On the Power of Unambiguity in Logspace*

Aduri Pavan† Raghunath Tewari‡ N. V. Vinodchandran§

January 12, 2010

Abstract

We report progress on the NL vs UL problem.

- We show unconditionally that the complexity class ReachFewL $\subseteq$ UL. This improves on the earlier known upper bound ReachFewL $\subseteq$ FewL.
- We investigate the complexity of min-uniqueness - a central notion in studying the NL vs UL problem.
  - We show that min-uniqueness is necessary and sufficient for showing NL = UL.
  - We revisit the class OptL$[\log n]$ and show that ShortestPathLength - computing the length of the shortest path in a DAG, is complete for OptL$[\log n]$.
  - We introduce UOptL$[\log n]$, an unambiguous version of OptL$[\log n]$, and show that (a) NL = UL if and only if OptL$[\log n]$ = UOptL$[\log n]$, (b) LogFew $\leq$ UOptL$[\log n] \leq$ SPL.
- We show that the reachability problem over graphs embedded on 3 pages is complete for NL. This contrasts with the reachability problem over graphs embedded on 2 pages which is logspace equivalent to the reachability problem in planar graphs and hence is in UL.

*Research supported in part by NSF grants CCF-0830730, CCF-0916525, CCF-0830479, CCF-0916797.

†Department of Computer Science, Iowa State University: email:pavan@cs.iastate.edu

‡Department of Computer Science and Engineering, University of Nebraska-Lincoln: email:rtewari@cse.unl.edu

§Department of Computer Science and Engineering, University of Nebraska-Lincoln: email:vinod@cse.unl.edu

1 Introduction

This paper is centered around the NL vs UL problem. Can nondeterministic space bounded computations be made unambiguous? This fundamental question was first raised by Reinhardt and Allender in the paper entitled "Making Nondeterminism Unambiguous" [RA00]. Reinhardt and Allender showed that in the non-uniform setting it is indeed possible to simulate any nondeterministic logspace computation by an unambiguous one (that is, NL/poly = UL/poly) thus giving the first strong evidence that this relation might hold in the uniform setting as well.

A nondeterministic machine is unambiguous if it has at most one accepting path on any input [Val76]. UL is the class of decision problems that are decided by unambiguous logspace bounded nondeterministic machines. Clearly UL is the natural logspace analog of UP [Val76], the unambiguous version of NP. Historically, several researchers have investigated this class (for example, [BHS93, AJ93, BJLR91, BDHM92]) in different contexts. But Buntrock et al. [BJLR91] are the first to conduct a focused study of the complexity class UL and its variations.

Since the above-mentioned paper due to Reinhardt and Allender, there has been significant progress reported on the NL vs UL problem. In [ARZ99], Allender, Reinhard, and Zou showed that, under the (very plausible) hardness assumption that deterministic linear space has functions that can not be computed by circuits of size $2^{\epsilon n}$, the constructions given by Reinhardt and Allender can be *derandomized* to show that NL = UL [ARZ99]. As the reachability problem for directed graphs is complete for NL, it is natural to investigate the space complexity of reachability for subclasses of directed graphs and indeed the recent progress has been in this direction. In [BTV09], it is shown that reachability for directed planar graphs is in UL. Subsequently, Thierauf and Wagner showed that reachability for $K_{3,3}$-free and K_5-free graphs can be reduced to planar reachability in logspace [TW09]. Kynčl and Vyskočil showed that reachability for bounded genus graphs also reduces to the planar case [KV09]. Thus reachability for these classes of graphs is also in UL.

These results provide significant evidence that NL equals UL and establishing this fundamental equivalence may be within the reach of current techniques.

Our Results

Complexity of ReachFewL

FewL, the logspace analog of the polynomial time class FewP [All86, CH90], is the class of languages that are decided by nondeterministic logpsace machines with the promise that on any input there are at most polynomially many accepting paths [BJLR91, BDHM92]. Is FewL = UL? As FewL $\subseteq$ NL, this is a very interesting restriction of NL = UL question (it is known that FewL is in $\mathsf{L}^{\mathsf{promiseUL}}$ [All06]). While we are unable to show that FewL $\subseteq$ UL , as our first result we show that the class ReachFewL $\subseteq$ UL.

Result 1. ReachFewL $\subseteq$ UL $\cap$ coUL.

ReachFewL is a restriction of FewL [BJLR91]. We call a nondeterministic machine M a *reach-few* machine, if for any input x and any configuration c of $M(x)$, the number of paths from the start configuration to c, is bounded by a polynomial. ReachFewL is the class of languages decided by a reach-few machine that is logspace bounded. Notice that for a machine accepting a FewL language there can be (useless) configurations which does not lead to any accepting configuration but still with exponentially many paths from the start configuration to them. For a reach-few machine, the number of paths from the start configuration to *any* configuration is bounded by a polynomial. It is worth noting that such distinctions are not meaningful in the polynomial time setting as there is enough space to store the entire computation path during a nondeterministic computation. Our result improves on the previous known trivial upper bound of $\mathsf{ReachFewL} \subseteq \mathsf{FewL}$.

The class ReachFewL was also investigated by Buntrock, Hemachandra, and Siefkes [BHS93] under the notation $\mathsf{Nspace\text{-}Ambiguity}(\log n, n^{O(1)})$. In [BHS93], the authors define, for a space bound s and an unambiguity parameter a, the class $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n))$ as the class of languages accepted by $s(n)$ space bounded nondeterministic machines for which the number of paths from the start configuration to any configuration is at most $a(n)$. They show that $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) \log a(n))$ (hence $\mathsf{Nspace\text{-}Ambiguity}(\log n, O(1)) \subseteq \mathsf{UL}$). Our method can be used to show that $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) + \log a(n))$, thus substantially improving their upper bound.

We extend our first result to show that in fact we can count the number of accepting paths of a ReachFewL computation using an oracle in $\mathsf{UL} \cap \mathsf{coUL}$ and this implies that $\mathsf{ReachLFew} \subseteq \mathsf{UL} \cap \mathsf{coUL}$ (ReachLFew is similar to the class Few [CH90] in the polynomial-time setting).

Complexity of Min-uniqueness

Our second consideration is the notion of *min-uniqueness* which is a central notion in the study of unambiguity in the logspace setting. Min-uniqueness was first used by Wigderson to show that $\mathsf{NL} \subseteq \oplus\mathsf{L}$ non-uniformly [Wig94]. For a directed graph G and two nodes s and t, G is called *st-min-unique* if the minimum length s to t path is unique (if it exists). G is min-unique with respect to s, if it is sv-min-unique for all vertices v. While st-min-uniqueness was sufficient for Wigderson's result, Reinhardt and Allender used the stronger version of min-uniqueness to show that $\mathsf{NL} \subseteq \mathsf{UL/poly}$. In particular, they essentially showed that a logspace algorithm that transforms a directed graph into a min-unique graph with respect to the start vertex can be used to design an unambiguous algorithm for reachability. This technique was subsequently used in [BTV09] to show that reachability for planar directed graphs is in UL. These results strongly indicate that understanding min-uniqueness is crucial to resolving the NL vs UL problem.

Our second set of results is aimed at understanding min-uniqueness from a complexity-theoretic point of view. First we observe that min-uniqueness is necessary to show that $\mathsf{NL} = \mathsf{UL}$: if $\mathsf{NL} = \mathsf{UL}$, then there is a UL algorithm that makes any directed graph min-unique with respect to the start vertex. It is an easy observation that Reinhardt and Allender's technique will work even if the algorithm that makes a directed graph min-unique is only UL computable. Thus min-uniqueness is necessary and sufficient for showing $\mathsf{NL} = \mathsf{UL}$.

Result 2: $\mathsf{NL} = \mathsf{UL}$ if and only if there is a polynomially-bounded UL-computable weight function f so that for any directed acyclic graphs G, $f(G)$ is min-unique with respect to s.

Graph reachability problems and logspace computations are fundamentally related. While, reachability in directed graphs characterizes NL, Reingold's break-through results implies that reachability in undirected graphs captures L [Rei08]. We ask the following question. Can we investigate the notion of min-uniqueness in the context of complexity classes? We introduce a logspace function class $\mathsf{UOptL}[\log n]$ towards this goal.

OptL is the function class defined by Àlvarez and Jenner (in [AJ93]) as the logpsace analog of Krentel's OptP [Kre88]. OptL is the class of functions whose values are the maximum over all the outputs of an NL-transducer. Àlvarez and Jenner showed that this class captures the complexity of some natural optimization problems in the logspace setting (eg. computing the lexicographically maximum path of length $\leq n$ from s to t in a directed graph).

We consider $\mathsf{OptL}[\log n]$, the restriction of OptL where the function values are bounded by a polynomial. Àlvarez and Jenner considered this restriction and showed that $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$. However, previously there were no completeness results known for this class. We show the first completeness result for $\mathsf{OptL}[\log n]$. Consider the problem: Given G and two nodes s and t. Compute the length of the shortest path from s to t (denoted by SHORTEST-PATHLENGTH). We show that SHORTESTPATHLENGTH is complete for the class $\mathsf{OptL}[\log n]$ (under metric reductions).

Result 3. SHORTESTPATHLENGTH is complete for $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$.

Motivated by this completeness result, we define a new unambiguous function class $\mathsf{UOptL}[\log n]$ (unambiguous OptL: the minimum is output on a unique computation path). We show that $\mathsf{NL} = \mathsf{UL}$ is equivalent to to the question whether $\mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n]$.

Result 4. $\mathsf{NL} = \mathsf{UL}$ if and only if $\mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n]$.

SPL, the 'gap' version of UL, is an interesting logspace class first studied in [ARZ99]. The authors showed that the 'matching problem' is contained in a non-uniform version of SPL. They also show that SPL is powerful enough to contain FewL. We show that $\mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{SPL}}[\log n]$. Thus any language that is reducible to $\mathsf{UOptL}[\log n]$ is in the complexity class SPL. This contrasts with the equivalence $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$. We also show that the class LogFew reduces to $\mathsf{UOptL}[\log n]$ (refer to the next section for the definition of LogFew).

Result 5. $\mathsf{LogFew} \leq \mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{SPL}}[\log n]$.

Figures 1 and 2 depict the relations among various unambiguous and 'few' classes known before and new relations that we establish in this paper, respectively. Definitions of these complexity classes are given in subsequent sections.

Three pages are sufficient for NL

Finally we consider the reachability problem for directed graphs embedded on 3 pages and show that it is complete for NL. This is in contrast with reachability for graphs on 2 pages which is logspace equivalent to reachability in grid graphs and hence is in UL by the result of [BTV09]. Thus in order to show that $\mathsf{NL} = \mathsf{UL}$, it is sufficient to extend the results of [BTV09] to graphs on 3 pages. It is also interesting to note that reachability for graphs on 1 page is equivalent to reachability in trees and is complete for L.

Result 6. Reachability in directed graphs embedded on 3 pages is complete for NL.

We use a combination of existing techniques for proving our results.

2 Logspace Complexity Classes

We assume familiarity with the basics of complexity theory and in particular the log-space bounded complexity class NL. It is well known that checking for st-connectivity for general

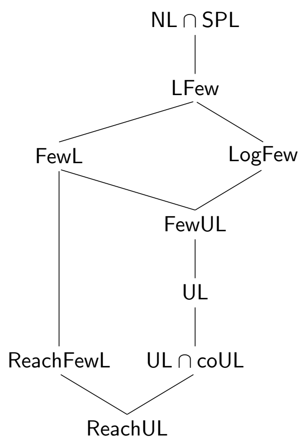

Figure 1: Relations known before.

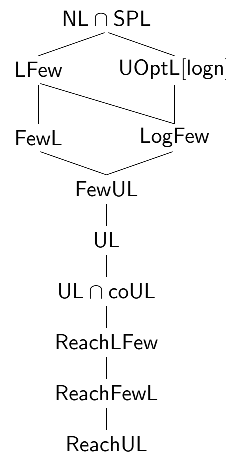

Figure 2: New relations.

directed graphs is NL-complete. We call a nondeterministic logspace machine an NL machine. For an NL machine M, let $acc_M(x)$ and $rej_M(x)$ denote the number of accepting computations and the number of rejecting computations respectively. Denote $gap_M(x) = acc_M(x) - rej_M(x)$.

We are interested in various restrictions of NL machines with few accepting paths. In the literature (eg [BJLR91, BDHM92, AJ93, ARZ99]) various versions of unambiguity and fewness have been studied. We first define them all here.

Definition 1. (**Unambiguous machines**) A nondeterministic logspace machine M is

- *reach-unambiguous* if for any input and for any configuration c, there is at most one path from the start configuration to c. (The prefix 'reach' in the term indicates that the property should hold for all configurations reachable from the start configuration).
- *unambiguous* if for any input there is at most one accepting path.
- *weakly unambiguous* if for any accepting configuration c there is at most one path from the start configuration to c.

Definition 2. (**Unambiguous classes**)

- ReachUL - class of languages that are decided by reach-unambiguous machines with at most one accepting path on any input.
- UL - class of languages that are decided by unambiguous machines.
- FewUL - class of languages that are decided by weakly unambiguous machines.
- LogFew - class of languages L for which there exists a weakly unambiguous machine M and a logspace computable predicate R such that $x \in L$ if and only if $R(x, acc_M(x))$ is true.

We could define a 'reach' version of FewUL. But that coincides with ReachUL as shown in [BJLR91]. The following containments are easy: ReachUL $\subseteq$ UL $\subseteq$ FewUL $\subseteq$ LogFew. It is also known that FewUL is L_d(UL) (logspace disjunctive truth-table closure of UL) [BJLR91].

By relaxing the unambiguity condition to a polynomial bound on the number of paths, we get analogous 'few' classes.

Definition 3. (**Few machines**) A nondeterministic logspace machine M is a

- *reach-few* machine if there is a polynomial p so that for any input x and for any configuration c, there are at most $p(|x|)$ paths from the start configuration to c.
- *few* machine if there is a polynomial p so that for any input x there are at most $p(|x|)$ accepting path.

Definition 4. (**Few classes**)

- ReachFewL - class of languages that are decided by reach-few machines.
- ReachLFew - class of languages L for which there exists a reach-few machine M and a logspace computable predicate R such that $x \in L$ if and only if $R(x, acc_M(x))$ is true.
- FewL - class of languages that are decided by few-machines.
- LFew - class of languages L for which there exists a few machine M and a logspace computable predicate R such that $x \in L$ if and only if $R(x, acc_M(x))$ is true.

As mentioned in the introduction, ReachFewL is the same class as Nspace−Ambiguity$(\log n, n^{O(1)})$ defined in [BHS93]. In [BJLR91], the authors observe that ReachFewL $\subseteq$ LogDCFL. This is because a depth first search of a reach-few machine can be implemented in LogDCFL.

The following containments follow from the definitions: ReachFewL $\subseteq$ FewL $\subseteq$ LFew. It is also clear that all the above-defined classes are contained in LFew and it is shown in [ARZ99] that LFew $\subseteq$ NL. Thus all these classes are contained in NL. Finally, we also consider the class SPL - the 'gap' version of UL. A language L is in SPL if there exists an NL-machine M so that for all inputs x, $gap_M(x) \in \{0, 1\}$ and $x \in L$ if and only if $gap_M(x) = 1$. SPL is contained in $\oplus$L (in fact all 'mod' classes) and it is big enough to contain LFew[ARZ99]. A nonuniform version of SPL contains the matching problem [ARZ99].

We will use *metric reductions* for functional reducibility. A function f is logspace metric reducible to function g, if there are logsapce computable functions h_1 and h_2 so that $f(x) = h_1(x, g(h_2(x)))$.

3 ReachFewL $\subseteq$ UL $\cap$ coUL

We will use the technique of Reinhardt and Allender to show the upper bound. We will state their theorem in a suitable form. But first we repeat the definition of min-uniqueness.

Definition 5. Let $G = (V, E)$ be a directed graph. For a pair of vertices s and t we say G is *st-min-unique* if there is a path from s to t in G, then the minimum length path from s to t is unique. G is called *min-unique* with respect to vertex s, if for all vertices v, G is sv-min-unique. G is called *min-unique* if it is min-unique with respect to all the nodes.

The following theorem from [RA00] states that the reachability problem can be solved unambiguously for classes of graphs that are min-unique with respect to the start vertex. Moreover, we can also check whether a graph is min-unique unambiguously.

Theorem 1 ([RA00]). *There is an unambiguous nondeterministic logspace machine M that on input a directed graph G and two vertices s and t such that*

1. *If G is not min-unique with respect to s, then M outputs 'not min-unique' on a unique path.*
2. *If G is min-unique with respect to s, then M accepts on a unique path if there is a directed path from s to t, and rejects on a unique path if there are no paths from s to t.*

We can also define the notion of min-uniqueness for weighted graphs. But this is equivalent to the above definition for our purposes if the weights are positive and polynomially bounded as we can replace an edge with weight k with a path of length k. In fact we will some times use this definition for weighted graphs without explicitly mentioning it. Thus for showing that $\mathsf{NL} = \mathsf{UL}$ it is sufficient to come up with a positive and polynomially bounded weight function that is UL-computable and makes a directed graph min-unique with respect to the start vertex.

Theorem 2. $\mathsf{ReachFewL} \subseteq \mathsf{UL} \cap \mathsf{coUL}$

Proof. Let L be in $\mathsf{ReachFewL}$ decided by the machine M. Let $G_{(M,x)}$ be the configuration graph of M on input x and s be the start configuration. Let t be the polynomial that bounds the number of paths from s to any configuration. Consider the edges in the lexicographical order. For the i^{th} edge give a weight 2^i. This is a very good weight function that assigns every path with unique weight. The problem is that this is not polynomially bounded. From this weight function we will give a polynomial number of weight functions that are logspace computable and polynomially bounded so that for one of them $G_{(M,x)}$ will be min-unique with respect to s. Since by Theorem 1 it is possible to check whether a given weight function makes the graph min-unique using a $\mathsf{UL} \cap \mathsf{coUL}$ computation, we can go through each weight function sequentially.

We will use the well known hashing technique introduced in [FKS84] for making the graph min-unique. Let N be the total number of configurations of $M(x)$. With respect to the above mentioned weight function, the weight of any path is bounded by 2^{N+1}. Let $p_1, p_2, \ldots, p_l$ be the first l distinct prime numbers so that $\prod_{i=1}^{l} p_i > 2^{N+1}t^2(N)$. Then $l \leq N^5$ and $p_l \leq N^6$. Hence each p_i has a logarithmic bit representation.

Let P be the set of all paths from s and w_i be the weight of the i^{th} path in P. Consider the product $\prod_{i,j}(w_i - w_j)$. This product is bounded by $2^{N+1}t^2(N)$ and is nonzero since for any pair i, j such that $i \neq j$, $w_i \neq w_j$. Thus $\prod_{i,j}(w_i - w_j) \neq 0(\mod \prod p_i)$. Hence there should be one (first) p_k with respect to which the product is non-zero and modulo this p_k, $w_i \neq w_j$ for all i, j. That is the weight function $w \mod p_k$ is a weight function which is UL-computable for which the configuration graph is min-unique with respect to the start configuration (UL-computable because, by Theorem 1, we can go through each prime and reject those which are not 'good' using a UL computation, until we reach p_k). □

Buntrock, Hemachandra, and Siefkes [BHS93] defined, for a space bound s and an unambiguity parameter a, the class $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n))$ as the class of languages accepted by $s(n)$ space bounded nondeterministic machines for which the number of paths from the start configuration to any configuration is at most $a(n)$. As one of their main theorems, the authors showed that $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) \log a(n))$

(hence $\mathsf{Nspace\text{-}Ambiguity}(\log n, O(1)) \subseteq \mathsf{UL}$). Our method can be used to show that $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) + \log a(n))$, thus substantially improving their upper bound.

Theorem 3. *For a space bound $s(n) \geq \log n$ and ambiguity parameter $a(n)$ computable in space $s(n)$ so that $a(n) = 2^{O(s(n))}$, $\mathsf{Nspace\text{-}Ambiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) + \log a(n))$.*

Theorem 4. *Let $L \in \mathsf{ReachFewL}$ accepted by a reach-few machine M. Then the $\#L$ function $acc_M(x)$ is computable in $\mathsf{FL}^{\mathsf{UL} \cap \mathsf{coUL}}$.*

Proof. The idea is to compute the number of paths from s to t of a $\mathsf{ReachFewL}$-computation with queries to $\mathsf{UL} \cap \mathsf{coUL}$ language using a logspace machine. If we make sure that all paths from s to t are of different weights then we can count them by making queries of the form "is there a path of length i from s to t" for all $i \leq N$ and by counting the number of positive answers.

We will use primes as before. But among polynomially many primes we have to reject those primes that does not give distinct weights to paths from s to t. Notice that Theorem 1 can only be used to rejects primes that do not make the graphs min-unique. It is possible that some prime makes the graph min-unique with respect to s but the graph may still have two paths from s to t of the same weight. For checking this more strict condition, we use the above result that $\mathsf{ReachFewL}$ is in $\mathsf{UL} \cap \mathsf{coUL}$.

Let L be a language in $\mathsf{ReachLFew}$ witnessed by a machine M and a polynomial q so that for every x, the number of paths from the start configuration of $M(x)$ to any configuration c is bounded $q(|x|)$. Let $G_{(M,x)}$ denote the standard layered configuration graph of $M(x)$. Then this graph also satisfy the property that the number of paths from the start configuration in the first layer to any configuration c is bounded by $q(|x|)$. Then the following language is in $\mathsf{UL} \cap \mathsf{coUL}$: $L = \{(x, c, i) \mid \text{there is a path of length } i \text{ from } s \text{ to } c \text{ in } G_{(M,x)}\}$.

In order to check whether p is a 'bad' prime, we need to check whether there are two paths from s to t of the same weight.
"p is bad $\Leftrightarrow \exists w \exists e = (c, c') \exists a \exists$ a path of length a from s to $c \wedge \exists$ a path of weight $w - w(e) - a$ from c' to t $\wedge \exists$ a path of length w from s to t in $G - e$"

This can be decided with polynomially many queries to L. Once we get a good prime p, we can use L as oracle to count the number of distinct paths from s to t using a deterministic logspace machine. This gives $\mathsf{ReachLFew} \subseteq \mathsf{UL} \cap \mathsf{coUL}$. □

Corollary 5. $\mathsf{ReachLFew} \subseteq \mathsf{UL} \cap \mathsf{coUL}$

4 Complexity of Min-uniqueness

Theorem 1 states that min-uniqueness is sufficient for showing $\mathsf{NL} = \mathsf{UL}$. Next we prove that if $\mathsf{NL} = \mathsf{UL}$ then there is a UL-computable weight function that makes any directed acyclic graph min-unique with respect to the start vertex. Thus min-uniqueness is necessary and sufficient for showing $\mathsf{NL} = \mathsf{UL}$.

Theorem 6. $\mathsf{NL} = \mathsf{UL}$ *if and only if there is a polynomially-bounded* UL*-computable weight function f so that for any directed acyclic graphs G, $f(G)$ is min-unique with respect to s.*

Proof. The reverse direction follows from the above theorem due to Reinhardt and Allender. For the other direction the idea is to compute a spanning tree of G rooted at s using reachability queries. Since NL is closed under complement, under the assumption that $\mathsf{NL} = \mathsf{UL}$, reachability is in $\mathsf{UL} \cap \mathsf{coUL}$. Thus the following language $A = \{(G, s, v, k) \mid$ there is a path from s to v of length $\leq k\}$ is in $\mathsf{UL} \cap \mathsf{coUL}$.

The tree can be described as follows. We say that a vertex v is in level k if the minimum length path from s to v is of length k. A directed edge (u, v) is in the tree if for some k (1) v is in level k (2) u is the lexicographically first vertex in level $k - 1$ so that (u, v) is an edge.

It is clear that this is indeed a well defined tree and deciding whether an edge $e = (u, v)$ is in this tree is in $\mathsf{L}^A \subseteq \mathsf{UL} \cap \mathsf{coUL}$.

Now for each edge in the tree give a weight 1. For the rest of the edges give a weight n^2. It is clear that shortest path from a vertex with respect to this weight function is min-unique with respect to s and it is computable using a UL-transducer.

□

Àlvarez and Jenner [AJ93] defines OptL as the logspace analog of Krental's OptP. They show that OptL captures the complexity of some natural optimization problems in the logspace setting (eg. computing lexicographically maximum path of length $\leq n$ from s to t in a directed graph). They also consider $\mathsf{OptL}[\log n]$ where the function values are bounded by a polynomial (hence has $O(\log n)$ bits representations). Here we revisit the class OptL [AJ93] and study them in relation to the notion of min-uniqueness. We define OptL as a minimization class and show that computing the minimum length path from s to t in a directed graph is complete (under metric reductions) for $\mathsf{OptL}[\log n]$.

Definition 6. An NL-transducer is a nondeterministic logspace bounded Turing machine with a one-way output tape in addition to its read-only input tape and read/write work tapes. We will assume that an NL-transducer will not repeat any configuration during its computation. Hence its configuration graph contains no cycles and all computation paths will halt with accepting or rejecting state after polynomially many steps. Let M be such a NL-transducer. An output on a computation path of M is valid if it halts in an accepting state. For any input x, $opt_M(x)$ is the minimum value over all valid outputs of M on x. If all the paths reject, then $opt_M(x) = \infty$. Further, M is called *min-unique* if for all x either $M(x)$ rejects on all paths or $M(x)$ outputs the minimum value on a unique path.

Definition 7. A function f is in OptL if there exists a NL-transducer M so that for any x, $f(x) = opt_M(x)$. A function f is in UOptL if there is a min-unique nondeterministic transducer M so that for any x, $f(x) = opt_M(x)$. Define $\mathsf{OptL}[\log n]$ and $\mathsf{UOptL}[\log n]$ as the restriction of OptL and UOptL where the output of the transducers are bounded by $O(\log n)$ bits.

If the output is unrestricted, then the computation path of an NL-transducer can be encoded in the output and hence all the output can be made distinct. Hence the classes OptL and UOptL are equivalent. But if we restrict the output to be of $O(\log n)$ bits the classes OptL and UOptL coincide if and only if $\mathsf{NL} = \mathsf{UL}$ as we show next.

We will need the following proposition shown in [AJ93]. $\mathsf{FL}^{\mathsf{NL}}[\log n]$ denotes the subclass of $\mathsf{FL}^{\mathsf{NL}}$ where the output length is bounded by $O(\log n)$.

Proposition 7 ([AJ93])**.** $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$.

Theorem 8. $\mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n]$ *if and only if* $\mathsf{NL} = \mathsf{UL}$.

Proof. $\mathsf{NL} = \mathsf{UL} \Rightarrow \mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n]$: Since NL is closed under complement, if $\mathsf{NL} = \mathsf{UL}$ then $\mathsf{NL} = \mathsf{UL} \cap \mathsf{coUL}$. Hence $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}} = \mathsf{FL}^{\mathsf{UL}\cap\mathsf{coUL}}$. For a function $f \in \mathsf{OptL}$, let M be FL machine that makes query to a language $L \in \mathsf{UL}\cap\mathsf{coUL}$ and computes f. Let N be the unambiguous machine that decided L. The min-unique transducer M' will simulate M and whenever a query y is made to L, it will simulate N on y and continue only on the unique path where it has an answer. In the end M' will output the value computed by M on a unique path.

$\mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n] \Rightarrow \mathsf{NL} = \mathsf{UL}$: Let $L \in \mathsf{NL}$. Since NL is closed under complement, there is a nondeterministic machine M that on input x accepts on some path and outputs '?' on all other paths if $x \in L$, and rejects on some paths and outputs '?' on all other paths if $x \notin L$. We will show that under the assumption $L \in \mathsf{coUL}$. Consider the NL-transducer which on input x simulates $M(x)$ and outputs 1 if M accepts and outputs 0 if M rejects and outputs a large value on paths with '?'. Let N be min-unique machine that computes this OptL function. Thus if $x \notin L$ then $N(x)$ has a unique path on which it outputs 0 (and there may be paths on which it outputs 1). If $x \in L$ then there is no path it outputs 0. Now consider the machine N' that simulates N and if N outputs 0 then it accepts. For all other values N' rejects. Clearly this is an unambiguous machine that decides $\overline{L}$.

□

Next we will exhibit a natural problem that is complete for $\mathsf{OptL}[\log n]$. Consider the computational problem ShortestPathLength

- ShortestPathLength: Given (G, s, t) where $G = (V, E)$ is a directed graph and s and t are two vertices in V. Compute the length of the shortest path from s to t. If no path exists then output ∞.

Theorem 9. ShortestPathLength *is complete for* $\mathsf{OptL}[\log n]$ *(under metric reductions)*

Proof. For the containment in $\mathsf{OptL}[\log n]$, consider the NL-transducer, which guesses a path of length $\leq n$ from s to t. It the guess succeeds then outputs the length of the path. Else it rejects. If G has a path from s to t, then the best path will be of length $\leq n$ hence the minimum among the outputs will be the length of the best path.

For the completeness, let f be a function in $\mathsf{OptL}[\log n]$ computed by an NL-transducer M. Since the output of M is of length $c \log n$ for some constant c, we will assume that M stores the intermediate value of the output on a separate work-tape (called the output work-tape) until the end of the computation, and before halting, M copies the contents of this work tape to the output tape deterministically and halts. Thus the configuration of this machine will also include the contend of this output work-tape. We will denote a typical configuration by the tuple (c, o) where o is the content of the output work tape. We will assume that at the start configuration the contents of this work-tape is 0.

Consider the following layered weighted graph $G_{(M,x)}$. $G_{(M,x)}$ has $p(|x|) + 1$ layers were p is the polynomial bounding the running time of M. For $1 \leq i \leq p(|x|)$, the i^{th} layer has vertices (i, c, o) where (c, o) is a configuration. The last layer which has just one vertex t. There is an edge from (i, c, o) to $(i + 1, c', o')$ if there is a valid move from the configuration (c, o) to (c', o'). The weight of this edge is $(o' - o) + n^k$ where k is a large constant so that $n^k > p(n) \times n^c$. We will also add edges from (i, c, o) to $(i + 1, c, o)$ if (c, o) is an accepting configuration. The weight of this edge is n^k. Finally we will add an edge with weight n^k from $(p(n), c, o)$ to t if (c, o) is an accepting configuration. For correctness, any computation path of M with an output o corresponds to a path in $G_{(M,x)}$ from the start configuration to t of weight $o + p(n)n^k$. Since the weights on the edges are positive and bounded by a polynomial, it is easy to replace to each edge with weight l with a path of length l. □

It can be verified that the standard reductions from directed graph reachability to other NL-complete problems also shows that a version of their optimization problems are $\mathsf{OptL}[\log n]$ complete. For example DFAShortestWordLength (Given a DFA M. Find the length of the shortest word that M accepts if $L(M)$ is nonempty) and WordGenLength (Given a set X with an associative binary operation, a subset $S \subseteq X$, and a word w over X. Find the length of the shortest generation sequence of w) are complete for $\mathsf{OptL}[\log n]$.

As $\mathsf{UOptL}[\log n] \subseteq \mathsf{OptL}[\log n]$, $\mathsf{UOptL}[\log n]$ is in $\mathsf{FL}^{\mathsf{NL}}[\log n]$. Here we show that $\mathsf{UOptL}[\log n]$ can be computed using a SPL oracle. Thus if NL reduces to $\mathsf{UOptL}[\log n]$, then $\mathsf{NL} \subseteq \mathsf{SPL}$.

Theorem 10. $\mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{SPL}}[\log n]$

Proof. Let $f \in \mathsf{UOptL}[\log n]$ and let M be the min-unique NL-transducer that witnesses that $f \in \mathsf{UOptL}[\log n]$ and let p be the polynomial bounding the value of f. Consider the following language L:

$$L = \{(x, i) \mid f(x) = i \text{ and } i \leq p(|x|)\}.$$

We will show that $L \in \mathsf{SPL}$. Then in order to compute f a logspace machine will ask polynomially many queries (x, i) for $1 \leq i \leq p(n)$.

Consider the following machine N which behaves as follows: N on input x and $i \leq p(n)$, simulates M on input x and accepts if and only if M halts with an output $\leq i$. Let $g(x, i)$ counts the number of accepting paths of N on input (x, i). Notice that for $i < f(x)$, $g(x, i) = 0$, for $i = f(x)$ then $g(x, i) = 1$, and for $i > f(x)$, $g(x, i) \geq 1$.

Now consider the GapL function $h(x, j) = g(x, j)\Pi_{i=1}^{j-1}(1 - g(x, i))$. It follows that $h(x, j) = 1$ exactly when $f(x) = i$. For the rest of i, $h(x, j) = 0$. Thus $L \in \mathsf{SPL}$. r □

Corollary 11. *If* $\mathsf{NL} \subseteq \mathsf{L}^{\mathsf{UOptL}[\log n]}$ *then* $\mathsf{NL} \subseteq \mathsf{SPL}$.

An interesting question is whether FewL reduces to UOptL. We are not able to show this, but we show that the class LogFew reduces to UOptL.

Theorem 12. $\mathsf{LogFew} \leq \mathsf{UOptL}[\log n]$ *(under metric reductions)*

Proof. Let L be a language in LogFew. Let M be a weakly unambiguous machine that decided L. Consider the NL-transducer N that on input x, computes the number of accepting paths of $M(x)$: $N(x)$ guess a l so that $1 \leq l \leq p(n)$ (where p is the polynomial bounding the number of accepting configurations) and then guess l distinct accepting paths in lexicographically increasing accepting configurations and accepts and outputs l if all of them accepts. Clearly N outputs $acc_M(x)$ on exactly one computation path and all other paths that accepts will have output $< acc_M(x)$. □

5 Three pages are sufficient for NL

We show that the reachability problem for directed graphs embedded on 3 pages is complete for NL. It can be shown that the reachability problem for graphs on 2 pages is equivalent to reachability in grid graphs and hence is in UL by the result of [BTV09]. Thus in order to show that NL = UL it is sufficient to extend the techniques of [BTV09] to graphs on 3 pages. It is also interesting to note that graphs embedded on 1 page are outer-planar and hence reachability for directed graphs on 1 page is complete for L [ABC$^+$06].

Definition 8. 3Page is the class of all graphs G, that can be embedded on 3 pages as follows: all vertices of G lie along the spine and the edges lie on exactly one of the two pages without intersection. Moreover all edges are directed from top to bottom. 3PageReach is the language consisting of tuples (G, s, t), such that $G \in$ 3Page, s and t are two vertices in G and there exists a path from s to t in G.

Theorem 13. 3PageReach *is complete for* NL*.*

Proof. Assume that we are given a topologically sorted DAG G, with $(u_1, u_2, \dots, u_n)$ being the topological ordering of the vertices of G. We want to decide if there is a path in G from u_1 to u_n. We define an ordering on the edges of G, say $\mathcal{E}(G)$. Given two edges e_1 and e_2, (i) if head of e_1 precedes head of e_2, then e_1 precedes e_2 in the ordering, (ii) if head of e_1 is the same as the head of e_2, then e_1 precedes e_2 in the ordering if tail of e_1 precedes tail of e_2. It is easy to see that $\mathcal{E}(G)$ can be constructed in logspace given G and in any path from s to t, if edge e_1 precedes e_2, then e_1 precedes e_2 in $\mathcal{E}(G)$ as well. Let m be the number edges in G.

We create $2m$ copies of each vertex in G and let v_i^j denote the jth copy of the vertex u_i, for $i \in [n]$ and $j \in [2m]$. We order the vertices along the spine of H from top to bottom as follows:
$(v_1^1, v_2^1, \dots, v_n^1, v_n^2, v_{n-1}^2, \dots, v_1^2, v_1^3, v_2^3, \dots, v_n^3, \dots, v_n^{2m}, \dots, v_1^{2m})$.

Next we need to connect all the $2m$ vertices corresponding to each u_i from the top to bottom. We use the first 2 pages to do that. Put the edge (v_i^j, v_i^{j+1}) in H, for each $i \in [n]$ and each $j \in [2m-1]$, using page 1 when j is odd and page 2 when j is even. For the kth edge in $\mathcal{E}(G)$, say $e_k = (u_{k_1}, u_{k_2})$, put the edge $(v_{k_1}^{2k-1}, v_{k_2}^{2k})$ in H, using page 3. It is clear that this can be done without any two edges crossing each other. We give an example of this reduction in Figure 3. The claim is, there exists a path from u_1 to u_n in G if and only if there exists a path from v_1^1 to v_n^{2m} in H.

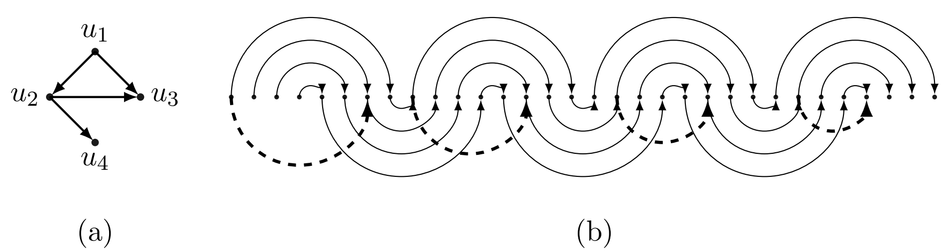

Figure 3: (a) Graph G. (b) The corresponding graph H. The dashed edges of H are on page 3.

Suppose there exists a path p from u_1 to u_n in G. Let $p = (e_{i_1}, \dots e_{i_l})$. For each $j \in [l]$, corresponding to e_{i_j} there exists an edge in page 3 of H by construction, say f_j. Also by construction and the ordering $\mathcal{E}(G)$, the tail of f_j lies above the head of f_{j+1} along the spine of H. Further, since the head of $e_{i_{j+1}}$ is the same as the tail of e_{i_j} for $j \in [l-1]$, there exists a path from the tail of f_j to the head of f_{j+1} (using edges from pages 1 and 2). Thus we get a path from v_1^1 to v_n^{2m} in H.

To see the other direction, let ρ be a path from v_1^1 to v_n^{2m} in H. Let $\rho_3 = (\alpha_1, \alpha_2, \ldots, \alpha_r)$ be the sequence of edges of ρ that lie on page 3. Note that each of the edges in ρ_3 has a unique pre-image in G by the property of the reduction. This defines a sequence of edges p' in G by taking the respective pre-images of the edges in ρ_3. Now the sub-path of ρ from the v_1^1 to the head of α_1 uses only edges from page 1 and 2 and thus by construction the head of α_1 is a vertex $v_1^{l_1}$ (for some $l_1 \in [2m]$). Similar argument establishes that the tail of α_r is a vertex $v_n^{l_2}$ (for some $l_2 \in [2m]$) and also that the tail of α_i and the head of α_{i+1} are the copies of the same vertex in G, for $i \in [r-1]$. Therefore p' is a path from u_1 to u_n in G. $\square$

6 Acknowledgments

We thank Eric Allender for an interesting email discussion and providing valuable suggestions that improved the presentation of the paper. We thank V. Arvind for interesting email exchanges on the topic of this paper. The last author deeply thanks Meena Mahajan and Thanh Minh Hoang for discussions on a related topic during a recent Dagstuhl workshop. We thank Samir Datta and Raghav Kulkarni for discussions which lead to a weaker version of Theorem 13 (namely, reachability for 4-page graphs is complete for NL).

References

[ABC+06] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Grid graph reachability problems. In *Annual IEEE Conference on Computational Complexity*, pages 299–313, 2006.

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

[All86] Eric Allender. The complexity of sparse sets in P. In A. Selman, editor, *Proc. Conference on Structure in Complexity Theory*, pages 1–11. Springer-Verlag, 1986.

[All06] Eric Allender. NL-printable sets and nondeterministic kolmogorov complexity. *Theor. Comput. Sci.*, 355(2):127–138, 2006.

[ARZ99] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-mod class. *Mathematical Systems Theory*, 25(3):223–237, 1992.

[BHS93] Gerhard Buntrock, Lane A. Hemachandra, and Dirk Siefkes. Using inductive counting to simulate nondeterministic computation. *Information and Computation*, 102(1):102–117, 1993.

[BJLR91] Gerhard Buntrock, Birgit Jenner, Klaus-Jörn Lange, and Peter Rossmanith. Unambiguity and fewness for logarithmic space. In *Proceedings of the 8th International Conference on Fundamentals of Computation Theory (FCT'91)*, Volume 529 Lecture Notes in Computer Science, pages 168–179. Springer-Verlag, 1991.

[BTV09] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):1–17, 2009.

[CH90] Jin-Yi Cai and Lane Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 1990.

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984.

[Kre88] Mark Krentel. The complexity of optimization problems. *J. of Computer and System Sciences*, 36:490–509, 1988.

[KV09] Jan Kynčl and Tomáš Vyskočil. Logspace reduction of directed reachability for bounded genus graphs to the planar case. *Electronic Colloquium on Computational Complexity*, (50), 2009.

[RA00] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal of Computing*, 29:1118–1131, 2000. An earlier version appeared in FOCS 1997, pp. 244–253.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[TW09] Thomas Thierauf and Fabin Wagner. Reachability in $K_{3,3}$-free graphs and K_5-free graphs is in unambiguous log-space. In *17th International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 5699, pages 323–334. Springer-Verlag, 2009.

[Val76] Leslie Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Wig94] Avi Wigderson. NL/poly $\subseteq \oplus$L/poly. In *Proceedings of the 9th Structures in Complexity conference*, pages 59–62, 1994.