DIMACS Technical Report 97-40
September 1997

The complexity of matrix rank and feasible systems of linear equations[1]

by

Eric Allender[2,3]
Dept. of Computer Science
Rutgers University
Piscataway, New Jersey 08855
allender@cs.rutgers.edu.

Robert Beals[4]
Department of Mathematics
University of Arizona
Tucson, AZ 85712
beals@math.arizona.edu

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

[1]A preliminary version of this work appeared in the proceedings of the 1996 ACM Symposium on Theory of Computing.
[2]Permanent Member
[3]Supported in part by NSF grant CCR-9509603. Part of this work was done while a visiting scholar at the Institute of Mathematical Sciences, Madras, India, and at the Wilhelm-Schickard Institut für Informatik, Universität Tübingen (supported by DFG grant TU 7/117-1).
[4]Research conducted while visiting IAS and DIMACS and supported in part by an NSF Mathematical Sciences Postdoctoral Fellowship.

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

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.

1.1 Complexity Classes for Counting and Enumeration

The counting hierarchy (sometimes denoted CH) is the complexity class $\mathrm{PP} \cup \mathrm{PP}^{\mathrm{PP}} \cup \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} \cup \ldots$. (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).[1] Similarly, the important question of whether or not the TC^0 hierarchy collapses is related to the question of whether or not CH collapses.

Since $\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#\mathrm{P}}$, an equivalent way to define CH is by $\mathrm{P} \cup \mathrm{P}^{\#\mathrm{P}} \cup \mathrm{P}^{\#\mathrm{P}^{\#\mathrm{P}}} \cup \ldots$. 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 (beginning with [Wa86]) have studied the class now called $\mathrm{C}_=\mathrm{P}$ (the set of all languages A with the property that there is an f in GapP such that $x \in A \Leftrightarrow f(x) = 0$). Note that $\mathrm{C}_=\mathrm{P}$ can also be characterized in terms of "exact counting"; a language A is in $\mathrm{C}_=\mathrm{P}$ 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 $\mathrm{C}_=\mathrm{P}^{\mathrm{C}_=\mathrm{P}}$, it follows that a third characterization of CH can be given in terms of $\mathrm{C}_=\mathrm{P}$.

1.2 Logspace Counting Classes

There is no *a priori* reason to expect that logspace analogs of the classes PP, #P, GapP, and $\mathrm{C}_=\mathrm{P}$ 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

[1] More precisely, there is a language in uniform NC^1 that requires uniform TC^0 circuits of size greater than $2^{\log^{O(1)} n}$, if and only if there is oracle separating PSPACE from CH.

6.5], Toda [To91, Theorem 2.1], and Damm [Da91]. (See also [MV97] and [Va92, Theorem 2]; further discussion may be found in [AO96].)

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 $\mathrm{C_{=}L}$. ($\mathrm{C_{=}L}$ can be defined in any of a number of equivalent ways; see [AO96]. Perhaps the simplest way is to say that a set A is in $\mathrm{C_{=}L}$ 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 $\mathrm{C_{=}L}$ 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 [AO96]. Although the hierarchies defined in terms of $\mathrm{C_{=}P}$, PP, and #P all coincide with CH, there seems to be little reason to believe that the hierarchies defined in terms of $\mathrm{C_{=}L}$, PL, and #L are equal. It was shown in [AO96] that these hierarchies correspond exactly to AC^0 reducibility, in the following sense[2]:

- The Exact Counting Logspace Hierarchy =
  $$\mathrm{C_{=}L}^{\mathrm{C_{=}L}^{\cdot^{\cdot^{\cdot^{\mathrm{C_{=}L}}}}}} = \mathrm{AC}^0(\mathrm{C_{=}L})$$
  = the class of problems AC^0-reducible to the set of singular integer matrices.
- The PL hierarchy = $\mathrm{PL}^{\mathrm{PL}^{\cdot^{\cdot^{\cdot^{\mathrm{PL}}}}}} = \mathrm{AC}^0(\mathrm{PL})$ = 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 = $\mathrm{L}^{\#\mathrm{L}^{\cdot^{\cdot^{\cdot^{\#\mathrm{L}}}}}} = \mathrm{AC}^0(\#\mathrm{L})$ = 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 (and hence are contained in 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.

[2]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.

1.3 Main results

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

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 $\mathrm{C_=L}$.

This is the first time that the complexity of these well-studied problems in linear algebra has been so precisely characterized. (Santha and Tan [ST94] studied these same computational problems using a coarser notion of reducibility that blurred the distinctions between the various levels of the exact counting logspace hierarchy and the Boolean hierarchy over $\mathrm{C_=L}$. The emphasis in [ST94] is on exploring the apparent difference in the complexity of such problems as verifying $det(M) = a$ and verifying that $M^{-1} = A$, although the complexity of *computing* the determinant is equivalent to that of matrix inversion.)

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

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

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

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

(*FSLE* stands for Feasible Systems of Linear Equations.)

Remark: We have defined these problems for integer matrices. It is perhaps worth mentioning that the corresponding problems for rational matrices are equivalent to these integer matrix problems under logspace reducibility. This follows easily from the observation that for any rational matrix A (with entries given as pairs of integers) and for any integers a and b, $det(A) = a/b$ if and only if $b(det(A)N) - aN = 0$ where integer N is the product of all of the denominators appearing in A. The function $b(det(A)N) - aN$ is easily seen to be in GapL, and thus checking if the determinant of a rational matrix is equal to a given value is reducible to checking if a zero-one integer matrix is singular. We will not mention rational matrices in the remainder of the paper.

We show that

- *FSLE* and *Comp.RANK* are complete for $\mathrm{L}^{\mathrm{C_=L}}$, 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, $\mathrm{AC}^0(\mathrm{C_=L})$, since it collapses to this level.)

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

2.1 Some Preliminaries

This section is largely a review and restatement of earlier work, although it is not intended to be a detailed survey of parallel computation for linear algebra. We refer the reader to the excellent survey article by von zur Gathen [vzG93] for more detailed coverage. We include this material since some of our constructions require an understanding of this previous work. (In particular, we need Corollaries 2.3 and 2.4, which appear to be new observations.)

As we review below, computing the rank of a matrix is intimately connected with computing the coefficients of the characteristic polynomial of a matrix. This, in turn is no more difficult than iterated matrix multiplication, as can be seen from the work of Berkowitz [Be84].

Theorem 2.1 *[Be84] Given an r-by-r matrix B, there is a logspace-computable reduction that constructs a sequence of m-by-m matrices D_i such that the coefficients $c_0, c_1, ..., c_r$ of the characteristic polynomial of B appear in positions $(1, r+1), ..., (1, 1)$ of the matrix $\prod_i D_i$.*

It is important for our applications to note that this reduction holds not only for integer matrices, but also for matrices over any ring with unity. In particular, this reduction has the property that the entries of the D_i's are either taken from B or taken from the constants $-1, 0, +1$; thus the reduction is also a reduction in the sense of [vzG93], and it is also a projection in the sense of [Va79].

There is also a reduction going the other way: iterated matrix multiplication is no more difficult than the determinant. The following construction goes back at least to [Va79] and the exposition below is similar to that in [To91].

Proposition 2.2 *There is a logspace-computable function that takes as input a sequence of matrices D_i and numbers (a, b), and produces as output a matrix H such that entry (a, b) of $\prod D_i$ is $det(H)$.*

As in Theorem 2.1, in addition to the entries of D_i, the constants we need are only $-1, 0, +1$, thus, this reduction holds for matrices over any ring with unity, and it is a reduction in the sense of [vzG93], and a projection in the sense of [Va79].

Proof: Consider each matrix D_i to be a bipartite graph on vertices arranged into two columns, where entry c in position (k, m) denotes an edge labeled c from vertex k in the first column to vertex m in the second column. The second column of D_i corresponds to the first column of D_{i+1}. Then entry (a, b) in $\prod D_i$ is simply the sum (over all paths from vertex a in the first column to vertex b in the last column) of the product of the labels on the edges on that path.

Now modify this graph by replacing each edge from x to y labeled c by a path of length two, consisting of an edge labeled 1 going from x to a new vertex z, and an edge labeled c going from z to y. (Note that this trivially makes the length of all paths from a to b of even length, without changing the value of the product of the values along any path.)

Next, add a self loop labeled 1 on all vertices *except* vertex b in the last column, and add an edge from vertex b to vertex a, and label this edge with 1. Let H be the matrix encoding this digraph. In the polynomial for the determinant of H, the only terms that don't vanish correspond to ways to cover the vertices by disjoint collections of cycles. In this graph, cycle covers one-to-one correspond to paths from a to b (with other vertices covered by self-loops), since the graph that results by deleting the self-loops and the edge from b to a is acyclic. In any such cycle cover, the single non-trivial cycle in the cover has odd length, and thus it is an even permutation. Thus $det(H)$ is simply the sum (over all paths from vertex a in the first column to vertex b in the last column) of the product of the labels on the edges on that path, as desired.

(We remark that a slightly more complicated construction given in [To91] provides a projection that does not make use of the constant -1, by introducing cycle covers corresponding to odd permutations.) $\square$

Corollary 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.*

(Again, this holds over any ring with unity.)

Proof: By Theorem 2.2, given M, there is a logspace reduction that produces sequence of matrices $D_1, \ldots, D_m$ such that entry $(1, n)$ of $\prod_i D_i$ is the determinant of M. The proof of Proposition 2.2 produces a graph H whose determinant is equal to entry $(1, n)$ of $\prod_i D_i$, and thus the determinant of H is equal to the determinant of M. Except for the edge $(n, 1)$ and the self-loops on vertices 1 through $n - 1$, the graph H is acyclic. Without loss of generality, if (i, j) is an edge, then $j > i$; thus the submatrix given by the first $n - 1$ rows and columns is upper triangular with 1's along the main diagonal, and thus the rank of H is at least $n-1$. $\Box$

It will be useful in later sections to call attention to a few more properties that follow from this same construction:

Corollary 2.4 *There is a logspace-computable function f that takes an r-by-r matrix M as input and outputs a sequence of n-vectors $V = v_1, \ldots, v_{n-1}$ with the following properties:*

- *The vector $(1, 0, 0, \ldots, 0)$ is spanned by V if and only if M is singular.*
- *f is a projection, in the sense that for all r, i, j, and for any r-by-r matrix M the jth coordinate of vector v_i in $f(M)$ is either $0, 1, -1$, or $M_{k,l}$, and this depends only on (r, i, j).*

(Again, this holds for any ring with unity.)

Let us now review some aspects of Mulmuley's algorithm for computing the rank [M87].

Let A be a matrix with entries from some ring K, and let A' be the matrix

$$\begin{bmatrix} 0 & A \\ A^t & 0 \end{bmatrix}$$

Let $B = YA'$ (where Y is the matrix with powers on indeterminate y on the diagonal: $Y_{i,i} = y^{i-1}$ for $1 \leq i \leq n$, $Y_{i,j} = 0$ for $i \neq j$).

As is explained in [vzG93], the following statements are equivalent:

1. $rank(A) \leq k$,
2. $rank(A') \leq 2k$,
3. $rank(B) \leq 2k$,
4. $algebraic.rank(B) \leq 2k$ (i.e., $rank(B^{2n}) \leq 2k$),
5. the first $2n - 2k$ coefficients of the characteristic polynomial of B are all zero, and
6. the first $2n-2k$ coefficients of the characteristic polynomial of B are all zero mod y^{4n^2},

where the entries in B (and hence the coefficients of the characteristic polynomial of B) are all in $K(y)$ (the ring of polynomials over K in indeterminant y). Using the reduction given by Theorem 2.1, it suffices to build a sequence of r-by-r matrices D_i (where r is polynomial in n), where the elements of the D_i are polynomials of degree at most $4n^2$ in $K(y)$, and determine if certain entries of $\prod D_i$ are all zero mod y^{4n^2}.

A polynomial in $K(y)$ of degree at most $d-1$ can be represented by a d-by-d Toeplitz matrix (see [vzG93]), and the zero element in $K(y)$ corresponds to the all-zero matrix. Thus determining if the rank of matrix A is at most k can be performed by building a sequence of $r(4n^2+1)$-by-$r(4n^2+1)$ matrices D'_i with entries from K, and determining if certain entries of $\prod D'_i$ are all zero.

Thus, for the particular case of integer matrices we have the following proposition, which in some sense is implicit in [vzG93] (see also [ST94]):

Proposition 2.5 *The set*

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

is complete for $C_{=}L$.

Proof: Hardness for $C_{=}L$ follows from Theorem 1.1 (even for the case $r = n$). Containment in $C_{=}L$ follows from the preceding discussion, along with the following observations:

- The problem of taking integer matrices D_l and indices i, j and determining if entry i, j of $\prod_l D_l$ is zero is in $C_{=}L$. (For details, see [To91].)
- Hence, the preceding discussion shows that the problem of determining if the rank is at most r is logspace conjunctive-truth-table reducible to a problem in $C_{=}L$.
- $C_{=}L$ is closed under logspace conjunctive-truth-table reductions [AO96].

□

2.1.1 A few comments regarding previous work

Von zur Gathen [vzG93] considers the problem INDEPENDENCE (given a set of vectors decide if they linearly independent), and specifically asks if INDEPENDENCE is reducible to SINGULAR (the set of singular matrices). For rational matrices, these problems are easily seen to be complete for co-$C_{=}L$ and for $C_{=}L$, respectively, so von zur Gathen's question in that setting can be viewed as asking if $C_{=}L$ is closed under complement.

However, in [vzG93] von zur Gathen is more interested in working in the algebraic setting over a given field F, and his notion of "reduction" is more restrictive than logspace reducibilities. More precisely, the reductions in [vzG93] are computed by constant-depth circuits with unbounded fan-in OR and $+$ gates, fan-in two AND and $\times$ gates, and (unbounded fan-in) oracle gates. It is not made clear in [vzG93] if NOT gates are also to be allowed. If NOT gates are allowed, then the restriction of bounded fan-in AND gates can be side-stepped using unbounded fan-in OR gates, via DeMorgan's laws. On the other hand, some of the reductions in [vzG93] (such as in Theorem 13.8) explicitly make use of NOT gates. Without using NOT gates at all, INDEPENDENCE_F (the subscript F indicates the language is the "field F"-version) is clearly many-one reducible to the question of whether a matrix has rank greater than r. We have seen (in the discussion preceding Proposition 2.5) that this problem in turn is many-one reducible to the question, given $D_1, \ldots, D_r$, of whether there is at least one non-zero value in certain positions of $\prod D_i$. Since each of these values can be represented as the determinant of a matrix (again, using a reduction in the sense of [vzG93], it follows that even without using NOT gates, INDEPENDENCE_F is reducible to the complement of SINGULAR_F using a reduction in the sense of [vzG93]. If NOT gates are allowed, then these problems are clearly interreducible, and they are also interreducible to the problem SING.NONSING_F (consisting of two matrices, the first of which is singular, the second nonsingular), and thus the reduction given in our Lemma 2.10 shows that all problems in RANK_F are reducible to $\mathrm{INDEPENDENCE}_F$ and to RANK_F, answering questions listed as open in [vzG93]. If NOT gates are not allowed, the situation remains unclear.

Santha and Tan [ST94] also considered complexity classes defined in terms of reducibility to problems in linear algebra over some field F. The reducibilities considered by Santha and Tan differ from those of [vzG93] in at least two respects: (1) unbounded fan-in AND gates are explicitly allowed, and (2) no path from input to output can encounter more than one oracle gate. (Thus these reductions are what are called AC^0_1 reductions in [AO96] and elsewhere.) The classes they study are DET (which would be called $\mathrm{AC}^0_1(\#\mathrm{L})$ in our notational scheme), and $V\text{-}DET$ (which by definition is $\mathrm{AC}^0_1(\mathrm{C_=L})$, and which we show coincides with the exact counting logspace hierarchy). Santha and Tan also consider problems that are many-one reducible to computing and verifying the determinant, and obtain the classes $m\text{-}DET$ (which is the same as our class GapL), and $m\text{-}V\text{-}DET$ (which is the same as our class $\mathrm{C_=L}$). Santha and Tan consider both *Boolean* and *arithmetic* versions of these problems; an arithmetic circuit computing the rank of n-by-n matrices must work correctly for *all* n-by-n matrices (regardless of the size of the entries), while a Boolean circuit takes the actual encodings of the entries of the matrix as input, and thus a larger circuit will handle n-by-n matrices with entries of 2^n bits than the circuit that handles matrices with entries of n bits. Our results show that *in the Boolean model*, for reductions to the problem $V\text{-}DET$, restriction (2) in the reducibilities of Santha and Tan [ST94] is redundant; the same class of problems results if this restriction is dropped. In the arithmetic case, however, this remains unknown (even in the case when F is the field of rational numbers).

Buntrock et al. [BDHM92] studied algebraic problems over GF[p] for prime p. The proof of Theorem 10 in [BDHM92] states that, over the ring of integers, computation of the determinant is NC^1-reducible to computation of the rank of a matrix, while in fact this remains an open question. (However, what is needed for the applications in [BDHM92] is that these problems are interreducible over GF[p^k], which is true. This is because computation of the determinant can be reduced to checking if it is exactly equal to one of a small number of values.)

2.2 The Complexity of Rank

In this section we present our results concerning the complexity of verifying the rank of integer matrices, building on Proposition 2.5, which characterizes the complexity of verifying that the rank of M is less than r.

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 $\mathrm{C_=L}$ 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 $\mathrm{C_=L}$, defined as the class of languages that can be formed by taking Boolean combinations of languages in $\mathrm{C_=L}$. Of particular interest to us will be the class that contains all sets that are the difference of two sets in $\mathrm{C_=L}$.

Definition 1 *Let* $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$ *be the set of all languages* A *such that there exist* $B \in \mathrm{C_=L}$ *and* $C \in \text{co-}\mathrm{C_=L}$ *such that* $A = B \cap C$.

Theorem 2.6 *The sets*

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

and

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

are complete for $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$.

Proof: It follows easily from Proposition 2.5 that these problems are in $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$. Thus it suffices to show completeness.

Let $A = B \cap C$ where $B \in \mathrm{C_=L}$ and $C \in \text{co-}\mathrm{C_=L}$. Since the set of singular matrices is complete for $\mathrm{C_=L}$, 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 $rank(M_3)$ is one less than full and the $rank(M_4)$ is full. (Note also that $x \notin A$ iff either $rank(M_3)$ is full or $rank(M_4)$ 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. □

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

Fact 2.7 *The language*

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

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

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

2.3 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 2.8 *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 $A^T y = 0$ and $b^T y = 1$. 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* NC^1_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 2.9 *The class of languages logspace many-one reducible to FSLE is closed under NC^1_1 reductions.*□

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

Lemma 2.10 *FSLE is logspace dtt reducible to the class* $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$.

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 2.7. □

Lemma 2.11 *Suppose A is logspace dtt reducible to* $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$. *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 (with respect to logspace many-one reductions) for $\mathrm{C_=L}$, Lemma 2.9 completes the proof. □

Theorem 2.12 *FSLE is complete for the class of languages logspace dtt reducible to* $\mathrm{C_=L} \wedge \text{co-}\mathrm{C_=L}$. *This class is closed under* NC_1^1 *reductions.*

Proof: Completeness follows from the preceding two Lemmas. Closure under NC_1^1 reductions is by Lemma 2.9. □

Corollary 2.13 *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 the complement of *Odd.RANK*, and by Proposition 2.8 *FSLE* is also reducible to *Odd.RANK* (and this problem, in turn, is trivially reducible to *Comp.RANK*).

Now we reduce *Comp.RANK* to *FSLE*. Given (A, i, b), let $S = \{j \le n \mid \text{bit } i \text{ of } j \text{ is equal to } b\}$. Then (A, i, b) is in *Comp.RANK* if and only if $\bigvee_{j \in S}(A, j) \in$ *Ver.RANK*. The result now follows by Lemma 2.11 and Theorem 2.6. □

2.4 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 $U_z \le V$, 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 $\mathrm{C_{=}L}$. Then A is logspace many-one reducible to the set of singular matrices over the rationals. In fact, this reduction has the properties outlined in Corollary 2.4. Thus (since the set of singular matrices is complete for $\mathrm{C_{=}L}$ under projections [To91]), we have that there is a logspace-computable f such that $f(x)$ is a system of linear equations of the form $My = b$ such that

1. $x \in A$ if and only if $My = b$ is feasible.
2. b is the vector $(1, 0, 0, \ldots., 0)$ (and in particular, b depends only on $|x|$).
3. Each entry in M is either $0, 1, -1$, or a literal x_i or $\overline{x_i}$ (and this also depends only on $|x|$).

Using the construction in the proof of Proposition 2.8, we see that for any $\overline{A}$ in co-$\mathrm{C_{=}L}$, an identical conclusion holds. (Note that $x \notin A$ if and only if $(bM)^T z = (1, 0, \ldots, 0)$ is feasible.) For any problem B in $\mathrm{C_{=}L} \wedge$ co-$\mathrm{C_{=}L}$ there is thus a logspace-computable f such that $f(x)$ is a system of linear equations of the form $My = b$ such that

1. $x \in B$ if and only if $My = b$ is feasible.
2. b is of the form $(1, 0, 0, \ldots., 0, 1, 0, \ldots, 0)$ (and in particular, b depends only on $|x|$).
3. Each entry in M is either $0, 1, -1$ or a literal x_i or $\overline{x_i}$ (and this also depends only on $|x|$).

If C is any set that is logspace dtt reducible to a set in $\mathrm{C_{=}L} \wedge$ co-$\mathrm{C_{=}L}$, a corresponding system of linear equations can be constructed by two more applications of the reduction of FSLE to its complement, and one application of taking the AND of several systems of linear equations. Thus we have proved the following:

Lemma 2.14 *Suppose A is logspace dtt reducible to $\mathrm{C_{=}L} \wedge$ co-$\mathrm{C_{=}L}$. Then A is logspace many-one reducible to FSLE, and this reduction has the following form: strings $x_1 x_2 \ldots x_n$ 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 either 0, 1, or a literal x_i or $\overline{x_i}$ (and this also depends only on n).*$\Box$

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 Lemma 2.14. 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 2.15 *A language $A \subseteq \{0,1\}^*$ has logspace uniform span programs over the rationals iff it is logspace reducible to FSLE.*$\Box$

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, it is an important characteristic of the span program model in the non-uniform setting that the only measure of interest is the *number* of vectors (and the size of each vector is not counted). For instance, it is shown in [KW93] that if "small" span programs exist for a problem, then span programs having a certain very restricted form must exist – but this restricted form uses vectors of *exponential* length. It is an important aspect of span programs that having extremely long vectors does not provide additional computational power. Our Theorem 2.15 does not immediately draw a connection between non-uniform span programs and non-uniform versions of $\mathrm{L}^{\mathrm{C_=L}}$. It is easy to see that the number of components in a vector is not a source of difficulty (since there are only a small number of rows in the matrix that are linearly independent); 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 $\mathrm{L}^{\mathrm{C_=L}}$/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.

2.5 Span Programs and the Matching Problem

The span program formalism was used recently in showing that, for every natural number k, the Perfect Matching problem is in the complexity class $\mathrm{Mod}_k\mathrm{L}$/poly [BGW96]. (That is, they show that, for every prime p, there are polynomial-size span programs over GF$[p]$ recognizing the Perfect Matching problem.) Vinay [Vi97] has pointed out that Perfect Matching is also in the class co-$\mathrm{C_{=}L}$/poly, via essentially the same argument. Let us sketch the details here; the main ideas stem from the work of [T47, L79, S80].

Given the adjacency matrix of a graph, replace the 1's in the matrix with indeterminates and negated indeterminates, to obtain the Tutte matrix for the graph. If there is no perfect matching, then the formal polynomial for the determinant is identically zero, and if there is a matching, then the formal polynomial is not identically zero. This polynomial has degree n. Consider random algorithm that (1) picks integers at random in some (exponentially-large) domain, (2) plugs them in for the indeterminates in several independent copies of the matrix, and (3) accepts if and only if all of the resulting matrices are nonsingular. If the determinant is not identically zero, this algorithm has probability exponentially close to 1 of finding a nonsingular matrix, and thus for each input size m there is a sequence of random choices with the property that, for all inputs of size m, the algorithm correctly solves the perfect matching problem when that sequence of random choices is used. This algorithm has the form of a nonuniform dtt reduction to the set of nonsingular matrices.

Proposition 2.16 *Perfect Matching is in* co-$\mathrm{C_{=}L}$*/poly.*

It is an empirical observation that most natural computational problems are complete for some natural complexity class. The Perfect Matching problem is one of the few important problems that has resisted all such attempts at being pigeonholed in this way. The problem is hard for NL. The reduction from 0-1 Network Flow to Perfect Matching given by Karp, Upfal, and Wigderson [KUW86] can be modified to show that the Directed Connectivity problem is reducible to the Perfect Matching problem. Since it is in $\mathrm{Mod}_k\mathrm{L}$ for *all* k (at least nonuniformly), it seems unlikely to be complete for *any* of the $\mathrm{Mod}_k\mathrm{L}$ classes. Similarly, if we assume for the moment that $\mathrm{C_{=}L}$ is not contained in any of the $\mathrm{Mod}_k\mathrm{L}$ classes, then Perfect Matching would seem not to be hard for co-$\mathrm{C_{=}L}$. However, the assumption that $\mathrm{C_{=}L}$ is not contained in $\mathrm{Mod}_k\mathrm{L}$ does not have much intuitive clout. It is known that NL is contained in the $\mathrm{Mod}_k\mathrm{L}$/poly classes [Wi94], and it is natural to ask if similar techniques might also apply to $\mathrm{C_{=}L}$.

3 Collapse of the hierarchy

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

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

Proof: Let M be an oblivious logspace machine witnessing that $A \in$ co-$\mathrm{C_{=}L}$, 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 $B \in$ co-C$_=$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 $\mathrm{L}^{\mathrm{C_=L}} = \mathrm{NC}^1(\mathrm{C_=L})$. *FSLE is complete for this class.*

Proof: The forward inclusion is obvious since $\mathrm{L}^{\mathrm{C_=L}}$ 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 $A \in$ co-C$_=$L. Let N be a nondeterministic Turing machine witnessing that A is in co-C$_=$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), we may assume that without loss of generality that each C_n is a tree (except that, of course, different input gates may be connected to the same input variable). It is also no loss of generality to assume that the only strings of length at most 3 in the oracle are those in the set $\{0, 001, 010, 011, 111\}$. Thus any AND gate with inputs g_1, g_2 can be replaced by an oracle gate with inputs $1, g_1, g_2$, each OR gate can be replaced by an oracle gate with inputs $0, g_1, g_2$, and each NOT gate is equivalent to a one-input oracle gate. Thus we may assume that each gate of C_n is either an input gate or an oracle gate. These assumptions do not affect logspace uniformity.

Now for each oracle gate g in C_n, we assign weight $R(g) = 2^m$, 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 thus 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 co-$\mathrm{C_=L}$ 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 co-$\mathrm{C_=L}$ 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 gates 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 weights 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 traversal 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. That is, C_n outputs 1 if and only if M' can generate nonzero gap on input (x, m_x). Therefore $x \in B$ iff for some $m \leq q(|x|)$, $(x, m) \in X_2$ and (for all $i > m$, $(x, i) \notin X_1$). Since X_1 and X_2 are in co-$\mathrm{C_=L}$, and since co-$\mathrm{C_=L}$ is closed under dtt reductions, this shows that B is logspace dtt reducible to $\mathrm{C_=L} \wedge$ co-$\mathrm{C_=L}$. Therefore, by Lemma 2.11, 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*. **Proof (sketch):** First, consider the case $i = 0$. The determinant of M is equivalent to 0 mod p if and only if the system of linear equations $Mx = b$ given in the proof of Corollary 2.4 has an (integer) solution mod p, which is equivalent to the existence of integer vectors x, y such that $Mx - py = b$, which can be posed as an instance of *IFSLE*. For $i \neq 0$, note that there is a logspace transformation that takes (M, p, i) as input and produces matrix M' as output, such that $\det(M') = \det(M) + (p - i)$. Thus $\det(M) \equiv i \pmod{p}$ if and only if $(\det(M') = 0 \pmod{p})$ and (there exist integer matrices X, Y such that $MX - pY = I$). This can be encoded as a many-one reduction to *IFSLE*.) This reduction works as long as p is at most polynomially large. Thus a P-uniform NC^1 reduction can use Chinese Remaindering to compute the exact value of the determinant [BCH86]. This shows that #L is P-uniform NC^1-reducible to *IFSLE*. In contrast, we do not know of any correspondingly efficient way to reduce computation of the determinant (or other #L-hard problems) to the problem *FSLE*.

5 Open Questions

The most obvious open question is: *Is* $\mathrm{C_{=}L}$ *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 $\mathrm{NC}^1(\mathrm{PL}) = \mathrm{AC}^0(\mathrm{PL})$*?* (This question has recently been answered, in the affirmative, by Beigel and Fu [Be95], who also show that $\mathrm{NC}^1(\mathrm{PP}) = \mathrm{AC}^0(\mathrm{PP})$.)

Acknowledgments

We wish to thank E. Kaltofen and L. Fortnow for insightful comments related to this paper. Dieter van Melkebeek gave useful feedback on an earlier draft. Kousha Ettasimi gave us an exposition of the NL-hardness of the Perfect Matching problem. The observation about the complexity of perfect matching is due to V. Vinay, and occurred during stimulating conversations the first author had with M. Mahajan, V. Arvind, and V. Vinodchandran at the Institute of Mathematical Sciences, in Chennai, India, where some of this work was done.

References

[AO96] E. Allender and M. Ogihara, Relationships among PL, #L, and the determinant. *RAIRO - Theoretical Information and Application*, **30** (1996), 1–21.

[BGW96] L. Babai, A. Gál, and A. Wigderson, Superpolynomial lower bounds for monotone span programs. DIMACS Technical Report 96-37.

[Ba91] J. Balcázar, Adaptive logspace and depth-bounded reducibilities. In *Proceedings of the sixth Conference on Structure in Complexity Theory*, IEEE Computer Society Press, 1991, 240–254.

[BCH86] P. Beame, S. Cook, and H. J. Hoover, Log depth circuits for division and related problems. *SIAM Journal on Computing* **15** (1996), 994–1003.

[Be95] R. Beigel and B. Fu, Circuits over PP and PL. To appear in *Proceedings of the 12th Conference on Computational Complexity*, 1997.

[Be84] S. Berkowitz, On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters* **18** (1984), 147–150.

[BDHM92] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel, Structure and importance of Logspace-MOD Class. *Mathematical Systems Theory* **25** (1992), 223–237.

[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 Journal on Computing* **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 Journal on Computing* **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 *Synthesis of Parallel Algorithms*, ed. J. Reif. Morgan Kaufmann, 1993, 574–615.

[Gi95] M. GIESBRECHT, Fast computation of the Smith normal form of an integer matrix. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ACM Press, 1995, 110–118.

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

[Go96] G. GOTTLOB, Collapsing Oracle-Tape Hierarchies. In *Proceedings of the 10th Conference on Computational Complexity*, IEEE Computer Society Press, 1996, 33–42.

[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 Journal on Computing* **17**, 1988, 935–938.

[Ju85] H. JUNG, On probabilistic time and space. In *Proceedings of the 12th International Conference on Automata, Languages, and Programming*, Lecture Notes in Computer Science **194**, Springer-Verlag, 1985, 310-317.

[Ka95] E. KALTOFEN, personal communication.

[KUW86] R. KARP AND E. UPFAL AND A. WIGDERSON, Construction a perfect matching is in random NC. *Combinatorica* **6** (1986), 35–48.

[KW93] M. KARCHMER AND A. WIGDERSON, On Span Programs. In *Proceedings of the 8th Conference on Structure in Complexity Theory*, IEEE Computer Society Press, 1993, 102–111.

[L79] L. LOVÁSZ, On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory* **2**, ed. L. BUDACH. Akademia-Verlag, Berlin, 1979, 565–574.

[MV97] M. MAHAJAN AND V. VINAY, A combinatorial algorithm for the determinant. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, ACM Press, 1997.

[M87] K. MULMULEY, A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica* **7** (1987), 101–104.

[NT95] N. NISAN AND A. TA-SHMA, Symmetric logspace is closed under complement. *Chicago Journal on Theoretical Computer Science*, 1995, article 1.

[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. In *Proceedings of the 28th Symposium on Theory of Computing*, ACM Press, 1996, 84–88.

[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. In *Proceedings of the 5th Symposium on Algorithms and Computation*, Lecture Notes in Computer Science **834**, Springer-Verlag, 1994, 65–73.

[S80] J. SCHWARTZ, Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM* **27** (1980), 701–717.

[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, Department of Computer Science and Information Mathematics, University of Electro-Communications, Tokyo, 1991.

[T47] W. TUTTE, The factorization of linear graphs. *Journal of the London Mathematics Society* **22** (1947), 107–111.

[Va79] L. VALIANT, Completeness classes in algebra. In *Proceedings of the 11th Symposium on Theory of Computing*, ACM Press, 1979, 249–261.

[Va92] L. VALIANT, Why is Boolean complexity theory difficult? In *Boolean Function Complexity*, ed. M. PATERSON. London Mathematical Society Lecture Notes Series **169**, Cambridge University Press, 1992, 84–94.

[Vi91] V. VINAY, Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of the 6th Conference on Structure in Complexity Theory*, IEEE Computer Society Press, 1991, 270–284.

[Vi97] V. VINAY, personal communication.

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

[Wi94] A. WIGDERSON, NL/poly $\subseteq \oplus$L/poly. In *Proceedings of the 9th Conference on Structure in Complexity Theory*, IEEE Computer Society Press, 1994, 59–62.