Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 52 (2008)

The Orbit problem is in the GapL Hierarchy

V. Arvind[1] and T. C. Vijayaraghavan[2]

[1] The Institute of Mathematical Sciences, Chennai 600 113, India
[2] Chennai Mathematical Institute, SIPCOT IT Park Padur PO, Siruseri 603103 India

Abstract. The *Orbit problem* is defined as follows: Given a matrix $A \in \mathbb{Q}^{n\times n}$ and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Q}^n$, does there exist a non-negative integer i such that $A^i\mathbf{x} = \mathbf{y}$. This problem was shown to be in deterministic polynomial time by Kannan and Lipton in [7]. In this paper we place the problem in the logspace counting hierarchy GapLH. We also show that the problem is hard for $\mathrm{C_=L}$ with respect to logspace many-one reductions.

1 Introduction

The *Orbit problem* is defined as follows.

> Given $A \in \mathbb{Q}^{n\times n}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Q}^n$, does there exist a non-negative integer i such that $A^i\mathbf{x} = \mathbf{y}$.

The goal of this paper is to give a new upper bound for the complexity of the orbit problem using logspace counting classes. We show that the orbit problem is in $\mathrm{AC}^0(\mathrm{GapL})$, and hence is in NC^2 as $\mathrm{AC}^0(\mathrm{GapL}) \subseteq \mathrm{TC}^1 \subseteq \mathrm{NC}^2$.

In a celebrated paper, Kannan and Lipton in [7] gave a polynomial time algorithm for the orbit problem. Their approach is to reduce it to the *Matrix power problem.* In the matrix power problem, we are given two matrices $B, D \in \mathbb{Q}^{n\times n}$ as input and we need to check if there exists a non-negative integer i such that $B^i = D$. Kannan and Lipton further show that (B, D) is a yes instance of the matrix power problem if and only if $B^i = q(B)$ for some nonnegative integer i, where $q(x) \in \mathbb{Q}[x]$ is a polynomial that depends on B and D and its coefficients can be computed in polynomial time. Here the degree of $q(x)$ is one less than the degree of the minimal polynomial of B. The rest of the algorithm in [7] focuses on checking if there is an $i \in \mathbb{Z}^+$ satisfying $B^i = q(B)$. Assume that we have computed the polynomial $q(x)$, and let α be an eigenvalue of B. Now, if there exists $i \in \mathbb{Z}^+$ such that $B^i = q(B)$ then $\alpha^i = q(\alpha)$. The algorithm in [7] uses this fact repeatedly while considering different cases: when $q(x)$ has a root that is not a root of unity, or when all roots of $q(x)$ are roots of unity with multiplicity 1, or the case when all the roots of $q(x)$ are roots of unity but with at least one root of multiplicity greater than 1. Kannan and Lipton design their algorithm based on this case analysis.

In this paper, we broadly follow the Kannan-Lipton algorithm [7], but differently analyze the complexity of the main steps involved in it. As a consequence,

ISSN 1433-8092

we modify several subroutines in the algorithm. Since these steps basically require linear algebraic computation over $\mathbb{Q}$, we obtain an upper bound in the GapL hierarchy. Some of the steps involve checking if a set of vectors are linearly independent over $\mathbb{Q}$, computing the determinant of a matrix over $\mathbb{Q}$, computing the inverse of a matrix, computing powers and the minimal polynomial of a rational matrix etc. We crucially use earlier work [1, 5, 6] classifying the complexity of various linear-algebraic problems using logspace counting classes. Among the new observations, we show that testing if all roots of a univariate polynomial over $\mathbb{Q}$ are complex roots of unity is in $\mathrm{AC}^0(\mathrm{GapL})$. Furthermore, if all roots of a polynomial are complex roots of unity we can factorize $p(x)$ into its irreducible factors in $\mathrm{AC}^0(\mathrm{GapL})$.

Finally, we show that the orbit problem is hard for $\mathrm{C}_=\mathrm{L}$ with respect to logspace many-one reductions.

2 Basic Results

In this section we recall basic definitions, notation, and results.

Definition 1. *A complex number θ is a n^{th} root of unity if $\theta^n - 1 = 0$. Furthermore, θ is a* primitive *n^{th} root of unity if θ is a n^{th} root of unity and $\theta^m - 1 \neq 0$ for all integers $0 < m < n$.*

Clearly, an n^{th} root of unity is of the form $e^{(2\pi\sqrt{-1})j/n}$ for $0 \leq j \leq (n-1)$. Also, $e^{(2\pi\sqrt{-1})j/n}$ is a primitive n^{th} root of unity if and only if $\gcd(j, n) = 1$.

Let $\varphi(j)$ denote the Euler totient function: the number of positive integers less than and relatively prime to j.

Definition 2. *Let $\theta_1, \ldots, \theta_{\varphi(j)}$ be primitive j^{th} roots of unity. Then, the j^{th} cyclotomic polynomial is defined as $C_j(x) = \prod_{i=1}^{\varphi(j)}(x - \theta_i)$.*

It is well-known that $C_j(x)$ is irreducible over $\mathbb{Q}$ and hence it must divide any polynomial $h(x) \in \mathbb{Q}[x]$ that has as root one of the primitive n^{th} roots of unity.

Proposition 1. *Let $h(x) \in \mathbb{Q}[x]$. If $h(\theta) = 0$ for a primitive n^{th} root of unity θ then $h(\theta') = 0$ for every other primitive n^{th} root of unity θ'.*

We recall the definition of logspace counting classes such as GapL and $\mathrm{C}_=\mathrm{L}$ from [2]. Many of the problems to follow are shown to be in the GapL hierarchy. We refer to [2] for more details.

Definition 3. *We define GapLH_1 to be GapL. For $i \geq 1$, let GapLH_{i+1} be the class of all functions $f : \Sigma^* \to \mathbb{Z}$, such that for some logspace-bounded nondeterministic oracle Turing machine M with a function $g \in \mathrm{GapLH}_i$ as oracle, we have $f(x) = acc_M(x) - rej_M(x)$. We denote the GapL hierarchy by GapLH. Here we assume the Ruzzo-Simon-Tompa oracle access mechanism as given in [2].*

Definition 4. *We say that a language $L \in \mathrm{AC}^0(\mathrm{GapL})$ if there exists a logspace uniform AC^0 oracle circuit family $\{C_n\}_{n\geq 1}$ with oracle gates computing a function $g \in \mathrm{GapL}$, such that on any input x of length n, we have $C_n(x) = 1$ if and only if $x \in L$.*

Theorem 1. [2] $\mathrm{GapLH} = \mathrm{AC}^0(\mathrm{GapL})$.

We assume that each rational entry of an input matrix $A \in \mathbb{Q}^{n\times m}$ is given in terms of its numerator and denominator. Also, we will assume that an algorithm computing $\det(A)$ for a rational matrix $A \in \mathbb{Q}^{n\times n}$ will output two integers p and q such that $\det(A) = p/q$. Furthermore, we will *not* require that p and q be relatively prime, that is $\gcd(p, q)$ need not be 1. This assumption is necessary because computing the GCD of two integers is not known to be in NC. This representation of rationals does not affect our algorithm so long as the size in binary of the two integers p and q is bounded by a polynomial in the size of the input. We will make a similar assumption for other computations involving rational inputs. We now recall the following results concerning rational matrices. These are usually stated for integer matrices.

Lemma 1. *Let $A \in \mathbb{Q}^{n\times m}$ be the given input rational matrix. Then,*

1. [2, 3, 8–10] *When $n = m$, computing the determinant of A denoted by $\det(A)$, computing the $(i,j)^{th}$ entry of A^{-1}, and computing the $(i,j)^{th}$ entry of A^l for a given positive integer l are complete for GapL under logspace many-one reductions.*
2. [1] *Checking if the set of column vectors of A are linearly dependent is complete for $\mathrm{C_=L}$ under logspace many-one reductions.*
3. [1] *Let $\mathbf{b} \in \mathbb{Q}^n$ be an n-dimensional rational vector. Then, determining if the system of linear equations $A\mathbf{x} = \mathbf{b}$ has a rational vector $\mathbf{x}$ as a solution is complete for $\mathrm{L}^{\mathrm{C_=L}}$ under logspace truth-table reductions.*
4. *Computing a maximal set of linearly independent columns from A is in $\mathrm{FL}^{\mathrm{C_=L}}$.*
5. [5] *Given $B \in \mathbb{Q}^{n\times n}$, we can compute the coefficients of the minimal polynomial of B in $\mathrm{AC}^0(\mathrm{GapL})$.*

Proof. Let $A \in \mathbb{Q}^{n\times m}$ be the given input rational matrix. Let $A_{ij} = p_{ij}/q_{ij}$, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Also, we can assume the size of each p_{ij} and q_{ij} is at most $\max(m, n)$. Let q be the product of all the denominators of the entries in A. It follows from [4] that, for any positive integer n, we can compute the i^{th} bit of the product of n integers, each of size n, using an NC^1 circuit and therefore we can compute q which is a product of nm integers in NC^1 as well. Let us consider the matrix (qA), obtained by multiplying each entry of A by q. Clearly (qA) is an integer matrix and $A = (qA)/q$. In problems involving an additional vector $\mathbf{b}$, we multiply q with the denominators of the entries occurring in $\mathbf{b}$ to reduce the problem to the case when the inputs are integer matrices. In all these cases, the size of q as well as entries of (qA) and $(q\mathbf{b})$ are bounded by a polynomial in the size of the input, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Thus we can compute the i^{th} bit of any entry of these matrices in logspace. The results stated above then follow by applying known complexity bounds (proven in the references appearing in the Theorem statement) on linear algebraic problems involving integer matrices to (qA), and $(q\mathbf{b})$.

The next lemma shows that a solution to a feasible system of linear equations over rationals can be computed in the GapL hierarchy.

Lemma 2. *Let $A \in \mathbb{Q}^{m\times n}$ and $\mathbf{b} \in \mathbb{Q}^n$. If the system of linear equations $A\mathbf{x} = \mathbf{b}$ is feasible, then a solution to it can be computed in* $\mathrm{AC}^0(\mathrm{GapL})$.

Proof. First, we can compute a maximal linearly independent set of columns of A with an $\mathrm{FL}^{\mathrm{GapL}}$ computation as follows: for each index i such that $2 \le i \le n$ we check if the i^{th} column A_i of A is linearly independent of the first $(i-1)$ columns $\{A_1, A_2, \ldots, A_{i-1}\}$. If so, we output the index i. Let $S \subseteq [n]$ denote the set of indexes output, and let A' denote the matrix formed by these linearly independent columns. It is easy to note that $A\mathbf{x} = \mathbf{b}$ is feasible if and only if $A'\mathbf{z} = \mathbf{b}$ is feasible, where $\mathbf{z}$ is an $n - |S|$ dimension vector of indeterminates. Furthermore, given a solution $\mathbf{z}$ for $A'\mathbf{z} = \mathbf{b}$ we can extend it to a solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ by setting $x_i = 0$ for $i \notin S$. Since the columns of A' are linearly independent, the solution $\mathbf{z}$, if it exists, is unique. In order to compute $\mathbf{z}$, we perform another $\mathrm{FL}^{\mathrm{GapL}}$ computation in which we output a maximal linearly independent set of *rows* of A' (using the same method as above). Let $T \subseteq [m]$ denote the set of $|S|$ row indexes output, and let B denote the corresponding $|S| \times |S|$ matrix. Furthermore, let $\mathbf{b}'$ denote the corresponding $|S|$-dimensional subvector of $\mathbf{b}$ picked out by index set T. Clearly, $A'\mathbf{z} = \mathbf{b}$ if and only if $B\mathbf{z} = \mathbf{b}'$ for any vector $\mathbf{z}$. Finally, since B is invertible we can compute B^{-1} using an $\mathrm{L}^{\mathrm{GapL}}$ computation to obtain the solution vector $\mathbf{z} = B^{-1}\mathbf{b}'$. The solution computed above can then be extended to obtain a solution for the input system $A\mathbf{x} = \mathbf{b}$. Since the number of levels of oracle calls involved while these steps are composed remains a constant (at the most bounded by 5), we get the required $\mathrm{AC}^0(\mathrm{GapL})$ upper bound.

We also need the following result of [6] to compute the GCD of two polynomials over $\mathbb{Q}$.

Lemma 3. [6] *Given polynomials $f(x)$, $g(x) \in \mathbb{Q}[x]$ as input their GCD can be computed in the complexity class* PL, *which is contained in* $\mathrm{L}^{\mathrm{GapL}}$.

3 Kannan-Lipton algorithm

We now proceed to show that the orbit problem is in GapLH, and hence in $\mathrm{AC}^0(\mathrm{GapL})$. We first describe the main steps in Kannan-Lipton algorithm [7] for the orbit problem. They first reduce the orbit problem to the *Matrix Power problem*: Given $B, D \in \mathbb{Q}^{n\times n}$ does there exists a non-negative integer i such that $B^i = D$.

We next describe the reduction. Let $(A, \mathbf{x}, \mathbf{y})$ be an instance of the orbit problem. Let $V \subseteq \mathbb{Q}^n$ denote the subspace spanned by $\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, \dots, A^{n-1}\mathbf{x}\}$. Clearly V is k-dimensional for the largest k such that $\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, \dots, A^{k-1}\mathbf{x}\}$ are linearly independent, and a basis for V is this set $\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, \dots, A^{k-1}\mathbf{x}\}$. We can compute this basis in $\mathrm{AC}^0(\mathrm{GapL})$: with an $\mathrm{L}^{\mathrm{GapL}}$ computation we can first compute $A^j\mathbf{x}$ for $1 \leq j \leq n-1$. The output of this machine is taken as input for another $\mathrm{L}^{\mathrm{GapL}}$ computation that finds the largest k such that $\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, \dots, A^{k-1}\mathbf{x}\}$ is linearly independent.

The subspace V is A-invariant. That is, $x \in V$ if and only if $A^i\mathbf{x} \in V$ for each $i \geq 0$. Consequently, $(A, \mathbf{x}, \mathbf{y})$ is a 'yes' instance for the orbit problem only if $\mathbf{y} \in V$. We can check if $\mathbf{y} \in V$ in $\mathrm{L}^{\mathrm{GapL}}$. If $\mathbf{y} \notin V$ then the reduction outputs the pair (O_n, I_n) of the matrix power problem, where O_n is the $n \times n$ zero matrix and I_n is the identity matrix. Therefore, in the sequel we can assume that $\dim(V) = k$ and $\mathbf{y} \in V$. Let

$$A^k\mathbf{x} = \sum_{j=0}^{k-1} \alpha_j A^j\mathbf{x}, \quad \mathbf{x} = \sum_{j=0}^{k-1} \beta_j A^j\mathbf{x}, \quad \mathbf{y} = \sum_{j=0}^{k-1} \gamma_j A^j\mathbf{x}.$$

Notice that $\beta_0 = 1$ and $\beta_j = 0$ for all $j > 0$. We can compute the scalars $\alpha_j, \beta_j, \gamma_j$ in $\mathrm{L}^{\mathrm{GapL}}$ by solving each of the above three systems of linear equations using Cramér's rule.

The $k \times k$ matrix for the linear transformation A from V to V has e_{j+1}, where $1 \leq j \leq k-1$, as its first $k-1$ columns and $(\alpha_0, \dots, \alpha_{k-1})^T$ as the last column.[3] Call this matrix A'. Likewise, let $\mathbf{x}' = (\beta_0, \dots, \beta_{k-1})^T$ and $\mathbf{y}' = (\gamma_0, \dots, \gamma_{k-1})^T$. Clearly, $(A', \mathbf{x}', \mathbf{y}')$ is a yes instance of the orbit problem if and only if $(A, \mathbf{x}, \mathbf{y})$ is a yes instance. This is because $A', \mathbf{x}', \mathbf{y}'$ are essentially A, $\mathbf{x}$, and $\mathbf{y}$ expressed using the basis $\mathbf{x}, A\mathbf{x}, \dots, A^{k-1}\mathbf{x}$ of V. Now, let C denote the $k \times k$ invertible matrix $[\mathbf{x}'|A'\mathbf{x}'|\cdots|A'^{k-1}\mathbf{x}']$. Similarly, let C' denote the $k \times k$ matrix $[\mathbf{y}'|A'\mathbf{y}'|\cdots|A'^{k-1}\mathbf{y}']$. Then, there exists an $i \geq 0$ such that $A'^i\mathbf{x}' = \mathbf{y}'$ if and only if $A'^iC = C'$, which we can rewrite as $A'^i = C'C^{-1}$ as C is invertible. Thus, $(A', C'C^{-1})$ is the instance of the matrix power problem to which we have reduced $(A, \mathbf{x}, \mathbf{y})$. We formally state this as a lemma.

Lemma 4. *The orbit problem can be reduced to the matrix power problem in* $\mathrm{AC}^0(\mathrm{GapL})$.

Proof. The correctness of the reduction follows from the above argument. To see that it is computable in $\mathrm{AC}^0(\mathrm{GapL})$, we note that a set of $\mathrm{L}^{\mathrm{GapL}}$ computations need to be carried out that involves a nesting of at most two levels of GapL queries.

We now turn to the matrix power problem. Let $B, D \in \mathbb{Q}^{n\times n}$ be an input instance. Following [7] we further reduce it to a more tractable problem.

[3] Here the vectors e_{j+1} denote the standard basis vectors of $\mathbb{R}^k$.

Lemma 5. *Given $B, D \in \mathbb{Q}^{n\times n}$, we can compute in* $\mathrm{AC}^0(\mathrm{GapL})$ *a polynomial $q(x) \in \mathbb{Q}[x]$ of degree at most $n-1$ such that there exists a non-negative integer i satisfying $B^i = D$ if and only if $B^i = q(B)$.*

Proof. Let $p(x)$ be the minimal polynomial of B (computable in $\mathrm{AC}^0(\mathrm{GapL})$ [5]). We have $p(B) = 0$ and $\deg(p(x)) = r \leq n$. If there exists an $i \geq 0$ such that $B^i = D$, then we show that there is a polynomial $q(x)$ of degree at most $n-1$ such that $D = q(B)$. We divide x^i by $p(x)$ and take the remainder as the polynomial $q(x)$. Thus, $q(x) \equiv x^i (\mathrm{mod}\ p(x))$, and $\deg(q(x)) \leq (\deg(p(x)) - 1) \leq (n-1)$. Therefore, (B, D) is a yes instance of the matrix power problem only if such a polynomial $q(x)$ exists. We can test this and compute the coefficients of $q(x)$ by solving the following system of n^2 linear equations over n variables: $\sum_{j=0}^{(r-1)} q_j B^j = D$ where the unknowns are the coefficients q_j of the polynomial $q(x)$. Given B and D as input, an $\mathrm{L}^{\mathrm{GapL}}$ computation will first compute B^j for $1 \leq j \leq n-1$ and pass it as input to another $\mathrm{L}^{\mathrm{GapL}}$ computation to check the feasibility of the above system and find a solution $q(x)$ using Lemma 2. Thus, the polynomial $q(x)$ can be computed in $\mathrm{AC}^0(\mathrm{GapL})$. Clearly, $B^i = q(B)$ if and only if $B^i = D$.

As mentioned previously, the overall reduction from the orbit problem involves composing computations, each of which is in some constant level of the GapL hierarchy. Since we will do only a constant number of such compositions the overall computation is still in a constant level of the GapL hierarchy. Continuing with the proof, as a consequence of Lemma 4 and Lemma 5, we obtain the following.

Corollary 1. *Given an instance $A \in \mathbb{Q}^{n\times n}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Q}^n$ of the orbit problem, for some $m \leq n$ we can compute a matrix $B \in \mathbb{Q}^{m\times m}$ and a polynomial $q(x) \in \mathbb{Q}[x]$ of degree at most $(m-1)$ in* $\mathrm{AC}^0(\mathrm{GapL})$*, such that $A^i\mathbf{x} = \mathbf{y}$ for some $i \geq 0$ if and only if $B^i = q(B)$.*

The following easy lemma is a useful property for the next step.

Lemma 6. *Suppose $p(x) \in \mathbb{Q}[x]$ is the minimal polynomial of matrix $B \in \mathbb{Q}^{n\times n}$. For any two polynomials $r(x), q(x) \in \mathbb{Q}[x]$ we have $r(B) = q(B)$ if and only if $r(x) = q(x)(mod\ p(x))$.*

In particular, it follows that $B^i = q(B)$ for some $i \geq 0$ if and only if $x^i = q(x)(\mathrm{mod}\ p(x))$. As a consequence of Corollary 1 and Lemma 6, it suffices to solve in $\mathrm{AC}^0(\mathrm{GapL})$ the problem of checking if $x^i = q(x)(\mathrm{mod}\ p(x))$ for some $i \geq 0$, where $p(x)$ is the minimal polynomial of the matrix B. Given polynomials $p, q \in \mathbb{Q}[x]$, where p is a monic, the goal is to test in $\mathrm{AC}^0(\mathrm{GapL})$ if $x^i = q(x)(\mathrm{mod}\ p(x))$ for some $i \geq 0$. Following [7], we need to handle different cases depending on the roots of $p(x)$. A crucial property is a bound from algebraic number theory [7, Theorem 3]. For a polynomial $f \in \mathbb{Q}[x]$ let $|f|$ denote the ℓ_2 norm of the vector of its coefficients.

Theorem 2. [7, Theorem 3] *There is a polynomial P such that for any algebraic number $\alpha \in \mathbb{C}$ that is not a root of unity and any polynomial $q(x) \in \mathbb{Q}[x]$, if $\alpha^i = q(\alpha)$ for $i \in \mathbb{Z}^+$ then $i \leq P(deg(f_\alpha), \log(|f_\alpha|), \log(|q|))$, where $f_\alpha \in \mathbb{Q}[x]$ is the minimal polynomial of α.*

Thus, if the given polynomial $p(x)$ has a root α that is not a root of unity then, by Theorem 2, we can test if there is an i such that $x^i = q(x) (\text{mod}\, p(x))$ by trying the polynomially many values of i in the range $i \leq P(\deg(f_\alpha), \log(|f_\alpha|), \log(|q|))$. Since f_α is an irreducible factor of $p(x)$, we know that $|f_\alpha|$ is polynomially bounded by $|p|$. Hence the range of values for i is indeed polynomially bounded by the input size. Indeed, since this test involves only division of polynomials, using the result of [4, Corollary 6.5] it can be carried out in logspace.

We now consider the case when all the roots of $p(x)$ are complex roots of unity. We use key properties of the cyclotomic polynomial $C_j(x)$. First we show that $C_j(x)$ can be computed in $\mathrm{AC}^0(\mathrm{GapL})$ by an algorithm that takes j in unary as input.

Lemma 7. *Given 1^j as input the j^{th} cyclotomic polynomial $C_j(x)$ can be computed in $\mathrm{AC}^0(\mathrm{GapL})$.*

Proof. The j^{th} cyclotomic polynomial $C_j(x) = \prod_{r=1}^{\varphi(j)}(x - \omega_r)$ where the ω_r are the $\varphi(j)$ different primitive j^{th} roots of unity and $C_j(x)$ is an irreducible factor of $x^j - 1$.

We first define the polynomial $t_j(x) = \prod_{i=1}^{j-1}(x^i - 1)$. The polynomial t_j is of degree $j(j-1)/2$ with rational coefficients. We can also compute the coefficients of this polynomial in logspace by substituting a large power of 2 for the indeterminate x and extracting the bits of the coefficients from the resulting value. Furthermore, it is clear that $b_j(x) = \gcd(t_j(x), x^j - 1)$ contains as roots precisely all non-primitive j^{th} roots of unity. Therefore, it follows that $C_j(x)$ is the quotient obtained on dividing $x^j - 1$ by $b_j(x)$. Given the coefficients of $t_j(x)$ we can apply Lemma 3 to compute $\gcd(t_j(x), x^j - 1)$ in $\mathrm{AC}^0(\mathrm{GapL})$. Therefore, the overall computation is clearly in $\mathrm{AC}^0(\mathrm{GapL})$.

We can easily show that testing if all roots of $p(x)$ are complex roots of unity is in $\mathrm{AC}^0(\mathrm{GapL})$.

Lemma 8. *Given $p(x) \in \mathbb{Q}[x]$ as input we can test in $\mathrm{AC}^0(\mathrm{GapL})$ if all roots of $p(x)$ are complex roots of unity, and if so we can factorize $p(x)$ into its irreducible factors in $\mathrm{AC}^0(\mathrm{GapL})$.*

Proof. Let $\deg(p(x)) = d$. We first compute $C_j(x), 1 \leq j \leq d$ using Lemma 7. Next, since division of polynomials with rational coefficients can be carried out in logspace using [4, Corollary 6.5], we can find the highest power of $C_j(x)$ that divides $p(x)$ in logspace. Putting it together will give us all the irreducible factors of $p(x)$, with multiplicity, from the set $C_j(x), 1 \leq j \leq d$.

After applying Lemma 8 we will know whether $p(x)$ has a root that is not a root of unity (in which case we can use the easy logspace algorithm based on

Theorem 2). Thus, we now consider only the case when $p(x) = \prod_{j=1}^{d} C_j(x)^{k_j}$, where $k_j \geq 0$. An easy and useful lemma is the following.

Lemma 9. *Let $q(x)$ be an arbitrary polynomial and let $C_j(x)$ be the j^{th} cyclotomic polynomial. The congruence $x^\ell \equiv q(x)$ $(mod\ C_j(x))$ holds for some nonnegative integer ℓ if and only if it holds for some* unique *ℓ in the range $0 \leq \ell \leq (j-1)$.*

Proof. Since $C_j(x)$ divides $x^j - 1$, it follows that $x^\ell \equiv q(x) \pmod{C_j(x)}$ implies $x^{\ell\ (mod\ j)} \equiv q(x) \pmod{C_j(x)}$.

Using the above result we first handle the case when $k_j \in \{0,1\}$ in $p(x) = \prod_{j=1}^{d} C_j(x)^{k_j}$.

Lemma 10. *If $p(x) = \prod_{j=1}^{d} C_j(x)^{k_j}$ for $k_j \in \{0,1\}$, then the problem of testing for a given polynomial $q(x) \in \mathbb{Q}[x]$ if $x^i \equiv q(x)$ $(mod\ p(x))$ for some positive integer i, is in* $\mathrm{AC}^0(\mathrm{GapL})$.

Proof. By the Chinese remainder theorem, it suffices to check if there is a positive integer i such that $x^i \equiv q(x) \pmod{C_j(x)}$ for every C_j such that $k_j = 1$. By Lemma 9 there is an $i \geq 0$ such that $x^i \equiv q(x) \pmod{C_j(x)}$ if and only if there is an $i_j \in \{0, 1, \cdots, j-1\}$ such that $x^{i_j} \equiv q(x) \pmod{C_j(x)}$. Notice that such an i_j, if it exists, has to be *unique*. If for some C_j such that $k_j = 1$ no such i_j exists we reject the input. Otherwise, we would have computed i_j for each C_j with $k_j = 1$. We only need to check if there exists a positive integer i such that

$$i \equiv i_j (\mathrm{mod}\ j) \tag{1}$$

for all j such that $k_j = 1$. We cannot directly apply the chinese remainder theorem to check this congruence as the different j's need not be relatively prime. However, since each such j is bounded by d, it follows that j has logarithmically many bits. Hence we can compute the prime factorization for each j such that $k_j = 1$ in deterministic logspace. Let $p_1, p_2, \ldots, p_k$ denote the set of all prime factors of any $j \leq d$. Clearly, each p_i is logarithmic in size and k is also logarithmic in the input size. Then we can rewrite the congruences in Equation 1 above as

$$i \equiv i_j (\mathrm{mod}\ p_\ell^{r_{j,\ell}}), \tag{2}$$

where $1 \leq \ell \leq k$ and j such that $k_j = 1$ and $j = \prod p_\ell^{r_{j,\ell}}$. Now, for each prime p_ℓ above we club together all congruences of the type $i \equiv i_j \pmod{p_\ell^{r_{j,\ell}}}$ for all the j's. Let j' be a value of j for which $r_{j',\ell}$ is maximum. Then, a necessary condition that Equation 2 has a solution for i is that $i_j = i_{j'} \pmod{p_\ell^{r_{j,\ell}}}$ for all j which we can check in logspace. Having checked this condition we can replace all the congruences in Equation 2 by the single congruence $i \equiv i_{j'} (\mathrm{mod}\ p_\ell^{r_{j',\ell}})$. Thus, for each p_ℓ we will have a single congruence and we can *now* invoke the chinese remainder theorem to check in logspace if there is a solution for Equation 1.

It now remains to handle the case when for some j, the exponent k_j of $C_j(x)$ is at least 2 in the factorization of $p(x)$.

Lemma 11. *Given $q(x) \in \mathbb{Q}[x]$ and a cyclotomic polynomial $C_j(x)$, we can compute in deterministic logspace a set $S_{q(x),j}$ of positive integers such that $|S_{q(x),j}|$ is polynomially bounded in $\log|q|$ and j, with the property that $x^i \equiv q(x)(mod\ C_j(x)^2)$ can have solutions only for $i \in S_{q(x),j}$.*

Proof. Suppose $x^i \equiv q(x)(\text{mod } C_j(x)^2)$. Then we have $x^i - q(x) = r(x)C_j(x)^2$. Taking the formal derivative on both sides we obtain $ix^{i-1} - q'(x) = 2C_j(x)r(x) + r'(x)C_j(x)^2$, implying that $ix^{i-1} - q'(x) \equiv 0 \pmod{C_j(x)}$, where $q'(x)$ and $r'(x)$ are the derivatives of $q(x)$ and $r(x)$ respectively. Let P_ℓ denote the polynomial $x^\ell \pmod{C_j(x)}$ for $0 \leq \ell \leq j-1$. Notice that each P_ℓ is of degree at most $\varphi(j) - 1$. Furthermore, let $q'_1(x) = q'(x) \pmod{C_j(x)}$. Thus, i is a candidate solution only if for some ℓ we have $iP_\ell = q'_1(x)$. We define the set $S_{q(x),j} = \{s \mid s = \frac{q'_1(x)}{P_\ell} \text{ for some } \ell\}$. Clearly, $|S_{q(x),j}| \leq j$ and can be computed in deterministic logspace.

We obtain the following corollary which limits the search space for the index i to such a set $S_{q(x),j}$.

Corollary 2. *Suppose $p(x) = \prod_{j=1}^{d} C_j(x)^{k_j}$ such that $k_{j'} \geq 2$ for some j'. Then $x^i \equiv q(x)\ (mod\ p(x))$ for some i if and only if $x^i \equiv q(x)\ (mod\ p(x))$ for some $i \in S_{q(x),j'}$.*

The rest of the algorithm is as follows: we need to check if there is an $i \in S_{q(x),j'}$ such that for each $k_j > 0$ we have $x^i \equiv q(x) \pmod{C_j(x)^{k_j}}$. Such an i is a solution. Notice that we cannot directly check this by division because $i \in S_{q(x),j'}$ may be an integer that is polynomially many bits long. Thus we need to devise a different test for checking if $x^i \equiv q(x) \pmod{C_j(x)^{k_j}}$ for a given i. This is described in our final lemma that will also complete the upper bound description.

Lemma 12. *Given as input a polynomial $q(x) \in \mathbb{Q}[x]$, and integer i (encoded in binary), a cyclotomic polynomial $C_j(x)$ and an integer k, where k and j are encoded in unary, we can test in deterministic logspace if $x^i \equiv q(x)\ (mod\ C_j(x)^k)$.*

Proof. Let ω denote a primitive j^{th} root of unity. Since $C_j(x)$ is irreducible it follows that $C_j(x)^k$ divides $x^i - q(x)$ if and only if $(x-\omega)^k$ divides $x^i - q(x)$. That means ω is a root of multiplicity k for $f(x) = x^i - q(x)$. Equivalently, we need to check if ω is a root of the ℓ^{th} formal derivative $f^{(\ell)}(x)$ of the polynomial $f(x)$ for each $0 \leq \ell \leq k-1$. Notice that $f^{(\ell)}(x)$ assumes the form $i(i-1)\cdots(i-\ell)x^{i-\ell} - q^{(\ell)}(x)$. Computing the coefficient $i(i-1)\cdots(i-\ell)$ is iterated integer multiplication that can be done in deterministic logspace. Furthermore, the ℓ^{th} derivative of the polynomial can be done term by term, which will also involve a similar iterated integer multiplication for each term and it can be done in deterministic logspace. Now, checking if ω is a root of $f^{(\ell)}(x)$ is equivalent to checking if $C_j(x)$ divides $f^{(\ell)}(x)$, again by the irreducibility of $C_j(x)$. But $f^{(\ell)}(x)$ has the nice form $i(i-1)\cdots(i-\ell)x^{i-\ell} - q^{(\ell)}(x)$ which is easy to divide by $C_j(x)$ as we can replace the exponent $i-\ell$ in the first term by $(i-\ell) \pmod{j}$. This completes the proof.

We now show that the orbit problem is hard for $\mathrm{C_{=}L}$ under logspace many-one reductions.

Theorem 3. *Orbit problem is hard for* $\mathrm{C_{=}L}$ *under logspace many-one reductions.*

Proof. Given a directed graph $G = (V, E)$, and vertices $u, v \in V$, the problem of checking is there is a directed path from u to v is NL-complete. In fact, this problem remains NL-complete for input graphs that are layered, directed, and acyclic, with u being in the first layer and the vertex v in the last layer. By a layered digraph we mean for each edge $(s,t) \in E$ in the graph if s is in layer i, then t is in layer $(i+1)$. The counting version of this problem: namely, counting the number of directed u-v paths is thus #L complete under logspace many-one reductions. Furthermore, verifying if the number of directed u-v paths is a given nonnegative integer m is $\mathrm{C_{=}L}$-complete under logspace many-one reductions. We now show a logspace many-one reduction from the above problem to the orbit problem.

Given a input layered digraph G as mentioned above, let the vertex u be indexed by 1, and vertex v be indexed by n. We then add a self-loop onto the vertex n in G, and denote the resulting graph by G'. Let A be the adjacency matrix of G', with vertex set $V = \{1, 2, \cdots, n\}$. We now check if the number of paths from 1 to n is m. Let the number of layers in G be $\ell + 1$. For every directed path from u to v in G, we have a directed walk from 1 to n in G' of length ℓ. Clearly, A is an $n \times n$ matrix with 0-1 entries, where rows and columns indexed by V. For $k \in \mathbb{Z}^+$, the $(i,j)^{th}$ entry of A^k is the number of walks from vertex i to vertex j in G. Since G is acyclic, all walks are directed paths. Define vectors $\mathbf{x} = (0, \ldots, 0, 1)^T \in \mathbb{Q}^{n\times 1}$, and $\mathbf{y} = (m, 0, \ldots, 0)^T \in \mathbb{Q}^{n\times 1}$. Since n is the unique node on the $(\ell+1)^{st}$ layer, the number of directed paths in G from 1 to n is m if and only if $A^\ell \mathbf{x} = \mathbf{y}$. More precisely, there is some $i \in \mathbb{Z}^+$ such that $A^i\mathbf{x} = \mathbf{y}$ if and only if there are exactly m directed paths in G from 1 to n.

4 Concluding remarks

The main open question here is to reduce the $\mathrm{AC}^0(\mathrm{GapL})$ upper bound shown above. Some other interesting questions also arise. In Lemma 8 we have shown that factoring uni-variate polynomials whose roots are all complex roots of unity is in $\mathrm{AC}^0(\mathrm{GapL})$. Using the LLL algorithm, all uni-variate polynomials over $\mathbb{Q}$ can be factored in polynomial time. To the best of our knowledge, there is no known P-hardness result for polynomial factorization. It would be interesting to either obtain a better complexity upper bound or show P-hardness.

Acknowledgments. Preliminary version of this paper appeared in COCOON '08. We thank the anonymous referees of COCOON '08 for their thoughtful comments.

References

1. Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.
2. Eric Allender and Mitsunori Ogihara. Relationships among PL, #L and the determinant. *RAIRO - Theoretical Informatics and Applications*, 30:1–21, 1996.
3. Carsten Damm. DET=L$^{\#L}$. Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universitat zu Berlin, 1991.
4. William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002.
5. Thanh Minh Hoang and Thomas Thierauf. The complexity of the characteristic and the minimal polynomial. *Theoretical Computer Science*, 295(1-3):205–222, 2003.
6. Thanh Minh Hoang and Thomas Thierauf. The complexity of the inertia and some closure properties of gapl. In *Proceedings of 20th IEEE Conference on Computational Complexity*, pages 28–37, 2005.
7. Ravi Kannan and Richard Lipton. Polynomial-time algorithm for the orbit problem. *Journal of the ACM*, 33(4):808–821, 1986.
8. Seinosuke Toda. Counting problems computationally equivalent to computing the determinant. Technical report 91-07, Department of Computer Science, University of Electro-Communications, Tokyo, Japan, 1991.
9. Leslie G. Valiant. Why is boolean complexity theory difficult? In *Proceedings of the London Mathematical Society symposium on Boolean function complexity*, pages 84–94, New York, NY, USA, 1992. Cambridge University Press.
10. V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *CCC '91: Proceedings of 6th Structure in Complexity Theory Conference*, pages 270–284, 1991.