comput. complex. **23** (2014), 85 – 98

1016-3328/14/010085-14
published online November 6, 2012
DOI 10.1007/s00037-012-0050-8

ReachFewL = ReachUL

Brady Garvin, Derrick Stolee,
Raghunath Tewari, and N. V. Vinodchandran

Abstract. We show that two complexity classes introduced about two decades ago are *unconditionally* equal. ReachUL is the class of problems decided by nondeterministic log-space machines which on every input have *at most one* computation path from the start configuration to any other configuration. ReachFewL, a natural generalization of ReachUL, is the class of problems decided by nondeterministic log-space machines which on every input have *at most polynomially many* computation paths from the start configuration to any other configuration. We show that ReachFewL = ReachUL.

Keywords. Log-space complexity, unambiguous computations, graph reachability.

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

1. Introduction

A nondeterministic machine is said to be *unambiguous* if for every input there is at most one accepting computation. UL is the class of problems decided by unambiguous log-space nondeterministic machines. Is this restricted version of log-space nondeterminism powerful enough to capture general log-space nondeterminism (the complexity class NL)? Recent research gives ample evidence to believe that the conjecture NL = UL is true (Allender *et al.* 1999; Bourke *et al.* 2009; Reinhardt & Allender 2000; Thierauf & Wagner 2009). However, researchers have yet to find a proof of this equality.

This paper considers a restricted version of log-space unambiguity called *reach-unambiguity*. A nondeterministic machine 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). ReachUL is the class of languages that are decided by log-space bounded reach-unambiguous machines, as defined by Buntrock *et al.* (1991).

ReachUL is a natural and interesting subclass of UL. As defined, ReachUL is a 'semantic' class. However, unlike most other semantic classes, ReachUL has a complete problem (see Lange 1997). In particular, Lange showed that the directed graph reachability problem associated with reach-unambiguous computations is ReachUL-complete. Subsequently, Allender & Lange (1998) showed that this reachability problem can be solved deterministically in space $O(\log^2 n/\log\log n)$ which is asymptotically better than Savitch's $O(\log^2 n)$ bound for the general reachability problem. Buntrock *et al.* (1991) showed that ReachUL is also known to be closed under complement.

The notion of *fewness* is a natural generalization of unambiguity that is of interest to researchers (see Allender 2006; Àlvarez & Jenner 1993; Buntrock *et al.* 1992, 1993, 1991; Pavan *et al.* 2010). Since an unrestricted log-space nondeterministic machine can have exponential number of accepting computations, *few* here means *polynomially* many. FewL is the class of problems decided by nondeterministic log-space machines which on any input have at most a *polynomial* number of accepting computations. Thus, FewL extends the class UL in a natural way. The analogous extension of ReachUL is the class ReachFewL–the class of problems decided by nondeterministic log-space machines which on any input have at most polynomial number of computation paths from the start configuration to *any* configuration (not just the accepting configuration). Can fewness be simulated by unambiguity? In particular, is FewL = UL? This is an interesting open question and a solution is likely to have implications on the NL versus UL question.

In this paper, we show that for reach-unambiguity, it is indeed the case that fewness does not add any power to unambiguity for log-space computations. That is, we show that ReachFewL= ReachUL. This theorem improves a recent upper bound that ReachFewL $\subseteq$ UL $\cap$ coUL shown in Pavan *et al.* (2010).

THEOREM 1.1 (Main Theorem). ReachFewL = ReachUL.

Proof outline. The proof is based on the well-known hashing technique due to Fredman *et al.* (1984) (see Theorem 2.11). Our goal is to reduce a ReachFewL computation to a ReachUL computation. Consider the configuration graph of a ReachFewL computation and consider the weighting scheme w where the i^{th} edge of this graph gets a weight 2^i. With respect to w, the graph is *distance isolated* (two distinct paths have different weights). By definition, the configuration graph of a ReachFewL computation has at most polynomially many paths from the start configuration to any other configuration. Hence, by the FKS-hashing theorem, there is an $O(\log n)$ bit prime number p so that with respect to the weight function w_p the graph is distance isolated, where $w_p(e) = w(e) \pmod p$. Now a standard layering technique will make this new weighted graph reach-unambiguous. This argument works for primes that are 'good'. For rejecting a bad prime, we use the result from Lange (1997) that checking whether a graph is reach-unambiguous with respect to a specific vertex can be done in ReachUL. Thus, we can cycle through all $O(\log n)$ bit numbers one by one, check whether it is prime, and if yes, check whether it is a good prime. For the first such good prime, we are guaranteed that the corresponding layered graph is reach-unambiguous. All these computations can be performed in log-space, and hence, we get that ReachFewL log-space Turing reduces to ReachUL. The theorem follows since ReachUL is closed under log-space Turing reductions (see Buntrock *et al.* 1991).

As a corollary to the main theorem, we get a new upper bound for the reachability problem over certain class of graphs that beats Savitch's $O(\log^2 n)$ space bound. Allender & Lange (1998) showed that the reachability problem over reach-unambiguous graphs can be solved in DSPACE$(\log^2 n/\log\log n)$. Our main theorem implies the same upper bound for the reachability problem over directed graphs that are *polynomially ambiguous*.

COROLLARY 1.2. *The s-t reachability problem over graphs with a promise that there are at most polynomially many paths from s to any other vertex can be solved in deterministic space* $O(\log^2 n/\log\log n)$.

The main theorem and the corollary can be slightly extended to get a $o(\log^2 n)$-space algorithm for the reachability problem over graphs with at most $2^{o(\log n \sqrt{\log \log n})}$ paths from the start vertex to any other vertex.

2. Definitions and necessary results

We only introduce the necessary definitions and notation related to log-space bounded complexity classes. For other standard complexity-theoretic definitions and notation that we use refer to the text book by Arora & Barak (2009).

In space complexity investigations, it is standard to view the computations as directed graphs on configurations. Given a Turing machine M and an input x, $G_{M,x}$ will denote the configuration graph of M on x.

L denotes deterministic log-space and NL denotes nondeterministic log-space. For a language A, L^A denotes the class of languages recognized by deterministic log-space machines with an oracle access to A. For a complexity class $\mathcal{C}$, $\mathsf{L}^{\mathcal{C}}$ denotes the class $\{\mathsf{L}^A \mid A \in \mathcal{C}\}$.

We are interested in log-space *unambiguous* complexity classes. There are mainly two versions of unambiguity that have been studied in the literature. The most general version gives rise to the class UL which is defined as follows.

Definition 2.1. *A language A is in the class UL if there exists a nondeterministic log-space machine M accepting A such that, for every instance x, M has at most one accepting computation on input x.*

The other form of log-space unambiguity that is studied in the literature is called *reach-unambiguity* (see Buntrock *et al.* 1991; Lange 1997). This notion gives rise to the class $\mathsf{ReachUL}$. We define reach-unambiguity as a general graph-theoretic notion.

Definition 2.2. *Let G be a graph, s be a vertex in G, and k be an integer. We say that G is k-*reach-unambiguous *with respect to s if for all vertices $x \in V(G)$, there are at most k paths from s to x. If $k = 1$, we say G is reach-unambiguous with respect to s.*

2.1. Definition and properties of ReachUL. Buntrock *et al.* (1991) defined ReachUL and showed that this class is closed under complement and log-space Turing reductions. Later, Lange (1997) showed that ReachUL (defined slightly differently) has complete problems. We will need these results to prove our main theorem.

Definition 2.3 (Buntrock *et al.* 1991). *A language L is in* ReachUL *if L is accepted by a nondeterministic log-space Turing machine M such that, on any input x, $M(x)$ has at most one accepting path and, in addition, $G_{M,x}$ is reach-unambiguous with respect to the start configuration.*

Thus, ReachUL is a subclass of UL by definition. Buntrock *et al.* also considered a variation of ReachUL, namely the class of languages that are accepted by reach-unambiguous machines without restricting the number of accepting paths. In particular, the reach-unambiguous machine deciding a language in this class is allowed to have more than one accepting computation each going to a different accepting configuration. But they showed that the resulting complexity class is same as ReachUL.

Lange (1997) considered ReachUL using the notation RUSPACE $(\log n)$ (or RUL) and with a slightly different definition. For a Turing machine M and input x, let s_x denote the start configuration and t_x denote the canonical accepting configuration (the accepting configuration where the state is the unique accepting state, all the tape heads are in the first cell of the respective tapes, and all the work tape contents are blanks).

Definition 2.4 (Lange 1997). *A language L is in* RUSPACE $(\log n)$ *if L is accepted by a nondeterministic log-space Turing machine M such that, on any input x, $G_{M,x}$ is reach-unambiguous with respect to the start configuration and (a) $x \in L \Rightarrow$ there is a path from s_x to t_x, (b) $x \notin L \Rightarrow$ there is no path from s_x to t_x.*

In Lange's definition, a string is accepted if there is a computation path from the start configuration to a fixed accepting configuration, while according to the definition of Buntrock et al, a string is accepted if there is a path from the start configuration to *some*

accepting configuration. It is easy to see that these two classes are same.

PROPOSITION 2.5. $\mathsf{ReachUL} = \mathsf{RUSPACE}(\log n)$.

PROOF. It is clear that $\mathsf{RUSPACE}(\log n) \subseteq \mathsf{ReachUL}$. To see the other containment, let L be a language in ReachUL witnessed by a reach-unambiguous machine M. Consider the machine M' which on input x simulates M on x. If M reaches an accepting configuration, M' moves to the canonical accepting configuration. Clearly, M' accepts x if and only if M accepts x, and as M is reach-unambiguous, M' is also reach-unambiguous. Moreover, since M has exactly one accepting computation path on positive instances, M' will also have exactly one path that leads to the canonical accepting configuration on such instances. □

We will use the name ReachUL to state results involving $\mathsf{RUSPACE}(\log n)$ from the literature. Lange (1997) proved that the graph reachability problem L_{ru} defined below is log-space many-one complete for ReachUL.

$$L_{ru} = \{\langle G, s, t\rangle \mid \text{ } G \text{ is a directed graph, there is a path from } s \text{ to } t, \text{ and } G \text{ is reach-unambiguous with respect to } s\}.$$

THEOREM 2.6 (Lange 1997). *L_{ru} is complete for* ReachUL *under log-space many-one reductions.*

The difficult part in the completeness proof is to show that L_{ru} is in ReachUL. Lange designed a clever ReachUL algorithm that checks whether a graph is reach-unambiguous with respect to the start vertex. We will use this algorithm in the proof of our main theorem.

2.2. Closure properties of ReachUL. We will use the fact that a log-space algorithm that queries a ReachUL language can be simulated in ReachUL. This is stated in Buntrock *et al.* (1991) without a proof. Given the fact that ReachUL is closed under complement, this is easy to prove. For the sake of completeness, we give a proof here.

LEMMA 2.7 (Buntrock *et al.* 1991). $\mathsf{L}^{\mathsf{ReachUL}} = \mathsf{ReachUL}$.

We will use the fact that ReachUL is closed under complement.

PROPOSITION 2.8 (Buntrock *et al.* 1991). ReachUL *is closed under complement.*

PROOF. (of Lemma 2.7). The containment $\mathsf{ReachUL} \subseteq \mathsf{L}^{\mathsf{ReachUL}}$ is immediate. Let L be a language in $\mathsf{L}^{\mathsf{ReachUL}}$ decided by a log-space oracle Turing machine M with access to a ReachUL oracle O. Since ReachUL is closed under complement, we can assume without loss of generality that O is accepted by a reach-unambiguous Turing machine N (a Turing machine whose configuration graph on any input is reach-unambiguous) with three types of halting configurations: 'accept', 'reject', and '?' so that for any input y (1) if $y \in O$ then there is a unique computation path that leads to an 'accept' configuration and all other computation paths lead to a '?' configuration and (2) if $y \notin O$ then there is a unique computation path that leads to a 'reject' configuration and all other computation paths lead to a '?' configuration. Moreover, since $O \in \mathsf{ReachUL}$, on any input, there is at most one path from the start configuration to any other configuration of N.

Consider the nondeterministic machine M' which on an input x, simulates $M(x)$ until a query configuration is reached with a query, say y. At this point M' will save the current configuration of M and simulate $N(y)$ until it halts. If $N(y)$ accepts y, then M' continues with the simulation of M with YES as the answer to the query y; if $N(y)$ rejects y, then M' continues with the simulation of M with NO as the answer the query y; and if $N(y)$ reaches a '?' halting configuration then, M' rejects the computation and halts. Finally M' accepts x if and only if M accepts x.

It is straightforward to verify that $M'(x)$ accepts if and only if $M(x)$ accepts and $G_{M',x}$ is reach-unambiguous with respect to the start configuration. □

DEFINITION 2.9. *A language L is in* ReachFewL *if L is accepted by a nondeterministic log-space Turing machine M such that, for some polynomial q and for any input x, $G_{M,x}$ is $q(|x|)$-reach-unambiguous with respect to the start configuration.*

2.3. Converting graphs with a few paths to distance isolated graphs.

DEFINITION 2.10. *Let G be a weighted graph on n vertices and let s be a vertex of G. We say that G is* distance isolated *with respect to s, if for every vertex $v \in V(G)$ and weight d there is at most one path of weight d from s to v, where weight of a path is the sum of the weights on its edges.*

We use the well-known hashing result due to Fredman, Komlós and Szemerédi to convert a graph with polynomially many paths to a distance isolated graph.

THEOREM 2.11 (Fredman *et al.* 1984). *For every constant c, there is a constant c' so that for every set S of n-bit integers with $|S| \leq n^c$ there is a $c' \log n$-bit prime number p so that for any $x \neq y \in S$ $x \not\equiv y \pmod{p}$.*

LEMMA 2.12. *Let G be a graph on n vertices and let s be a vertex of G. Let $E(G) = \{e_1, e_2, \ldots, e_\ell\}$ be the set of edges of G. Let q be a polynomial. If G is $q(n)$-reach-unambiguous with respect to s, then there is a prime $p \leq n^k$, for some constant k, such that the weight function $w_p : E(G) \to \{1, \ldots, p\}$ given by $w_p(e_i) = 2^i \pmod{p}$ defines a weighted graph G_{w_p} which is distance isolated with respect to s.*

PROOF. Let $q(n) \leq c_1 n^{k_1}$ for all $n \geq 1$. Also let w be the edge weight function that assigns the weight 2^i to the edge e_i, for $i \in [\ell]$. Let S_v be the set of weights of all paths from s to v, and $S = \cup_{v \in V(G)} S_v$. Then, $|S| \leq c_1 n^{k_1+1}$. By Theorem 2.11 there is a $c' \log n$-bit prime p, for some constant c', such that for any $x \neq y \in S$ $x \not\equiv y \pmod{p}$. Then, with respect to the prime p, we get the weight function w_p, which defines the weighted graph G_{w_p}, that is, distance isolated with respect to s. □

The graph G_{w_p} in Lemma 2.12 can be converted to an unweighted, distance isolated graph by replacing an edge having weight ℓ by a path of length ℓ.

2.4. Converting distance isolated graphs to unambiguous graphs. Given a distance isolated graph, we can form a reach-unambiguous graph by applying a standard layering transformation.

DEFINITION 2.13. *Let G be a directed graph on n vertices. The* layered graph *$lay(G)$ induced by G is the graph on vertices $V(G) \times \{0, 1, \ldots, n\}$, and for all edges (x, y) of G and $i \in \{0, 1, \ldots, n-1\}$, the edge $(x, i) \to (y, i+1)$ is in $lay(G)$.*

LEMMA 2.14. *If G is an acyclic and distance isolated graph with respect to a vertex s, then $lay(G)$ is reach-unambiguous with respect to $(s, 0)$, and there is a path of length d from s to v in G if and only if there is a path from $(s, 0)$ to (v, d) in $lay(G)$.*

PROOF. Since all edges in $lay(G)$ pass between consecutive layers, paths of length d from s to v in G are in bijective correspondence with paths from $(s, 0)$ to (v, d) in $lay(G)$. Since there exists at most one path of each length from s to any vertex v in G, there exists at most one path from $(u, 0)$ to any other vertex (v, d) in $lay(G)$. □

3. ReachFewL = ReachUL

We have sufficient tools to prove Theorem 1.1.

THEOREM 3.1. ReachFewL $\subseteq$ ReachUL.

PROOF. Let L be a language in ReachFewL. Then there is a constant c and a nondeterministic log-space machine M deciding L, so that for any input x, $G_{M,x}$ has at most $|x|^c$ paths from the start configuration to any other configuration. Note that, without loss of generality, we can assume that there is a single accepting configuration for a ReachFewL computation. Thus, in $G_{M,x}$, let s be the vertex corresponding to the start configuration and t be the vertex corresponding to the accepting configuration. For determining

membership of x in L, we need to decide whether there is a path from s to t in $G_{M,x}$.

Input: (G, s, t) such that G has at most n^c paths from s to any other vertex.

Output: If there is a path from s to t in G output True, else output False.

foreach $p \in \{1, \ldots, n^{c'}\}$ *such that* p *is a prime* **do**
  Define $w_p(e_i) = 2^i \pmod p$;
  Construct G_{w_p};
  Construct $lay(G_{w_p})$;
  foreach $d \in \{1, \ldots, |V(G_{w_p})|\}$ **do**
    if $\langle lay(G_{w_p}), (s, 0), (t, d)\rangle \in L_{ru}$ **then return** True;
  end
  return False;
end
return False;

Algorithm 1: ReachFewSearch(G, s, t)

Consider the algorithm ReachFewSearch(G, s, t) given in Algorithm 1. This is a log-space algorithm that queries the ReachUL complete language L_{ru} defined in Section 2. We will argue that there is a path from s to t in $G_{M,x}$ if and only if ReachFewSearch$(G_{M,x}, s, t)$ returns True. This will imply that ReachFewL $\subseteq$ $\mathsf{L}^{\mathsf{ReachUL}}$. Since $\mathsf{L}^{\mathsf{ReachUL}}$ equals ReachUL by Lemma 2.7, the theorem will follow.

For the rest of the discussion by G, we mean $G_{M,x}$. For constant c, let c' be the constant given by Theorem 2.11.

We say that a prime p is *good*, if G_{w_p} is distance isolated. By Lemma 2.12, there exists a good prime $p \in \{1, \ldots, n^{c'}\}$. For this good prime, $lay(G_{w_p})$ is reach-unambiguous with respect to $(s, 0)$ by Lemma 2.14. Moreover, there is a path from s to t in G, if and only if there is a d such that there is a path from $(s, 0)$ to (t, d) in this layered graph. So if there is a path from s to t in G, for this good prime $\langle lay(G_{w_p}), (s, 0), (t, d)\rangle \in L_{ru}$ and the algorithm returns True. Note that for a prime p that is not good, $lay(G_{w_p})$

will not be reach-unambiguous and $\langle lay(G_{w_p}), (s,0), (t,d)\rangle \notin L_{ru}$ for any d. □

Allender & Lange (1998) showed that ReachUL $\subseteq$ DSPACE $(\log^2 n/\log\log n)$ by showing $L_{ru} \in \text{DSPACE}(\log^2 n/\log\log n)$. It is not clear how to directly extend their techniques to ReachFewL. However, our main result implies the same upper bound for the reachability problem associated with ReachFewL computations.

COROLLARY 3.2. *The s-t reachability problem over graphs with a promise that there are at most polynomially many paths from s to any other vertex can be solved in deterministic space* $O(\log^2 n/\log\log n)$.

3.1. Extension. Buntrock *et al.* (1993) investigated the class ReachFewL using the notation NspaceAmbiguity$(\log n, n^{O(1)})$ which is defined below.

DEFINITION 3.3. *For a space bound s and unambiguity parameter a, a language L is said to be in the class* NspaceAmbiguity$(s(n), a(n))$ *if L is accepted by an* $s(n)$ *space bounded nondeterministic Turing machine M, such that on any input x,* $G_{M,x}$ *is* $a(|x|)$*-reach-unambiguous with respect to the start configuration.*

Buntrock *et al.* (1993) showed that NspaceAmbiguity$(s(n), a(n)) \subseteq$ USPACE$(s(n)\log a(n))$ (hence NspaceAmbiguity$(\log n, O(1)) \subseteq$ UL). This result was recently improved by Pavan *et al.* (2010) who showed that NspaceAmbiguity$(s(n), a(n)) \subseteq$ USPACE$(s(n) + \log a(n))$. Here we further improve this upper bound.

DEFINITION 3.4. *For a space bound s, a language L is said to be in the class* ReachUSPACE$(s(n))$ *if L is accepted by an* $s(n)$ *space bounded nondeterministic Turing machine M, such that on any input x,* $G_{M,x}$ *is reach-unambiguous with respect to the start configuration.*

The proof of the following theorem is identical to the proof of Theorem 3.1 except for the parameters.

THEOREM 3.5. NspaceAmbiguity$(s(n), a(n)) \subseteq$ ReachUSPACE$(s(n) + \log a(n))$.

PROOF. First using FKS-hashing with $O(\log a(n))$ bit primes, we can show that $\mathsf{NspaceAmbiguity}(s(n), a(n))$ can be simulated in $\mathsf{DSPACE}(s(n) + \log a(n))$ using L_{ru} as an oracle (using Algorithm 1 on the configuration graph of $\mathsf{NspaceAmbiguity}(s(n), a(n))$ computation). Then, using identical arguments as in Lemma 2.7 (except for the parameters), it follows that $\mathsf{DSPACE}(s(n) + \log a(n))^{\mathsf{ReachUL}} \subseteq \mathsf{ReachUSPACE}(s(n) + \log a(n))$. □

Allender & Lange (1998) showed that $\mathsf{ReachUSPACE}(s(n)) \subseteq \mathsf{DSPACE}(\log^2 s(n)/\log\log s(n))$. Combining this result with the above upper bound, we get a class of graphs for which the reachability problem can be solved in deterministic space bound which is asymptotically better than Savitch's $O(\log^2 n)$ bound.

COROLLARY 3.6. *The s-t reachability problem in graphs where the number of paths from the start vertex to any other vertex is* $2^{o(\log n\sqrt{\log\log n})}$ *can be decided in* $\mathsf{DSPACE}(o(\log^2 n))$.

4. Discussion

Can we show that $\mathsf{FewL} = \mathsf{UL}$? Reinhardt & Allender (2000) showed that the reachability problem for graphs where there is a unique *minimum length* path from the source to any other vertex can be solved in UL. Given the configuration graph G of a FewL computation, the hashing lemma implies that there exists a small prime p so that in G_{w_p} all the paths from the start configuration to the accepting configuration will be of distinct weights. This implies that G_{w_p} have a unique minimum length path between this pair of configurations. However, the UL algorithm mentioned above requires that the input graph has a unique minimum length path from the start vertex to *any other vertex*, not just the terminating vertex. Managing this gap appears to be a serious technical difficulty for showing FewL=UL.

Acknowledgements

We thank Eric Allender for pointing to an error in an earlier version of the paper. We thank Tyler Seacrest for discussions in the Advanced Complexity course at UNL which led to the main result

in this paper. We thank the second reviewer for pointing out a subtle difference between the definitions of ReachUL (defined in Buntrock *et al.* 1991) and RUSPACE($\log n$) (defined in Lange 1997). We thank the reviewers for valuable comments that improved the presentation of the paper.

References

Eric Allender (2006). NL-printable sets and nondeterministic Kolmogorov complexity. *Theoretical Computer Science* **355**(2), 127–138.

Eric Allender & Klaus-Jörn Lange (1998). RUSPACE($\log n$) $\subseteq$ DSPACE($\log^2 n/\log\log n$). *Theory of Computing Systems* **31**, 539–550.

Eric Allender, Klaus Reinhardt & Shiyu Zhou (1999). Isolation, Matching, and Counting Uniform and Nonuniform Upper Bounds. *Journal of Computer and System Sciences* **59**(2), 164–181. ISSN 0022-0000.

Carme Àlvarez & Birgit Jenner (1993). A very hard log-space counting class. *Theoretical Computer Science* **107**, 3–30.

Sanjeev Arora & Boaz Barak (2009). *Computational Complexity - A Modern Approach*. Cambridge University Press. ISBN 978-0-521-42426-4.

Chris Bourke, Raghunath Tewari & N. V. Vinodchandran (2009). Directed Planar Reachability Is in Unambiguous Log-Space. *ACM Transactions on Computation Theory* **1**(1), 1–17.

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.

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

Klaus-Jörn Lange (1997). An Unambiguous Class Possessing a Complete Set. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS'97)*, 339–350.

A. Pavan, Raghunath Tewari & N. V. Vinodchandran (2010). On the Power of Unambiguity in Logspace To appear in Computational Complexity.

Klaus Reinhardt & Eric Allender (2000). Making nondeterminism unambiguous. *SIAM Journal on Computing* **29**(4), 1118 – 1131. ISSN 0097-5397.

Thomas Thierauf & Fabian Wagner (2009). Reachability in $K_{3,3}$-Free Graphs and K_5-Free Graphs Is in Unambiguous Log-Space. In *Proceedings of the 26th International Conference on Fundamentals of Computation Theory (FCT'09)*, 323–334.

Manuscript received 9 May 2011

Brady Garvin
Department of Computer Science and Engineering,
University of Nebraska-Lincoln,
Lincoln, NE 68588, USA.
bgarvin@cse.unl.edu

Derrick Stolee
Department of Computer Science and Engineering,
University of Nebraska-Lincoln,
Lincoln, NE 68588, USA.
dstolee@cse.unl.edu

Raghunath Tewari
Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur,
Kharagpur 721302, India.
raghunath@cse.iitkgp.ac.in

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