Electronic Colloquium on Computational Complexity, Report No. 99 (2010)

A Note on Closure Properties of ModL

Thirumalai C. Vijayaraghavan*
Department of Computer Science
Ramakrishna Mission Vivekananda College
Evening College(Autonomous)
Mylapore, Chennai 600004, INDIA.

20^{th} June 2010

Abstract

Recently in [Vij09, Corollary 3.7] the complexity class ModL has been shown to be closed under complement assuming NL = UL. In this note we continue to show many other closure properties of ModL which include the following.

1. ModL is closed under $\leq^{\mathrm{L}}_{m}$ reduction, $\vee$(join) and $\leq^{\mathrm{UL}}_{m}$ reduction,
2. ModL is closed under $\leq^{\mathrm{L}}_{1-\mathrm{tt}}$ and $\leq^{\mathrm{UL}}_{1-\mathrm{tt}}$ reduction assuming NL = UL,
3. $\mathrm{ModL}^{\mathrm{UL}} = \mathrm{ModL}$ assuming UL = coUL,
4. $\mathrm{UL}^{\mathrm{ModL}}_{1-\mathrm{tt}} = \mathrm{ModL}^{\mathrm{UL}} = \mathrm{ModL}$ assuming NL = UL,
5. if $l \in \mathbb{Z}^+$ such that $l \geq 2$ and ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ reduction then $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$ for all $k \in \mathbb{Z}^+$ such that $k \geq 6$ is a composite number and k has at least 2 and at most l distinct prime divisors, and
6. if ModL is closed under $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reduction then $\mathrm{coC_{=}L} \subseteq \mathrm{ModL}$.

Also using [Vij09, Corollary 3.7] we show that if NL = UL and ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ and $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reduction then ModL is also closed under $\leq^{\mathrm{L}}_{l-\mathrm{ctt}}$ and $\leq^{\mathrm{L}}_{\mathrm{ctt}}$ reduction respectively.

We also show a proof of the well known result that the determinant of a matrix with entries in $\mathbb{Z}$ is computable in L-uniform TC^1 from which it follows that ModL $\subseteq$ L-uniform TC^1.

*Email:tcvijay@gmail.com,tcvasu@vsnl.net

ISSN 1433-8092

1 Introduction

The complexity class ModL was defined by Arvind and Vijayaraghavan in [AV04] (more precisely in [Vij08, Definition 1.4.1] and [AV, Definition 3.1]) to tightly classify the complexity of solving a system of linear equations modulo a composite number k (called LCON) where k is given in terms of its prime factorization such that every distinct prime power divisor that occurs in the prime factorization of k is given in the unary representation. This problem was shown to be in $\mathrm{L}^{\mathrm{GapL}}/\mathrm{poly}$ in [AV05]. Using the seminal result of [Tod91, Vin91, Dam92, Val92] that computing the determinant of an integer matrix is complete for GapL under logspace many-one reductions and the Chinese remainder theorem it follows from the definition of ModL that $\mathrm{L}^{\mathrm{ModL}} = \mathrm{L}^{\mathrm{GapL}}$ [AV04]. As a consequence we obtain $\mathrm{LCON} \in \mathrm{L}^{\mathrm{ModL}}/\mathrm{poly}$. The LCON problem has been shown to be logspace Turing reducible to many other problems on Abelian permutation groups in [Vij08, AV] and as a consequence [Vij08, AV] also show the upper bound of $\mathrm{L}^{\mathrm{ModL}}/\mathrm{poly}$ for these problems also.

1.1 Motivation and recent progress

The LCON problem was shown to be logspace many-one hard for ModL in [Vij08, Chapter 3]. However subsequently the proof of this result was found to be incorrect and recently in [AV] it is shown that LCON is logspace Turing hard for ModL. The logspace many-one hardness of LCON for ModL is yet to be shown to be true and it serves as one of the main motivation for results shown in this note.

More recently in [Vij09, Corollary 3.7] it is shown that ModL is closed under complement under the assumption that NL = UL. To prove this result it is shown in [Vij09, Theorem 3.6] that if we assume NL = UL and we have a language $L \subseteq \Sigma^*$ with $L \in \mathrm{ModL}$ then we can decide whether an input $x \in \Sigma^*$ is in L using $f \in \#\mathrm{L}$ and $g \in \mathrm{FL}$ where $g(x)$ is a prime number $p \in \mathbb{Z}^+$ that is output by g in the unary representation. The proof of [Vij09, Theorem 3.6] is similar to the result that $\mathrm{Mod}_{p^e}\mathrm{P} = \mathrm{Mod}_p\mathrm{P}$ shown in [BG92].

However to implement this proof in the logspace setting, it is shown in [Vij09, Theorem 3.4] by assuming NL = UL that if $f \in \#\mathrm{L}$ and $g \in \mathrm{FL}$ such that $g(x)$ is the unary representation of a non-negative integer in $\mathbb{Z}^+$, where $x \in \Sigma^*$ is the input then the number of ways of choosing exactly $|g(x)|$ distinct paths from amongst the $f(x)$ accepting computation paths of the NL machine corresponding to f is also in #L.

One of the main components in the proof of [Vij09, Theorem 3.4] that is shown in in [Vij09, Corollary 3.3] is that if we are given an instance G of LDAG-st-CON with vertices s and t in G such that every vertex or edge in G is in at least one path from the vertex s to the vertex t then the problem of deciding if the number of paths from s to t is at least $p \in \mathbb{Z}^+$ is in UL where p is bounded by a polynomial in the size of G. The proof of [Vij09, Corollary 3.3] is based on deterministically isolating polynomially many distinct paths from the vertex

s to the vertex t in G using a deterministic weight assigment method to the edges of G which is based on the results shown in [GK87, AHT07, Vij08]. Also [Lan97] shows many other problems that are complete for UL under logspace many-one reductions.

Let Σ be the input alphabet. While we know that if $L \subseteq \Sigma^*$ and $L \in \mathrm{NL}$ then there exists a function $f \in \#\mathrm{L}$ such that on any input $x \in \Sigma^*$ we have $x \in L$ if and only if $f(x) \geq 1$ the results shown in [Vij09, Lemma 3.1 and Lemma 3.2] (which leads to [Vij09, Corollary 3.3]) also show that we can decide if $f(x) \geq p$ for some $p \in \mathbb{Z}^+$ where p is bounded by a polynomial in $|x|$ is in NL. This seems interesting and shows a subtle difference between the complexity classes NL and PL since the results shown in [AO96] imply that if a language $L' \subseteq \Sigma^*$ and $L' \in \mathrm{PL}$ then there exists $f' \in \mathrm{GapL}$ and $g' \in \mathrm{FL}$ such that on any input $x \in \Sigma^*$ we have $x \in L'$ if and only if $f'(x) \geq |g'(x)|$ where $g'(x)$ is a positive integer in the unary representation.

1.2 Our results

In this note we show in Theorem 3.6 that ModL is closed under $\leq^{\mathrm{L}}_{m}$ reduction and $\vee$(join). Also using [Vij09, Theorem 3.6 and Corollary 3.7] we show in Theorem 3.6(2) that ModL is closed under $\leq^{\mathrm{L}}_{1-\mathrm{tt}}$ assuming NL = UL. From these closure properties of ModL we also show in Corollary 3.9 that ModL is closed under $\leq^{\mathrm{UL}}_{m}$ reduction and in Corollary 3.10 that if we assume NL = UL then ModL is closed under $\leq^{\mathrm{UL}}_{1-\mathrm{tt}}$ reduction. The known relations between ModL and other logspace counting classes is shown in Figure 1 at the end of Section 3.1. The closure properties of ModL that are known is listed in Table 1 at the end of Section 3.2.

We also show that $\mathrm{ModL}^{\mathrm{UL}} = \mathrm{ModL}$ assuming UL = coUL in Corollary 3.12. As a consequence of these results we show in Corollary 3.13 that $\mathrm{UL}^{\mathrm{ModL}}_{1-\mathrm{tt}} = \mathrm{ModL}^{\mathrm{UL}} = \mathrm{ModL}$ assuming NL = UL.

While we know that $\mathrm{Mod}_{p^e}\mathrm{L} \subseteq \mathrm{ModL}$ if $p^e \in \mathbb{Z}^+$ is a prime power, it is not known whether $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$ when $k \in \mathbb{Z}^+$ and $k \geq 6$ is a composite number that has more than one distinct prime divisor. Similarly it is unknown if $\mathrm{coC}_{=}\mathrm{L} \subseteq \mathrm{ModL}$. We show that these inclusions would follow if ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ and $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reductions respectively where $l \in \mathbb{Z}^+$, $l \geq 2$ is such that the number of distinct prime divisors of k is at most l. Some other implications of the closure of ModL under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ and $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reductions are also discussed in Section 4.

It is known that the determinant of a matrix with entries in $\mathbb{Z}$ is computable in L-uniform TC^1. We show a proof of this result in Theorem 3.2 and using results on integer multiplication and integer division from [Vol99, HAB02, CDL01] we also show in Corollary 3.3 that ModL $\subseteq$ L-uniform TC^1.

2 Preliminaries

We start by recalling the definitions of logspace counting classes.

Definition 2.1. [AJ93] *Let Σ be the input alphabet. The complexity class* #L *is defined to be the class of functions $f : \Sigma^* \to \mathbb{Z}^+$ such that there exists a* NL *Turing machine M for which we have $f(x) = acc_M(x)$ where $acc_M(x)$ denotes the number of accepting computation paths of M on any input $x \in \Sigma^*$.*

The complexity class GapL was defined in [AO96] to precisely classify the seminal result of [Tod91, Vin91, Dam92, Val92] that computing the determinant of an integer matrix is complete for GapL under logspace many-one reductions.

Definition 2.2. [AO96] *Let Σ be the input alphabet. The complexity class* GapL *is defined to be the class of functions $f : \Sigma^* \to \mathbb{Z}$ such that there exists a* NL *Turing machine M for which we have $f(x) = acc_M(x) - rej_M(x)$ where $acc_M(x)$ and $rej_M(x)$ denote the number of accepting and the number of rejecting computation paths of M on any input $x \in \Sigma^*$ respectively. We also denote $(acc_M(x) - rej_M(x))$ by $gap_M(x)$.*

Definition 2.3. [BD$^+$92] *Let Σ be the input alphabet. Also let $k \in \mathbb{Z}^+$ and let $k \geq 2$. We say that $L \subseteq \Sigma^*$ is a language in the complexity class* Mod_kL *if there exists a function $f \in$ #L such that for any input $x \in \Sigma^*$ we have $x \in L$ if and only if $f(x) \not\equiv 0 (mod\ k)$.*

Definition 2.4. [AO96] *Let Σ be the input alphabet. We say that $L \subseteq \Sigma^*$ is a language in the complexity class* C$_=$L *if there exists a function $f \in$ GapL such that for any input $x \in \Sigma^*$ we have $x \in L$ if and only if $f(x) = 0$.*

Definition 2.5. [BJ$^+$91] *Let Σ be the input alphabet. We say that $L \subseteq \Sigma^*$ is a language in the complexity class* UL *if there exists a function $f \in$ #L such that for any input $x \in \Sigma^*$ we have $f(x) = 1$ if $x \in L$ and $f(x) = 0$ if $x \notin L$.*

Definition 2.6. [Vol99] *The complexity class* TC^0 *is the class of all sets $A \subseteq \{0,1\}^*$ such that there exists Boolean circuits of size $p(n)$ and depth $d \in \mathbb{N}$ containing $\neg$ gates, unbounded fan-in $\vee$, $\wedge$ and* MAJ *gates where n is the size of the input and $p(n)$ is a polynomial in n.*

Definition 2.7. [Vol99] *The complexity class* NC^1 *is the class of all sets $A \subseteq \{0,1\}^*$ such that there exists Boolean circuits of size $p(n)$ and depth $O(log\ n)$ containing $\neg$ gates and $\vee$, $\wedge$ gates that have fan-in 2 where n is the size of the input and $p(n)$ is a polynomial in n.*

Definition 2.8. [Vol99] *The complexity class* TC^1 *is the class of all sets $A \subseteq \{0,1\}^*$ such that there exists Boolean circuits of size $p(n)$ and depth $O(log\ n)$ containing $\neg$ gates, unbounded fan-in $\vee$, $\wedge$ and* MAJ *gates where n is the size of the input and $p(n)$ is a polynomial in n.*

For the results in this note we also need the notion of uniformity of circuits. We refer to [Vol99, HAB02] for a clear and detailed exposition of uniformity of circuits. The following inclusions are known among these complexity classes (we assume the circuit complexity classes are L-uniform): TC$^0 \subseteq$ NC$^1 \subseteq$ L $\subseteq$ UL $\subseteq$ Mod$_p$L $\subseteq$ Mod$_k$L where $p, k \in \mathbb{Z}^+$ and p is a prime that divides k. Also UL $\subseteq$ coC$_=$L.

Oracle access mechanism

In the results to be shown we assume that any NL oracle Turing machine submits the queries to its oracle according to the Ruzzo-Simon-Tompa oracle access mechanism described in [AO96, ABO99]. According to this mechanism any NL Turing machine can write its queries on the oracle tape in a deterministic manner only. As a result it follows that any NL Turing machine can submit at most polynomially many queries to its oracle and the queries can be submitted to the oracle in a deterministic manner having known only the size of the input before it starts performing any computation on the given input.

3 ModL

The complexity class ModL is defined in [AV04] (more precisely in [Vij08, Definition 1.4.1] and [AV, Definition 3.1]).

Definition 3.1. *Let Σ be the input alphabet. A language $L \subseteq \Sigma^*$ is in the complexity class* ModL *if there is a function $f \in$ GapL and a function $g \in$ FL such that on any input $x \in \Sigma^*$,*

- *$g(x) = 1^{p^e}$ for some prime p and a positive integer e, and*
- *$x \in L \Leftrightarrow f(x) \not\equiv 0 (mod\ p^e)$.*

In the definition of ModL the FL function g is assumed to output a prime in the unary representation for every input $x \in \Sigma^*$. Also the output of g can vary with the input x. It is easy to note that ModL generalizes the complexity class Mod_kL when $k = p^e \in \mathbb{Z}^+$ where p is a prime. Clearly $\text{Mod}_{p^e}\text{L} \subseteq \text{ModL}$.

Due to the seminal result of [Tod91, Vin91, Dam92, Val92] that computing the determinant of an integer matrix is complete for GapL under logspace many-one reductions we obtain the following canonical complete problem for ModL under logspace many-one reductions: $\text{ModDet} = \{\langle M, 1^{p^e}\rangle | \det(M) \not\equiv 0 (\text{mod } p^e), \text{where } M \in \mathbb{Z}^{n\times n} \text{ and } p, e \in \mathbb{Z}^+, p \text{ is a prime and } e \geq 1\}$.

The question of whether $\text{Mod}_k\text{L} \subseteq \text{ModL}$ if $k \in \mathbb{Z}^+$ and $k \geq 6$ is a composite number that has at least two distinct prime divisors is open (in fact it is also not known if $\text{Mod}_k\text{P} \subseteq \text{ModP}$ [KT96]).

3.1 ModL $\subseteq$ L-uniform TC^1

The results of Berkowitz in [Ber84] show that we can compute the determinant of a matrix $A \in \mathbb{Z}^{n\times n}$ in NC^2. Also subsequently the results of [Tod91] show that the problem of computing the $\det(A)$ is logspace many-one reducible to the problem of computing the $(i,j)^{th}$ entry of the k^{th} power of a matrix $B \in \mathbb{Z}^{m\times m}$ where $1 \leq i, j \leq m$ and k, m are bounded by a polynomial in n. A proof of this reduction given in [ABO99, HT03] considers the input matrix A to be the adjacency matrix of a weighted directed graph G where A_{ij} denotes the weight of the directed edge (i, j) in G. Also it is easy to note that this many-one

reduction is in fact computable by a L-uniform AC^0 circuit. We use the above mentioned reduction and the result that it is possible to compute the iterated sum and the iterated product of a set of integers given as input in L-uniform TC^0 [Vol99] (which also follows from the results shown in [HAB02, CDL01]) to show an upper bound of L-uniform TC^1 for computing the determinant of an integer matrix. Using this upper bound for computing the determinant and using the result that we can compute the quotient and remainder upon dividing a by b in L-uniform TC^0 [HAB02, CDL01], where $a, b \in \mathbb{Z}^+$ and $a, b \geq 1$, it follows that ModL $\subseteq$ L-uniform TC^1.

Theorem 3.2. *Given a matrix $A \in \mathbb{Z}^{n \times n}$ as input we can compute the* $\det(A)$ *in* L-*uniform* TC^1.

Proof. We use the L-uniform AC^0 many-one reduction from the determinant of an integer matrix to computing the entries of powers of an integer matrix [Tod91] to obtain a $m \times m$ matrix $B = (b_{ij})$, $k \in \mathbb{Z}^+$ and the pair (i, j) as the output of this reduction from the input matrix $A \in \mathbb{Z}^{n \times n}$ such that $B^k_{ij} = \det(A)$ where $k \geq 1$ and $1 \leq i, j \leq m$ and $m = p(n)$ for some polynomial $p(n)$.

The following results are computable in L-uniform TC^0 [Vol99, HAB02]:

- computing the sum of a set of n input integers $a_1, \ldots, a_n \in \mathbb{Z}$, and
- computing the product of $a, b \in \mathbb{Z}$ given as input.

As a consequence of these results it is easy to note that if $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^n$ then the inner product of $\mathbf{u}$ and $\mathbf{v}$ is computable in L-uniform TC^0.

From these results it is clear that we can compute the $(i, j)^{th}$ entry in the product of an input pair of square integer matrices in L-uniform TC^0. As a consequence if we are given a set of n square integer matrices as input where every entry in each of these matrices is of size n then we can compute the entries in product of these matrices by combining these TC^0 circuits in a pairwise manner to obtain a Boolean circuit that contains $\neg$ gates, unbounded fan-in $\vee$,$\wedge$ and MAJ gates and whose size is a polynomial in n and depth $O(\log n)$. Clearly this shows that the problem of computing the $(i, j)^{th}$ entry of the powers of an input matrix that has entries in $\mathbb{Z}$ is in L-uniform TC^1. Combining the L-uniform AC^0 circuit that many-one reduces the problem of computing the determinant of an integer matrix to computing the entries of powers of an integer matrix with the L-uniform TC^1 circuit we have described above and since L-uniform $AC^0 \subseteq$ L-uniform TC^1 it follows that we can compute the $\det(A)$ in L-uniform TC^1. •

Corollary 3.3. ModL $\subseteq$ L-*uniform* TC^1.

Proof. Let Σ be the input alphabet and let $L \subseteq \Sigma^*$ be such that $L \in$ ModL. Also let $f \in$ GapL and $g \in$ FL be functions using which we decide if an input string $x \in \Sigma^*$ is in L. Let $x \in \Sigma^*$ be the input string and let $g(x) = 1^{p^e}$ where $p, e \in \mathbb{Z}^+$ with $p \geq 2$ being a prime and $e \geq 1$.

Since computing the determinant of an integer matrix is complete for GapL under logspace many-one reductions [Tod91, Vin91, Dam92, Val92] it follows

that we can obtain the matrix $A \in \mathbb{Z}^{n \times n}$ from the input x using a FL function such that $\det(A) = f(x)$. Clearly we can obtain the prime power $p^e = |g(x)|$ by computing the size of the output of the FL function g when it is given the input x. Then $x \in L$ if and only if $f(x) \not\equiv 0 (\mathrm{mod}\ p^e)$.

We have shown in Theorem 3.2 that it is possible to compute $\det(A)$ in L-uniform TC^1. Also since FL $\subseteq$ L-uniform TC^1 it follows that it is possible to compute $p^e = |g(x)|$ and also test if $\det(A) \not\equiv 0 (\mathrm{mod}\ p^e)$ in L-uniform TC^1 since we can compute the quotient and remainder obtained upon dividing a by b in L-uniform TC^0 [HAB02], where $a, b \in \mathbb{Z}^+$ and $b \geq 1$. This shows that we can decide if $x \in L$ in L-uniform TC^1 and therefore ModL $\subseteq$ L-uniform TC^1. •

Since we can define every logspace counting class based on computing the determinant of a matrix with entries in $\mathbb{Z}$, it is a standard consequence of Theorem 3.2 that all the logspace counting classes (and in fact the logspace counting hierarchy #LH [AO96]) are contained in L-uniform TC^1. The relation between ModL and other logspace counting classes is shown in Figure 1 below.

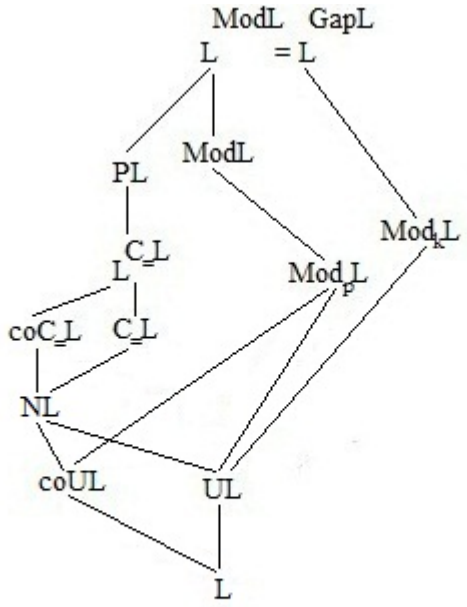

Figure 1: Known relations among logspace counting classes. Here p≥2 is is a prime and k≥6 is a composite number that has more than one distinct prime divisor.

3.2 Closure under $\leq_m^{\mathrm{L}}$, $\vee$ (join), $\leq_m^{\mathrm{UL}}$, $\leq_{1-\mathrm{tt}}^{\mathrm{L}}$ and $\leq_{1-\mathrm{tt}}^{\mathrm{UL}}$

Definition 3.4. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. We say that L_1 is logspace 1-truth-table reducible to L_2, denoted by $L_1 \leq_{1-\mathrm{tt}}^{\mathrm{L}} L_2$ if there exists a $O(\log n)$ space bounded Turing machine M that has access to L_2 as an oracle and which decides if any given input $x \in \Sigma^*$ is in L_1 by making exactly*

one query to the L_2 oracle where $n = |x|$.

Definition 3.5. *Let Σ be the input alphabet such that $\{0,1\} \subseteq \Sigma$ and let $L_1, L_2 \subseteq \Sigma^*$. We define the join of L_1 and L_2 to be $L_1 \vee L_2 = \{x1|x \in L_1\} \cup \{y0|y \in L_2\}$.*

Theorem 3.6. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. Also let $L_2 \in$ ModL. Now*

1. *if $L_1 \leq_m^{\mathrm{L}} L_2$ then $L_1 \in$ ModL,*
2. *assume that* NL = UL. *If $L_1 \leq_{1-\mathrm{tt}}^{\mathrm{L}} L_2$ then $L_1 \in$ ModL, and*
3. *if $\{0,1\} \subseteq \Sigma$ and $L_1 \in$ ModL then $L_1 \vee L_2 \in$ ModL.*

Proof.

1. Let $L_2 \in$ ModL and let $f_2 \in$ GapL and $g_2 \in$ FL be functions using which we decide if an input $x \in \Sigma^*$ is in L_2. Also let $L_1 \leq_m^{\mathrm{L}} L_2$ using $f \in$ FL. If $x \in \Sigma^*$ is the input then $x \in L_1$ if and only if $f(x) \in L_2$. But $L_2 \in$ ModL and therefore $f(x) \in L_2$ if and only if $f_2(f(x)) \not\equiv 0 (\mathrm{mod}\ p^e)$ where $p^e = |g_2(f(x))| \in \mathbb{Z}^+$ where $p, e \in \mathbb{Z}^+$ with p being a prime and $e \geq 1$. Using results from [AO96] it is easy to note that $(f_2 \circ f) \in$ GapL and $(g_2 \circ f) \in$ FL. Clearly $g_2(f(x))$ is a prime power in the unary representation for any $x \in \Sigma^*$. As a result if $x \in \Sigma^*$ then $x \in L_1$ if and only if $f_2(f(x)) \not\equiv 0 (\mathrm{mod}\ p^e)$ where $p^e = |g_2(f(x))|$ which shows $L_1 \in$ ModL.

2. Let $L_1 \leq_{1-\mathrm{tt}}^{\mathrm{L}} L_2$ using $f \in$ FL that makes exactly one query to the L_2 oracle. In other words f correctly decides if an input $x \in \Sigma^*$ is in L_1 by making exactly one query to the L_2 oracle. Also let $q_x \in \Sigma^*$ be the query string that is generated by f for the input $x \in \Sigma^*$. Clearly f decides if $x \in L_1$ based on x and the reply of the L_2 oracle when it is given the query q_x as input.

   Therefore let $f' \in$ FL be the function that generates and outputs the query q_x generated by f when it is given the input x. Also let $g' \in$ FL be such that $g'(x, \chi_{L_2}(q_x)) = f(x)$ where χ_{L_2} is the characteristic function of L_2. It is clear that g' correctly decides whether any input $x \in L_1$ if the reply of the L_2 oracle is also given when we submit the query q_x to the L_2 oracle.

   Since $L_2 \in$ ModL and we have assumed NL = UL, it follows from [Vij09, Theorem 3.6] that there exists $f_2 \in \#\mathrm{L}$ and $g_2 \in$ FL such that on any input $x \in \Sigma^*$ we have $x \in L_2$ if and only if $f_2(x) \not\equiv 0 (\mathrm{mod}\ p)$ where $g_2(x) = 1^p$, $p \in \mathbb{Z}^+$ and p is a prime. We know that $\#\mathrm{L} \subset$ GapL. As a result we assume without loss of generality that on any input $x \in \Sigma^*$ the NL machine corresponding to f_2 has at least one accepting computation path and at least one rejecting computation path.

   Let us consider $(g' \circ f_2 \circ f')$ and $(g_2 \circ f')$. Since $f', g' \in$ FL and since FL $\subseteq$ GapL it follows that $(g' \circ f_2 \circ f') \in$ GapL. It is also clear that

$(g_2 \circ f') \in$ FL. Let M denote the NL Turing machine corresponding to $(g' \circ f_2 \circ f')$. Clearly the output of M along any computation path depends on the input x and $\chi_{L_2}(q_x)$. We however note that the output of M is the same along all the accepting computation paths of the NL Turing machine corresponding to $(f_2 \circ f')$. In other words, M accepts along all the accepting computation paths of the NL Turing machine corresponding to $(f_2 \circ f')$ or M rejects along all the accepting computation paths of the NL Turing machine corresponding to $(f_2 \circ f')$. It is clear that the same observation is true for M along all the rejecting computation paths of the NL Turing machine corresponding to $(f_2 \circ f')$. As a result it follows that $|gap_M(x)| = |gap_{(f_2 \circ f')}(q_x)|$ on any input $x \in \Sigma^*$. Also note that depending on $\chi_{L_2}(q_x)$ the output of M along any computation path is either a logspace many-one reduction to L_2 or it complements $\chi_{L_2}(q_x)$ as its output along any of its computation path.

Let $x \in \Sigma^*$ be the input and let $g_2(f'(x)) = 1^p$. We now define a NL Turing machine M' that makes $(p-1)$ simulations of M on the input x. In addition M' also keeps track of whether the computation path chosen is the lexicographically least in its computation tree on any input $x \in \Sigma^*$. Now if the output of M along any of its computation paths is the same as the output of the computation path of the NL Turing machine corresponding to $(f_2 \circ f')$ on the input x then M' stops with the output of M on input x after making $(p-1)$ simulations of M on input x. On the other hand if we have the output of M along any of its computation paths is the complement of the output of the computation path of the NL Turing machine corresponding to $(f_2 \circ f')$ on a given input $x \in \Sigma^*$ then M' makes $(p-1)$ simulations of M on input x. In these $(p-1)$ simulations of M by M', along any of the computation paths of M' that is not the lexicographically least computation path we assume that M' stops with the output of M on input x. However along the lexicographically least computation path that we obtain in these $(p-1)$ simulations of M by M' we assume that M' makes sufficiently many non-deterministic choices at the end of this computation path so that we obtain $gap_{M'}(x) = (gap_M(x))^{p-1} + (p-1)$.

We know that $(g_2 \circ f') \in$ FL and $g_2(f'(x)) = 1^p$ where $p \in \mathbb{Z}^+$ and p is a prime. It is easy to note that M' can determine if the output of M along any of its computation path is the same or it is the complement of the output of the NL Turing machine corresponding to $(f_2 \circ f')$ on any input $x \in \Sigma^*$ using constant space. Also M' can simultaneously keep track of whether the computation path is the lexicographically least in its computation tree on input x using constant space. As a result since $p \in O(\log n)$ where $n = |x|$ it follows that M' can make sufficiently many non-deterministic choices along the lexicographically least computation path using space at most $O(\log n)$ to obtain $gap_{M'}(x) = (gap_M(x))^{p-1} + (p-1)$. It is therefore clear that M' is also a NL Turing machine. Clearly if for a given input $x \in \Sigma^*$ if $L_1 \leq^{\mathrm{L}}_{1-\mathrm{tt}} L_2$ is similar to a many-one reduction

from L_1 to L_2 then $x \in L_1$ if and only if $q_x \in L_2$. In this case using our observations and [Vij09, Theorem 3.6] it follows that $x \in L_1$ if and only if $gap_{M'}(x) \not\equiv 0 (\text{mod } p)$. Otherwise if the $\leq^{\text{L}}_{1-\text{tt}}$ is such that the output of M complements the output of the NL Turing machine corresponding to $(f_2 \circ f')$ then from the definition of M' it follows from [Vij09, Theorem 3.6 and Corollary 3.7] that if $x \in L_1$ then $gap_{M'}(x) \equiv (p-1) (\text{mod } p)$ and if $x \notin L_1$ then $gap_{M'}(x) \equiv 0 (\text{mod } p)$. This shows $L_1 \in$ ModL.

3. Let $L_i \in$ ModL and assume that we decide if an input string $x \in \Sigma^*$ is in L_i using functions $f_i \in$ GapL and $g_i \in$ FL where $1 \leq i \leq 2$ respectively. We assume without loss of generality that the size of any input string x is at least 1.

   Let $x = x_1 x_2 \cdots x_{n+1}$ and let $y = x_1 \cdots x_n$ where $n \in \mathbb{Z}^+$. Also let

   $$f(x) = \begin{cases} f_1(y) & \text{if } x_{n+1} = 1 \\ f_2(y) & \text{otherwise if } x_{n+1} = 0, \end{cases}$$

   and

   $$g(x) = \begin{cases} g_1(y) & \text{if } x_{n+1} = 1 \\ g_2(y) & \text{otherwise if } x_{n+1} = 0. \end{cases}$$

   It is easy to note that $f \in$ GapL and $g \in$ FL. Since $|g_1(x)|$ or $|g_2(x)|$ is always a prime power on any input string $x \in \Sigma^*$ it follows that $|g(x)|$ is also a prime power. Now $f(x) \not\equiv 0 (\text{mod } |g(x)|)$ for any input $x = yx_{n+1}$ if and only if exactly one of the following is true:

   - $(x_{n+1} = 1$ and $y \in L_1)$,
   - $(x_{n+1} = 0$ and $y \in L_2)$.

   Clearly this shows that $L_1 \vee L_2 \in$ ModL and that we can decide if any input string $x \in \Sigma^*$ is in $L_1 \vee L_2$ using $f \in$ GapL and $g \in$ FL which completes the proof.

•

Definition 3.7. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. We say that $L_1 \leq^{\text{UL}}_m L_2$ if there exists a* FNL *Turing machine M such that on any input $x \in \Sigma^*$ the number of accepting computation paths of $M(x)$ is at most* 1. *Also $x \in L_1$ if and only if there exists an accepting computation path of M on input x such that if we obtain $y \in \Sigma^*$ as the output along the accepting computation path then $y \in L_2$. If $x \notin L_1$ then either M does not have any accepting computation path on input x or if there exists an accepting computation path of M on input x and $y \in \Sigma^*$ is the output along this computation path then $y \notin L_2$.*

Definition 3.8. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. We say that $L_1 \leq^{\text{UL}}_{1-\text{tt}} L_2$ if there exists a* NL *Turing machine M such that on any input $x \in \Sigma^*$ the number of accepting computation paths of $M(x)$ is at most* 1. *Also M submits exactly one query $q_x \in \Sigma^*$ to the L_2 oracle on any input $x \in \Sigma^*$*

to decide if $x \in L_1$. We assume that the query is submitted by the NL *Turing machine M in a deterministic manner to the L_2 oracle according to the Ruzzo-Simon-Tompa oracle access mechanism as described in* [AO96, ABO99]*. Here if $x \in L_1$ then there exists exactly one accepting computation path of M on input x. Otherwise if $x \notin L_1$ then M rejects the input x on all of its computation paths.*

Corollary 3.9. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. Also let $L_2 \in$ ModL. Now if $L_1 \leq_m^{\mathrm{UL}} L_2$ then $L_1 \in$ ModL.*

Proof. Proof is similar to Theorem 3.6(1). We however use [Vij09, Lemma 3.5] for L_2 and assume that there exists $f \in \#\mathrm{L}$ and $g \in \mathrm{FL}$ such that on any input $x \in \Sigma^*$ we have $g(x) = 1^{p^e}$ where $p, e \in \mathbb{Z}^+$, p is a prime, $e \geq 1$ and $x \in L_2$ if and only if $f(x) \not\equiv 0 (\mathrm{mod}\ p^e)$.

Also we note that the the $\leq_m^{\mathrm{UL}}$ reduction has at most one accepting computation path. Therefore if the computation path we obtain is an accepting computation path and $y \in \Sigma^*$ is output along this accepting computation path, then simulating the NL Turing machine for $L_2 \in$ ModL with y as the input shows that the congruence relation based on the #L function f and the FL function g that is used to decide if any input $x \in \Sigma^*$ is in L_2 can also be used for deciding if $x \in L_1$. Otherwise if the computation path of the $\leq_m^{\mathrm{UL}}$ reduction that we obtain is a rejecting computation path then our NL Turing machine makes sufficiently many non-deterministic choices so that the number of accepting computation paths that we obtain at the end of each of these rejecting computation paths is divisible by $|g(x)|$. This shows $L_1 \in$ ModL. •

Corollary 3.10. *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. Also assume that* NL = UL *and let $L_2 \in$ ModL. Now if $L_1 \leq_{1-\mathrm{tt}}^{\mathrm{UL}} L_2$ then $L_1 \in$ ModL.*

Proof. Proof is similar to Theorem 3.6(2). We however need to use [Vij09, Lemma 3.5] and consider the accepting computation paths of the NL Turing machine corresponding to L_2 as in the proof of Corollary 3.9. We also use [Vij09, Theorem 3.6] as in Theorem 3.6(2) and thereby assume that the FL function that is used to decide if any input string is in L_2 outputs a prime number in the unary representation.

Based on whether the $\leq_{1-\mathrm{tt}}^{\mathrm{UL}}$ reduction is a $\leq_m^{\mathrm{UL}}$ reduction or if the output of the computation paths of the UL Turing machine is the complement of the output of the computation paths of the NL Turing machine for the language L_2 on the oracle query string, we define the NL Turing machine for L_1 similar to the proof of Theorem 3.6(2). To be precise if the output of computation paths of the NL Turing machine for L_2 on the input query string is the complement of the output of the computation paths of the UL Turing machine on the given input then we use [Vij09, Theorem 3.6 and Corollary 3.7] and ensure that our NL Turing machine for L_1 simulates the NL Turing machine for L_2 sufficiently many times and that it also makes sufficiently many non-deterministic choices at the end of the lexicographically least computation path in its computation tree (we need to note that non-deterministic choices that the NL Turing machine for L_1

we define does not depend on whether the lexicographically least computation path ends in an accepting state or in a rejecting state. We only ensure that sufficiently many accepting computation paths are added to the computation tree at the end of this computation path for this NL Turing machine). The proof that the Turing machine we define is $O(\log n)$ space bounded and that the computation tree is defined suitably to decide if any input $x \in \Sigma^*$ is in L_1 based on the congruence relation is similar to the proof of Theorem 3.6(2). •

The following table lists the closure properties of ModL that are known.

Closure property	*True/False*	*Reference*
complement	true (assuming NL = UL)	[Vij09, Corollary 3.7]
$\leq_m^{\mathrm{L}}$	true	Theorem 3.6(1)
$\leq_{1-\mathrm{tt}}^{\mathrm{L}}$	true (assuming NL = UL)	Theorem 3.6(2)
$\vee$	true	Theorem 3.6(3)
$\leq_m^{\mathrm{UL}}$	true	Corollary 3.9
$\leq_{1-\mathrm{tt}}^{\mathrm{UL}}$	true (asuming NL = UL)	Corollary 3.10

Table1: Closure properties of ModL

3.3 ModL and UL

Theorem 3.11. *Assume that* UL = coUL. *Then* $\mathrm{ModL}^{\mathrm{UL}} \subseteq \mathrm{ModL}$.

Proof. Assume that UL = coUL. Let Σ be the input alphabet such that $L_1, L_2 \subseteq \Sigma^*$. Also let $L_2 \in \mathrm{UL}$ and $L_1 \in \mathrm{ModL}^{L_2}$. As a result using [Vij09, Lemma 3.5] for L_1 it follows that on any input $x \in \Sigma^*$ there exists a NL machine M that accesses L_2 as an oracle such that if $f \in \#\mathrm{L}$ and $f(x)$ denotes the number of accepting computation paths of M on input x and $g \in \mathrm{FL}$ then we have $x \in L_1$ if and only if $f(x) \not\equiv 0 (\mathrm{mod}\ |g(x)|)$ where $g(x) = 1^{p^e}$ with $p, e \in \mathbb{Z}^+$, p being a prime and $e \geq 1$.

Here we assume that the queries that are submitted to the L_2 oracle by M are generated in a deterministic manner according to the Ruzzo-Simon-Tompa oracle access mechanism as stated in [AO96, ABO99]. As a consequence we assume that M submits the queries to the L_2 oracle and also obtains the reply of the oracle on the oracle tape before it starts any other computation on the input. In addition since M is $O(\log n)$ space bounded for any given input $x \in \Sigma^*$ such that $n = |x|$ it is clear that the size of each query is at most n^j and the number of queries that M can submit to L_2 is at most n^k where $j, k \in \mathbb{Z}^+$. Let us denote the i^{th} query that is generated by M as $q_x^{(i)} \in \Sigma^*$ where $1 \leq i \leq n^k$.

The NL Turing machine M' that we define for L_1 simulates M on the input x. However unlike M the NL Turing machine M' does not have access to any oracle. However M' needs to obtain the reply of the L_2 oracle for the queries that M submits in its computation path. For this purpose we follow the proof of the result that $\mathrm{L}^{\oplus\mathrm{L}} = \oplus\mathrm{L}$ in [HRV00] which also shows $\oplus\mathrm{L}^{\oplus\mathrm{L}} = \oplus\mathrm{L}$ in [HRV00] and assume that M' generates the i^{th} query $q_x^{(i)}$ in a deterministic manner during its simulation of M on input x where $1 \leq i \leq n^k$.

For every query $q_x^{(i)}$ that is generated by M', to determine if $q_x^{(i)} \in L_2$, the NL Turing machine M' simulates the UL Turing machine M_{L_2} corresponding to L_2. Clearly if $q_x^{(i)} \in L_2$ then M_{L_2} has exactly one accepting computation path. If the simulation of M_{L_2} by M' results in this accepting computation path then M' assumes the output of the oracle query that M would have obtained when it submits the query $q_x^{(i)}$ as input to the L_2 oracle to be "yes" and it continues its simulation of M on input x. Otherwise let us assume that the simulation of M_{L_2} by M' results in a rejecting computation path. In this case to test if $q_x^{(i)} \in \overline{L_2}$ we use the assumption that UL = coUL and hence allow M' to simulate the UL machine $M_{\overline{L_2}}$ corresponding to $\overline{L_2}$ on input $q_x^{(i)}$ to test if this simulation results in an accepting computation path. If M' ends in an accepting computation path in this simulation then it implies $q_x^{(i)} \in \overline{L_2}$. Therefore M' assumes that the reply of the L_2 oracle is "no" on the query $q_x^{(i)}$ and continues its simulation of M on input x. Otherwise if both these simulations of M' on input $q_x^{(i)}$ results only in rejecting computation paths then M' makes sufficiently many non-deterministic choices in the computation tree and ends in a rejecting state in all these computation paths.

For any query string $q_x^{(i)}$ that is generated by M' along any of its computation paths we either have $q_x^{(i)} \in L_2$ or we have $q_x^{(i)} \in \overline{L_2}$ where $1 \leq i \leq n^k$. Now it follows from the definition of M' that we can determine if $q_x^{(i)} \in L_2$ by simulating M_{L_2} or if necessary $M_{\overline{L_2}}$ on $q_x^{(i)}$ during its computation on the input x where $1 \leq i \leq n^k$. Due to our assumption that UL = coUL it also follows that the number of computation paths in these simulations that correctly determine if $q_x^{(i)} \in L_2$ is exactly one where $1 \leq i \leq n^k$. Apart from simulating M_{L_2} and $M_{\overline{L_2}}$ since the computation of M' is identical to M on input x it follows that $acc_{M'}(x) = f(x)$. As a result we have $x \in L_1$ if and only if $acc_{M'}(x) \not\equiv 0 (\text{mod } |g(x)|)$. This shows $L_1 \in \text{ModL}$. •

Corollary 3.12. *Assume that* UL = coUL. *Then* $\text{ModL}^{\text{UL}} = \text{ModL}$.

Proof. The containment $\text{ModL} \subseteq \text{ModL}^{\text{UL}}$ follows immediately from the definition of ModL. Conversely if we assume UL = coUL then we have shown in Theorem 3.11 that $\text{ModL}^{\text{UL}} \subseteq \text{ModL}$ which completes the proof. •

Corollary 3.13. *Assume that* NL = UL. *Then* $\text{UL}_{1-\text{tt}}^{\text{ModL}} = \text{ModL}^{\text{UL}} = \text{ModL}$.

Proof. Since we know from the Immerman-Szelepscényi Theorem [Pap94, Theorem 7.6] that NL is closed under complement, our assumption that NL = UL implies UL = coUL. Proof of our result now follows from Corollary 3.10 and Corollary 3.12. •

4 Closure under $\leq_{l-\text{dtt}}^{\text{L}}$ and $\leq_{\text{dtt}}^{\text{L}}$ reductions

Definition 4.1. [ABO99] *Let Σ be the input alphabet and let $L_1, L_2 \subseteq \Sigma^*$. We say that L_1 is logspace k-disjunctive truth-table reducible to L_2 for some $k \in \mathbb{Z}^+$,*

$k \geq 1$, *denoted by* $L_1 \leq^{\mathrm{L}}_{k-\mathrm{dtt}} L_2$, *if there exists* $f \in \mathrm{FL}$ *such that given an input* $x \in \Sigma^*$ *of length* n *we have* $f(x) = \{y_1, \ldots, y_k\}$ *and* $x \in L_1$ *if and only if* $y_i \in L_2$ *for at least one* $1 \leq i \leq k$.

Definition 4.2. [ABO99] *Let* Σ *be the input alphabet and let* $L_1, L_2 \subseteq \Sigma^*$. *We say that* L_1 *is logspace disjunctive truth-table reducible to* L_2, *denoted by* $L_1 \leq^{\mathrm{L}}_{\mathrm{dtt}} L_2$, *if there exists* $f \in \mathrm{FL}$ *such that given an input* $x \in \Sigma^*$ *of length* n *we have* $f(x) = \{y_1, \ldots, y_{p(n)}\}$ *and* $x \in L_1$ *if and only if* $y_i \in L_2$ *for at least one* $1 \leq i \leq p(n)$ *where* $p(n)$ *is a polynomial in* n.

It follows from the definition of ModL that if $p \in \mathbb{Z}^+$ is a prime then $\mathrm{Mod}_p\mathrm{L} \subseteq \mathrm{ModL}$. However if $k \in \mathbb{Z}^+$ such that $k \geq 6$ is a composite number that has more than one distinct prime divisor then it is not known if $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$. We show that if ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ reductions for some $l \in \mathbb{Z}^+$ such that $l \geq 2$ then $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$ for all $k \in \mathbb{Z}^+$ such that $k \geq 6$ is a composite number that has at most l distinct prime divisors.

Theorem 4.3. *If* ModL *is closed under* $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ *reductions where* $l \in \mathbb{Z}^+$ *and* $l \geq 2$ *then* $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$ *for all* $k \in \mathbb{Z}^+$ *such that* $k \geq 6$ *is a composite number that has at least* 2 *and at most* l *distinct prime divisors.*

Proof. Let us assume that ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ reductions for some $l \in \mathbb{Z}^+$ such that $l \geq 2$. Let Σ be the input alphabet and let $\sharp \notin \Sigma$. Also let $L \in \Sigma^*$ and assume that $L \in \mathrm{Mod}_k\mathrm{L}$ where $k = p_1^{e_1} \cdots p_m^{e_m}$ is a composite number such that p_i is a prime, $e_i \in \mathbb{Z}^+$ with $e_i \geq 1$ and $p_i \neq p_j$ for all $1 \leq i < j \leq m \leq l$.

It is shown in [BD$^+$92] that if $A \in \mathbb{Z}^{n \times n}$ then determining if $\det(A) \not\equiv 0 (\mathrm{mod}\ k)$ is complete for $\mathrm{Mod}_k\mathrm{L}$ under logspace many-one reductions. Let us denote this language that is complete for $\mathrm{Mod}_k\mathrm{L}$ by $\mathrm{Mod}_k\mathrm{Det}$. Also [BD$^+$92] have shown that $\det(A) \not\equiv 0 (\mathrm{mod}\ k)$ if and only if $\det(A) \not\equiv 0 (\mathrm{mod}\ p_i)$ for at least one of the primes $p_i | k$, where $1 \leq i \leq m$. Based on these observations let us define $L_k = \{\langle A \sharp 1^{p_i} \rangle | \det(A) \not\equiv 0 (\mathrm{mod}\ |g(\langle A \sharp 1^{p_i} \rangle)|)\ \forall\ p_i | k$ where $g \in \mathrm{FL}$ and $g(\langle A \sharp 1^{p_i} \rangle) = 1^{p_i}\}$. It follows from Definition 3.1 that $L_k \in \mathrm{ModL}$.

Therefore if $A \in \mathbb{Z}^{n \times n}$ is the input then $A \in L$ if and only if $\langle A \sharp 1^{p_i} \rangle \in L_k$ for at least one prime p_i where $1 \leq i \leq m$. But it is easy to note that given A as input, a $O(\log |x|)$ space bounded Turing machine M can obtain the prime factorization of k and also output $\langle A \sharp 1^{p_i} \rangle$ for all primes $p_i | k$ where $1 \leq i \leq m$. If $m < l$ then we assume that M outputs the last query $\langle A \sharp 1^{p_j} \rangle$ that it generates with repetition sufficiently many times to output l strings where $1 \leq j \leq m \leq l$. Now $A \in L$ if and only if $\langle A \sharp 1^{p_j} \rangle \in L_k$ for at least one $1 \leq j \leq l$. However we have assumed that ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ reductions which implies $L \in \mathrm{ModL}$. This shows that $\mathrm{Mod}_k\mathrm{L} \subseteq \mathrm{ModL}$ whenever $k \geq 6$ is a composite number such that the number of distinct prime divisors of k is at least 2 and at most l. •

Theorem 4.4. *If* ModL *is closed under* $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ *reductions then* $\mathrm{coC_{=}L} \subseteq \mathrm{ModL}$.

Proof. Let us assume that ModL is closed under $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reductions. We know that ModDet is a canonical complete language for ModL under logspace many-one reductions. Also given $A \in \mathbb{Z}^{n\times n}$ as input the problem of determining if $\det(A) \neq 0$ is complete for coC$_=$L under logspace many-one reductions [AO96]. Using the Chinese remainder theorem we know that the $\det(A)$ is uniquely determined by its residues modulo all the primes from 2 and n^4. Therefore $\det(A) \neq 0$ if and only if $\det(A) \not\equiv 0 (\mathrm{mod}\ p)$ for some prime $2 \leq p \leq n^4$.

Similar to the proof of Lemma 4.3 it is easy to note that there exists a $O(\log n)$ space bounded Turing machine M that when given the matrix A as input computes all the primes from 2 to n^4 and also outputs the pairs $\langle A\sharp 1^{p_i}\rangle$ for all of the primes from 2 to n^4. Clearly $\det(A) \neq 0$ if and only if $\langle A\sharp 1^{p_i}\rangle \in \mathrm{Mod}_p\mathrm{Det}$ for at least one of the primes $p \in \{2,\ldots,n^4\}$. However we have assumed that ModL is closed under $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reductions from which it follows that we can determine if $\det(A) \neq 0$ in ModL and this implies coC$_=$L $\subseteq$ ModL. •

Corollary 4.5. *Assume that* NL = UL. *If* ModL *is closed under* $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ *reductions then* C$_=$L $\subseteq$ ModL.

Proof. Proof follows from Theorem 4.4 and [Vij09, Corollary 3.7]. •

4.1 Implications for $\leq^{\mathrm{L}}_{l-\mathrm{ctt}}$ and $\leq^{\mathrm{L}}_{\mathrm{ctt}}$ reductions

Definition 4.6. [ABO99] *Let* Σ *be the input alphabet and let* $L_1, L_2 \subseteq \Sigma^*$. *We say that* L_1 *is logspace* k*-disjunctive truth-table reducible to* L_2*, denoted by* $L_1 \leq^{\mathrm{L}}_{k-\mathrm{ctt}} L_2$*, if there exists* $f \in$ FL *such that given an input* $x \in \Sigma^*$ *of length* n $f(x) = \{y_1 \ldots, y_k\}$ *and* $x \in L_1$ *if and only if* $y_i \in L_2$ *for every* $1 \leq i \leq k$.

Definition 4.7. [ABO99] *Let* Σ *be the input alphabet and let* $L_1, L_2 \subseteq \Sigma^*$. *We say that* L_1 *is logspace conjunctive truth-table reducible to* L_2*, denoted by* $L_1 \leq^{\mathrm{L}}_{\mathrm{ctt}} L_2$*, if there exists* $f \in$ FL *such that given an input* $x \in \Sigma^*$ *of length* n *we have* $f \in$ FL *such that* $f(x) = \{y_1 \ldots, y_{p(n)}\}$ *and* $x \in L_1$ *if and only if* $y_i \in L_2$ *for every* $1 \leq i \leq p(n)$ *where* $p(n)$ *is a polynomial in* n.

Lemma 4.8. *Assume that* NL = UL. *If* ModL *is closed under* $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ *reductions where* $l \in \mathbb{Z}^+$ *and* $l \geq 2$ *then* ModL *is closed under* $\leq^{\mathrm{L}}_{l-\mathrm{ctt}}$ *reductions.*

Proof. Let Σ be the input alphabet and let $\sharp \notin \Sigma$. Also let $L_1, L_2 \subseteq \Sigma^*$ and let $L_2 \in$ ModL be such that $L_1 \leq^{\mathrm{L}}_{l-\mathrm{ctt}} L_2$ using a function $f \in$ FL. Therefore if $x \in \Sigma^*$ is the input then we have $f(x) = \langle y_1\sharp\cdots\sharp y_l\rangle$ such that $x \in L_1$ if and only if $y_i \in L_2$ for all $1 \leq i \leq l$.

However this is equivalent to $x \notin L_1$ if and only if $y_i \in \overline{L_2}$ for at least one $1 \leq i \leq l$ where $\overline{L_2}$ denotes the complement of L_2. But since we have assumed NL = UL it follows from [Vij09, Corollary 3.7] that $\overline{L_2} \in$ ModL. These observations show that $\overline{L_1} \leq^{\mathrm{L}}_{l-\mathrm{dtt}} \overline{L_2}$ and since we have assumed ModL is closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ we have $\overline{L_1} \in$ ModL. Once again from our assumption that NL = UL using the result that ModL is closed under complement shown in [Vij09, Corollary 3.7] it follows that $L_1 \in$ ModL. •

Theorem 4.9. *Assume that* NL = UL. *If* ModL *is closed under* $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ *reductions then* ModL *is closed under* $\leq^{\mathrm{L}}_{\mathrm{ctt}}$ *reductions.*

Proof. Proof of this result is similar to the proof of Lemma 4.8. We need to observe that the number of instances of L_2 that can be output by a $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ reduction is at most a polynomial in the size of the input. Since we have assumed NL = UL it follows from [Vij09, Corollary 3.7] that ModL is closed under complement and by using this property of ModL we obtain this result. •

5 Remarks

As mentioned in [Vij09, Section 4] in an earlier submission to the STACS 2009 conference I had claimed and given proofs of many of the closure properties of ModL. These properties followed from the statement that we need not restrict the FL function g in the definition of ModL to output a prime power in the unary representation but that it can be a composite number in the unary representation that has more than one distinct prime divisor. However once again as it is mentioned in [Vij09, Section 4] the proof of this statement was found to be incorrect.

In this note we have shown that ModL is closed under $\leq^{\mathrm{L}}_{m}$ reduction, $\vee$(join) and $\leq^{\mathrm{UL}}_{m}$ reduction. Also using [Vij09, Theorem 3.6 and Corollary 3.7] we have shown that ModL is closed under $\leq^{\mathrm{UL}}_{1-\mathrm{tt}}$ and $\leq^{\mathrm{L}}_{1-\mathrm{tt}}$ reductions assuming NL = UL. Following this under the assumption that UL = coUL we are also able to show that $\mathrm{ModL}^{\mathrm{UL}}$ = ModL. This in turn shows that $\mathrm{UL}^{\mathrm{ModL}}_{1-\mathrm{tt}}$ = $\mathrm{ModL}^{\mathrm{UL}}$ = ModL assuming NL = UL.

As we had mentioned in Section 1.1 the LCON problem was claimed to be logspace many-one hard for ModL in [Vij08]. Using [Vij09, Theorem 3.6] it follows that LCON is logspace many-one hard for ModL under the assumption that NL = UL. The results shown in this note do not seem to unconditionally show that LCON is logspace many-one hard for ModL.

In this note we have also considered the possibility of ModL being closed under $\leq^{\mathrm{L}}_{l-\mathrm{dtt}}$ and $\leq^{\mathrm{L}}_{\mathrm{dtt}}$ and shown its implications to other logspace counting classes such as $\mathrm{Mod}_k\mathrm{L}$ and $\mathrm{C}_=\mathrm{L}$ where $k, l \in \mathbb{Z}^+$ and $k \geq 6$ is a composite number that has more than one and at most l distinct prime divisors. Also [HT04] have shown results that deal with the closure of GapL under division. We remark that other than showing the closure of ModL under different types of reductions exploring the closure properties of GapL also leads to showing many relations among logspace counting classes.

5.1 Open problems

While we know that UL $\subseteq \oplus$L $\subseteq$ ModL we are yet to show NL $\subseteq$ ModL. This containment is believed to be true especially since we know due to the results shown in [Wig94, GW96] that NL/poly = $\oplus$L/poly and in [RA00] that

NL/poly = UL/poly and that these classes coincide in the uniform setting also which follow by using well known derandomization techniques for constructing pseudo-random generators that depend on the existence of functions in DSPACE(n) that require circuits of exponential size to be computed which are also believed to exist. A believably far more difficult problem of showing $\mathrm{L}^{\mathrm{ModL}} = \mathrm{ModL}$ is also left open in [Vij09].

Interestingly many of these closure properties and containments have already been examined in the polynomial time setting and analogous problems are open for the complexity class ModP (see [KT96]). It seems very likely that proof of any of these closure properties for ModP will also imply the same property to be true for ModL (possibly with some assumptions used in [Vij09] and in the results shown in this note such as NL = UL or UL = coUL).

References

[ABO99] Eric Allender, Robert Beals and Mitsunori Ogihara. The complexity of matrix rank and feasible system of linear equations. *Computational Complexity*, 8:99-126, 1999.

[AHT07] Manindra Agrawal, Thanh Minh Hoang and Thomas Thierauf. The polynomially bounded perfect matching problem is in NC^2. In *STACS '07: Proceedings of the* 24^{th} *Annual Symposium on Theoretical Aspects of Computer Science*, pages 489-499, 2007. Also published as ECCC Report No.129(2006).

[AJ93] Carme Àlvarez and Birgit Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107:3-30, 1993.

[AO96] Eric Allender and Mitsunori Ogihara. Relationships among PL, #L and the determinant. *RAIRO - Theoretical Informatics and Applications*, 30:1-21, 1996.

[AV04] V. Arvind and T.C. Vijayaraghavan. Abelian Permutation Group Problems and Logspace Counting Classes. *CCC '04: Proceedings of the 19th IEEE Annual Conference on Computational Complexity*, pages 204-214, 2004.

[AV05] V. Arvind and T.C. Vijayaraghavan. The Complexity of Solving Linear Equations over a Finite Ring. *STACS '05: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science*, pages 472-484, 2005.

[AV] V. Arvind and T.C. Vijayaraghavan. Classifying Problems on Linear Congruences and Abelian Permutation Groups using Logspace Counting Classes. To appear in *Computational Complexity*.

[BD+92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf and Christoph Meinel. Structure and Importance of Logspace-MOD Classes. *Mathematical Systems Theory*, 25(3):223-237, 1992.

[Ber84] Stuart Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147-150, 1984.

[BG92] Richard Beigel and John Gill. Counting classes: Thresholds, Parity, Mods and Fewness. *Theoretical Computer Science*, 103(1):3-23, 1992.

[BJ+91] Gerhard Buntrock, Birgit Jenner, Klaus-Jörn Lange and Peter Rossmanith. Unambiguity and fewness for logarithmic space. In *FCT '91: Proceedings of the* 8^{th} *International Conference on Fundamentals of Computation Theory*, LNCS 529, pages 168-179, Springer, 1991.

[CDL01] Andrew Chiu, George Davida and Bruce Litow. Division in logspace-uniform NC^1. *RAIRO - Theoretical Informatics and Applications*, 35:259-276, 2001.

[Dam92] Carsten Damm. $DET=L^{\#L}$. Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universitat zu Berlin, 1991.

[GK87] Dimitri Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanent is in NC. In *FOCS '87: Proceedings of the* 28^{th} *Annual IEEE Symposium on Foundations of Computer Science*, pages 166-172, IEEE Computer Society Press, 1987.

[GW96] Anna Gál and Avi Wigderson. Boolean complexity classes versus their arithmetic analogs. *Random Structures and Algorithms*, 9(1-2):99-111, 1996.

[HAB02] 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.

[HT03] Thanh Minh Hoang and Thomas Thierauf. The complexity of the characteristic and the minimal polynomial. *Theoretical Computer Science*, 295(1-3):205-222, 2003.

[HT04] Thanh Minh Hoang and Thomas Thierauf. *On Closure Properties of* GapL. Electronic Colloquium on Computational Complexity Report No.24(2004).

[HRV00] Ulrich Hertrampf, Steffan Reith and Heribert Vollmer. A Note on Closure Properties of Logspace MOD Classes. *Information Processing Letters*, 75(3):91-93 2000

[KT96] Johannes Köbler and Seinosuke Toda. On the power of generalized MOD-classes. *Mathematical Systems Theory*, 29(1):33-46, 1996.

[Lan97] Klaus-Jörn Lange. An unambiguous class possessing a complete set. In *Proceedings of the* 14^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS '97)*, LNCS 1200, pages 339-350, Springer, 1997.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

[RA00] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal on Computing*, 29(4):1118-1131, 2000.

[Tod91] Seinosuke Toda. *Counting problems computationally equivalent to computing the determinant.* Technical Report CSIM 91-07, Department of Computer Science, University of Electro-Communications, Tokyo, Japan, May 1991.

[Val92] Leslie G. Valiant. Why is boolean complexity theory difficult? *Proceedings of the London Mathematical Society symposium on Boolean function complexity*, pages 84-94, Cambridge University Press, 1992.

[Vij08] T.C. Vijayaraghavan. *Classifying certain algebraic problems using logspace counting classes.* Ph.D Thesis, The Institute of Mathematical Sciences, Homi Bhabha National Institute, December 2008.

[Vij09] T.C. Vijayaraghavan. *Characterization of* ModL *using Prime Modulus.* Electronic Colloquium on Computational Complexity Report No.82(2009).

[Vin91] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Structure in Complexity Theory Conference*, pages 270-284, 1991.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.

[Wig94] Avi Wigderson. NL/poly $\subseteq \oplus$L/poly. In *Proceedings of the* 9^{th} *Structure in Complexity Theory Conference*, pages 59-62, 1994.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de