VERIFYING THE DETERMINANT IN PARALLEL

Miklos Santha and Sovanna Tan

Abstract. In this paper we investigate both in the Boolean arithmetic circuit and the Boolean circuit model the complexity of the verification of problems whose computation is equivalent to the determinant. We observe that for a few problems there exists an easy (NC^1) verification algorithm. To characterize the harder ones, we define under two different reductions the class of problems which are reducible to the verification of the determinant and establish a list of complete problems in these classes. In particular we prove that computing the rank is equivalent under AC^0 reductions to verifying the determinant. We show in the Boolean case that none of the complete problems can be recognized in NC^1 unless $L = NL$. On the other hand we show that for functions, even if they are hard to verify, there exists an NC^1 checker and that they can be extended into functions whose verification is easy.
Key words. Parallel computing; complexity classes; verification; program checking; linear algebra.
Subject classifications. 68Q15, 68Q25.

1. Introduction

The classification of the computational complexity of problems which can be solved rapidly in parallel is one of the main topics in recent complexity theory. As the result of the work of several people, Csanky[9], Berkowitz[3], Chistov[7], Ibarra et al.[15], Borodin et al.[6][5], von zur Gathen[14], Mulmuley[17], it is known that most problems in linear algebra, (among others iterated matrix multiplication, characteristic polynomial, inversion, rank) admit fast parallel algorithms. More precisely they can be solved in functional NC^2, that is by circuits with a polynomial number of gates in depth $O(\log^2 n)$. To capture the exact complexity of these problems, Cook[8] has defined the class DET as the set of problems which can be reduced by circuits in depth $O(\log n)$ to the computation of the determinant. Actually it is not hard to see that all the problems identified by Cook in that class are already reducible by polynomial size unbounded fan-in constant depth (AC^0) circuits to the determinant, and it is that way we will define DET. He has observed that the above algorithms imply that most parallelizable problems in linear algebra are in that class, and many of them are equivalent to the computation of the determinant. (Many problems which are reducible to the determinant but are not known to be equivalent to it are equivalent to computing the rank. Some authors (von zur Gathen[14]) even define a similar "natural" class $RANK$ which is included in DET.) More recently (Toda[19],Vinay[23],Damm[10],Valiant[22]) DET was extended by several counting problems in graph theory. It is a major open problem in complexity theory if the class DET fully captures the computational power of functional NC^2.

The starting point of this paper is a very simple observation. Computing the determinant and the inverse of a matrix in parallel are computationally equivalent problems. But, if we look at the verification of these problems, that is if we want to decide for a given matrix A, if another matrix B is its inverse, and if a value a is its determinant then the situation looks quite different. For the inversion we just multiply A and B and compare the result to the identity matrix which is arguably a much simpler (NC^1)algorithm than the inversion. But for the determinant, there doesn't seem to be a simpler way to verify it than the computation itself.

Although we cannot prove such a claim – we don't even conjecture that it is true – in this paper we will establish that verifying some problems in the class DET (like the determinant) is indeed harder than verifying others from the same class (like inversion), unless the complexity classes deterministic logspace and non-deterministic logspace coincide. We also have a complementary result: functions in DET which are hard to verify have an extended version which is easy. Most of these extended functions are not really natural. Inversion is a rare example of a "natural" extended problem: it is the extension of the computation of a single element of the inverse matrix which is just as hard to verify as the determinant.

Whenever we do linear algebraic computation in parallel, there are at least two natural circuit models to consider. The first one is the standard Boolean circuit model which can deal with integer matrices. The other one is the arithmetic Boolean circuit model which deals with computations over an arbitrary field K. In this model in addition to the Boolean operations, there are also gates for the field operations. Results often do translate from one model to the other but it is not always the case. The difficulty in Boolean circuits stems for example from the fact that multiplying two integers takes logarithmic depth, whereas two field elements can be multiplied by definition in depth one in the arithmetic model. On the other hand some algorithms (for example computing the rank) are much harder over finite fields than over the integers. In the paper we will deal first with linear algebra over an arbitrary field K in the Boolean arithmetic model. Then in a separate section we will point out the differences with the Boolean case and the additional results.

We will use a method similar to Cook's and we will define $V\text{-}DET$ as the class of decision problems which are reducible to the problem of verifying the determinant by AC^0 circuits. It will turn out useful to consider also AC^0 many-one reductions, and define analogously the class $m\text{-}V\text{-}DET$. It was already observed in the case of the class DET (Allender and Ogihara[1], Toda[19]) that unlike most other "natural" complexity classes, the class of problems reducible to the determinant does depend on the particular reduction chosen.

In this paper we identify several complete problems for the classes $V\text{-}DET$ and $m\text{-}V\text{-}DET$. Most of these problems are verifications of functions from DET but the list also contains computational problems from linear algebra. In particular we will show that computing the rank is AC^0 equivalent to verifying the determinant which implies that the classes $V\text{-}DET$ and $RANK$ coincide. The completeness results about the verification problems give a rather satisfying complexity theoretical characterization of our original observation: in the Boolean case it follows that none of the $V\text{-}DET$ complete problems can be recognized as easily as the verification of the inversion unless $L = NL$. Still, the existence of easy (NC^1) verification procedures for some problems in DET implies that DET-complete functions are easy to verify if we define verification in a less restricted way: they all have NC^1 checkers in the sense of Blum and Kannan[4].

The paper is organized as follows: after some preliminaries in Section 2, Sections 3 and 4 treat the Boolean arithmetic circuit model. Section 3 deals with decision problems which are AC^0 many-one equivalent to verifying the determinant. In Section 4 we prove that computing the rank is AC^0 equivalent to verifying the determinant and that deciding the similarity of two matrices reduces to it. Section 5 deals with the Boolean case, and in Section 6 we explain the relationship between checkers and our problems. Here we also show how to extend every complete function in DET into an easily verifiable one.

2. Preliminaries

For a a square matrix A let $\det(A)$ denote the determinant of A, and unless otherwise stated, A_{ij} the element of A in row i and column j. Let $A(i|j)$ be the minor obtained from A by removing the i^{th} row and the j^{th} column.

Throughout the paper K will denote an arbitrary field, with 0 and 1 respectively the neutral and unit elements. The Boolean universe contains $\mathbf{T}, \mathbf{F}$, where $\mathbf{T}$ stands for "true" and $\mathbf{F}$ for "false". When we do computations only in the Boolean universe (no field is involved) then we will also write 0 for $\mathbf{F}$ and 1 for $\mathbf{T}$.

An *integer problem* or *relation* is a subset of $\{0,1\}^* \times \{0,1\}^*$. An *arithmetic problem* is a subset of $(K^* \times \{\mathbf{T},\mathbf{F}\}^*) \times (K^* \times \{\mathbf{T},\mathbf{F}\}^*)$, where $K^* = \cup_{n=1}^{\infty} K^n$. A problem R can be given by an *Input-Output* specification, where *Input* $=$ *domain*(R) and for every $x \in$ *Input*, we have *Output*$(x) = \{y : (x,y) \in R\}$. Whenever we will specify an arithmetic problem, we will mark it with the subscript K, the corresponding integer problem will not have any subscript. A *function* f is a relation such that for every x there is exactly one y such that $(x,y) \in f$. As usual, we will denote that y by $f(x)$. A *decision problem* is a Boolean function, and can be identified by the set of elements for which the function is $\mathbf{T}$. An algorithm *computes* a problem R if for every $x \in$ *Input*, it produces $y \in$ *Output*(x). The *verification* of a relation R is the decision problem V-R where by definition V-$R(x,y) = \mathbf{T}$ iff $(x,y) \in R$.

A Boolean arithmetic circuit is a circuit which contains gates for the Boolean operations $\neg$, $\vee$, $\wedge$ and the field operations $+$, $-$, $\times$, $/$. Two further gates connect the arithmetic and the Boolean components. A *test gate* has an arithmetic input x and a Boolean output y such that $y = \begin{cases} \mathbf{T} & \text{if} \quad x \neq 0, \\ \mathbf{F} & \text{if} \quad x = 0. \end{cases}$ A *selection gate* has two arithmetic inputs x_1 and x_2, a Boolean input y, and an arithmetic output z such that $z = \begin{cases} x_1 & \text{if} \quad y = \mathbf{T}, \\ x_2 & \text{if} \quad y = \mathbf{F}. \end{cases}$ A *Boolean circuit* contains only gates for the Boolean operations. In a natural way a Boolean arithmetic circuit computes an arithmetic problem, and a Boolean circuit computes an integer problem. In the case of arithmetic problems polynomial size integers are encoded in unary by $\mathbf{T}$ and $\mathbf{F}$.

For every i, let NC_K^i be the set of arithmetic decision problems which can be computed by a logspace uniform, polynomial size, bounded fan-in family of Boolean arithmetic circuits in depth $O(\log^i n)$ [14]. Here logspace uniform means that there exists a deterministic Turing machine working in space $O(\log n)$ which on input 0^n computes the description of the n^{th} circuit in the family [18]. Similarly AC_K^i is the set of arithmetic decision problems which can be computed by a logspace uniform, polynomial size, depth $O(\log^i n)$ family of Boolean arithmetic family where the Boolean gates have unbounded fan-in. For us it will be sufficient to suppose that the only field constants used by the logspace uniform circuits are 0 and 1. NC^i and AC^i denote the analogous classes for integer problems.

A relation R is AC^0_K *reducible* to a relation Q if there exists a polynomial size constant depth unbounded Boolean fan-in logspace uniform family of circuits (in short an AC^0_K family of circuits) with gates for Q which computes R. We require that there is at most one occurrence of a gate for Q along any input-output path. (This reduction is sometimes called a constant depth truth table reduction.) A function f is AC^0_K *many-one reducible* to a function g if there exists an AC^0_K family of circuits C such that for every x we have $f(x) = g(C(x))$. In many of our reductions the reducing circuits will output inputs from the source problem as well as constants. In these cases the circuits do not perform arithmetic operations nor selection nor test operations.

The following is a first list of basic problems considered in linear algebra.

1. DETERMINANT$_K$
Input: $A \in K^{n\times n}$.
Output: $\det(A)$ = the determinant of A.
2. POWER$_K$
Input: $A \in K^{n\times n}$, and $1 \leq m \leq n$.
Output: A^m.
3. CONSECUTIVE POWERS$_K$
Input: An $A \in K^{n\times n}$.
Output: The powers A^i for $i = 1, \ldots, n$.
4. POWER ELEMENT$_K$
Input: $A \in K^{n\times n}$, and $1 \leq i, j, m \leq n$.
Output: The $(i,j)^{\text{th}}$ element of A^m.
5. INVERSE$_K$
Input: An invertible matrix $A \in K^{n\times n}$.
Output: A^{-1}.
6. INVERSE ELEMENT$_K$
Input: An invertible matrix $A \in K^{n\times n}$, and $1 \leq i, j \leq n$.
Output: The $(i,j)^{\text{th}}$ element of A^{-1}.
7. CHARACTERISTIC POLYNOMIAL$_K$
Input: $A \in K^{n\times n}$.
Output: $\chi(A)$ = the characteristic polynomial of A.

DEFINITION 2.1. Let DET_K be the set of problems which are AC^0_K reducible to DETERMINANT$_K$. A problem in DET_K is *complete* if the DETERMINANT$_K$ is AC^0_K reducible to it. Let m-DET_K be the set of functions which are AC^0_K many-one reducible to DETERMINANT$_K$. A function in m-DET_K is *complete* if DETERMINANT$_K$ is AC^0_K many-one reducible to it.

It is immediate from the definition that $m\text{-}DET_K \subseteq DET_K$, and that $m\text{-}DET_K$ complete problems are also complete in DET_K. The following three theorems were obtained by several people including Csanky[9], Chistov[7], Berkowitz[3], Borodin et al.[6], and Ben-Or and Cleve[2].

THEOREM 2.2. *All the above problems are complete in* DET_K.

THEOREM 2.3. *The functions* DETERMINANT$_K$ *and* POWER ELEMENT$_K$ *are complete in* $m\text{-}DET_K$.

THEOREM 2.4. *We have the inclusions*

$$NC_K^1 \subseteq m\text{-}DET_K \subseteq DET_K \subseteq NC_K^2.$$

It is an easy observation that some problems in DET_K can be verified in NC_K^1.

THEOREM 2.5. *The problems* V-INVERSE$_K$ *and* V-CONSECUTIVE POWERS$_K$ *are in* NC_K^1.

PROOF. To verify that B is the inverse of A, we multiply A with B and check if the result is the identity matrix I. To verify that $(A, (B_1, \ldots, B_n)) \in$ CONSECUTIVE POWERS$_K$, we check in parallel if $A = B_1$ and $AB_i = B_{i+1}$ for $i = 1, \ldots, n-1$. $\square$

In order to classify the complexity of verifying the other problems from the list we introduce two new complexity classes.

DEFINITION 2.6. Let $V\text{-}DET_K$ be the set of problems which are AC_K^0 reducible to V-DETERMINANT$_K$. A problem in $V\text{-}DET_K$ is *complete* if the problem V-DETERMINANT$_K$ is AC_K^0 reducible to it. Let $m\text{-}V\text{-}DET_K$ be the set of decision problems which are AC_K^0 many-one reducible to V-DETERMINANT$_K$. A problem in $m\text{-}V\text{-}DET_K$ is *complete* if V-DETERMINANT$_K$ is AC_K^0 many-one reducible to it.

Again, by definition we have

$$m\text{-}V\text{-}DET_K \subseteq V\text{-}DET_K \subseteq DET_K.$$

Also, an $m\text{-}V\text{-}DET_K$ complete problem is complete in $V\text{-}DET_K$.

3. Complete problems for m-V-DET_K

THEOREM 3.1. *The problems* V-DETERMINANT$_K$ *and* V-POWER ELEMENT$_K$ *are complete for* m-V-DET_K.

PROOF. Let g be DETERMINANT$_K$ or POWER ELEMENT$_K$, and let f be a function in m-DET_K. Since g is complete in m-DET_K, there exists an AC^0_K family of circuits such that $f(x) = g(C(x))$. Therefore $(x, y) \in f$ if and only if $(C(x), y) \in g$. $\square$

We now turn to our first problem on graphs which is as hard to compute as the determinant. The advantage of this problem is that it will enable us to derive the Sum Lemma which will be a main tool for proving completeness results for the verification classes.

A *weighted graph* G is a directed graph where every edge has a weight from K associated with it. It can be represented by its adjacency matrix whose $(i, j)^{\text{th}}$ element is 0 if there is no edge from i to j, and the weight of the edge (i, j) otherwise. The weight of a path is defined as the product of the weights of the edges in the path. Given two vertices i and j we define $\text{wpath}(G, i, j)$ as the sum of the weights of all i-j paths in G.

8. WEIGHTED PATHS$_K$
Input: A directed acyclic graph G with n vertices, and two vertices $1 \leq i, j \leq n$.
Output: $\text{wpath}(G, i, j)$.

THEOREM 3.2. WEIGHTED PATHS$_K$ *is complete in* m-DET_K.

PROOF. First we show that WEIGHTED PATHS$_K$ is in m-DET_K by reducing it to DETERMINANT$_K$. The idea of the proof goes back to Valiant[21]. Given a directed acyclic graph G with two distinguished vertices i and j first we construct a directed acyclic graph H as follows: We replace every edge $e = (u, v)$ by two consecutive edges (u, u_e) and (u_e, v) where u_e is a new vertex. The weight of (u, u_e) is the weight of (u, v), and the weight of (u_e, v) is 1. We add an edge (j, i) with weight 1. Finally we put a self-loop on each vertex except i and j with weight 1.

As an illustration, we discuss here briefly why this reduction is uniform. The subcircuits computing the different outputs are obtained by cycling through the vertices and the possible edges of the input graph. Their numbers depend only on the size of the input. While cycling trough the vertices, only edges with uniform weight 1 are constructed. While cycling trough the edges, outputs with uniform weight 1 or with weight equal to the given input value are constructed. These two cases can be distinguished by a trivial counter.

Let M be the adjacency matrix of H, we claim that $\det(M) = \mathrm{wpath}(G, i, j)$. The value of a permutation is non zero exactly when it induces a cycle cover of H. Any cycle cover of H consists of a large cycle containing a path from i to j, and self-loops for the other vertices. Clearly there is a 1-1 correspondence between i-j paths in G and i-j paths in H, and the value of a non zero permutation is the weight of the corresponding path in G. Moreover the sign of these permutations is $+1$ since they consist of cycles of odd length.

We show now that WEIGHTED PATHS$_K$ is complete by reducing POWER ELEMENT$_K$ to it. This construction is a modification of Toda's idea[19]. Given an $n \times n$ matrix A and $1 \leq i, j, m \leq n$, we define a graph H as follows. The vertices of H are the pairs $[k, h]$ for $1 \leq k \leq n$ and $0 \leq h \leq m$. For every $1 \leq k, l \leq n$ and $0 \leq h < m$ there is an edge from $[k, h]$ to $[l, h+1]$ exactly when $A_{kl} \neq 0$, and the weight of this edge is A_{kl}. Then there is one-one correspondence between the i-j paths of length m in the graph whose adjacency matrix is A and the paths in H from $[i, 0]$ to $[j, m]$ which preserves the weight. Therefore $A^m_{ij} = \mathrm{wpath}(H, [i, 0], [j, m])$. This reduction is again uniform since basically the Turing machine on input n and $m \leq n$ given in unary has to enumerate all the 4-tuples $([k, h], [k', h'])$ with $1 \leq k, k' \leq n$ and $0 \leq h, h' \leq m$ which can easily be done in space $O(\log n)$. $\square$

This theorem implies that V-WEIGHTED PATHS$_K$ is m-V-DET_K complete. More importantly it helps us to derive the Sum Lemma.

LEMMA 3.3. (SUM LEMMA) *For every m-DET_K complete function f there exists a family of AC^0_K circuits which on input (x, y) outputs z such that $f(x) + f(y) = f(z)$.*

PROOF. It is enough to prove the statement for one m-DET_K complete function, and we will choose WEIGHTED PATHS$_K$. Given two instances (G_1, i_1, j_1) and (G_2, i_2, j_2) to it we construct H as follows. H contains all the vertices and edges of G_1 and G_2. It has additional vertices i and j and edges (i, i_1), (i, i_2), (j_1, j) and (j_2, j) with weights 1. Then $\mathrm{wpath}(G_1, i_1, j_1) + \mathrm{wpath}(G_2, i_2, j_2) = \mathrm{wpath}(H, i, j)$. $\square$

As an immediate consequence we get that verifying the determinant is as hard as deciding if a matrix is singular.

9. SINGULAR$_K$

Input: An $n \times n$ matrix A.
Question: Is A singular?

COROLLARY 3.4. SINGULAR$_K$ *is* m-V-DET_K *complete.*

PROOF. To decide if $\det(A) = a$ we construct a matrix B which is the identity except the first element of the diagonal is $-a$. By the Sum Lemma we can find a matrix C whose determinant is $\det(A) - a$. □

With the Sum Lemma we are also able to show that verifying just one element of the inverse matrix is probably much harder than verifying the whole matrix.

THEOREM 3.5. V-INVERSE ELEMENT$_K$ *is* m-V-DET_K *complete.*

PROOF. We first show that V-INVERSE ELEMENT$_K$ is in m-V-DET_K. Given a matrix A, integers i and j, and a field element a, let us suppose without loss of generality that $i + j$ is even. Then we have $A^{-1}_{ij} = a$ if and only if $\det(A(j|i))/\det(A) = a$. Let B be the $(n+1) \times (n+1)$ matrix whose first row and first column is everywhere 0 except $B_{11} = -a$, and whose lower right $n \times n$ submatrix is A. Then $\det(B) = -a\det(A)$. Thus we have to verify if $\det(A(j|i)) + \det(B) = 0$. By the Sum Lemma we can find a matrix C such that $\det(A) + \det(B) = \det(C)$ and verifying if $\det(C) = 0$ is in m-V-DET_K.

For the hardness we show that POWER ELEMENT$_K$ is AC^0 many-one reducible to INVERSE ELEMENT$_K$. This is actually immediate from the standard many-one reduction of POWER$_K$ to INVERSE$_K$: For an $n \times n$ matrix A, let us consider the $(mn + n) \times (mn + n)$ matrix B, consisting of blocks of size $n \times n$ which are all zero except for $m+1$ copies of the identity matrix in the diagonal and m copies of $-A$ above the diagonal. Then the inverse of B contains A^m in its upper right corner. □

For a function f we define EQ-f the decision problem *equality* of f as EQ-$f = \{(x, y) : f(x) = f(y)\}$. For example EQ-DETERMINANT$_K$ is the pairs of matrices with the same determinant.

THEOREM 3.6. *Let f be a function which is complete in m-DET_K. Then* EQ-f *is complete in m-V-DET_K.*

PROOF. The proof is similar of that of the previous theorem. First we show that EQ-f is in m-V-DET_K. Since $f \in m$-DET_K, given x and y we can find in AC^0_K an element z such that $f(x) = f(y)$ if and only if $f(z) = 0$.

Now we reduce V-DETERMINANT$_K$ to EQ-f. Given a matrix A and a field element a, first we construct in AC^0_K a matrix B which is the identity matrix except it has a in its 1st row 1st column. Then clearly $\det(A) = a$ exactly when $\det(A) = \det(B)$. Since f is m-DET_K complete, we can find also in AC^0_K elements x and y such that $\det(A) = f(x)$ and $\det(B) = f(y)$. Thus $\det(A) = a$ if and only if $f(x) = f(y)$. □

THEOREM 3.7. EQ-INVERSE ELEMENT$_K$ *is* m-V-DET_K *complete.*

PROOF. The problem is m-V-DET_K hard since as it was proven already in Theorem 3.5, INVERSE ELEMENT$_K$ is complete for V-DET_K. We therefore only have to show that it is in m-V-DET_K. For two matrices A and B, we have $A^{-1}_{ij} = B^{-1}_{i'j'}$ if and only if $\det(A(j|i))/\det(A) = \det(B(j'|i'))/\det(B)$. (We suppose without loss of generality that $i + j$ and $i' + j'$ have the same parity). This in turn is equivalent to $\det(A(j|i))\det(B) = \det(A)\det(B(j'|i'))$. Given two matrices one can easily find in AC^0_K a third one whose determinant is the product of the determinants. Therefore we finally have to check if two matrices have the same determinant which is indeed a problem in m-V-DET_K. □

4. Complete problems for V-DET_K

In the previous section we have proven that verifying a single element in an inverse matrix or in a power is hard for the class m-V-DET_K. The verification of the whole inverse matrix can be done in NC^1_K. Our first theorem about V-DET_K says that power remains hard to verify even if the whole matrix is concerned.

THEOREM 4.1. V-POWER$_K$ *is* V-DET_K *complete.*

PROOF. For the easiness, we observe that V-POWER$_K$ is AC^0_K reducible to V-POWER ELEMENT$_K$ since we have to verify that all the elements of the result are correct. For the hardness, we will reduce V-POWER ELEMENT$_K$ to V-POWER$_K$. Actually this can be done even with a many-one reduction. Let $A \in K^{n\times n}$, and let the $(i,j)^{\text{th}}$ element of A^m be a. For $1 \le k,l \le n$, we define the $n \times n$ matrix Z_{kl} which is everywhere 0 except the $(k,l)^{\text{th}}$ element, which is 1. Then $Z_{1i} \times A^m \times Z_{jn}$ is an $n \times n$ everywhere 0 matrix except the $(1,n)^{\text{th}}$ element, which is a. Let B be the $(mn + 3n) \times (mn + 3n)$ matrix consisting of blocks of size $n \times n$ which are all zero except the $m + 2$ blocks above the diagonal. The matrices in these blocks are in descending order Z_{1i}, m copies of A, and Z_{jn}. Then B^{m+2} is an everywhere 0 matrix except the element in the first line last column, which is a. □

We now turn to computing problems which are equivalent to the verification of the determinant. Our first example is the rank of a matrix.

10. RANK$_K$
Input: $A \in K^{n \times n}$.
Output: $r(A) =$ the rank of A.

THEOREM 4.2. RANK$_K$ *is V-DET_K complete.*

PROOF. Since the rank of an $n \times n$ matrix is at most n, computing it is AC^0_K equivalent to verifying it. Therefore to show that RANK$_K$ is in V-DET_K it is enough to show that V-RANK$_K$ is in the class. We will show that by reducing V-RANK$_K$ to V-POWER ELEMENT$_K$. Actually the reduction will follow from Mulmuley's algorithm[17] reducing RANK$_K$ to POWER ELEMENT$_K$. It turns out that this reduction doesn't really need to compute the elements of a well chosen matrix power, it only has to verify if these elements are 0, and therefore it can be easily transformed into a reduction in the verification scenario.

Given a matrix A we can suppose without loss of generality that A is symmetric. (Otherwise we can replace A by A' , where A' is $\boxed{\begin{array}{cc} 0 & A \\ A^t & 0 \end{array}}$ and $r(A') = 2r(A)$.) Let y be an indeterminate over K, and let Y be the $n \times n$ diagonal matrix with $Y_{ii} = y^{i-1}$ for $i = 1, \dots n$. Let $B(y) = YA$. Then Mulmuley has shown that $r(A) = r(B(y)) = r(B^2(y))$, and that $r(B(y)) = n - k$ if and only if k is the largest integer such that x^k divides $\chi(B(y))$. The coefficients of the characteristic polynomial of $B(y)$ are polynomials in y of degree at most n^2, let $b_i(y)$ be the coefficient of x^i. Berkowitz's algorithm[3] reduces the computation of $\chi(B(y))$ to the power $E(y) = (D(y))^m$, where $D(y)$ is a $t \times t$ matrix over $K[y]$, with t is a polynomial in n and $m \leq t$. Moreover the elements of $D(y)$ are either elements of $B(y)$ or the constants 0 or 1. The coefficients of the characteristic polynomial can be found for example in the last column of $E(y)$, the coefficients of x^j for $j > n$ being 0.

In order to make this reduction into one where the matrices are over the ground field we can use the representation of polynomials by upper triangular Toeplitz matrices as it is suggested by von zur Gathen[14] . For any polynomial $p(y) = c_0 + c_1 y + \dots$, and integer d let $\tau_d(p)$ be the $d \times d$ matrix C over K with

$$C_{i,j} = \begin{cases} 0, & \text{if} \quad i > j, \\ c_k, & \text{if} \quad i = j + k. \end{cases}$$

Then τ_d is a ring homomorphism from $K[y]$ to $K^{d \times d}$ with kernel y^d. Let us write the matrix $D(y)$ as $\sum_{j=0}^{t-1} D_j y^j$, where all D_j are $t \times t$ matrices over K. Then if we consider $\tau_{t^2}(D(y))$ as an $t^3 \times t^3$ matrix over K then the power $(\tau_{t^2}(D(y)))^m$ contains in its last column for every $0 \leq i \leq n^2$ the numerical coefficients of $b_i(y)$ one by one. Therefore we can check in AC^0 with an oracle for V-POWER ELEMENT$_K$ if for $i = 0, \ldots k-1$ all the coefficients of $b_i(y)$ are 0, and if there exists a coefficient of $b_k(y)$ which is different from 0.

To finish the proof we show that RANK$_K$ is V-DET hard by reducing SINGULAR$_K$ to it. This is immediate since a $n \times n$ matrix A is singular if and only if $r(A) < n$. □

As a consequence we obtain a list of new computational problems equivalent to V-DETERMINANT$_K$.

11. MAXMINOR$_K$
Input: $A \in K^{n \times n}$.
Output: A maximal nonsingular submatrix of A.
12. BASIS$_K$
Input: $A \in K^{n \times n}$.
Output: A maximal set of linearly independent columns of A.
13. SOLVABILITY$_K$
Input: $A \in K^{n \times n}$ and $b \in K^n$.
Question: Does $Ax = b$ have a solution in K^n?
14. INDEPENDENCE$_K$
Input: $a_1, \ldots, a_i \in K^n$.
Question: Are they linearly independent?

THEOREM 4.3. *MAXMINOR$_K$, BASIS$_K$, SOLVABILITY$_K$ and INDEPENDENCE$_K$ are V-DET complete.*

PROOF. The circular reduction given by von zur Gathen[14] between RANK$_K$ and the first three of the above problems establishes the result for them. INDEPENDENCE$_K$ is in V-DET since it can be reduced to RANK$_K$: to check if $a_1, \ldots, a_i$ are linearly independent we append $n - i$ zero vectors to them and check if the matrix has rank i. It is also hard since SINGULAR$_K$ can be reduced to it: a matrix A is singular if and only if its column vectors are not independent. □

We end this section with another well known problem. SIMILARITY$_K$ is reducible to V-DETERMINANT$_K$ but we don't know if it is also equivalent to it.

15. SIMILARITY$_K$
Input: $A, B \in K^{n \times n}$.
Question: Are A and B similar?

THEOREM 4.4. SIMILARITY$_K$ *is in* V-DET_K.

PROOF. We will use Dixon's criterion [11] which was already applied by Zalcstein and Garzon[13] to show that SIMILARITY$_K \in NC^2_K$. Let $\otimes$ denote the tensor product, and for two $n \times n$ matrices A and B let $q(A,B) = r(A \otimes I - I \otimes B)$ where I is the $n \times n$ identity matrix. Then Dixon's theorem says that A and B are similar if and only if $q^2(A,B) = q(A,A)q(B,B)$. The tensor product is in AC^0_K. To check Dixon's criterion we verify if there exist $0 \leq k, l, m \leq n^2$, such that $q(A,B) = k, q(A,A) = l, q(B,B) = m$ and $k^2 = lm$. Since integer multiplication in unary can easily be done in AC^0, we have an AC^0 reduction of the problem to RANK$_K$. □

5. The Boolean case

Most of the arithmetic problems defined in the previous sections do naturally translate into integer problems by replacing the input matrices in $K^{n \times n}$ with $n \times n$ matrices of n-bit integers. We will consider these matrices over the field of rational numbers. Since the inverse of integer matrices are not necessarily integer matrices, and the solutions of integer linear equations are not necessarily integer vectors, we modify the problems of inversion and solvability accordingly. We will also consider a few additional problems about graphs.

5'. INVERSE
Input: An invertible $n \times n$ matrix A of n-bit integers.
Output: A matrix B of $2n^2$-bit integers, and an integer d also on $2n^2$ bits such that $A^{-1} = B/d$.
6'. INVERSE ELEMENT
Input: An invertible $n \times n$ matrix A of n-bit integers, and $1 \leq i, j \leq n$.
Output: Two $2n^2$-bit integers c, d such that the $(i,j)^{\text{th}}$ element of A^{-1} is c/d.
13'. SOLVABILITY
Input: An $n \times n$ matrix A and a length n column vector b of n-bit integers.
Question: Does $Ax = b$ have a solution in the rational numbers?
16. PATHS
Input: A directed acyclic graph G with n vertices, and two vertices $1 \leq i, j \leq n$.
Output: $\text{path}(G, i, j)$, the number of paths in G from i to j.
17. PATHS DIFFERENCE
Input: Two directed acyclic graphs G_1, G_2 with n vertices, and $1 \leq i_1, i_2, j_1, j_2 \leq n$.
Output: $\text{path}(G_1, i_1, j_1) - \text{path}(G_2, i_2, j_2)$.

18. SHORTEST PATHS
Input: A directed acyclic graph G with n vertices, and $1 \leq i, j \leq n$.
Output: $\mathrm{spath}(G, i, j)$, the number of shortest paths in G from i to j.

The two problems which were easy to verify remain so also in the Boolean model.

THEOREM 5.1. *The problems* V-INVERSE *and* V-CONSECUTIVE POWERS *are in NC^1.*

The results about Boolean $m\text{-}V\text{-}DET$ are summarized in the following two theorems. The fact that V-PATHS is complete for $m\text{-}V\text{-}DET$ and the proof direction was pointed out to us by Eric Allender.

THEOREM 5.2.

(a) *The problems* V-DETERMINANT, V-POWER ELEMENT, V-WEIGHTED PATHS *and* V-SINGULAR *are complete for $m\text{-}V\text{-}DET$.*

(b) *If f is complete in $m\text{-}DET$ then* EQ-f *is complete in $m\text{-}V\text{-}DET$.*

(c) V-PATHS DIFFERENCE *is complete for $m\text{-}V\text{-}DET$.*

(d) V-INVERSE ELEMENT *is complete for $m\text{-}V\text{-}DET$.*

(e) V-POWER *is complete for $m\text{-}V\text{-}DET$.*

PROOF. The results in **(a)** and **(b)** follow from the boolean analogs of theorems 3.1, 3.2, 3.6 and corollary 3.4. Part **(c)** follows from the completeness of PATHS DIFFERENCE in $m\text{-}DET$ which was proven in [19] [10] [22]. For part **(d)**, we follow the proof of theorem 3.5. For the easiness, let A be an integer matrix, and again let us suppose without loss of generality that we are given two indices i and j such that $i + j$ is even. Then $A^{-1}_{ij} = c/d$ if and only if $d \det(A(j|i)) - c \det(A) = 0$. Using the Sum Lemma, we cand find a matrix C such that $d \det(A(j|i)) - c \det(A) = \det(C)$. Verifying that $\det(C) = 0$ is in $m\text{-}V\text{-}DET$. For the hardness, let A be an integer matrix, and like in theorem 3.5, we can construct a matrix B whose inverse contains A^m as a submatrix. Therefore we can verify if a is an element of A^m by verifying if a (represented for example as $a/1$) is in the inverse matrix at the corresponding place. Finally, for part **(e)**, as we will discuss subsequently, the class $m\text{-}V\text{-}DET$ is closed under AC^0 conjunctive truth table reduction, therefore V-POWER is in $m\text{-}V\text{-}DET$. Its hardness under many-one reduction can be proven like it was done for the arithmetic case in theorem 4.1. $\square$

THEOREM 5.3.

(a) V-PATHS *is complete for* m-V-DET.

(b) V-SHORTEST PATHS *is complete for* m-V-DET.

PROOF. **(a)** We will give an AC^0 many-one reduction from V-PATHS DIFFERENCE to V-PATHS. The basic idea of our reduction is the proof in [12], which shows that $GapP = \#P - FP$. We just add some technical refinements in order to ensure that the reduction used in that proof can be performed in AC^0.

Let G_1 and G_2 be two directed acyclic graphs with vertices respectively labeled $1, 2, ..., n_1$ and $1, 2, ..., n_2$. We will construct a graph G with $O(n_1 + n_2^5)$ vertices such that PATHS DIFFERENCE$((G_1, 1, n_1), (G_2, 1, n_2)) =$ PATHS$(G, s, t) - 2^{n_2^2}$. Since $2^{n_2^2}$ in binary can be computed in AC^0 from n_2 in unary, and since addition is also in AC^0, this will imply the result.

Let p_2 denote PATHS$(G, 1, n_2)$. First we transform G_2 into a graph H with n_2^2 vertices of maximal outdegree 2 such that the number of paths between 1 and n_2^2 is equal to p_2. We associate to each vertex u of G_2, n_2 vertices $u_0, u_1, ..., u_{n_2-1}$ in H. If k edges are going from u to nodes $v_1, v_2, ..., v_k$ in G_2, H has the edges (u_i, u_{i+1}) for $0 \leq i < k$ and (u_i, v_i) for $1 \leq i \leq k$.

Then we transform H into a leveled graph I following a construction similar to the one described in theorem 3.2. The vertices of I are the pairs $[i, l]$ for $1 \leq i \leq n_2^2$ and $0 \leq l \leq n_2^2$. For every $1 \leq i < n_2^2$, $1 \leq j \leq n_2^2$ and $0 \leq l < n_2^2$, there is an edge from $[i, l]$ to $[j, l+1]$ in I when (i, j) is an edge of H. We do not have to take into account the edges originating from n_2^2 since they can not appear in a path from 1 to n_2^2. In I there are also edges for every $0 \leq l < n_2^2$, from $[n_2^2, l]$ to $[n_2^2, l+1]$. There are exactly p_2 paths from $[1, 0]$ to $[n_2^2, n_2^2]$ in I.

From I we construct K which is also a layered graph with $n_2^2 + 1$ levels such that all its vertices have outdegree exactly 2 except the vertices of the last layer which have outdegree 0. The graph K contains I with some additional subgraphs, and in K there will be exactly 2^l paths of length l starting from $[1, 0]$. We now describe the additions. Let D_n be a directed acyclic graph with $2n+1$ vertices $u, v_1, v_2, ..., v_n, w_1, w_2, ..., w_n$ with edges (u, v_1), (u, w_1), and (v_l, v_{l+1}), (v_l, w_{l+1}), (w_l, v_{l+1}) and (w_l, w_{l+1}) for every $1 \leq l \leq n-1$. The vertex u is called the *root* of D_n. In order to construct K, we add to each vertex $[i, l]$, $1 \leq i \leq n_2^2$, $0 \leq l < n_2^2$ with no outgoing edge a subgraph $D_{n_2^2 - l}$ identifying the vertex $[i, l]$ with the root of $D_{n_2^2 - l}$. We also add to each vertex $[i, l]$, $1 \leq i \leq n_2^2$, $0 \leq l < n_2^2$, with one outgoing edge another edge connected to the root of a subgraph $D_{n_2^2-l-1}$. Observe that in K we still have p_2 paths between $[1,0]$ and $[n_2^2, n_2^2]$.

The graph G is built from G_1 and K. It has two supplementary vertices s and t, and additional edges from s to the node 1 of G_1, from s to the node $[1,0]$ of K, from the node n_1 of G_1 to t, and from every node of K at level n_2^2 to t except from the node $[n_2^2, n_2^2]$. We have PATHS DIFFERENCE$((G_1, 1, n_1), (G_2, 1, n_2))$ $=$ PATHS$(G, s, t) - 2^{n_2^2}$.

(b) We first prove that V-PATHS reduces to V-SHORTEST PATHS. The reduction is similar to the one in theorem 3.2. Let G be a directed acyclic graph on n vertices, and let us suppose without loss of generality that we want to verify if $\text{path}(G, 1, n) = a$. Let us define a graph H as follows. The vertices of H are the pairs $[k, m]$ for $1 \le k \le n$ and $0 \le m \le n$. For every $1 \le k < n$, $1 \le l \le n$, $0 \le m < n$ there is an edge from $[k, m]$ to $[l, m+1]$ exactly when (k, l) is an edge of G. We do not need to consider the edges that originate from n because they are not involved in paths from 1 to n since G is acyclic. Let us add to the graph H for every $0 \le m < n$, an edge from $[n, m]$ to $[n, m+1]$. The number of paths from 1 to n in G is equal to the number of paths from $[1,0]$ to $[n, n]$ in H and moreover all paths from $[1,0]$ to $[n,n]$ are shortest paths. Hence $\text{path}(G, 1, n) = a$ if and only if $\text{spath}(H, [1,0], [n,n]) = a$.

Let us prove now that V-SHORTEST PATHS reduces to V-PATHS. Again, in the reduction we will use a similar construction. Let G be a directed acyclic graph on n vertices, and let us suppose without loss of generality that we want to verify if $\text{spath}(G, 1, n) = a$. For $j = 1, \ldots, n$ we define the graphs G_j as follows. The vertices of G_j are the pairs $[k, m]$ for $1 \le k \le n$ and $0 \le m \le j$. For every $1 \le k, l \le n$ and $0 \le m < j$ there is an edge from $[k, m]$ to $[l, m+1]$ exactly when (k, l) is an edge of G. Now observe that if the length of the shortest path between 1 and n in G is i then $\text{spath}(G, 1, n) = \text{spath}(G_i, [1,0], [n, i])$. Moreover, in that case for every $j < i$ we have $\text{spath}(G_j, [1,0], [n, j]) = 0$. Therefore $\text{spath}(G, 1, n) = a$ if and only if there exists $1 \le i \le n$ such that $((G_i, [1,0], [n, i]), a) \in$ V-SHORTEST PATHS and for every $1 \le j < i$ we have $((G_j, [1,0], [n, j]), 0) \in$ V-PATHS. $\Box$

The class *m-V-DET* has recently been also defined and investigated by Allender and Ogihara [1] from a somewhat different, structural complexity perspective. Let us recall a few standard definitions from complexity theory before we briefly explain their results. In the following discussion by reduction we always mean AC^0 many-one reduction.

For a nondeterministic Turing machine (NTM) M which halts on every input, let $acc_M(x)$ be the number of accepting computation paths of M on input x. The complexity class $\#L$ is defined as the set of functions f for which there exists a logspace NTM M such that for every input x, the equality $f(x) = acc_M(x)$ holds. It is well known that PATHS is complete in $\#L$. The class $\#L$ is the counting version of NL, the class of decision problems which can be computed by a NTM in logspace. The generic complete problem for NL is REACHABILITY.

19. REACHABILITY
Input: A directed acyclic graph G with n vertices, and two vertices $1 \leq i, j \leq n$.
Question: Is there a path in G from i to j?

DEFINITION 5.4. (Allender and Ogihara)
$GapL = \{f : f(x) = g(x) - h(x) \text{ for some } g, h \in \#L\}$.
$C_=L = \{A \subseteq \{0,1\}^* : \exists f \in GapL,\ \forall x \in \{0,1\}^* \quad x \in A \iff f(x) = 0\}$.

These classes were defined as the analogs of the well known classes $GapP$ and and $C_=P$ (Torán[20]) by replacing polynomial time with logarithmic space. Many properties of $C_=P$ remain true for $C_=L$, including closure under AC^0 conjunctive, disjunctive and positive truth table reductions (Fenner et al.[12]) where *positive* refers to reductions via monotone circuit families.

Since PATHS is complete in $\#L$, the function PATHS DIFFERENCE is clearly complete in $GapL$. Therefore $GapL$ coincides with the class m-DET. Also, V-PATHS DIFFERENCE is complete for $C_=L$. This in turn implies that m-V-DET and $C_=L$ are in fact the same classes. The fact that V-DETERMINANT is complete in $C_=L$ was already pointed out by Allender and Ogihara[1].

They have also noted that $NL \subseteq m\text{-}V\text{-}DET$ since $NL = co\text{-}NL$ and the complement of the NL complete problem REACHABILITY just requires to verify if there is no path between the specified vertices. Since $NC^1 \subseteq L$, we have the following sequence of inclusions.

THEOREM 5.5.

$$NC_1 \subseteq L \subseteq NL \subseteq m\text{-}V\text{-}DET \subseteq V\text{-}DET \subseteq NC^2.$$

An immediate corollary of this theorem explains why is it highly unlikely that verifying the determinant (or any other V-DET complete problem) will be as easy as verifying inversion.

COROLLARY 5.6. *If* V-INVERSE *and any V-DET complete problem are equivalent under logspace reduction then $L = NL$.*

Let us say here a few words about the reducibilities we have chosen. The classes m-DET and m-V-DET actually remain the same under several reducibilities, such as logspace uniform projection, AC^0 many-one, logspace uniform NC^1 many-one, or logspace many-one reduction. However this is not known (and doesn't seem to be true) for DET and V-DET. We have chosen AC^0 reduction in the definition because this is the strongest standard non many-one reduction under which complete problems are known for DET and we were able to find complete problems for V-DET. If one would like to avoid using a reduction with a uniformity condition less restricted than the reduction itself, one could define V-DET with logspace reduction and all the result would carry through. Nonetheless, as we said, it might be that we define this way a larger class.

Finally let us remark that as far as the class V-DET goes, theorems 4.2, 4.3 and 4.4 carry through the integer case.

THEOREM 5.7.

(a) *The problems* RANK, MAXMINOR, BASIS, SOLVABILITY *and* INDEPENDENCE *are complete for* V-DET.

(b) SIMILARITY *is in* V-DET.

6. Verification and checking

In this section we will discuss the relationship between our results on the parallel verification of problems in DET and the theory of program checking. We will work with integer problems. For motivations and the general ideas underlying program checking we refer to [4]. First we recall the definition of a checker which is due to Blum. A probabilistic algorithm C_R with oracle is a *checker* for the problem R if for every deterministic program (algorithm) P used as an oracle, for every $x \in Input$, and for every integer k we have:

- If for every $z \in Input$, $(z, P(z)) \in R$ then with probability at least $1 - 1/2^k$, $C_R^P(x, k) = correct$.
- If $(x, P(x)) \notin R$ then with probability at least $1 - 1/2^k$, $C_R^P(x, k) = buggy$.

A checker is interesting if it belongs to a complexity class $\mathcal{F}$ such that the problem it is checking is not known to be in $\mathcal{F}$. If the verification V-R of a problem R belongs to a complexity class $\mathcal{F}$ then we have (a rather trivial) deterministic checker C_R for R in $\mathcal{F}$. On input x, C_R just calls the oracle P

with x and then says *correct* if and only if V-$R(x, P(x)) = \mathbf{T}$. Such a trivial checker was already considered by Blum and Kannan[4] for the extended gcd problem.

Since we can verify it in NC^1, there exists an deterministic NC^1 checker for INVERSE. More importantly, we will prove that every complete function in DET has such a checker. Before stating the theorem, we need a few definitions. If f and g are two integer functions we define the diamond composition $f \diamond g$ by $f \diamond g(n) = \max_{1 \leq t \leq g(n)}\{f(t)\}$. A class of integer functions is *robust* if it is closed under sum, product and if the diamond composition of any two functions can be bounded from above by a function in the class. A complexity class is *robust* when the class of integer resource functions that defines the complexity class is robust. The complexity class AC^0 is robust since the polynomial functions that bound the size of the circuits and the constant functions that bound their depth are robust.

The next theorem is a scaled down version of Beigel's theorem[16] which states that if R_1 and R_2 are polynomial time equivalent decision problems, then from a polynomial time checker for R_1 it is possible to construct a polynomial time checker for R_2. Kannan [16] has sketched an extension of that construction to the case when the reduction is taken from a robust complexity class. Here the checker will be in NC^1, and the reduction in AC^0.

In Beigel's construction, for the given input two solutions are computed using the program in two different ways. An important step is then to decide if they are both valid outputs. The existing checker guarantees that this problem has to be solved only when at least one of the solutions is indeed a valid output. In the case of decision problems (or in general in case of functions) then the problem reduces to the trivial question if the two solutions are equal. Here we consider not only decision problems but also relations which might have several outputs for a given input. We will be able to extend the construction for those relations for which the above problem can also be decided in NC^1. We say that the *conditional verification* of a relation R is the promise decision problem CV-R, where on input (x, y_1, y_2), with the promise that $(x, y_1) \in R$, by definition CV-$R(x, y_1, y_2) = \mathbf{T}$ iff $(x, y_2) \in R$. Observe that if R is a function then CV-$R(x, y_1, y_2) = \mathbf{T}$ iff $y_1 = y_2$.

THEOREM 6.1. *Let R_1 and R_2 two AC^0 equivalent problems. If there exists an NC^1 deterministic checker for R_1 and* CV-$R_2 \in NC^1$ *then there exists also an NC^1 deterministic checker for R_2.*

PROOF. For the sake of completeness, we describe the construction. Let $R_{1,2}$ be the AC^0 circuit family for the reduction of R_1 to R_2 and $R_{2,1}$ the AC^0

circuit family for the reduction of R_2 to R_1. Let C_1 be the NC^1 checker for R_1, we will construct an NC^1 checker C_2 for R_2, that is an NC^1 boolean circuit family with oracle gates for R_2 that satisfies the checker definition.

Given a program (oracle) P_2 for R_2, first we define a program P_1 for R_1. On input y, P_1 uses the reduction of R_1 to R_2 given by the $R_{1,2}$ circuit family. The oracle queries in the circuits are answered by P_2.

Now we describe the computation of the checker C_2 on input x with oracle P_2.

1. It queries P_2 on x, let the answer be r_x.
2. Also on input x, it uses the reduction of R_2 to R_1 given by the $R_{2,1}$ circuit family. Let's suppose that the oracle queries in the circuit are $y_1, \ldots, y_t$. All these queries are answered by P_1 as defined above. Let the result of the reduction be s_x.
3. It computes $C_1^{P_1}(y_i)$ for $i = 1, \ldots, t$.

Given all these computations, the result $C_2^{P_2}(x)$ is evaluated as follows. If there exists $1 \leq i \leq t$ such that $C_1^{P_1}(y_i) = buggy$ then $C_2^{P_2}(x) = buggy$. Otherwise $C_2^{P_2}(x) = correct$ if and only if CV-$R_2(x, s_x, r_x) = \mathbf{T}$.

We now justify simultaneously that the checker C_2 satisfies the definition, and that whenever in the evaluation it has to compute CV-$R_2(x, s_x, r_x)$, its promise is satisfied. Let's first suppose that P_2 computes R_2, that is for every u, we have $(u, P_2(u)) \in R_2$. In particular we have then $(x, r_x) \in R_2$. This also implies that P_1 computes R_1, since the reduction $R_{1,2}$ computes correctly R_1 if the oracle queries are answered according to R_2. Therefore $(x, s_x) \in R_2$, since the reduction $R_{2,1}$ computes correctly R_2 if the oracle queries are answered according to R_1. Thus CV-$R_2(x, s_x, r_x) = \mathbf{T}$ and $C_2^{P_2}(x) = correct$. Let's suppose now that $(x, r_x) \notin R_2$. If there exists $1 \leq i \leq t$ such that $C_1^{P_1}(y_i) = buggy$ then $C_2^{P_2}(x) = buggy$. Otherwise $(y_i, P_1(y_i)) \in R_1$ for $1 \leq i \leq t$, and $(x, s_x) \in R_2$ since again, in the reduction $R_{2,1}$ the oracle queries are answered according to R_1. Therefore the promise is satisfied, and since CV-$R_2(x, s_x, r_x) = \mathbf{F}$, in that case also we have $C_2^{P_2}(x) = buggy$.

Since the checker C_1 and the promise problem CV-R_2 are in NC^1, and both reductions $R_{1,2}$ and $R_{2,1}$ are in the robust class AC^0 which is included in NC^1, the checker C_2 belongs to NC^1. □

COROLLARY 6.2. *Every complete problem in DET whose conditional verification is in NC^1 has a deterministic NC^1 checker. In particular every DET-complete function, and also* INVERSE ELEMENT *have a deterministic NC^1 checker.*

PROOF. This is trivially true for CONSECUTIVE POWERS and INVERSE since their verification is in NC^1 by theorem 5.1. By theorem 6.1 it is also true for all the other complete problems in DET whose conditional verification is in NC^1. For every function f, CV-f is trivially in NC^1. Finally, CV-INVERSE ELEMENT is also in NC_1, since if we know that c/d is the inverse element then c'/d' is also the inverse element if and only if $cd' = c'd$. □

In the NC^1 checkers constructed for DET, the property that the calls made to the oracle are independent from each other leads to another result. If we extend our problem by including the inputs to the calls to the oracle then we will have a new problem which solves the original one and which can be verified in NC^1. We formalize this in the following definition. The relation R' is an NC^1 *extension* of the relation R if there exists an NC^1 family of circuits C and a polynomial $k(= k(n))$ such that for every x_0 of size n, $C(x_0) = (x_1, \ldots, x_k)$ and $R' = \{((x_0, \ldots, x_k), (y_0, \ldots y_k)) : (x_1, \ldots, x_k) = C(x_0)$ and $(x_i, y_i) \in R$ for $0 \leq i \leq k\}$.

THEOREM 6.3. *Every complete problem in DET whose conditional verification is in NC^1 has an NC^1 extension whose verification is in NC^1.*

The easily verifiable extensions of most problems in DET are not natural. For example CONSECUTIVE POWER is an extension of POWER. A rare exception is INVERSE: this "natural" problem is an extension of the DET complete problem INVERSE ELEMENT.

Let us mention finally an open problem: V-CHARACTERISTIC POLYNOMIAL is in m-V-DET but we were unable to prove any non-trivial hardness result for it.

Acknowledgements

We are very grateful to Eric Allender for several conversations and suggested improvements on the paper. We also thank the anonymous referees for many valuable comments.
This research was supported by the ESPRIT Project 21726 RAND2.

References

[1] E. ALLENDER AND M. OGIHARA. Relationships Among PL, #L , and the Determinant. In *Proc. 9th IEEE Structure in Complexity Theory Conference*, 1994, 267–278.

[2] M. Ben-Or and R. Cleve. Computing Algebraic Formulas using a Constant Number of Registers. In *Proc. Twentieth ACM Symp. Theor. Comput.*, 1988, 254–257.

[3] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.* **18**, 1984, 147–150.

[4] M. Blum and S. Kannan. Designing Programs that Check Their Work. *J. Assoc. Comput. Mach.* **42**, 1995, 269–291.

[5] A. Borodin, S. Cook and N. Pippenger. Parallel computation for well-endowed ring and space-Bounded probabilistic machines. *Inform. and Control* **48**, 1983, 113–136.

[6] A. Borodin, J. von zur Gathen and J. Hopcroft. Fast parallel matrix and GCD computation. *Inform. and Control* **52**, 1982, 241–256.

[7] A. L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. *Proc. International Conference on Foundations of Computation Theory, Lecture Notes in Computer Science* **199**, 1985, 63–69.

[8] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Inform. and Control* **64**, 1985, 2–22.

[9] L. Csanky. Fast Parallel Matrix Inversion Algorithms. *SIAM J. Comput.* **5**(4), 1976, 618–623.

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

[11] J. Dixon. An isomorphism criterion for modules over a principal ideal domain. *Linear and Multilinear Algebra* **8**, 1979, 69–72.

[12] S. Fenner, L. Fortnow and S. Kurtz. Gap-definable counting classes. In *Proc. 6th IEEE Structure in Complexity Theory Conference*, 1991, 30–42.

[13] M. Garzon and Y. Zalcstein. An NC^2 algorithm for testing similarity of matrices. *Inform. Process. Lett.* **30**, 1989, 253–254.

[14] J. von zur Gathen. Parallel Linear Algebra. In *Synthesis of Parallel Algorithms*, chapter 13, ed. J. H. Reif. Morgan Kaufmann Publishers, 1993, 574–615.

[15] O.H. Ibarra, S. Moran, and L.E. Rosier. A note on the parallel complexity of computing the rank of order n matrices. *Inform. Process. Lett.* **11**, 1980, 162.

[16] S. KANNAN. *Program Checkers for algebraic problems*. PhD thesis, PhD thesis, International Computer Science Institute, Berkeley, CA, 1989.

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

[18] W. RUZZO. On uniform circuit complexity. *J. Comput. System Sci.* **22**, 1981, 365–383.

[19] S. TODA. *Counting Problems Computationally Equivalent to Computing the Determinant*. Tech. Rep. CSIM 91–07, 1991.

[20] J. TORÁN. Complexity classes defined by counting quantifiers. *J. Assoc. Comput. Mach.* **38**, 1991, 753–744.

[21] L.G. VALIANT. Completeness classes in algebra. In *Proc. Eleventh Ann. ACM Symp. Theor. Comput.*, 1979, 249–261.

[22] L.G. VALIANT. Why is Boolean complexity theory difficult? In *London Mathematical Society Lecture Notes Series 16 9*, ed. M. S. PATERSON. Cambridge University Press, 1992.

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

Manuscript received 24 August 1995

MIKLOS SANTHA
CNRS, URA 410,
Université Paris-Sud
91405 Orsay, France
santha@lri.fr

SOVANNA TAN
Univ. Paris XII-Val de Marne
61, av. du Général de Gaulle
94 010 Créteil Cedex, France
sovanna@univ-paris12.fr