end{pmatrix}$$

where the matrices S_t, R_t, and M_t are the following submatrices of A:

$$R_t \;=\; \begin{pmatrix} a_{t,t+1} & \cdots & a_{t,n} \end{pmatrix}$$

$$S_t = \begin{pmatrix} a_{t+1,t} \\ \vdots \\ a_{n,t} \end{pmatrix} \quad M_t \;=\; \begin{pmatrix} a_{t+1,t+1} & \cdots & a_{t+1,n} \\ \vdots & \vdots & \vdots \\ a_{n,t+1} & \cdots & a_{n,n} \end{pmatrix}$$

Define D_t to be the $(n+1) \times (n+1)$ matrix given by C_t in the upper left corner, extended by 0's to fill it out. Now, the determinant of A is entry $(n+1, 1)$ of the product $\prod_{i=1}^{n} D_t$. It thus suffices to show that the D_t's can be written uniformly as iterated products of fops of A. This may be obvious to the reader. If not, observe that we can first compute all the powers of M_t that we need by multiplying the "diagonal" matrices:

$$\begin{pmatrix} M_t & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{pmatrix} * \begin{pmatrix} M_t & 0 & \dots & 0 \\ 0 & M_t & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{pmatrix} * \cdots * \begin{pmatrix} M_t & 0 & \dots & 0 \\ 0 & M_t & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M_t \end{pmatrix}$$

Next, to get the relevant coordinates: $R_t M_t^i S_t$, we multiply on the left and right by the matrices:

$$\begin{pmatrix} L & 0 & \dots & 0 \\ 0 & L & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & L \end{pmatrix}, \quad \begin{pmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{pmatrix}$$

where L is the square matrix whose first row is R_t and whose other rows are zero; and R is the square matrix with first column S_t and the other columns zero.

So now we have a diagonal matrix containing all the coefficients of D_t. Next by increasing the dimension an additional factor of n^2 we can put any of these coefficients anywhere we want them: first by copying the diagonal entries into all relevant columns and then performing the relevant elementary row operations.

Note that the computational power needed to determine for each entry whether it is 0 or 1 or a particular entry of A requires at most addition on coordinate numbers. Thus the whole set of iterated products is a first-order projection as claimed. ∎

6 Conclusions and Conjectures

We have defined first-order projections and shown that natural iterated multiplication problems are complete for various low-level complexity classes via these reductions. We feel that the reductions are sufficiently weak, and preserve enough of the algebraic nature of these problems to permit solutions to some of the problems listed below.

Remark 6.1 (Does the Coding Matter?) *When dealing with very low-level reductions we have to explicitly define the coding of the problem. Our intuitive feeling when we started this work was that the coding does not matter very much* *as long as it is sensibly done. However, upon further reflection, we found that the coding really can matter in some cases. An example from the literature is Theorem 4.2 from [HIM] which says that there are P-complete and NP-complete sets via logspace reductions that – because of the coding – are not complete via one-way logspace reductions. Closer to home, we were unable to prove Theorem 5.2 the way we had originally encoded it: With the coding $R(x,i,j)$, meaning that permutation x maps i to j, it is not at all clear how to show that $\prod(X_n)$ is complete for L via many-one reductions. However, this was easy once we decided to recode by looking at separate bits of $\pi(i)$, rather than forcing the very simple reduction to produce the correct j.*

In some sense the issues of complexity and coding are orthogonal. Once you have proved that a problem S_0 is complete for some complexity class, $\mathcal{C}$, via fops, you have exposed the essence of $\mathcal{C}$ as being identical to that of S_0. However, by contrast, another problem S_1 that is complete for $\mathcal{C}$ via say logspace reductions may fail to be complete for $\mathcal{C}$ via fops because of the encoding, or perhaps for other reasons. From our point of view, this may not matter. What is important is that the complexity class has been tightly captured: for any problem T, T is in $\mathcal{C}$ if and only if $T \leq_{\text{fop}} S_0$. Some of the problems below suggest possible ways of exploiting this sort of situation.

Remark 6.2 (fop's versus qfp's) *Because of the simple form of projections, the difference between fops and qfps is neither more nor less than the numeric predicates such as BIT that are available in the language. Thus, in the presence of the numeric predicates: BIT, PLUS, PLUS MOD 2 OF THE ODD NUMBERED BITS, all of the results of §5 go through for qfps. On the other hand, it is not clear that the relation $\leq_{\text{qfp}}$ remains transitive in the presence of these numeric predicates. We have thus made the choice in this paper to stick to fops as our extremely low-level reduction of choice. Our reason is pragmatic: fops are much easier to handle than qfps and they should be low-level enough for anyone.*

What is now needed is an expanded theory of fops. In particular we would love to see some proofs that no fop exists between certain interesting pairs of problems.

Conjectures, and Open Problems:

1. We conjecture that fop(DET) = DET* = log(DET). (This last class is the set of problems many-one, logspace reducible to DET.) This amounts to proving a qfp normal form theorem for $(\text{FO} + \prod(M_n(\mathbf{Z})))$, cf. Definition 4.4. Note that the difficulty of proving this has to do with separating individual bits of an n-bit entry of some matrix. A typical case is how to write the formula $(\exists y)\prod(M_n(\mathbf{Z}))(\varphi(y))[i,j,r]$ in the form $\prod(M_n(\mathbf{Z}))(\varphi')[i',j',r']$. The first formula says that for some y, the r^{th} bit of the $(i,j)^{\text{th}}$ entry of a product of matrices is 1. The question is then how to write this existential

statement as an arithmetic expression.

Note also that this is intimately connected with a similar kind of bit separation that is important in Toda's theorem, [Tod89]. A solution to this problem would give a simplified and generalized version of that theorem. See [GKT] for related work.

2. $\prod(S_n) \not\leq_{\text{fop}} \prod(S_5)$: This is true iff NC^1 is strictly contained in L.

3. $\prod(\mathbf{Z}) \leq_{\text{fop}} \prod(S_5)$: This is true iff $\prod(\mathbf{Z}) \in \mathrm{NC}^1$.

4. $\prod(S_5) \leq_{\text{fop}} \prod(\mathbf{Z})$: This would imply that NC^1 is contained in polynomial-time uniform ThC^0.

5. We conjecture that DET is complete for ThC^1 and thus that DET* is equal to ThC^1. This would nicely linearize the containments of Figure 1.

6. We conjecture that the integer permanent problem is not a first-order projection of the integer determinant problem, in symbols, PERM $\not\leq_{\text{fop}}$ DET. This is true iff PERM $\notin \mathrm{fop}(\prod(M_n(\mathbf{Z})))$. Thus this result would imply among other things that NL does not contain #P, cf. [vzG, Cai].

7. We conjecture that a low-level version of the "isomorphism conjecture" [BH], [You] should hold: For the complexity classes $\mathcal{C}$ = NL, P, NP, all problems complete for $\mathcal{C}$ via first-order projections are logspace isomorphic.[4]

8. It would be very nice to consider "more algebraic" versions of the reductions we are using, i.e. perhaps some versions of these reductions that are actually homomorphisms would suffice. This would allow more algebraic theory to be brought to bear in deciding when such reductions exist.

9. In a similar vein, it might be fruitful to restrict our attention to groups rather than monoids. This can presumably be done for the monoids of matrices by restricting attention to the special linear groups – the submonoid of invertible matrices with determinant one.

10. We would like to add the following monoids to our chart between problems $\prod(M_n(\text{bool}))$ and $\prod(M_n(\mathbf{Z}))$: (1) $M_n(\mathbf{F}_p)$, for fixed p, and, (2) $M_n(\mathbf{F}_p)$, for $p \leq n$. It is obvious that (1) $\leq_{\text{fop}}$ (2) $\leq_{\text{fop}} \prod(M_n(\mathbf{Z}))$. It is also interesting to note that Chinese remaindering plus (2) gives $\prod(M_n(\mathbf{Z}))$, in symbols: $\prod(\mathbf{Z}) + (2) = \prod(M_n(\mathbf{Z}))$. We do not know whether any of the containments between these classes are strict.

11. We would like to know whether or not $\prod(M_n(\text{bool})) \leq_{\text{fop}} \prod(M_n(\mathbf{F}_2))$, cf. Corollary 4.6.

[4] Very recently this conjecture has been proved, in fact, all problems complete for such $\mathcal{C}$ via fops are first-order isomorphic, [ABI].

We hope that first-order projections become a useful tool for applying algebraic methods to prove complexity theoretic lower bounds. For example, suppose that T is complete for a complexity class $\mathcal{C}$ via first-order projections and let S be any problem. Then S is a member of $\mathcal{C}$ if and only if $S \leq_{\text{fop}} T$. Furthermore, a proof that there is no k-ary first-order projection from S to T is a lower bound on the complexity of S.

Let us make this last statement more precise. First, generalize the notion of k-ary projection to be a map from structures of size n to structures of size $O[n^k]$, where the new universe is a bounded union of k-tuples. Now, suppose that T is complete for linear time via linear first-order projections. It should follow that S is a k-ary first-order projection of T iff S is doable in time n^k. For this reason we feel that first-order projections are a promising approach for obtaining non-trivial lower bounds.

Acknowledgments. Thanks to two anonymous referees who saved us embarrassment by pointing out some gaps in the logic of an earlier version of this paper. Thanks to Eric Allender, David Mix Barrington, and Zhi-Li Zhang for helpful comments and corrections.

References

[All88] Eric Allender (1988), Isomorphisms and 1-L Reductions, *J. Computer Sys. Sci. 36*, 336-350.

[All89] Eric Allender (1989), P-Uniform Circuit Complexity, *JACM,* **36**, 912 - 928.

[ABI] E. Allender,N. Immerman,J. Balcázar (1993), A First-Order Isomorphism Theorem, *STACS*, 163-174.

[Bar] David Barrington (1989), Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1, *JCSS 38*), 150-164.

[BIS] D. Barrington, N. Immerman, H. Straubing (1990), On Uniformity Within NC^1, *JCSS 41, No. 3*, 274 - 306.

[BCH] Paul Beame, Steve Cook, H. James Hoover (1986), Log Depth Circuits for Division and Related Problems, *SIAM J. Comput. 15:4*, 994-1003.

[BM] Martin Beaudry and Pierre McKenzie (1992), Circuits, matrices, and nonassociative computation, *7th Structure in Complexity Theory Symp.*, 94-106.

[BLM] F. Bédard, F. Lemieux and P. McKenzie (1990), Extensions to Barrington's M-program model, *5th IEEE Structure in Complexity Theory Symp.*, 200-210. Revised version to appear in *Theoret. Comp. Sci.*

[BC] Michael Ben-Or and Richard Cleve (1992), Computing Algebraic Formulas Using A Constant Number of Registers, *SIAM J. Comput. 21*, 54-58.

[Ber] Stuart Berkowitz (1984), On Computing Determinant in Small Parallel Time Using a Small Numbers of Processors, *Information Processing Letters,* **18**, No. 3, 147-150.

[BH] Len Berman and Juris Hartmanis (1977), On Polynomial Time Isomorphisms of Complete Sets, *Proc. Third Theor. Comput. Sci. GI Conf.,* (Springer-Verlag LNCS No. 48), 1-15.

[BDHM] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel (1992), Structure and Importance of Logspace-MOD Class, *Math. Systems Theory*25, 223-237.

[Cai] Jin-yi Cai (1990), A Note on the Determinant and Permanent Problem, *Information and Computation,* **84**, No. 1, 119-127.

[Coo71] Stephen Cook (1971), The Complexity of Theorem¡ Proving Procedures, *Proc. Third Annual ACM STOC Symp.*, 151-158.

[Coo85] Stephen Cook (1985), A Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control* **64**, 2-22.

[CM] Stephen Cook and Pierre McKenzie (1987), Problems Complete for Deterministic Logarithmic Space, *J. Algorithms* **8**, 385-394.

[Dah] Elias Dahlhaus (1984), Reduction to NP-Complete Problems by Interpretations, Lecture Notes In Computer Science, Springer-Verlag, 357-365.

[End] Herbert Enderton (1972), *A Mathematical Introduction to Logic,* Academic Press.

[GKT] F. Green, J. Köbler,J. Torán (1992), The Power of the Middle Bit, *Seventh IEEE Structure in Complexity Theory Symp.*, 111-117.

[HIM] Juris Hartmanis, Neil Immerman, and Stephen Mahaney (1978), One-Way Log Tape Reductions, *19th IEEE FOCS Symposium*, 65-72.

[Imm87] Neil Immerman (1987), Languages That Capture Complexity Classes, *SIAM J. Comput.* **16**, No. 4, 760-778.

[Imm88] Neil Immerman (1988), Nondeterministic Space is Closed Under Complementation, *SIAM J. Comput.* **17**, No. 5, 935-938.

[Imm89] Neil Immerman (1989), Descriptive and Computational Complexity, *Computational Complexity Theory,* ed. J. Hartmanis, *Proc. Symp. in Applied Math.,* 38, American Mathematical Society, 75-91.

[Imm89b] Neil Immerman (1989), Expressibility and Parallel Complexity, *SIAM J. of Comput* **18**, 625-638.

[Jon] Neil Jones (1973), Reducibility Among Combinatorial Problems in Log n Space, *Proc. Seventh Annual Princeton Conf. Info. Sci. and Systems*, 547-551.

[Kar] Richard Karp (1972), Reducibility Among Combinatorial Problems, in *Complexity of Computations,* R.E.Miller and J.W.Thatcher, eds., Plenum Press,85-104.

[Lin] Stephen Lindell (1992), A Purely Logical Characterization of Circuit Uniformity, *Seventh IEEE Structure in Complexity Theory*, 185-192.

[LG] Laszlo Lovász and Peter Gács (1977), Some Remarks on Generalized Spectra, *Zeitchr. f. math, Logik und Grundlagen d. Math,* Bd. 23, 547-554.

[Rei] John Reif (1987), On Threshold Circuits and Polynomial Computation, *Second Annual Structure in Complexity Theory Symp.*, 118-123.

[Ruz] Larry Ruzzo (1981), On Uniform Circuit Complexity, *J. Comp. Sys. Sci.,* **21**, No. 2, 365-383.

[Ste] Iain Stewart (1992), Using the Hamiltonian Operator to Capture NP, to appear in *J. Comput. System Sci.*

[Sze] Róbert Szelepcsényi (1988), The Method of Forced Enumeration for Nondeterministic Automata, *Acta Informatica* **26**, 279-284.

[Tod89] Seinosuke Toda (1989), On the Computational Power of PP and $\oplus$P,*30th IEEE FOCS Symp.*, 514-519.

[Val] L.G. Valiant (1982), Reducibility By Algebraic Projections, *L'Enseignement mathématique,* **28**, 3-4, 253-68.

[vzG] Joachim von zur Gathen (1987), Permanent and Determinant, *Linear Algebra Appl.* **96**, 87-100.

[You] Paul Young (1990), Juris Hartmanis: Fundamental Contributions to Isomorphism Problems, in *Complexity Theory Retrospective,* Alan Selman, ed., Springer-Verlag, 28-58.