The complexity of matrix rank and feasible systems of linear equations (Extended Abstract)

Eric Allender* Robert Beals† Mitsunori Ogihara‡

Abstract

We characterize the complexity of some natural and important problems in linear algebra. In particular, we identify natural complexity classes for which the problems of (a) determining if a system of linear equations is feasible and (b) computing the rank of an integer matrix, (as well as other problems), are complete under logspace reductions.

As an important part of presenting this classification, we show that the "exact counting logspace hierarchy" collapses to near the bottom level. (We review the definition of this hierarchy below.) We further show that this class is closed under NC^1-reducibility, and that it consists of exactly those languages that have logspace uniform span programs (introduced by Karchmer and Wigderson) over the rationals.

In addition, we contrast the complexity of these problems with the complexity of determining if a system of linear equations has an *integer* solution.

1 Introduction

The motivation for this work comes from two quite different sources. The first and most obvious source is the desire to understand the complexity of problems in linear algebra; our results succeed in meeting this goal. The other, less obvious, source is the desire to understand the power of threshold circuits and enumeration problems. Although our results do not actually help much in this regard, this motivation is responsible for some of the notation used later, and thus we start by explaining this side of things.

*Department of Computer Science, Rutgers University, P.O. Box 1179, Piscataway, NJ 08855-1179, allender@cs.rutgers.edu Supported in part by NSF grant CCR-9509603.

†DIMACS, Rutgers University, PO Box 1179, Piscataway, NJ 08855-1179, and School for Mathematics, Institute for Advanced Study, Olden Lane, Princeton, NJ 08540, rbeals@dimacs.rutgers.edu. Supported in part by an NSF Mathematical Sciences Postdoctoral Fellowship.

‡Department of Computer Science, University of Rochester, Rochester, NY, 14627, ogihara@cs.rochester.edu

STOC'96, Philadelphia PA,USA

1.1 Complexity Classes for Counting and Enumeration

The counting hierarchy (sometimes denoted CH) is the complexity class $PP \cup PP^{PP} \cup PP^{PP^{PP}} \cup \ldots$. (Here, PP is unbounded-error probabilistic polynomial time [Gi77].) Although the counting hierarchy was originally defined in order to classify the complexity of various problems [Wa86], another reason to study CH comes from the connection with threshold circuits. Using the analogous correspondence between constant-depth circuits and the polynomial hierarchy established by [FSS84], it is known that constructing an oracle separating PSPACE from CH is essentially the same problem as showing that NC^1 properly contains TC^0 (the class of problems computable by constant-depth threshold circuits of polynomial size). Similarly, the important question of whether or not the TC^0 hierarchy collapses is related in this way to the question of whether or not CH collapses.

Since $P^{PP} = P^{\#P}$, an equivalent way to define CH is by $P \cup P^{\#P} \cup P^{\#P^{\#P}} \cup \ldots$. In proving results about the complexity of PP and #P and related classes, it has often proved more convenient to use the related class of functions GapP [FFK], which is the set of functions that can be expressed as the difference of two #P functions.

One final complexity class related to CH needs to be defined. A number of authors have studied the class $C_=P = \{A \mid \exists f \in \text{GapP}, \text{such that } x \in A \Leftrightarrow f(x) = 0\}$. Note that $C_=P$ can also be characterized in terms of "exact counting"; a language A is in $C_=P$ iff there is an NP machine M and a poly-time-computable g such that, for all $x, x \in A$ iff the number of accepting computations of M on input x is exactly $g(x)$. Since PP is contained in $C_=P^{C_=P}$, it follows that a third characterization of CH can be given in terms of $C_=P$.

1.2 Logspace Counting Classes

There is no *a priori* reason to expect that logspace analogs of the classes PP, #P, GapP, and $C_=P$ should be interesting, and in fact, with the exception of PL, the related logspace classes remained uninvestigated until fairly recently, when independent discoveries by Vinay, Toda, and Damm showed that #L actually characterizes the complexity of the determinant quite well. More precisely, the following result is essentially shown by Vinay [Vi91, Theorem 6.5], Toda [To91, Theorem 2.1], and Damm [Da91]. (See also [Va92, Theorem 2]; further discussion may be found in [AO94].)

Theorem 1.1 *A function f is in* GapL *iff f is logspace many-one reducible to the determinant.*

It follows immediately from this characterization that a complete problem for PL is the set of integer matrices whose determinant is positive (originally proved by [Ju85]). Of course, checking if the determinant is positive is not nearly as important a problem as checking if the determinant is exactly equal to zero, and it is equally immediate from the foregoing that the set of singular matrices is complete for the complexity class $C_=L$. ($C_=L$ can be defined in any of a number of equivalent ways; see [AO94]. Perhaps the simplest way is to say that a set A is in $C_=L$ if there is an NL machine such that x is in A iff the machine has exactly the same number of accepting and rejecting paths on input x.) Although the machine model for $C_=L$ is not as natural as some, the fact that it exactly characterizes the complexity of the singular matrices makes this a better-motivated class than, say, PL.

Logspace versions of the counting hierarchy were considered in [AO94]. Although the hierarchies defined in terms of $C_=P$, PP, and #P all coincide with CH, there seems to be little reason to believe that the hierarchies defined in terms of $C_=L$, PL, and #L are equal. It was shown in [AO94] that these hierarchies correspond exactly to AC^0 reducibility, in the following sense[1]:

- The Exact Counting Logspace Hierarchy =

$$C_=L^{C_=L^{\cdots^{C_=L}}} = AC^0(C_=L)$$

  = the class of problems AC^0-reducible to the set of singular integer matrices.

- The PL hierarchy = $PL^{PL^{\cdots^{PL}}} = AC^0(PL)$ = the class of problems AC^0-reducible to the problem of computing the high-order bit of the determinant of integer matrices.

- The #L hierarchy = $L^{\#L^{\cdots^{\#L}}} = AC^0(\#L)$ = the class of problems AC^0-reducible to computing the determinant of integer matrices.

Note that all of these classes contain NL and are contained in $TC^1 \subseteq NC^2$. Ogihara [Og96] recently proved that the PL hierarchy collapses to PL.

AC^0 reducibility is a restricted form of the NC^1 reducibility defined and studied in [Co85]. For example, Cook defined DET to be the class of problems NC^1 reducible to the determinant, and thus his class DET contains the #L hierarchy.

1.3 Main results

We show that the exact counting logspace hierarchy collapses to $L^{C=L}$. It collapses all the way to $C_=L$ if and only if $C_=L$ is closed under complement. We further show that $NC^1(C_=L) = L^{C=L}$, and that this class consists of exactly those language with logspace uniform span programs over the rationals (cf. [KW93]).

[1]When defining classes in terms of space-bounded oracle Turing machines, one needs to be careful how access to the oracle is provided. We use the "Ruzzo-Simon-Tompa" access mechanism [RST], which dictates that a nondeterministic Turing machine must behave deterministically while *writing* on its oracle tape. One consequence of using this definition is that we may assume without loss of generality that the list of queries asked by the machine depends only on the input x and does not depend on the answers given by the oracle [RST]. The correspondence with AC^0 reducibility helps justify this choice.

We show that testing feasibility of a system of linear equations is complete for this hierarchy. Another complete problem for this class is computing the rank of a matrix (or even determining the low order bit of the rank).

In contrast, verifying that a matrix has a particular rank is complete for a level of the Boolean hierarchy over $C_=L$.

This is the first time that the complexity of these well-studied problems in linear algebra has been so precisely characterized.

It should be noticed that there are several other classes $\mathcal{C}$ for which it has been shown that $NC^1(\mathcal{C})$ is equal to $L^{\mathcal{C}}$. In particular, there is a superficial resemblence between our result showing $NC^1(C_=L) = L^{C=L}$, and the result of [Og95] that $NC^1(C_=P)$ is equal to $L^{C=P}$. Also, Gottlob [Go96] has recently studied the question of which classes $\mathcal{C}$ satisfy $AC^0(\mathcal{C}) = L^{\mathcal{C}}$. (Our results imply that $C_=L$ has this property.) However the techniques of [Og95, Go96] do not carry over to complexity classes with small space bounds such as $C_=L$, and thus our proofs are correspondingly more complex.

2 Complexity of Problems in Linear Algebra

We will focus mainly on the following problems concerning integer matrices: verifying that the rank of a matrix is r, computing the rank of a matrix, and determining if a system of linear equations is feasible.

$$Ver.RANK = \{(A, r) \mid A \in \mathbf{Z}^{m\times n}, r \in \mathbf{N}, \mathrm{rank}(A) = r\}.$$

$Comp.RANK =$
$\{(A, i, b) \mid A \in \mathbf{Z}^{m\times n}, \mathrm{rank}(A) = r, \text{ and bit } i \text{ of } r \text{ is } b\}$

$FSLE =$
$\{(A, b) \mid A \in \mathbf{Z}^{m\times n}, b \in \mathbf{Z}^{m\times 1}, \exists x \in \mathbf{Q}^{n\times 1} \mid Ax = b\}.$

(*FSLE* stands for Feasible Systems of Linear Equations.) (The analogous problems for rational matrices have the same complexity in our setting; we will not mention them further in this extended abstract.)

We show that

- *FSLE* and *Comp.RANK* are complete for $L^{C=L}$, and these problems are equivalent to the problem of determining if the rank is odd. (Note that all of these problems are thus complete for the entire Exact Counting Logspace Hierarchy, $AC^0(C_=L)$, since it collapses to this level.)

- *Ver.RANK* is complete for the second level of the Boolean Hierarchy above $C_=L$ (i.e., the class of all sets expressible as the intersection of a set in $C_=L$ and a set in $coC_=L$).

2.1 The Complexity of Rank

In this section we present the basic results concerning the complexity of verifying the rank of integer matrices. We will make frequent use of the many results that are already known about the parallel complexity of problems in linear algebra. An excellent survey article covering this area has been written by von zur Gathen [vzG93].

Proposition 2.1 *The set*

$$\{(M, r) \mid M \in Z^{n \times n} \text{ and } rank(M) < r\}$$

is complete for $C_=L$.

Proof: Recall that the set of singular matrices is complete for $C_=L$, and thus this set remains hard for this class even when $r = n$. To see that it is in $C_=L$, let χ_M denote the characteristic polynomial of M, with coefficients $c_0, \ldots, c_n$, so $\chi_M(x) = \sum_{i=0}^{n} c_i x^i$. It was observed in [IMR80] that the problem of determining if $\text{rank}(M) \leq r$ is reducible to the problem of whether the coefficients $c_0, c_1, \ldots, c_{n-r-1}$ in $\chi_{M'}$ (for some M' easily computable from M) are all zero. (In fact, even over arbitrary fields F, checking if $\text{rank}(M) \leq r$ is reducible to checking to see if several coefficients of a characteristic polynomial are zero. This is mentioned in [ST94, Theorem 10], and can be shown to follow from [vzG93, Theorem 13.7].[2]) Since it follows easily from observations in [To91] that the function $f(M, i) ::=$ the coefficient c_i of χ_M is a function in GapL, and hence that the set $\{M, i \mid f(M, i) = 0\}$ is in $C_=L$, it follows that the set of matrices with rank at most r is logspace-ctt-reducible to a set in $C_=L$. Since $C_=L$ is closed under logspace-ctt reductions [AO94], the result follows.□

A more interesting question than asking if the rank of M is less than r is asking if it is *equal* to r (and even more interesting is the problem of *computing* the rank). In order to classify the problem of verifying the rank, it is necessary to define some additional complexity classes.

It is not known if $C_=L$ is closed under complement. Thus, just as has been done with complexity classes such as NP [CGH*88, CGH*89], one can define the *Boolean Hierarchy* over $C_=L$, defined as the class of languages that can be formed by taking Boolean combinations of languages in $C_=L$.[3] Of particular interest to us will be the class that contains all sets that are the difference of two sets in $C_=L$.

Definition 1 *Let* $C_=L \wedge coC_=L$ *be the set of all languages* A *such that there exist* $B \in C_=L$ *and* $C \in coC_=L$ *such that* $A = B \cap C$.

Theorem 2.2 *The sets*

$$\{(M, r) \mid M \in Z^{n \times n} \text{ and } rank(M) = r\}$$

and

$$\{M \mid M \in Z^{n \times n} \text{ and } rank(M) = n - 1\}$$

are complete for $C_=L \wedge coC_=L$.

Proof: It follows easily from Proposition 2.1 that these problems are in $C_=L \wedge coC_=L$. Thus it suffices to show completeness. To do this, it will be useful to have the following lemma, which is perhaps interesting in its own right.

[2]The full paper will contain a detailed proof of this observation, as well as a discussion clarifying the complexity of the problem "INDEPENDENCE" (i.e., determining if a set of vectors is linearly independent or not). In the integer setting, this is easily seen to be complete for $coC_=L$. In the more general setting working over rings with unity, [vzG93] lists as an open problem the question of whether INDEPENDENCE is reducible to the problem of whether a matrix is singular. The full paper will show that this can be done with a conjunctive truth-table reduction.

[3]It can be verified that this Boolean hierarchy over $C_=L$ in fact coincides with the class V-DET defined and studied in [ST94].

Lemma 2.3 *There is a logspace-computable function* f *such that if* M *is a matrix of full rank, then so is* $f(M)$, *and if* M *is a matrix with determinant zero, then* $f(M)$ *is a matrix of rank exactly one less than full.*

First we will show that the lemma provides a proof, and then afterward we will prove the lemma.

Let $A = B \cap C$ where $B \in C_=L$ and $C \in coC_=L$. Since the set of singular matrices is complete for $C_=L$, on input x we can compute matrices M_1 and M_2 such that $x \in A$ iff $det(M_1) = 0$ and $det(M_2) \neq 0$. By Lemma 2.3 we can compute matrices M_3 and M_4 such that $x \in A$ iff $\text{rank}(M_3)$ is one less than full and the $\text{rank}(M_4)$ is full. (Note also that $x \notin A$ iff either $\text{rank}(M_3)$ is full or $\text{rank}(M_4)$ is one less than full.) Thus $x \in A$ if and only if the matrix

$$\begin{bmatrix} M_3 & & \\ & M_4 & \\ & & M_4 \end{bmatrix}$$

has rank one less than full. This completes the proof of the theorem. □

Proof: (of Lemma 2.3) Since the determinant of a matrix is a GapL function, it follows that given a matrix M one can easily construct directed acyclic graphs G and H such that the determinant of M is (the number of s-t paths in G) − (the number of s-t paths in H), and it is easy to construct these graphs such that any s-t path in either graph has even length. Now build a graph K consisting of G and H, together with two new vertices a and b, where K has all of the edges of G and H, along with self-loops on all vertices *except* a, and also having edges from a to the start vertices of G and H, and from the end vertex of G to b, from b to a, and from the end vertex of H to a.

It is shown in [To91] (modifying an older argument due to [Va79]) that the determinant of K (i.e., the determinant of the adjacency matrix of the graph K) is equal to the determinant of M. (The reason is that the only non-zero terms in the canonical expression of the determinant of K correspond to disjoint cycles that cover the vertices of K. However, the only way to cover K by disjoint cycles is to have one cycle corresponding to an s-t path in either G or H, with all other vertices on self-loops. Closer examination shows that, since the length of cycles through G have odd length and the cycles through H have even length, the positive contribution to the determinant comes entirely from the s-t paths in G, and the negative contribution comes from the s-t paths in H.)

It is not clear what the rank of K is. However, let us now create a graph K' consisting of K along with two new vertices, giving exactly one new s-t path in G and one new s-t path in H. That is, let the two new vertices be c, d, and put self-loops on c and d, and add edges from the start vertex of G to c, from the start vertex of H to d, from c to the end vertex of G, and from d to the end vertex of H. Note that the determinant of K' is equal to the determinant of K, since we have added exactly one new s-t path on each side of K to obtain K'. Thus if M has full rank, so does K'. However if M has determinant zero, so does K', but if we delete the edge leaving vertex d then we obtain a matrix with nonzero determinant (since we've removed one of the s-t paths in H but not in G). Note that this implies that K' must have had rank only one less than full, since changing one bit changes only one of the columns of K'. That completes the proof of the lemma. □

It will be useful later on to observe that the following fact holds.

Fact 2.4 *The language*

$$\{(A, B, r) \mid r \text{ is the rank of both } A \text{ and } B\}$$

is in $\mathrm{C_=L} \wedge co\mathrm{C_=L}$.

Proof: This can easily be expressed as the intersection of sets checking (1) $\mathrm{rank}(A) = r$, and (2) $\mathrm{rank}(B) = r$. Note that $\mathrm{C_=L} \wedge \mathrm{coC_=L}$ is easily seen to be closed under intersection. □

2.2 Feasible Systems of Linear Equations

In this section we introduce one of the complete languages for $\mathrm{L(C_=L)}$, and give some preliminary reductions. The proof of completeness is in the next section.

Definition 2 *Let FSLE denote the language* $\{(A, b) \mid A \in \mathbf{Z}^{n\times n}, b \in \mathbf{Z}^{n\times 1}, \exists x \in \mathbf{Q}^{n\times 1} \mid Ax = b\}$.

Proposition 2.5 *The language FSLE is logspace many-one reducible to its complement.*

Proof: Consider the system of equations $Ax = b$, and let W be the subspace spanned by the columns of A. The system is feasible iff $b \in W$. From elementary linear algebra we know that b can be written uniquely as $b = v + w$, where v is perpendicular to W (i.e., $v^T A = 0$) and $w \in W$. If $v \neq 0$, then since $v^T w = 0$ we have $v^T b = v^T v > 0$, and we may let $y = (1/v^T v)v$. We have shown that if $Ax = b$ is infeasible, then there exists y such that $y^T A = 0$ and $y^T b = 1$. Conversely, if such a y exists then $Ax = b$ is infeasible. The linear equations specifying y are logspace-computable from A and b, as desired. □

The above shows how to "negate" a system of linear equations. We remark that other logical operations can in some sense be performed on systems of linear equations. For example, suppose that we are given two systems, $Ax = b$ and $Cy = d$, and we wish to make a system that is feasible iff both original systems are feasible (i.e., we wish to compute the logical AND of the two systems). The system

$$\begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}$$

is exactly what we want. To construct the logical OR of two systems, we note that an OR gate can be built out of three negation gates and an AND gate. It is useful to carry this observation a little further, for which we need the following:

Definition 3 *A* logspace dtt reduction *from A to B is a function f, computable in logspace, such that for all x, f(x) produces a list of strings* $(y_1, y_2, \ldots, y_r)$*, with the property that* $x \in A$ *iff at least one of the* y_i *is in B ("dtt" stands for "disjunctive truth table" reducibility). Similarly, one defines "conjunctive truth table reducibility" (ctt reductions). A more general type of reduction is the following. An* $\mathrm{NC^1}$ reduction *[Ba91] is a uniform sequence of circuits* $\{C_n\}$ *of size* $n^{O(1)}$ *and depth* $O(\log n)$*, consisting of fan-in two AND and OR gates, NOT gates, and "oracle gates", with the property that no path from input to output goes through more than one oracle gate.*

Expanding on the observations in the previous paragraph easily shows:

Lemma 2.6 *The class of languages logspace many-one reducible to FSLE is closed under* $\mathrm{NC^1}$ *reductions.*□

We now give some relationships between *FSLE* and $\mathrm{C_=L}$, using the results on rank from the previous section.

Lemma 2.7 *FSLE is logspace dtt reducible to the class* $\mathrm{C_=L} \wedge co\mathrm{C_=L}$.

Proof: Note that $Ax = b$ is feasible iff the matrices A and (Ab) have the same rank. So feasibility can be expressed as a disjunction, for all $0 \leq r \leq n$, of the statement that A and (Ab) have rank r. The lemma now follows by Fact 2.4. □

Lemma 2.8 *Suppose A is logspace dtt reducible to* $\mathrm{C_=L} \wedge co\mathrm{C_=L}$. *Then A is logspace many-one reducible to FSLE.*

Proof: Let M be a square matrix. Then M is nonsingular iff there exists a square matrix X such that $MX = I$, where I is the identity matrix. Observe that this is a system of linear equations in the entries of X. Since testing singularity of a matrix is complete (wrt logspace many-one reductions) for $\mathrm{C_=L}$, Lemma 2.6 completes the proof. □

Theorem 2.9 *FSLE is complete for the class of languages logspace dtt reducible to* $\mathrm{C_=L} \wedge co\mathrm{C_=L}$. *This class is closed under* $\mathrm{NC^1}$ *reductions.*

Proof: Completeness follows from the preceding two Lemmas. Closure under $\mathrm{NC^1}$ reductions is by Lemma 2.6. □

Corollary 2.10 *Comp.RANK, Odd.RANK, and FSLE are equivalent under logspace many-one reductions.*

Proof: First we reduce *FSLE* to *Odd.RANK*. As noted above, the system $Ax = b$ is feasible iff A and (Ab) have the same rank. In addition, if $Ax = b$ is infeasible, then the rank of (Ab) is exactly one more than the rank of A. Therefore, $Ax = b$ is feasible iff the rank of

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A & b \end{pmatrix}$$

is even. Thus, *FSLE* is reducible to *Odd.RANK* (and this problem, in turn, is trivially reducible to *Comp.RANK*).

Now we reduce *Comp.RANK* to *FSLE*. Let M be an $m \times n$ matrix. Consider the following set of systems of linear equations: for $0 \leq i < n$ let A^i denote the first i columns of M, and let b^i denote column $i + 1$ of M. Then $\mathrm{rank}(M)$ is exactly the number of infeasible systems among the systems $A^i x = b^i$. Since counting can be done in $\mathrm{NC^1}$, by Lemma 2.6 any bit (or negation of a bit) of $\mathrm{rank}(M)$ can be expressed as the feasibility of a logspace-computable system of linear equations, as desired. □

2.3 Span programs

The span program model of computation was introduced by Karchmer and Wigderson [KW93]. A span program on n Boolean variables $x_1, \ldots, x_n$ consists of a target vector b in some vector space V, together with a collection of $2n$ subspaces $U_z \leq V$, for each literal $z \in \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ (each subspace is represented by a possibly redundant generating set). The language accepted by the span program is the set of n-bit strings for which b lies in the span of the union of the U_z, for those true literals z. The complexity of

the span program is the sum of the dimensions of the U_z for all z.

For a language A, it is clear that if the n-bit strings of A are accepted by a logspace computable span program over the rationals, then A is logspace reducible to *FSLE*. We shall see that the converse is true as well. In what follows, we will continue to use x_i to denote the bits of a binary string (which may or may not be in some language A). We will use $y_1, \ldots, y_\ell$ to denote the variables in a system $My = b$ obtained from x such that $x \in A$ iff $My = b$ is feasible (So the matrix M is a function of the x_i).

To begin with, let A be a language in $C_=L$. Then A is logspace many-one reducible to the set of singular matrices over the rationals. In fact, the reduction for #L (cf. [To91]) has the property that strings $x_1x_2\ldots x_n$ of length n are transformed into a matrix whose entries are all linear functions of the x_i: indeed, only literals and constants in $\{0,1\}$ occur.

By examining the constructions used in the proof of Lemma 2.8, we conclude:

Lemma 2.11 *Suppose A is logspace dtt reducible to $C_=L \wedge coC_=L$. Then A is logspace many-one reducible to FSLE, and this reduction has the following form: strings $x_1x_2\ldots x_n$ of length n are reduced to a system $My = b$, where the vector b is constant (i.e., depends only on n) and the matrix entries are linear functions of the x_i with logspace uniform coefficients.*□

To arrive at a span program for A, we need to pursue this a little further. A span program is essentially a system $My = b$ where b is a constant and each column of M depends only on a single variable x_i. The space U_{x_i} is spanned by the columns which depend on x_i, evaluated at $x_i = 1$, while $U_{\neg x_i}$ is spanned by these same columns evaluated at $x_i = 0$. We wish to obtain such a system by modifying the system $My = b$ from the above Lemma. Our construction will increase the number of rows and columns polynomially: if M is an $m \times \ell$ matrix, then we will obtain a matrix M' with $n\ell$ columns and $m + (n-1)\ell$ rows.

For simplicity we begin with the $\ell = 1$ case of the construction, so assume M is a single column. We can easily represent M as a sum $M = v_1 + v_2 + \ldots + v_n$, such that each v_i depends only on x_i. Then $My = b$ is feasible iff b is a linear combination of the v_i with all coefficients equal. So we are trying to solve the following system:

$$\sum y_i v_i = b,$$

$$y_1 = y_2 = \ldots = y_n.$$

This amounts to adding $n-1$ variables and $n-1$ constraints to the original system. This generalizes to the $\ell \geq 1$ case quite naturally: each column of M is replaced by n columns, each variable in y is replaced by n variables, which are constrained to be equal by appending $n-1$ rows to the matrix.

We have shown:

Theorem 2.12 *A language $A \subseteq \{0,1\}^*$ has logspace uniform span programs over the rationals iff it is logspace reducible to FSLE.*□

Since the span program model is also studied in the setting of non-uniform circuit complexity, we should say a few words about non-uniform span programs. In particular, since the only measure of interest in the non-uniform model is the *number* of vectors (and the size of each vector is not counted) these results do not immediately draw a connection between non-uniform span programs and non-uniform versions of $L^{C=L}$. It is easy to see that the number of components in a vector is not a source of difficulty; a potentially more difficult problem is posed by span programs with entries with large numerators and/or denominators. If we measure the size of a (non-uniform) span program over the rationals as the sum of (1) the sum of the dimensions of the U_z for all z, and (2) the maximum number of bits required to represent any single entry in the program, then polynomial-size span programs over the rationals characterize $L^{C=L}$/poly, which is also equal to the class of languages reducible to the set of singular matrices via non-uniform AC^0 or NC^1 reductions.

3 Collapse of the hierarchy

In this section we prove the collapse of the $C_=L$ hierarchy, by showing that $L^{C=L} = NC^1(C_=L)$. We shall make use of the following:

Lemma 3.1 *Let $A \in coC_=L$. Then there is a $B \in coC_=L$ such that A is logspace many-one reducible to B, and there is a machine N witnessing that $B \in coC_=L$ such that the the input tape of N is one-way.*

Proof: Let M be an oblivious logspace machine witnessing that $A \in coC_=L$, and let p be a polynomial such that on inputs of length n, M scans the input tape $p(n)$ times. Let N be a one-way machine that takes an input $x_1\#x_2\#\ldots\#x_m\#$, and simulates M, using x_i for the ith scan of M's input tape. If the strings x_i do not all have the same length, or if $m \neq p(|x_1|)$, then N generates both an accepting and rejecting computation. Otherwise, N accepts iff the simulation of M does. Let B be the set of inputs for which N has nonzero Gap. Then $B \in coC_=L$, and A is reducible to B via the reduction $x \mapsto x\#x\#\ldots x\#$, where the string $x\#$ is repeated $p(|x|)$ times. □

Theorem 3.2 $L^{C=L} = NC^1(C_=L)$. *FSLE is complete for this class.*

Proof: The forward inclusion is obvious since $L^{C=L}$ is easily contained in the $C_=L$ hierarchy, and since every AC^0 reduction is an NC^1 reduction.

Let B be logspace-uniform NC^1 reducible to a language $A \in coC = L$. Let N be a nondeterministic Turing machine witnessing that A is in coC=L. By Lemma 3.1, we may assume that N has a one-way input tape.

Let $\{C_n\}_{n\geq 1}$ be a logspace-uniform NC^1-circuit family that reduces B to A. For simplicity, let n be fixed and let $x \in \Sigma^n$ be a string whose membership in B we are testing. Without loss of generality, we may assume that constants 0 and 1 are given as input bits in addition to the actual input string x.

By definition of $NC^1(C_=L)$, the product of the degrees of the nodes on any root-leaf path of C_n is bounded by a polynomial in n (i.e., the sum of the logs of the degrees is $O(\log n)$. Therefore, (by duplicating polynomially many gates) we may assume that C_n is a tree. For simplicity, we may assume that each gate of C_n is an oracle gate. These assumptions do not affect logspace uniformity.

Now for each oracle gate g in C_n, we assign weight $R(g)$ of 2^m, where m is the number of oracle gates in C_n between

g and the root (the output gate). Clearly, $R(g)$ is bounded by some polynomial in n and so, the sum of the weights is bounded by some polynomial in n. Let $q(n)$ be a polynomial bounding the sum of the weights.

Define M to be the machine that, on input (x, m), behaves as follows: First, M sets variable s to m. Next M guesses the output of C_n. Then M starts traversing the tree C_n by a depth first search. When M visits a new node, say g, M guesses the output of g and does the following:

- If the guessed output of g is 1, then M subtracts $R(g)$ from s and starts simulating N on the input of g. Since N is one-way on the input tape, the simulation is done by visiting the children of g from left to right. When M proceeds to a new bit of g's input, the subtree rooted at the corresponding child of g is visited, and on returning to g the guessed bit is used in the simulation of N.
- If the guessed output of g is 0, then M traverses the trees corresponding to the inputs of g, but does not simulate N.
- If g is an input gate or an additional constant gate, then g checks whether the guessed bit for g is correct. If not, then M aborts all the simulations and tree traversing and then guesses one bit r to accept if and only if $r = 0$.

Also, M holds a one-bit parity counter *par*, which is set to 0 at the beginning. When M finishes one simulation of N, if M ends up in a rejecting state, then *par* is flipped. When M finishes traversing all the nodes in C_n, then if the counter s is not equal to zero, then M flips one more bit b and accepts iff $b = 1$. Otherwise, if s is equal to zero, then M accepts if and only if *par* is 0.

Note that M can be logspace bounded: the space required by the simultaneous simulation of several computations of N's is bounded by $O(\mathrm{Depth}(C_n))$; only $O(\log n)$ many guessed bits have to be maintained, and traversing the tree also requires only $O(\log n)$ many bits.

Define X_1 to be the language in coC$_=$L defined by the gap function with respect to M: (x, m) belongs to X_1 if and only if M on (x, m) has a nonzero gap. Let m_x be the largest m such that (x, m) is in X_1. Also, define M' to be the machine that behaves as M does except for guessing 1 as the output of C_n and define X_2 to be the language in coC=L characterized by the gap of M'. Then we will see that $x \in B$ if and only if $(x, m_x) \in X_2$, which implies $B \in \mathrm{L}(\mathrm{C_=L})$.

Note that M can be viewed as a machine that, on input x, m, guesses a collection H of oracle gates in C_n so that the sum of the weight of the gates in H equal to m (the collection H is exactly the set of gates with guessed value 1). For a fixed H, the Gap generated by M is $\mathrm{Gap}_N(y_1) \cdots \mathrm{Gap}_N(y_m)$, where $g_1, \cdots, g_m$ is an enumeration of all the gate in H, and the string y_i is the string appearing in the gate g_i if exactly those gates in H output 1.

Let Z_x be the collection of all oracle gates of C_n that output 1 on input x and let n_x be the sum of the weight of all gates in Z_x. We will show that $n_x = m_x$.

If M guesses Z_x as H, then the gap generated for H is non-zero, since all of the y_i will belong to A and therefore the factor $\mathrm{Gap}_N(y_i)$ will be nonzero. Let Z be a collection not equal to Z_x whose weight sum is at least n_x. By construction, the weight of any gate is greater than the sum of the weights of all of its ancestors. Therefore, there is a gate g in $Z \setminus Z_x$ such that for every gate h below g, h is in Z_x if and only if h is in Z. Let u be the string that is assumed to be the input for the gate g in the simulation of N when M guesses Z_x as H. Clearly, u is the actual query string. So, $gap_N(u) = 0$. On the other hand, when M guesses Z as Z_x, by the assumption that each oracle gate below g is in $(Z \cap Z_x) \cup ((\bar{Z}) \cap (\bar{Z}_x))$, the input string that M simulates is u. So, the gap generated with respect to Z becomes 0 whether or not the traverse is finished.

Thus, $n_x = m_x$. Now, the only difference between M and M' is that M' guesses 1 as the output of C_n. So, C_n outputs 1 if and only if M' can generate nonzero gap on input (x, m_x). So, $x \in B$ iff for some $m \leq q(|x|)$, $(x, m) \in X_2$ and (for all $i \geq m$, $(x, i) \notin X_1$). Since X_1 and X_2 are in coC$_=$L, and since C$_=$L is closed under conjunctive and disjunctive reductions, this shows that B is logspace dtt reducible to C$_=$L $\wedge$ coC$_=$L. Therefore, by Lemma 2.8, B is logspace many-one reducible to *FSLE*. □

4 Integer Solutions

In contrast to the problems considered above, the problem of determining if a system of linear equations has an *integer* solution (*IFSLE*) is not known to have a parallel algorithm at all. This problem is at least as hard as determining if two integers are relatively prime, since the equation $ax+by = 1$ has an integer solution iff $(a, b) = 1$. In fact, Kaltofen [Ka95] has pointed out to us that recent work by Giesbrecht [Gi95] can be used to show that *IFSLE* is RNC-equivalent to the problem of determining if $\mathrm{GCD}(x_1, \ldots, x_n) = \mathrm{GCD}(y_1, \ldots, y_n)$.

In addition, it is not too hard to show that the problem of determining if the determinant of an integer matrix is equivalent to i mod p is many-one reducible to *IFSLE*. Thus a P-uniform NC^1 reduction can use Chinese Remaindering to compute the exact value of the determinant. This shows that #L is P-uniform NC^1-reducible to *IFSLE*, in contrast to what we are able to show for *FSLE*.

5 Open Questions

The most obvious open question is: *Is* C$_=$L *closed under complement?* This happens if and only if the set of singular matrices can be reduced to the set of nonsingular matrices. Just as the complementation results of [Im88, Sz88, NT95] have led to useful insights, we believe that a positive answer to this question would be extremely interesting.

Does the #L *hierarchy collapse?* Given the collapse of the other two logspace counting hierarchies, it is tempting to guess that this hierarchy also collapses. Recall that this hierarchy is the class of problems AC^0-reducible to the determinant.

Is NC^1(PL) = AC^0(PL)*?* (This question has recently been answered, in the affirmative, by Beigel [Be95], who also shows that NC^1(PP) = AC^0(PP).)

Acknowledgments

We wish to thank E. Kaltofen and L. Fortnow for insightful comments related to this paper.

References

[AO94] E. Allender and M. Ogihara, *Relationships among PL, #L, and the determinant* Proc. 9th IEEE

Structure in Complexity Theory Conference, 1994, pp. 267–278.

[Ba91] J. Balcázar, *Adaptive logspace and depth-bounded reducibilities*, Proc. 6th IEEE Structure in Complexity Theory Conference, 1991, pp. 240–254.

[Be95] R. Beigel, personal communication.

[CGH*88] J-Y Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, *The Boolean Hierarchy I: Structural Properties*, SIAM J. Comput. **17** (1988) 1232-1252.

[CGH*89] J-Y Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, *The Boolean Hierarchy II: Applications*, SIAM J. Comput. **18** (1989) 95-111.

[Co85] S. Cook, *A taxonomy of problems with fast parallel algorithms*, Information and Control **64** (1985) 2–22.

[Da91] C. Damm, $DET = L^{\#L}$?, Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universität zu Berlin, 1991.

[FFK] S. Fenner, L. Fortnow, and S. Kurtz, *Gap-definable counting classes*, Journal of Computer and System Sciences 48 (1994) 116–148.

[FSS84] M. Furst, J. Saxe, and M. Sipser, *Parity, circuits, and the polynomial-time hierarchy*, Mathematical Systems Theory **17** (1984) 13–27.

[vzG93] J. von zur Gathen, *Parallel linear algebra*, in J. Reif, ed, *Synthesis of Parallel Algorithms*, Morgan Kaufmann, 1993, pp. 574–615.

[Gi95] M. Giesbrecht, *Fast computation of the Smith normal form of an integer matrix*, Proceedings ISSAC '95.

[Gi77] J. Gill, *Computational complexity of probabilistic Turing machines*, SIAM Journal on Computing 6 (1977) 675–695.

[Go96] G. Gottlob, *Collapsing Oracle-Tape Hierarchies*, to appear in Proceedings of Complexity '96.

[IMR80] O. Ibarra, S. Moran, and L. Rosier, *A note on the parallel complexity of computing the rank of order n matrices*, Information Processing Letters 11 (1980) 162.

[Im88] N. Immerman, *Nondeterministic space is closed under complementation*, SIAM J. Comput. **17** (1988) 935–938.

[Ju85] H. Jung, *On probabilistic time and space*, Proc. 12th ICALP, Lecture Notes in Computer Science 194, 1985, pp. 310–317.

[Ka95] E. Kaltofen, personal communication.

[KW93] M. Karchmer, A. Wigderson, *On Span Programs* Proc. 8th IEEE Structure in Complexity Theory Conference, 1993, pp. 102–111.

[NT95] N. Nisan and A. Ta-Shma, *Symmetric logspace is closed under complement*, STOC '95, pp. 140–146.

[Og95] M. Ogihara, *Equivalence of* NC^k *and* AC^{k-1} *closures of* NP *and other classes*, Information and Computation **120** (1995) 55–58.

[Og96] M. Ogihara, *The PL Hierarchy collapses*, these proceedings.

[RST] W. Ruzzo, J. Simon, and M. Tompa, *Space-bounded hierarchies and probabilistic computation*, Journal of Computer and System Sciences 28 (1984) 216–230.

[ST94] M. Santha and S. Tan, *Verifying the determinant*, Proc. COCOON '94, LNCS.

[Sz88] R. Szelepcsényi, *The method of forced enumeration for nondeterministic automata*, Acta Informatica **26** (1988) 279–284.

[To91] S. Toda, *Counting problems computationally equivalent to the determinant*, Technical Report CSIM 91-07, Dept. Comp. Sci. and Inf. Math., Univ. of Electro-Communications, Tokyo, 1991.

[Va79] L. G. Valiant, *Completeness classes in algebra*, Proc. 11th ACM Symposium on the Theory of Computation, 1979, 249–261.

[Va92] L. Valiant, *Why is Boolean complexity theory difficult?* in *Boolean Function Complexity*, edited by M. S. Paterson, London Mathematical Society Lecture Notes Series 169, Cambridge University Press, 1992.

[Vi91] V. Vinay, *Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits*, Proc. 6th IEEE Structure in Complexity Theory Conference (1991) 270–284.

[Wa86] K. Wagner, *The complexity of combinatorial problems with succinct input representation*, Acta Informatica 23 (1986) 325–356.