ReachFewL = ReachUL

Brady Garvin[1⋆], Derrick Stolee[1⋆⋆], Raghunath Tewari[1⋆⋆⋆], and N. V. Vinodchandran[1†]

Department of Computer Science
University of Nebraska–Lincoln
{bgarvin,dstolee,rtewari,vinod}@cse.unl.edu

Abstract. We show that two complexity classes introduced about two decades ago are 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.

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 conjucture NL = UL is true [ARZ99, RA00, BTV09, TW09]. However, researchers are 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 [BJLR91].

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 [Lan97]. In particular, Lange showed that the directed graph reachability problem associated with reach-unambiguous computations is

⋆ This author is supported in part by NSF grant CFDA#47.076 and AFOSR grant FA9550-10-1-0406.
⋆⋆ This author is supported in part by NSF grants CCF-0916525 and DMS-0914815.
⋆⋆⋆ This author is supported in part by NSF grant CCF-0916525.
† This author is supported in part by NSF grant CCF-0916525.

ReachUL-complete. Subsequently Allender and Lange 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 [AL98]. 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 [BJLR91, BDHM92, ÀJ93, BHS93, All06, PTV10]. 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 *polynomial* number of accepting computations. Thus FewL extends the class UL in a natural way. The analogus 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.

Theorem 1 (Main Theorem). ReachFewL = ReachUL

This theorem improves a recent upper bound that ReachFewL $\subseteq$ UL $\cap$ coUL shown in [PTV10]. We combine several existing techniques to prove our main result. In Section 2, we prove certain necessary results to prove the Theorem 1. In Section 3, we prove Theorem 1.

2 Definitions and Necessary Results

We begin by defining graph properties which characterize the configuration graphs of reach-unambiguous computations. Given a Turing machine M and an input x of M, let $G_{M,x}$ denote the configuration graph of M on x. Let $M(x)$ denote the computation of M on x.

Definition 1. *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.*

Definition 2. *A language L is in* ReachUL *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.*

Definition 3. *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.*

We now state certain critical properties of ReachUL that we use in the proof of Theorem 1. Lange proved that the associated graph reachability problem is complete for ReachUL [Lan97]. Define,

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

Theorem 2 ([Lan97]). *L_{ru} is complete for* ReachUL*.*

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 also need the fact that ReachUL is closed under complement [BJLR91].

Proposition 1 ([BJLR91]). ReachUL *is closed under complement.*

2.1 ReachUL as an Oracle

We first show that a log-space algorithm that queries a ReachUL language can be simulated in ReachUL. Given the fact that ReachUL is closed under complement, this is easy to prove. We give a sketch of the proof here.

Lemma 1. $\mathsf{L}^{\mathsf{ReachUL}} = \mathsf{ReachUL}$

Proof. 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.

2.2 Converting Graphs with a Few Paths to Distance Isolated Graphs

Definition 4. *Let G be a graph on n vertices and 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 $d \in \{1, \ldots, n\}$, there is at most one path of weight d from s to v.*

It is straight forward to extend this definition to graphs with positive integer 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 3 ([FKS84]). *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 $x \neq y \in S$ $x \not\equiv y \pmod{p}$.*

The next lemma follows easily from Theorem 3.

Lemma 2. *Let G be a graph on n vertices and 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.*

The graph G_{w_p} in Lemma 2 can be converted to an unweighted, distance isolated graph by replacing an edge having weight ℓ by a path of length ℓ.

2.3 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 5. *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 3. *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.

Theorem 4. 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 $G_{M,x}$ has at most n^c paths from the start configuration to any other configuration. Let s be the vertex corresponding to the start configuration and t be the vertex corresponding to the accepting configuration (without loss of generality we can assume that there is a single accepting configuration for a ReachFewL computation) in $G_{M,x}$. We need to decide whether there is a path from s to t.

The algorithm ReachFewSearch(G, s, t) given in Algorithm 1 is a log-space algorithm that queries the ReachUL complete languages L_{ru} defined in Section 2 and decides whether there is a path from s to t. This gives the inclusion ReachFewL $\subseteq \mathsf{L}^{\mathsf{ReachUL}}$. Since $\mathsf{L}^{\mathsf{ReachUL}}$ equals ReachUL by Lemma 1, the theorem follows. For the constant c, let c' be the constant given by Theorem 3.

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 ∈ {1, ..., n^{c'}} such that p is a prime do
    Define w_p(e_i) = 2^i (mod p);
    Construct G_{w_p};
    Construct lay(G_{w_p});
    foreach d ∈ {1, ..., |V(G_{w_p})|} do
        if ⟨lay(G_{w_p}), (s,0), (t,d)⟩ ∈ L_ru then return True;
    end
    return False;
end
return False;
```

Algorithm 1: ReachFewSearch(G, s, t)

We say that a prime p is *good* if G_{w_p} is distance isolated. By Lemma 2, 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 3. 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). So for this good prime $\langle lay(G_{w_p}), (s, 0), (t, d)\rangle \in L_{ru}$ and the algorithm accepts. 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.

4 Discussion

Allender and Lange showed that $\mathsf{ReachUL} \subseteq \mathrm{DSPACE}(\log^2 n/\log\log n)$ [AL98]. It is not clear how to directly extend this upper bound to ReachFewL. However our main result implies the same upper bound for the reachability problem associated with ReachFewL.

Corollary 1. *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* $O(\log^2 n/\log\log n)$.

Can we show that $\mathsf{FewL} = \mathsf{UL}$? Reinhardt and Allender [RA00] 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 pairs 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.

Acknowledgement

We would like to thank Eric Allender for pointing to an error in an earlier version of the paper. We would also like to thank Tyler Seacrest for discussions in the Advanced Complexity course which led to the result in this paper.

References

[ÀJ93] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoret. Comput. Sci.*, 107:3–30, 1993.

[AL98] Eric Allender and Klaus-Jörn Lange. $\mathsf{RUSPACE}(\log n) \subseteq \mathsf{DSPACE}(\log^2 n/\log\log n)$. *Theory of Computing Systems*, 31:539–550, 1998. Special issue devoted to the 7th Annual International Symposium on Algorithms and Computation (ISAAC'96).

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

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolation, Matching, and Counting Uniform and Nonuniform Upper Bounds. *Journal of Computer and System Sciences*, 59(2):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 Transactions on Computation Theory*, 1(1):1–17, 2009.

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

[Lan97] Klaus-Jörn Lange. An unambiguous class possessing a complete set. In *STACS '97: Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, pages 339–350, 1997.

[PTV10] A. Pavan, Raghunath Tewari, and N. V. Vinodchandran. On the power of unambiguity in logspace. 2010. To appear in Computational Complexity.

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

[TW09] Thomas Thierauf and Fabian Wagner. Reachability in $K_{3,3}$-free graphs and K_5-free graphs is in unambiguous log-space. In *FCT*, pages 323–334, 2009.