ON THE POWER OF UNAMBIGUITY IN LOG-SPACE

A. Pavan, Raghunath Tewari,
and N. V. Vinodchandran

Abstract. We report progress on the NL versus UL problem.

- We show that counting the number of s-t paths in graphs where the number of s-v paths for any v is bounded by a polynomial can be done in FUL: the unambiguous log-space function class. Several new upper bounds follow from this including $\mathsf{ReachFewL} \subseteq \mathsf{UL}$ and $\mathsf{LFew} \subseteq \mathsf{UL}^{\mathsf{FewL}}$.
- We investigate the complexity of min-uniqueness – a central notion in studying the NL versus UL problem. In this regard we revisit the class $\mathsf{OptL}[\log n]$ and introduce $\mathsf{UOptL}[\log n]$, an unambiguous version of $\mathsf{OptL}[\log n]$. We investigate the relation between $\mathsf{UOptL}[\log n]$ and other existing complexity classes.
- We consider the unambiguous hierarchies over UL and $\mathsf{UOptL}[\log n]$. We show that the hierarchy over $\mathsf{UOptL}[\log n]$ collapses. This implies that $\mathsf{ULH} \subseteq \mathsf{L}^{\mathsf{promiseUL}}$ thus collapsing the UL hierarchy.
- 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 log-space equivalent to the reachability problem in planar graphs and hence is in UL.

Keywords. Log-space complexity, unambiguous computations, graph reachability, log-space optimization, hardness.

Subject classification. 68Q05, 68Q10, 68Q15, 68Q17.

1. Introduction

This paper is centered around the NL versus UL problem. Can nondeterministic space bounded computations be made unambiguous? This fundamental question was first raised by Reinhardt & Allender (2000) in the paper enti-

tled "Making Nondeterminism Unambiguous". They showed that in the non-uniform setting it is indeed possible to simulate any nondeterministic log-space computation by an unambiguous one (that is, $\mathsf{NL/poly} = \mathsf{UL/poly}$) 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 (Valiant (1976)). UL is the class of decision problems that are decided by unambiguous log-space bounded nondeterministic machines. Clearly, UL is the natural log-space analog of UP (Valiant (1976)), the unambiguous version of NP. Historically, several researchers have investigated this class (for example, by Àlvarez & Jenner (1993); Buntrock *et al.* (1992, 1993, 1991)) in different contexts. But Buntrock *et al.* (1991) 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 progress reported on the NL versus UL problem. Allender *et al.* (1999b) 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 & Allender (2000) can be *derandomized* to show that $\mathsf{NL} = \mathsf{UL}$. 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. Bourke *et al.* (2009) showed that reachability for directed planar graphs is in UL. Subsequently, Thierauf & Wagner (2009) showed that reachability for $K_{3,3}$-free and K_5-free graphs can be reduced to planar reachability in log-space. Kynčl & Vyskočil (2010) showed that reachability for bounded genus directed graphs also reduces to the planar case. 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.

Reachability in graphs with few paths. FewL, the log-space analog of the polynomial time class FewP (Allender (1986); Cai & Hemachandra (1990)), is the class of languages that are decided by nondeterministic log-space machines that have the property that on any input there are at most polynomially many accepting paths (Buntrock *et al.* (1992, 1991)). Is $\mathsf{FewL} = \mathsf{UL}$? As $\mathsf{FewL} \subseteq \mathsf{NL}$, this is a very interesting restriction of the $\mathsf{NL} = \mathsf{UL}$ question (Allender (2006) showed that FewL is in $\mathsf{L}^{\mathsf{promiseUL}}$). While we are unable to show that $\mathsf{FewL} \subseteq \mathsf{UL}$, we prove new *unambiguous* upper bounds for complexity classes related to

FewL.

As one of our main results, we show that counting the number of s-t paths in graphs where the number of paths from s to any node is bounded by a polynomial is in the unambiguous function class FUL.

This result immediately implies a new upper bound ReachFewL $\subseteq$ UL.[1] ReachFewL is a restriction of FewL (Buntrock *et al.* (1991)). A nondeterministic machine M is called a *reach-few* machine, if on 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 log-space bounded. Our result improves on the previous known trivial upper bound of ReachFewL $\subseteq$ FewL.

The class ReachFewL was also investigated by Buntrock *et al.* (1993) under the notation NspaceAmbiguity$(\log n, n^{O(1)})$. Buntrock *et al.* (1993) define, for a space bound s and an unambiguity parameter a, the class NspaceAmbiguity$(s(n), a(n))$ 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 NspaceAmbiguity$(s(n), a(n)) \subseteq$ Uspace$(s(n) \log a(n))$ (hence NspaceAmbiguity$(\log n, O(1)) \subseteq$ UL). Our method can be used to show that NspaceAmbiguity$(s(n), a(n)) \subseteq$ Uspace$(s(n)+\log a(n))$, thus substantially improving their upper bound.

Even though our techniques do not lead to a new upper bound on FewL, we show a new upper bound for LFew (LFew is the counting verstion of FewL and is analogous to the class Few (Cai & Hemachandra (1990)) in the polynomial-time setting). We show that LFew $\subseteq$ UL$^{\text{FewL}}$. This puts LFew in the second level of FewL hierarchy.

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 log-space setting. Min-uniqueness was first used by Wigderson (1994) to show that NL $\subseteq \oplus$L/poly. 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 NL $\subseteq$ UL/poly. In particular, they essentially showed that a log-space 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

[1] Very recently in the paper by Garvin *et al.* (2011), we improve this result to show that ReachFewL actually collapses to ReachUL.

subsequently used by Bourke *et al.* (2009) 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 versus 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 NL = UL: if NL = UL, then there is a UL algorithm that gives a reachability preserving mapping from any directed graph to another graph that is 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 NL = UL.

Graph reachability problems and log-space computations are fundamentally related. While reachability in directed graphs characterizes NL, the breakthrough result of Reingold (2008) implies that reachability in undirected graphs captures L̇We ask the following question: Can we investigate the notion of min-uniqueness in the context of complexity classes? We introduce a log-space function class UOptL$[\log n]$ towards this goal.

OptL is the function class defined by Àlvarez & Jenner (1993) as the log-space analog of OptP, defined by Krentel (1988). 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 log-space setting.

Consider OptL$[\log n]$, the restriction of OptL where the function values are bounded by a polynomial. Àlvarez & Jenner (1993) considered this restriction and showed that OptL$[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$. Tantau (2003) showed that "given a directed graph G and two nodes s and t, computing the length of the shortest path from s to t" is complete for OptL$[\log n]$.

Here we define a new unambiguous function class UOptL$[\log n]$ (unambiguous OptL: the minimum is output on a unique computation path), and show that NL = UL is equivalent to the question whether OptL$[\log n]$ = UOptL$[\log n]$.

SPL, the 'gap' version of UL, is an interesting log-space class first studied by Allender *et al.* (1999b). 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 UOptL$[\log n] \subseteq \mathsf{FL}^{\mathsf{SPL}}[\log n]$. Thus any language that is reducible to UOptL$[\log n]$ is in the complexity class SPL. This contrasts with the equivalence OptL$[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$. We also show that the class LogFew reduces to UOptL$[\log n]$ (refer to the next section for the definition of LogFew).

Finally, we also observe that UOptL$[\log n]$ is contained in $\mathsf{FL}^{\mathsf{promiseUL}}$. A very

interesting open question is to show that FewL reduces to UOptL[log n].

Figure 1.1 depicts the relations among various unambiguous and 'few' classes that were known before and the new relations that we establish in this paper. Definitions of these complexity classes are given in subsequent sections.

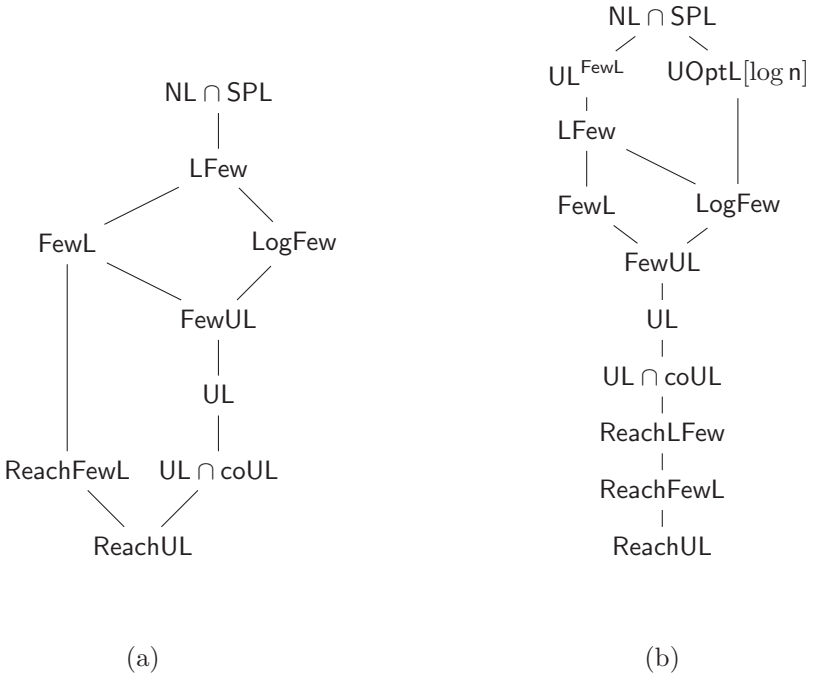

Figure 1.1: (a) Relations known before. (b) New Relations.

Unambiguous hierarchies. Since it is not known whether UL is closed under complement, it is interesting to investigate ULH; the unambiguous log-space hierarchy over UL. We first consider the hierarchy over UOptL[log n] and the show that the UOptL[log n] hierarchy collapses: $\mathsf{UOptL}[\log n]^{\mathsf{UOptL}[\log n]} \leq \mathsf{UOptL}[\log n]$. Since $\mathsf{UL} \leq \mathsf{UOptL}[\log n]$ (with relativization) it follows from this collapse result and the result that $\mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{promiseUL}}$, in fact $\mathsf{ULH} \subseteq \mathsf{L}^{\mathsf{promiseUL}}$.

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 log-space equivalent to reachability in grid graphs and hence is in UL by the

result of Bourke *et al.* (2009). Thus in order to show that NL = UL, it is sufficient to extend the results of Bourke *et al.* (2009) 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.

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

2. Log-space 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 reachability in general directed graphs is NL-complete. We call a nondeterministic log-space 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 (for instance Allender *et al.* (1999b); Àlvarez & Jenner (1993); Buntrock *et al.* (1992, 1991)) various versions of unambiguity and fewness have been studied. We first define them all here.

Definition 2.1. *(***Unambiguous machines***) A nondeterministic log-space 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 input and any accepting configuration c there is at most one path from the start configuration to c.*

Definition 2.2. *(***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 log-space 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 by Buntrock *et al.* (1991). The following containments are easy: ReachUL $\subseteq$ UL $\subseteq$ FewUL $\subseteq$ LogFew. It is also shown by Buntrock *et al.* (1991) that FewUL is $\mathsf{L}_d(\mathsf{UL})$ (log-space disjunctive truth-table closure of UL).

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

We are interested in graphs with a bound on the number of paths. We use the following notation due Buntrock *et al.* (1993) to quantify *ambiguity* in a graph.

Definition 2.3. *For a directed acyclic graph G and a node s, we say G is k-ambiguous with respect to s, if for any node v, the number of paths from s to v is bounded by k.*

Definition 2.4. *(***Few machines***) A nondeterministic log-space machine M is a*

- reach-few *machine if there is a polynomial p so that on any input x the configuration graph of M on x is $p(|x|)$-ambiguous with respect to the start configuration.*
- few *machine if there is a polynomial p so that on any input x there are at most $p(|x|)$ accepting paths.*

Definition 2.5. *(***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 log-space 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 log-space computable predicate R such that $x \in L$ if and only if $R(x, acc_M(x))$ is true.*

As mentioned earlier, ReachFewL is the same class as NspaceAmbiguity$(\log n, n^{O(1)})$ defined by Buntrock *et al.* (1993). Buntrock *et al.* (1991) 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 by Allender *et al.* (1999b) that $\mathsf{LFew} \subseteq \mathsf{NL}$. Thus all these classes are contained in NL. 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 on any input x, $gap_M(x) \in \{0,1\}$ and $x \in L$ if and only if $gap_M(x) = 1$. SPL is contained in $\oplus\mathsf{L}$ (in fact all 'mod' classes) and it is big enough to contain LFew (Allender *et al.* (1999b)). Allender *et al.* (1999b) also showed that a nonuniform version of SPL contains the matching problem. We use the facts that $\mathsf{L}^{\mathsf{UL}\cap\mathsf{coUL}} = \mathsf{UL} \cap \mathsf{coUL}$ and $\mathsf{FUL}^{\mathsf{UL}\cap\mathsf{coUL}} \subseteq \mathsf{FUL}$ in our paper. The proof of these uses standard techniques, refer to Thierauf & Wagner (2010) for a proof of the former equivalence and the latter equivalence can be shown similarly.

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

3. Reachability in graphs with few paths

In this section we show new upper bounds on the space needed by an unambiguous machine for reachability problems over graphs with a polynomial number of paths. Our main technical tool is the following theorem due to Reinhardt and Allender. First we give the definition of min-uniqueness.

Definition 3.1. *Let $G = (V, E)$ be a directed graph. For a pair of vertices s and t we say G is* st-min-unique *if, provided there is a path from s to t in G, the minimum length path from s to t is unique. G is called* min-unique *with respect to the 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 by Reinhardt & Allender (2000) 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 3.2 (Reinhardt & Allender 2000). *There is an unambiguous non-deterministic log-space machine M that given a directed graph G and two vertices s and t as input, does the following:*

(i) If G is not min-unique with respect to s, then M outputs 'not min-unique' on a unique path.

(ii) 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. For graphs with a polynomial number of paths, we can use known hashing techniques to make the graph min-unique. In particular, we will use the following well known scheme based on primes for our proofs.

LEMMA 3.3 (Fredman *et al.* 1984). *For every constant c there is a constant c' such that for every set S of n-bit integers with $|S| \leq n^c$ the following holds: There is a $c' \log n$-bit prime number p so that for any $x \neq y \in S$ we have $x \not\equiv y \pmod{p}$.*

All our upper bounds in this section are based on the following theorem.

THEOREM 3.4. *For any polynomial $q(n)$, there is a nondeterministic log-space bounded Turing machine M so that, for any reachability instance $\langle G, s, t\rangle$, if G is $q(n)$-ambiguous with respect to s, then M will output the number of paths from s to t on a unique path (all other paths reject).*

PROOF. First we show that the reachability question in a $q(n)$-ambiguous graph can be decided in an unambiguous manner. We do this by making such graphs min-unique with respect to s and applying Theorem 3.2.

THEOREM 3.5. *For any polynomial $q(n)$, there is a nondeterministic log-space bounded Turing machine M so that, for any reachability instance $\langle G, s, t\rangle$, if G is $q(n)$-ambiguous with respect to s, then M will accept on a unique path if there is a path from s to t. If there are no s to t paths, M will reject on a unique path. All other paths will output '?'.*

PROOF. (*of Theorem 3.5*). Let $\langle G, s, t\rangle$ be an instance of reachability. Consider the edges of G 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 it is not polynomially bounded. We will give a polynomial number of weight functions that are log-space computable and polynomially bounded so that for one of them G will be min-unique with respect to s. Since by Theorem 3.2 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. Let p_j be the j^{th} prime number. Then the j^{th} weight function (for $1 \leq j \leq q'(n)$ for an appropriate polynomial q' dictated by Lemma 3.3) is $w_j(e_i) = 2^i(\mod p_j)$. It follows from Lemma 3.3 that under some w_j all paths from s to t will have different weights. Hence the graph is min-unique under this weight function. □

(*Proof of Theorem 3.4 cont.*) Let $\mathcal{G}$ be the class of weighted graphs which are $q(n)$-ambiguous with respect to a fixed vertex s, such that every path starting at s has a distinct weight. Let A be the 'promise language' consisting of tuples (G, s, t, i), given the promise that $G \in \mathcal{G}$ such that there exists a path of length i from s to t. In particular, such a graph G is min-unique. Note that A is in $\mathsf{promiseUL}$,[2] that is, there exists an NL machine that has zero or one accepting path on every input that satisfies the promise. Also note that, given a $q(n)$-ambiguous graph G, with respect to one of the weight functions defined in Theorem 3.5, G is in $\mathcal{G}$.

In the above proof, a 'good' weight function actually does more than making the graph min-unique: it makes weights of every path distinct. With this stronger property we can count the number of paths by making queries of the form "is there a path of length i from s to t" to the language A, for all $i \leq N$ and by counting the number of positive answers (where N is the maximum weight possible and is bounded by a polynomial). It is important to observe that whenever we make a query to A, the query does satisfy the necessary promise.

But among polynomially many weight functions we have to reject those that do not give distinct weights to paths from s to t. Theorem 3.2 can only be used to reject weight functions that do not make the graph min-unique. It is possible that some weight function 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. We use the unambiguous machine for deciding reachability in order to check this more strict condition.

Let G' denote the standard layered graph of G: G' will have n layers. For a vertex u of G there will be copy u_i in the i^{th} layer of G'. There is an edge from u_i in the i^{th} layer to v_{i+1} in the $(i+1)^{th}$ layer if (u, v) is an edge in G. Notice that no new paths are added in this layered graph. The following claim is straightforward to see.

Claim 3.6. *If G is k-ambiguous, then G' is also k-ambiguous. Moreover, there*

[2] We define $\mathsf{promiseUL}$ later in Definition 4.9

is an s to t path of length i in G if and only if there is an s_1 to t_i path in G'.

Hence deciding reachability in G' can also be done unambiguously and checking whether G has an s to t path of weight exactly i can be done by reachability from s_1 to t_i in G'.

In order to check whether w is a 'bad' weight function, we need to check whether there are two paths from s to t of the same weight. We can use the following equivalence for checking this condition. α and β are integer values bounded by a polynomial.

$$w \text{ is bad} \Leftrightarrow \exists\alpha\exists(e=uv)\exists\beta[\exists \text{ a path of length } \beta \text{ from } s \text{ to } u] \wedge [\exists \text{ a path of weight } \alpha - w(e) - \beta \text{ from } v \text{ to } t] \wedge [\exists \text{ a path of length } \alpha \text{ from } s \text{ to } t \text{ in } G - \{e\}]$$

Note that the total number of possible values of α, β and e are bounded by polynomials in n. Thus we can decide whether a weight function w is 'bad' or not by making polynomially many reachability queries (that is for each choice of α, β and e). Once we get a good weight function w, we can again use reachability queries to compute the number of distinct paths from s to t using a deterministic log-space machine. Now using the fact that $\mathsf{L}^{\mathsf{UL}\cap\mathsf{coUL}} = \mathsf{UL}\cap\mathsf{coUL}$ (Thierauf & Wagner (2010)) we get the desired result. This proves the theorem. □

THEOREM 3.7. *Let $L \in$ ReachFewL be 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. By definition, the configuration graph of a machine accepting a ReachFewL language is $q(n)$-ambiguous for some fixed polynomial q. □

COROLLARY 3.8. $\mathsf{ReachFewL} \subseteq \mathsf{ReachLFew} \subseteq \mathsf{UL} \cap \mathsf{coUL}$

Our method can be used to show that $\mathsf{NspaceAmbiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n)+\log a(n))$. This substantially improves the earlier known upper bound by Buntrock *et al.* (1993) that
$\mathsf{NspaceAmbiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n)\log a(n))$.

THEOREM 3.9. *For a space bound $s(n) \geq \log n$ and ambiguity parameter $a(n)$ computable in space $s(n)$ so that $a(n) = 2^{O(s(n))}$,* $\mathsf{NspaceAmbiguity}(s(n), a(n)) \subseteq \mathsf{Uspace}(s(n) + \log a(n))$.

THEOREM 3.10. *Let $L \in \mathsf{FewL}$ be accepted by a few-machine M. Then the $\#L$ function $acc_M(x)$ is computable in $\mathsf{FUL}^{\mathsf{FewL}}$.*

PROOF. For an input x, let G denote the configuration graph of $M(x)$ and let c_s be the start configuration and c_t be the unique accepting configuration. Let p be the polynomial bounding the number of paths from c_s to c_t. First we will present an $\mathsf{FL}^{\mathsf{FewL}}$ computation that outputs a graph H that is $p(n)$-ambiguous with respect to c_s which preserves the number of paths from c_s to c_t. Combining this reduction with the unambiguous machine from Theorem 3.4 we will get the required upper bound.

We say a configuration c is *useful* if it is in some c_s to c_t path and c is *useless* if it is not useful. In the reduced graph H, we will remove all the useless nodes from G, and the edges incident on them. Clearly, all the c_s-to-c_t paths in G will be preserved in H. Moreover, H will be $p(n)$-ambiguous.

We will design a FewL language for detecting whether a configuration is useful or not. For a configuration $c \in G$, consider the following graph G_c. Take two copies of G: G_1 and G_2. In G_1 delete all the outgoing edges from c. In G_2, delete all the incoming edges to c. Now add a directed edge from the copy of c in G_1 to the copy of c in G_2 to get a single graph H. The following claim is easy to verify.

CLAIM 3.11. *c is useless if and only if there is a path from c_s to c_t in G'_x. Moreover, if c is useful, then the number of paths from c_s to c_t is at most $p(n)$.*

Thus the following language $L' = \{(x, c) \mid$ there is a path from c_s to c_t in $H_{x,c}\}$ is in FewL. The log-space machine, for each configuration c, checks whether c is useful or not by querying L' and delete it from G if it is useless. The output graph G' will not have any useless nodes and hence will be $p(n)$-ambiguous. □

As a corollary, we get the following new upper bound on the complexity class LFew. Earlier it was known to be in $\mathsf{NL} \cap \mathsf{SPL}$ by Allender *et al.* (1999b).

COROLLARY 3.12. $\mathsf{LFew} \subseteq \mathsf{UL}^{\mathsf{FewL}}$

Similar ideas together with the fact that planar reachability is in $\mathsf{UL} \cap \mathsf{coUL}$ (Bourke *et al.* (2009)) also gives the following upper bound on couting the number of paths in planar directed acyclic graphs with a polynomial bound on the number of paths. This improves the upper bound of $\mathsf{UAuxPDA}$ for this problem given by Nutan Limaye & Nimbhorkar (2010).

THEOREM 3.13. *For any polynomial p, there is an unambiguous machine M that given a planar directed acyclic graph G and two nodes s and t as input, (a) outputs the number of s to t paths if the number of such paths are bounded by $p(n)$ (b) outputs "more than $p(n)$ paths" if there are more than $p(n)$ s to t paths.*

PROOF. Consider the edge weight functions defined in Theorem 3.5. With respect to each weight function, we can check whether the number of s to t paths is bounded by $p(n)$ in an unambiguous manner by considering two cases: (i) if none of the weight functions are good, which can be checked unambiguously, then clearly there are more than $p(n)$ number of paths, (ii) if a weight function is good, then we can in fact count the number of paths from s to t and see if it is greater than $p(n)$ by an unambiguous algorithm that makes reachability queries (in $\mathsf{UL} \cap \mathsf{coUL}$ for planar graphs). Since $\mathsf{FUL}^{\mathsf{UL} \cap \mathsf{coUL}}$ is in FUL the theorem follows. □

4. Complexity of Min-uniqueness

Let f be a polynomially bounded, positive-valued, edge weight function (that is a function that takes an edge as input and outputs an integer which we call the weight of the edge). Then by an abuse of notation, for any directed graph G, we shall denote $f(G)$ to be the weighted directed graph obtained by taking every edge e in G and replacing it with the weighted edge having weight $f(e)$.

Theorem 3.2 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 4.1. $\mathsf{NL} = \mathsf{UL}$ *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 UL (since $\mathsf{UL} = \mathsf{coUL}$ under the above assumption). Thus the following language $A = \{(G, s, v, k) \mid$ there is a path from s to v of length $\leq k\}$ is in UL.

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 $\mathsf{L}^A \subseteq \mathsf{UL}$.

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 the 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 & Jenner (1993) define OptL as the log-space analog of OptP, which was defined by Krentel (1988). They show that OptL captures the complexity of some natural optimization problems in the log-space setting (e.g. computing lexicographically maximum path of length at most 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 $O(\log n)$ bit representation). Here we revisit the class OptL and study it in relation to the notion of min-uniqueness.

Definition 4.2. *An NL-transducer is a nondeterministic log-space 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 an NL-transducer. An output on a computation path of M is valid if it halts in an accepting state. For any input x, $\mathrm{opt}_M(x)$ is the minimum value over all valid outputs of M on x. If all the paths reject, then $\mathrm{opt}_M(x) = \infty$. 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 4.3. *A function f is in OptL if there exists a NL-transducer M so that for any x, $f(x) = \mathrm{opt}_M(x)$. 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.*

Remark 4.4. *We can also define the class $\mathsf{OptL}[\log n]$ (similarly $\mathsf{UOptL}[\log n]$) in terms of a function that gives the maximum value over all valid outputs of M on x, instead of the minimum value. It is easy to see that the two classes (corresponding to maximum and minimum) of functions we thus obtain are*

equivalent via a log-space reduction by subtracting the value of the $f(x)$ from a sufficiently large polynomial whose value is greater than the maximum possible output value of f. For the sake of convenience, we use both the notions in this paper.

We next observe that if we restrict the output to be of $O(\log n)$ bits, the classes OptL and UOptL coincide if and only if $\mathsf{NL} = \mathsf{UL}$.

We will need the following proposition shown by Àlvarez & Jenner (1993). $\mathsf{FL}^{\mathsf{NL}}[\log n]$ denotes the subclass of $\mathsf{FL}^{\mathsf{NL}}$ where the output length is bounded by $O(\log n)$.

PROPOSITION 4.5 (Àlvarez & Jenner 1993). $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n]$.

THEOREM 4.6. $\mathsf{OptL}[\log n] = \mathsf{UOptL}[\log n]$ *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 $\mathsf{OptL}[\log n] = \mathsf{FL}^{\mathsf{NL}}[\log n] = \mathsf{FL}^{\mathsf{UL}\cap\mathsf{coUL}}[\log n]$. For a function $f \in \mathsf{OptL}$, let M be the FL machine that makes queries 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 $L \in \mathsf{coUL}$. 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 a 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 on which 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 $\overline{L}$. □

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

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 log-space machine will ask polynomially many queries (x, i) for $1 \leq i \leq p(n)$.

Consider the following machine N: on input x and $i \leq p(n)$, it simulates M on input x and accepts if and only if M halts with an output at most i. Let $g(x, i)$ be 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 $h(x, i) = g(x, i) \prod_{j=1}^{i-1}(1 - g(x, j))$ (to know more about closure properties of GapL functions see the paper by Allender & Ogihara (1994)). It follows that $h(x, i) = 1$ exactly when $f(x) = i$. For the rest of i, $h(x, i) = 0$. Thus $L \in \mathsf{SPL}$. □

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

PROOF. In this proof we define the class $\mathsf{UOptL}[\log n]$ as a maximization class (see Remark 4.4). Let L be a language in LogFew. Let M be a weakly unambiguous machine that decides L. Consider the NL-transducer N that on input x computes the number of accepting paths of $M(x)$: $N(x)$ guesses an integer l so that $1 \leq l \leq p(n)$ (where p is the polynomial bounding the number of accepting configurations) and then guesses l distinct accepting paths to l accepting configurations, in a lexicographically increasing order, and accepts and outputs l if all of them accept. Clearly N outputs $acc_M(x)$ on exactly one computation path and all other paths that accept will have output less than $acc_M(x)$. □

DEFINITION 4.9. *(**Promise classes**)*

- *A* promise language *is a tuple (I_y, I_n), where I_y and I_n are disjoint subsets of $\{0, 1\}^*$ (collectively known as the promise instances).*

- *A promise language* (I_y, I_n) *is said to be in* promiseUL *if there is an* NL *machine* M*, such that* M *has a unique accepting path for instances in* I_y*, and no accepting paths for instances in* I_n*, but could have any number of accepting paths for all other instances.*

- *A language* A *is said to be* consistent *with a promise language* (I_y, I_n)*, if* $x \in I_y \Rightarrow x \in A$ *and* $x \in I_n \Rightarrow x \notin A$*.*

- f *is said to be in* $\mathsf{FL}^{\mathsf{promiseUL}}$ *if there exists a promise language* (I_y, I_n) *in* promiseUL *and a log-space transducer* M *such that for every language* A *consistent with* (I_y, I_n)*,* $f(x) = M^A(x)$ *for all* $x \in \{0,1\}^*$*.*

We next show that a function in $\mathsf{UOptL}[\log n]$ is also contained in $\mathsf{FL}^{\mathsf{promiseUL}}$.

THEOREM 4.10. $\mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{promiseUL}}$

PROOF. In this proof we define the class $\mathsf{UOptL}[\log n]$ as a maximization class (see Remark 4.4). Let f be a $\mathsf{UOptL}[\log n]$ function computed by a $\mathsf{UOptL}[\log n]$ machine. We will first define a promiseUL problem.

The instances of the problem are: $\langle M, x, k\rangle$ where M is a nondeterministic log-space bounded transducer, k is an integer and x is an input to M. The promise language (I_y, I_n) of 'Yes' and 'No' instances is defined as follows.

$$\begin{aligned} I_y &= \{\langle M, x, k\rangle \text{ so that } M \text{ is a } \mathsf{UOptL}[\log n] \text{ machine and } opt_M(x) = k\} \\ I_n &= \{\langle M, x, k\rangle \text{ so that } M \text{ is a } \mathsf{UOptL}[\log n] \text{ machine and } opt_M(x) < k\} \end{aligned}$$

Now an NL machine on input $\langle M, x, k\rangle$ simulates $M(x)$ and accepts if the output is k and rejects otherwise.

On I_y instances it accepts on a unique path. I_n instances it rejects on all paths.

Now consider a $\mathsf{UOptL}[\log n]$ function computed by a machine M. An FL machine asks queries $\langle M, x, k\rangle$ starting from the largest possible k and comes down until it gets a yes answer at which point it outputs that k. The FL machine is only asking queries in the promised region. □

5. Unambiguous Hierarchies

Since UL is not known to be closed under complement, it is interesting to study the complexity class hierarchy over UL which can be defined in natural way: $\mathsf{ULH}_1 = \mathsf{UL}$ and $\mathsf{ULH}_{(i+1)} = \mathsf{UL}^{\mathsf{ULH}_i}$. Then $\mathsf{ULH} = \bigcup_i \mathsf{ULH}_i$. We show that $\mathsf{ULH} \subseteq \mathsf{L}^{\mathsf{promiseUL}}$. For showing this we in fact first show that the hierarchy over $\mathsf{UOptL}[\log n]$ collapses. We then show that $\mathsf{UOptL}[\log n] \subseteq \mathsf{FL}^{\mathsf{promiseUL}}$.

We assume RST-relativizations when dealing with nondeterministic log-space oracle classes. When the machine enters the query state, it behaves deterministically until the entire query is written. One important consequence of this is that, since the number of configurations of a log-space machine is polynomial, the total number of queries that such a machine can make on any input is polynomially bounded. Moreover, the set of all potential queries that can be asked by a nondeterministic machine on any specific input can be computed by a deterministic log-space machine.

THEOREM 5.1. $\mathsf{UOptL}[\log n]$ *hierarchy collapses. That is,* $\mathsf{UOptL}[\log n]^{\mathsf{UOptL}[\log n]} \leq \mathsf{UOptL}[\log n]$ *under metric reductions.*

PROOF. We use an enhanced census technique, similar to the ones that are used to prove collapses of hierarchies over log-space classes (Allender *et al.* (1999a); Hemachandra (1989); Ogihara (1995); Schning & Wagner (1988)). But since we are dealing with function classes we need to be a bit more careful. Also, in this proof we define the class $\mathsf{UOptL}[\log n]$ as a maximization class (see Remark 4.4).

Let f be a $\mathsf{UOptL}[\log n]$ function computed by an oracle machine M making oracle calls to a $\mathsf{UOptL}[\log n]$ function g. Let N be a $\mathsf{UOptL}[\log n]$ machine computing g. We will show that f reduces to a $\mathsf{UOptL}[\log n]$ function h. An important consideration (which makes the proof a bit more complicated) is that f and g could be partial and on the inputs where the value is not defined, the corresponding machines reject on all paths (and hence do not have the unambiguous behavior).

Consider the computation of $M(x)$. Let $Q_x = \{q_1, \ldots, q_{n^c}\}$ be all the potential queries of $M(x)$ to the function g. Let $D_x = \{q \mid q \in Q_x \text{ and } g(q) \text{ is defined}\}$. Let $S_x = \sum_{q \in D_x} g(x)$. That is, S_x is the sum of all the values of the function g on queries on which g is defined. This value is polynomially bounded.

Consider the function h, which has two components, defined as $h(x) = \langle S_x, f(x) \rangle$ (obtained by concatenating S_x and $f(x)$). Clearly given x and $h(x)$, we can decode $f(x)$ in log-space and hence $f \leq h$. We will show that $h(x)$ is a $\mathsf{UOptL}[\log n]$ function.

Consider the following nondeterministic transducer M_h which operates in two phases, on input x. In the first phase, on input x, M_h tries to compute the sum S_x. Towards this M_h initializes a sum $S = 0$ and guesses a subset $A \subseteq Q_x$ of potential queries and simulates N on this subset: that is, M_h generates the potential queries $q \in Q_x$ one by one, for each of q, it guesses 0 or 1. If it guessed 0 (meaning g is not defined) it goes to the next query. If the guess is 1 then

it guesses a computation path ρ of N on q. If the path rejects, M_h rejects. Otherwise it updates $S = S + \rho_N(x)$ (that is, it adds the value computed by $N(q)$ on this path ρ to S). We claim that after the first phase, M_h will have computed S_x on a unique path, and all other paths the value computed will be less than S_x. M_h will output this sum S as the first component of the function. Since the highest value S_x is output on a unique path, for the second phase we will ignore the computation on any path that is a continuation of the paths that compute a value $S < S_x$.

In the second phase M_h will start simulating $M(x)$. For each query q asked by M, M_h will simulate the answer as follows (if $g(q)$ is defined, then M_h could have just guessed a path of N and continued; but a problem arises on queries for which the oracle function is undefined and hence the computation does not have an unambiguous behavior; we need to take care of this). M_h again guesses a subset A of queries as in phase one and for each of the queries in A, it also guesses a computation path ρ, of N and computes the sum S of values for each of the queries. It continues the computation only on the unique path where $S = S_x$. For this path if q is not in A, then M_h treats the answer to the query as "not defined" and continues. If q is in A then M_h treats the value computed by $N(q)$ on the path ρ as answer to the oracle query q. Finally, M_h outputs $\langle S_x, v\rangle$ where v is the value that N computes on a path. □

Corollary 5.2. $\mathsf{ULH} \subseteq \mathsf{L}^{\mathsf{promiseUL}}$

6. 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 Bourke *et al.* (2009). Thus in order to show that NL = UL it is sufficient to extend the techniques of Bourke *et al.* (2009) 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 as shown by Allender *et al.* (2009).

Definition 6.1. 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 three pages without intersection. Moreover all edges are directed from top to bottom.* 3PageReach *is the language consisting of tuples (G, s, t, f), such that $G \in$ 3Page, s and t are two vertices in G and there exists*

a path from s to t in G, and f is an embedding of G on 3 pages (that is, f defines the ordering of the vertices along the the spine and in which page an edge lies on).

THEOREM 6.2. 3PAGEREACH *is complete for* NL.

PROOF. To show that 3PAGEREACH is in NL we need to verify that, given an instance (G, s, t, f), if f is an embedding of G on 3 pages. Note that, for any two edges (u_1, v_1) and (u_2, v_2) in G that lies in the same page, the edges cross each other if and only if either (i) u_2 lies in between u_1 and v_1, or (ii) v_2 lies in between u_1 and v_1, in the ordering of the vertices along the spine. This condition can be checked in NL and therefore whether f is indeed an embedding of G on 3 pages or not can also be verified in NL.

To show that 3PAGEREACH is hard for NL, assume that we are given a topologically sorted DAG G, with $(u_1, u_2, \ldots, u_n)$ 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 the head of e_1 precedes the head of e_2, then e_1 precedes e_2 in the ordering, (ii) if the head of e_1 is the same as the 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 log-space 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.

For any integer k, let $[k]$ denote the set of integers $\{1, \ldots, k\}$. 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:
$(v_1^1, v_2^1, \ldots, v_n^1, v_n^2, v_{n-1}^2, \ldots, v_1^2, v_1^3, v_2^3, \ldots, v_n^3, \ldots, v_n^{2m}, \ldots, v_1^{2m})$.

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 6.1. 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.

Suppose there exists a path p from u_1 to u_n in G. Let $p = (e_{i_1}, \ldots e_{i_l})$. 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

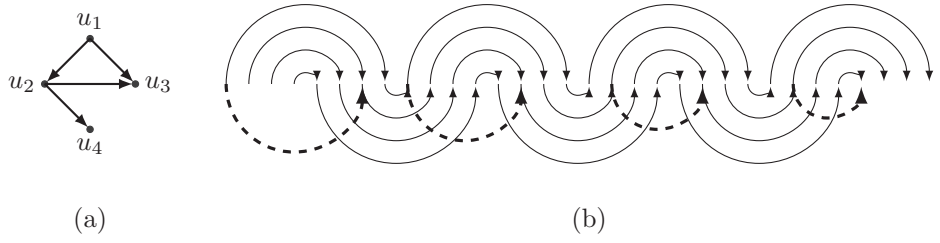

Figure 6.1: (a) Graph G. (b) The corresponding graph H. The dashed edges of H are on page 3.

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 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 $l_1 \in [2m]$). A 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. □

Acknowledgements

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 6.2 (namely, reachability for 4-page graphs is complete for NL). We would also like to acknowledge the suppert of the NSF grants CCF-0830730, CCF-0916525, CCF-0830479, CCF-0916797. We also thank the anonymous reviewers

for their helpful comments and suggestions, and in particular for pointing to the paper by Tantau (2003) on log-space optimization classes.

References

E. Allender & M. Ogihara (1994). Relationships among PL, #L, and the determinant. In *Structure in Complexity Theory Conference, 1994., Proceedings of the Ninth Annual*, 267 –278.

Eric Allender (1986). The complexity of sparse sets in P. In *Proc. of the conference on Structure in complexity theory*, 1–11. ISBN 0-387-16486-3.

Eric Allender (2006). NL-printable sets and nondeterministic Kolmogorov complexity. *Theor. Comput. Sci.* **355**(2), 127–138.

Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta & Sambuddha Roy (2009). Planar and Grid Graph Reachability Problems. *Theory Comput. Syst.* **45**(4), 675–723.

Eric Allender, Robert Beals & Mitsunori Ogihara (1999a). The complexity of matrix rank and feasible systems of linear equations. *Comput. Complex.* **8**, 99–126. ISSN 1016-3328. URL `http://portal.acm.org/citation.cfm?id=329550.329552`.

Eric Allender, Klaus Reinhardt & Shiyu Zhou (1999b). Isolation, Matching, and Counting: Uniform and Nonuniform Upper Bounds. *Journal of Computer and System Sciences* **59**, 164–181.

Carme Àlvarez & Birgit Jenner (1993). A Very Hard Log-space Counting Class. *Theoretical Computer Science* **107**, 3–30.

Chris Bourke, Raghunath Tewari & N. V. Vinodchandran (2009). Directed Planar Reachability Is in Unambiguous Log-Space. *ACM Trans. Comput. Theory* **1**(1), 1–17. ISSN 1942-3454.

Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf & Christoph Meinel (1992). Structure and Importance of Logspace-MOD Class. *Mathematical Systems Theory* **25**(3), 223–237.

Gerhard Buntrock, Lane A. Hemachandra & Dirk Siefkes (1993). Using Inductive Counting to Simulate Nondeterministic Computation. *Information and Computation* **102**(1), 102–117.

Gerhard Buntrock, Birgit Jenner, Klaus-Jörn Lange & Peter Rossmanith (1991). 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, 168–179. Springer-Verlag.

Jin-Yi Cai & Lane Hemachandra (1990). On the power of parity polynomial time. *Mathematical Systems Theory* .

Michael L. Fredman, János Komlós & Endre Szemerédi (1984). Storing a Sparse Table with $O(1)$ Worst Case Access Time. *J. ACM* **31**(3), 538–544.

Brady Garvin, Derrick Stolee, Raghunath Tewari & N. V. Vinodchandran (2011). ReachFewL=ReachUL. Manuscript.

L. Hemachandra (1989). The strong exponential hierarchy collapses. *J. of Computer and System Sciences* **39**(3), 299–322.

Mark Krentel (1988). The complexity of optimization problems. *J. of Computer and System Sciences* **36**, 490–509.

Jan Kynčl & Tomáš Vyskočil (2010). Logspace Reduction of Directed Reachability for Bounded Genus Graphs to the Planar Case. *ACM Trans. Comput. Theory* **1**(3), 1–11. ISSN 1942-3454.

Meena Mahajan Nutan Limaye & Prajakta Nimbhorkar (2010). Longest Paths in Planar DAGs in Unambiguous Log-Space. *Chicago Journal of Theoretical Computer Science* **2010**(8).

M. Ogihara (1995). Equivalence of NCk and ACk - 1 Closures of NP and Other Classes. *Information and Computation* **120**(1), 55 – 58. ISSN 0890-5401. URL `http://www.sciencedirect.com/science/article/B6WGK-45NJJWP-43/2/4fad4e8bff85772430de575c68b164aa`.

Omer Reingold (2008). Undirected connectivity in log-space. *J. ACM* **55**(4), 1–24. ISSN 0004-5411.

Klaus Reinhardt & Eric Allender (2000). Making nondeterminism unambiguous. *SIAM Journal of Computing* **29**, 1118–1131. An earlier version appeared in FOCS 1997, pp. 244–253.

Uwe Schning & Klaus Wagner (1988). Collapsing oracle hierarchies, census functions and logarithmically many queries. In *STACS 88*, Robert Cori & Martin Wirsing, editors, volume 294 of *Lecture Notes in Computer Science*, 91–97. Springer Berlin / Heidelberg. URL `http://dx.doi.org/10.1007/BFb0035835`. 10.1007/BFb0035835.

Till Tantau (2003). Logspace Optimisation Problems and their Approximation Properties. Technical Report TR03-077, Electronic Colloquium on Computational Complexity.

Thomas Thierauf & Fabian Wagner (2010). The Isomorphism Problem for Planar 3-Connected Graphs Is in Unambiguous Logspace. *Theor. Comp. Sys.* **47**, 655–673. ISSN 1432-4350. URL `http://dx.doi.org/10.1007/s00224-009-9188-4`.

Thomas Thierauf & Fabin Wagner (2009). 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, 323–334. Springer-Verlag.

Leslie Valiant (1976). The Relative Complexity of Checking and Evaluating. *Information Processing Letters* **5**, 20–23.

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

Manuscript received 5 October 2010

A. Pavan
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
`pavan@cs.iastate.edu`

Raghunath Tewari
Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588 USA
`rtewari@cse.unl.edu`

N. V. Vinodchandran
Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588 USA
`vinod@cse.unl.edu`