Electronic Colloquium on Computational Complexity, Report No. 154 (2010)

Space-Efficient Algorithms for Reachability in Surface-Embedded Graphs

Derrick Stolee*
Department of Computer Science
Department of Mathematics
University of Nebraska–Lincoln
dstolee@cse.unl.edu

N. V. Vinodchandran†
Department of Computer Science
University of Nebraska–Lincoln
vinod@cse.unl.edu

October 8, 2010

Abstract

We consider the reachability problem for a certain class of directed acyclic graphs embedded on surfaces. Let $\mathcal{G}(m, g)$ be the class of directed acyclic graphs with $m = m(n)$ source vertices embedded on a surface (orientable or non-orientable) of genus $g = g(n)$. We give a log-space reduction that on input $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m, g)$ and u and v are two vertices of G, outputs $\langle G', u', v'\rangle$ where G' is directed graph, and u', v' are vertices of G', so that (a) there is a directed path from u to v in G if and only if there is a directed path from u' to v' in G' and (b) G' has $O(m + g)$ vertices.

By a direct application of Savitch's theorem on the reduced instance we get a deterministic $O(\log n + \log^2(m + g))$-space algorithm for the reachability problem for graphs in $\mathcal{G}(m, g)$. By setting m and g to be $2^{O(\sqrt{\log n})}$ we get that the reachability problem for directed acyclic graphs with $2^{O(\sqrt{\log n})}$ sources embedded on surfaces of genus $2^{O(\sqrt{\log n})}$ is in L (deterministic logarithmic space). Earlier, in this direction, deterministic log-space algorithms were known only for *planar* directed acyclic graphs with $O(\log n)$ sources. Hence our result drastically improves the class of directed graphs for which we now know how to decide reachability in deterministic logarithmic space. By setting m and g to be $n^{o(1)}$ we get a deterministic algorithm for reachability for directed acyclic graphs embedded on surfaces with *sub-polynomial genus* and with *sub-polynomial number of sources*, that asymptotically beats Savitch's $O(\log^2 n)$ space bound.

Our reduction can also be combined with a simple depth-first search to achieve new simultaneous time-space upper bounds for reachability in a large class of directed acyclic graphs embedded on surfaces. In particular, for any $\epsilon < 1$, by performing a depth-first search on the reduced instance, we get a polynomial time algorithm for reachability over graphs in $\mathcal{G}(O(n^\epsilon), O(n^\epsilon))$ that uses $O(n^\epsilon)$ space. This beats the best upper bound of polynomial time and $O(n/2^{\sqrt{\log n}})$ space for this class of graphs known earlier.

*This author is supported in part by the NSF grants CCF-0916525 and DMS-0914815.
†This author is supported in part by the NSF grants CCF-0830730 and CCF-0916525.

ISSN 1433-8092

1 Introduction

Graph reachability problems are central to computational complexity theory. Different versions of this problem very closely characterize several important space complexity classes. The problem of deciding whether there is a path from a node u to v in a directed graph is the canonical complete problem for non-deterministic log-space (NL). Recent break-through result of Reingold implies that for undirected graphs, the reachability problem characterizes deterministic log-space computation (L) [Rei08]. It is also known that some restricted promise versions of the directed reachability problem characterize randomized log-space computations (RL) [RTV06]. Because of its fundamental role, designing space efficient algorithms for reachability problems is of great significance to complexity theory.

Prior Results

Savitch's $O(\log^2 n)$ space bound for the directed reachability problem [Sav70], Saks and Zhou's $O(\log^{3/2} n)$ bound for reachability problems characterizing RL computations [SZ99], and Reingold's log-space algorithm for the undirected reachability problem [Rei08] are the three most significant results in this topic. Clearly, designing algorithm for general reachability problem that beats Savitch's bound is one of the most important open questions in this area. While this appears to be a difficult problem, investigating classes of directed graphs for which we can design space efficient algorithms is an important research direction that appears to have promise. Recently, there has been progress reported along this theme. Jacoby, Liśkiewicz, and Reischuk, and, Jacoby and Tantau show that various reachability and optimization questions for *series-parallel* graphs admit deterministic log-space algorithms [JLR06, JT07]. Series-parallel graphs are a very restricted subclass of planar directed acyclic graphs (DAGs). In particular, such graphs have a single source and a single sink. Allender, Barrington, Chakraborty, Datta, and Roy [ABC$^+$09] extended Jacoby *et al.* 's result to show that the reachability problem for planar DAGs with single source and multiple sinks (SMPDs) can be decided in log-space. Building on this work, in [SBV10], the authors show that in fact, for planar DAGs with $O(\log n)$ sources, reachability can be decided in log-space. Investigating tree-width restricted graphs has also resulted in new space-efficient algorithms. Elberfeld, Jakoby, and Tantau present a log-space algorithm for reachability (and other problems) over graphs with constant tree-width [EJT10]. Another interesting class of graphs for which we know an algorithm that beats Savitch's bound is the class of *mangroves* – digraphs where every pair of vertices are connected by at most one path. Allender and Lange showed that reachability in mangroves can be solved in deterministic $O(\log^2 n/\log\log n)$ space [AL98].

Designing algorithms for reachability with simultaneous time and space bound is another important direction that has been of considerable interest in the past. Since a depth first search can be implemented in linear time and $O(n)$ space, the goal here is to improve the space bound while maintaining running time to be a polynomial. The most significant result here is Nisan's $O(\log^2 n)$ space, $n^{O(1)}$ time bound for RL [Nis95]. The best upper bound for general directed reachability is the $O(n/2^{\sqrt{\log n}})$ space, $n^{O(1)}$ time algorithm due to Barnes, Buss, Ruzzo and Schieber [BBRS92] However, these results were discovered nearly two decades ago and there appears to be not much recent progress reported on this topic. In this paper we give new bounds that beats the Barnes *et al.* bound for a large class of directed acyclic graphs.

Our Results

We consider the reachability problem over a large class of directed acyclic graphs embedded on surfaces. Since the graph is acyclic, there exist vertices with no incoming edge, called *sources*.

Define $n = n(G)$ to be the number of vertices in the input graph. Let $\mathcal{G}(m, g)$ denote the class of directed acyclic graphs (DAGs) with at most $m = m(n)$ source vertices embedded on a surface (orientable or non-orientable) of genus at most $g = g(n)$. Our main technical contribution is the following log-space reduction that *compresses* an instance of reachability for such surface-embedded DAGs.

Theorem 1.1. *There is a log-space reduction that given an instance $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m, g)$ and u, v vertices of G, outputs an instance $\langle G', u', v'\rangle$ where G is a directed graph and u', v' vertices of G', so that*

(a) there is a directed path from u to v in G if and only if there is a directed path from u' to v' in G',
(b) G' has $O(m + g)$ vertices.

Combining this reduction with known algorithms leads to many new space complexity upper bounds for the reachability problem.

By a direct application of Savitch's theorem on the reduced instance we get the following result.

Theorem 1.2. *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in deterministic $O(\log n + \log^2(m + g))$ space.*

By setting $m = g = 2^{O(\sqrt{\log n})}$ we get a deterministic log-space algorithm for reachability in DAGs with $2^{\sqrt{\log n}}$ sources embedded on a surface of genus $2^{\sqrt{\log n}}$. This drastically improves the class of graphs for which we now know how to decide reachability in deterministic logarithmic space.

Corollary 1.3. *The reachability problem for directed acyclic graphs with $2^{O(\sqrt{\log n})}$ sources embedded on surfaces of genus $2^{O(\sqrt{\log n})}$ can be decided in deterministic logarithmic space.*

A more relaxed setting of parameters leads to deterministic algorithms that asymptotically beat the Savitch's bound of $O(\log^2 n)$. By setting m and g to be $n^{o(1)}$ we get the following.

Corollary 1.4. *The reachability problem for directed acyclic graphs embedded on surfaces with* sub-polynomial genus *and with* sub-polynomial number of sources *can be decided in deterministic space $o(\log^2 n)$.*

Combining our reduction with a simple depth-first search gives us better simultaneous time-space bound for reachability over a large class of graphs than known before.

Theorem 1.5. *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in polynomial time using $O(\log n + m + g)$ space.*

In particular, for any $\epsilon < 1$, we get a polynomial time algorithm for reachability over graphs in $\mathcal{G}(n^\epsilon, n^\epsilon)$ that uses $O(n^\epsilon)$ space. This beats the Barnes *et al.* upper bound of polynomial time and $O(n/2^{\sqrt{\log n}})$ space for this class of graphs.

Corollary 1.6. *For any ϵ with $0 < \epsilon < 1$, the reachability problem for graphs in $\mathcal{G}(O(n^\epsilon), O(n^\epsilon))$ can be decided in polynomial time using $O(n^\epsilon)$ space.*

We note that the upper bound on space given in Theorem 1.5 can be slightly improved to $O\left(\frac{m+g}{2^{\sqrt{\log(m+g)}}}\right)$ by using Barnes *et al.* algorithm instead of the depth-first search, which will give $o(n^\epsilon)$ space bound in the above corollary.

Theorem 1.7. *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in polynomial time using $O\left(\log n + \frac{m+g}{2^{\sqrt{\log(m+g)}}}\right)$ space.*

Overview of the Construction

This main theorem is proven in several parts. In Section 2, we present a simple structural decomposition called the *forest decomposition* of the given directed acyclic graph. Based on this decomposition, we classify the edges as local and global. Reachability using only local edges can be decided in log-space. In order to pinpoint how the global edges interact, we define the notion of *topological equivalence* among global edges. We show that the number of possible equivalence classes is bounded by $O(m + g)$ where m is the number of sources and g is the genus of the surface. Then, Section 3 describes a finite list of *patterns* that characterize how paths pass through these equivalence classes. We also analyze the structure of these patterns. In particular, for each pattern type we identify a pair of log-space computable edges in the corresponding equivalence class that has certain canonical properties. In Section 4, we describe a graph $\mathrm{P}(G, F)$ on $O(m+g)$ vertices (where F is the forest decomposition) called the *pattern graph* whose vertices are described by (pattern-equivalence class) pairs. We finally show that this pattern graph is computable in log-space and preserves reachability between a given pair of vertices.

Before we begin, we note that through out this paper certain known log-space primitives are frequently used as subroutines without explicit reference to them. In particular, Reingold's log-space algorithm for undirected reachability is often used, for example to identify connected components in certain undirected graphs.

2 Preliminaries

We mainly deal with directed graphs. A directed edge $e = xy$ has the direction from x to y and we call x the *tail* and y the *head*. Sometimes we use the notation $x = \mathrm{Tail}(e)$, and $y = \mathrm{Head}(e)$.

We assume that G is a graph and the embedding is given as part of the input. If G is planar, this can be detected in log-space and the embedding can be generated using a log-space reduction. We assume that the input graph G is embedded on a surface S where every face is homeomorphic to an open disk. Such embeddings are called *2-cell embeddings*. We also assume that the graph is presented as a *combinatorial embedding* where for each vertex v the circular ordering of the edges incident to v is specified. In the case of a non-orientable surface, the signature of an edge is also given, specifying if the orientation of the rotation switches across this edge.

Let G be a graph with n vertices and e edges embedded on a surface S with f faces, then by the well known *Euler's Formula* we have $n - e + f = \chi_S$, where χ_S is the Euler characteristic of the surface S. The number of faces in a graph is log-space computable from a combinatorial embedding (for a proof, see [KV10]), so χ_S is also computable in log-space. The genus g_S of the surface S is given by the equation $\chi_S = 2 - 2g_S$ for orientable surfaces and $\chi_S = 2 - g_S$ for non-orientable surfaces.

Let C be a simple closed curve on S given by a cycle in the underlying undirected graph of G. C is called *surface separating* if the removal of C disconnects S. A surface separating curve C is called *contractible* if removal of C disconnects S where at least one of the connected components is homeomorphic to a disc. Given a cycle C it is possible to detect the type of C in log-space (for example by using log-space algorithm for undirected reachability to find the connected components, then calculating the Euler characteristic for each component).

We assume that the given graph is acyclic. Lemma 2.1 gives a technique for converting a source-bounded reachability algorithm on graphs promised to be acyclic into a cycle-detection algorithm without asymptotically increasing the space requirement.

Lemma 2.1. *Let $s(n, m, g) = \Omega(\log n)$. If there exists an $O(s(n, m, g))$-space bounded algorithm for testing uv-reachability over graphs in $\mathcal{G}(m, g)$ then there exists an $O(s(n, m, g))$-space bounded algorithm to test if a graph is acyclic, given that it has at most m sources and is embedded in a surface of genus at most g.*

Proof. Let $A(G, u, v)$ be the algorithm for testing uv-reachability on $G \in \mathcal{G}(m, g)$. Fix an incoming edge at each non-source vertex, making a set $F \subseteq E(G)$. By taking reverse walks from each vertex, it can be verified that F has no cycles.

Order the edges $E(G)$ as $\{e_1, \ldots, e_{|E(G)|}\}$. For each $i \in \{0, 1, \ldots, |E(G)|\}$, let G_i be the subgraph of G where an edge e_j is present in G_i if $e_j \in F$ or $j \leq i$. Iterate through all such i and test if $A(G_i, \text{Head}(e_{i+1}), \text{Tail}(e_{i+1}))$ ever returns with success. If any returns True, then there is a cycle including the edge e_{i+1}. Note that A gives the correct response, since G_0 was cycle free and by iteration, G_i is cycle free. Each G_i is acyclic for $i \in \{1, \ldots, |E(G)|\}$ if and only if G is acyclic and all queries $A(G_i, \text{Head}(e_{i+1}), \text{Tail}(e_{i+1}))$ return False. □

Forest Decomposition, Edge Classification, and Topological Equivalence

A simple structural decomposition, called a *forest decomposition*, of a directed acyclic graph forms the basis of our algorithm. This forest decomposition has been utilized in previous works [ABC$^+$09, SBV10].

Let G be a directed acyclic graph and let u, v be two vertices. Our goal is to decide whether there is directed path from u to v. Let $u, s_1, \ldots, s_m$ be the sources of G. If u is not a source, make it a source by removing all the incoming edges. This will not affect uv-reachability, increases the number of sources by at most one, and only reduces the genus of the embedding.

Definition 2.2 (Forest Decomposition)**.** Let A be a deterministic log-space algorithm that on input of a non-source vertex x, outputs an incoming edge yx (for example, selecting the lexicographically-first vertex y so that yx is an edge in G). This algorithm defines a set of edges

$$F_A = \{yx : x \in V(G) \setminus \{u, v, s_1, \ldots, s_m\},\ y = A(x)\},$$

called a *forest decomposition* of G.

Note that since G is acyclic, the reverse walk $x_1, x_2, \ldots$, where $x_1 = x$ and $x_{i+1} = A(x_i)$, must terminate at a source s_j, u, or v, so the edges in F_A form a forest subgraph. For the purposes of the forest decomposition, v is treated as a source since no incoming edge is selected. If a vertex x is in the tree with source v, then all non-tree edges entering x are deleted. This does not affect uv-reachability, since G is acyclic and does not increase the number of sources or the genus of the surface. Each connected component in F_A is a tree rooted at a source vertex, called a *source tree*. The forest forms a typical *ancestor* and *descendant* relationship within each tree.

For the remainder of this work, we fix an acyclic graph $G \in \mathcal{G}(m, g)$ embedded on a surface S (defined by the combinatorial embedding) and $F = F_A$ a log-space computable forest decomposition.

Definition 2.3 (Tree Curves)**.** Let x and y be two vertices in some source tree T of F. The *tree curve* at xy is the curve on S formed by the unique undirected path in T from x to y. If xy is an edge, then the closed curve formed by xy and the tree curve at xy is called the *closed tree curve* at xy.

Edges in G that are not included in F can be partitioned into two classes, *local* and *global*. We use this classification to create subgraphs of G which are locally embedded on a disk.

Definition 2.4 (Local and Global Edges)**.** Given an S-embedded graph G and a forest decomposition F, an edge xy in $E(G) \setminus F$ is classified as

- *local*[1] if the following three conditions hold:
  1. x and y are on the same tree in F.
  2. The closed tree curve at xy is contractible (i.e. the curve cuts S into a disk and another surface).
  3. No sources lie on the interior of the surface which is homeomorphic to a disk.

  If S is the sphere, then the curve cuts S into two disks and xy is local if one of the disks contains no source in the interior.
- *global* otherwise.

Note that the subgraph of G given by the edges in F and the local edges has each connected component a DAG with a single source.

Definition 2.5 (The Region of a Source Tree)**.** Let T be a connected component in the forest decomposition F along with the local edges between vertices in T. The *region* of T, denoted $\mathcal{R}[T]$ is the portion of the surface S given by the faces enclosed by the tree and local edges in T.

The faces that compose $\mathcal{R}[T]$ form a disk, since $\mathcal{R}[T]$ can contract to the source vertex by contracting the disks given by the local edges into the tree, and then contracting the tree into the source vertex. This disk is oriented using the combinatorial embedding at the source by the right-hand rule. Reachability in such subgraphs T can be decided using the SMPD algorithm [ABC$^+$09], in log-space. Note that the restriction of a 2-cell embedding implies all global edges are incident to vertices on the outer curve of the disk $\mathcal{R}[T]$. Our figures depict source trees as circles, with the source placed in the center, with tree edges spanning radially away from the source[2].

The following notion of topological equivalence plays a central role in our algorithms. It was originally presented in [SBV10] for planar graphs, but we extend it to arbitrary surfaces.

Definition 2.6 (Topological Equivalence)**.** Let G be a graph embedded on a surface S. Let F be a forest decomposition of G. We say two (undirected) global edges xy and wz are *topologically equivalent* if the following two conditions are satisfied:

- They span the same source trees in F (assume x and w are on the same tree).
- The closed curve in the underlying undirected graph formed by (1) the edge xy, (2) the tree curve from y to z, (3) the edge zw, and (4) the tree curve from w to x bounds a connected portion of S, denoted $D(xy, wz)$, that is homeomorphic to a disk and no source lies within $D(xy, wz)$.

Topological equivalence is an equivalence relation. For the sake of the reflexive property, we take as convention that a single edge is topologically equivalent to itself. The symmetry property is implied by the symmetry of the definition. For transitivity, consider the following lemma.

[1] This definition of *local* differs from the use in [ABC$^+$09] and [SBV10].

[2] This visualization of source trees was crucial to the development of this work, and is due to [ABC$^+$09].

Lemma 2.7. *Let e_1, e_2 be topologically equivalent global edges and e_3 a global edge.*

1. *If e_3 has an endpoint in $D(e_1, e_2)$, then e_3 is equivalent to both e_1 and e_2.*
2. *If e_3 is equivalent to e_2, then one of the following cases holds:*
   (a) *e_1 is in $D(e_2, e_3)$.*
   (b) *$D(e_1, e_2)$ and $D(e_2, e_3)$ intersect at the curve given by e_2 and the ancestor paths from its endpoints to their respective sources, and $D(e_1, e_3) = D(e_1, e_2) \cup D(e_2, e_3)$.*

   In both cases (a) and (b), e_1 is topologically equivalent to e_3.

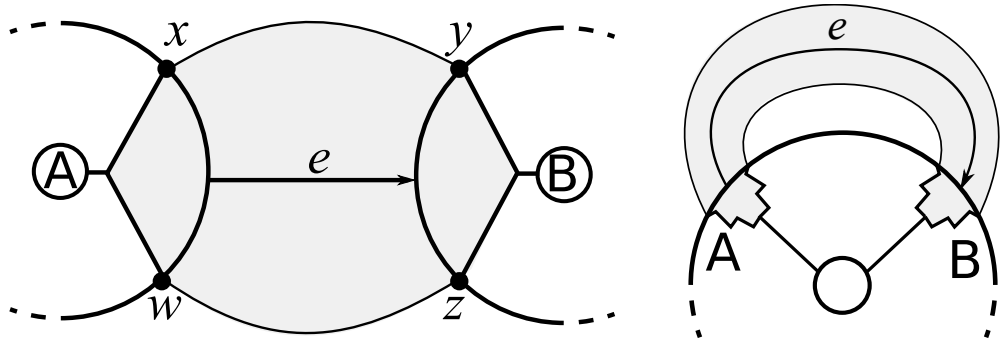
Let E' be an equivalence class of global edges containing an edge e, where e spans two different source trees. Consider the subgraph of G given by the vertices in the source trees containing the endpoints of e, along with all local edges in those trees and the edges in E'. This subgraph is embedded in a disk on S. We shall make explicit use of this locally-planar embedding.

For an equivalence class of global edges spanning vertices in the same tree, a similar subgraph and embedding is formed by considering the ends of the equivalence class to be different copies of that source tree.

The lexicographically-least edge e in a topological equivalence class of global edges is log-space computable. By counting how many global edges which are lexicographically smaller than e and are the lexicographically-least in their equivalence classes, the equivalence class containing e is assigned an index i. The class E_i is the ith equivalence class in this ordering. We shall use this notation to label the equivalence classes.

Definition 2.8 (The Region of an Equivalence Class)**.** Let E_i be an equivalence class of global edges. Define the *region enclosed by E_i* as

$$\mathcal{R}[E_i] = \bigcup_{e_1, e_2 \in E_i} D(e_1, e_2).$$



(a) A region of edges spanning different trees.

(b) A region of edges spanning a single tree.

Figure 1: Regions of an equivalence class. The representative edge e defines the A-end and B-end of $\mathcal{R}[E_i]$.

The region $\mathcal{R}[E_i]$ has some properties which are quickly identified.

1. Two regions $\mathcal{R}[E_i]$ and $\mathcal{R}[E_j]$ on different classes E_i and E_j intersect only on the boundary paths. The vertices on the boundary are not considered *inside* the region, since they may be in multiple regions.

2. There are two edges $e_a, e_b \in E_i$ so that $\mathcal{R}[E_i] = D(e_a, e_b)$. These outer edges define the *sides* of $\mathcal{R}[E_i]$.
3. The *boundary* of $\mathcal{R}[E_i]$ is given by these two edges and their ancestor paths in F on all four endpoints.
4. Let T_A and T_B be the two source trees containing the tail and head, respectively, of the representative edge in E_i. The vertices in $\mathcal{R}[E_i]$ are partitioned into two *ends*, A and B, where the vertices are placed in an end determined by containment in $\mathcal{R}[T_A] \cap \mathcal{R}[E_i]$ and $\mathcal{R}[T_B] \cap \mathcal{R}[E_i]$ when there trees T_A and T_B different or by the two connected components of $\mathcal{R}[T_A] \cap \mathcal{R}[E_i]$ when the trees T_A and T_B are equal.
5. There is an ordering $e_a = e_1, e_2, \ldots, e_k = e_b$ of E_i so that the endpoints of the e_j on the A-end appear in a clockwise order in that tree.

Since global edges appear on the boundary of $\mathcal{R}[T]$ for a given source tree T, there is a natural clockwise ordering on these edges, with respect to the orientation of T. Further, we can order the incident equivalence classes (with possibly a single repetition, in the case of global edges with both endpoints in T) by the clockwise order the ends $\mathcal{R}[E_i] \cap \mathcal{R}[T]$ appear on the boundary of $\mathcal{R}[T]$.

The resource bounds we prove directly depends on the number of equivalence classes. The following lemma bounds the number of equivalence classes.

Lemma 2.9. *Let G be a graph embedded on a surface S with Euler characteristic χ_S with a forest decomposition F with m sources. There are at most $3(m + |\chi_S|)$ topological equivalence classes of global edges. If g_S is the genus of S, $|\chi_S| = O(g_S)$ and there are $O(m + g_S)$ equivalence classes of global edges.*

Proof. Consider a graph G which has a maximal number of equivalence classes and remove all but one representative of each class. Create a new multigraph H on the m sources with edges given by the representatives of each class, with the edges embedded in S by following the undirected path composed of the tree path from the first source to the edge, the edge, then the tree path from the edge to the second source. There are m vertices, and let e be the number of edges, f the number of faces. Subdivide these edges twice to get a simple graph embedded in S. Note that Euler's formula holds in this graph on $m + 2e$ vertices, $3e$ edges, and f faces. Hence,

$$\begin{aligned}\chi_S &= (m + 2e) - (3e) + f \\ &= m - e + f.\end{aligned}$$

Moreover, each face must have at least three equivalence classes, and each edge is incident to two faces, so $2e \leq 3f$ and $f \leq \frac{2}{3}e$. This gives

$$\begin{aligned}\chi_S = m - e + f &\leq m - \frac{1}{3}e \\ \Rightarrow e &\leq 3m - 3\chi_S \leq 3(m + |\chi_S|).\end{aligned}$$

□

An important consequence of this lemma is that it requires only $O(\log(m + g_S))$ bits to store the binary representation of the index i of an equivalence class E_i.

Remark. Now that all tree and local edges are embedded in disks of the form $\mathcal{R}[T]$ and global edges are in $O(m + g)$ disks of the form $\mathcal{R}[E_i]$, we are able to abandon all other portions of S. The important information from S is that the ends of regions incident to a given source tree appear

in a clockwise order on the boundary of $\mathcal{R}[T]$ and that there are $O(m+g)$ equivalence classes of global edges. Each source tree looks like a disk ($\mathcal{R}[T]$) with strips ($\mathcal{R}[E_i]$ for incident classes E_i) stretching radially away from it (as long as the other end of the strip $\mathcal{R}[E_i]$ is not considered).

In the next section, we investigate how global edges are used by paths in G by utilizing this locally-planar embedding. The paths are broken down into parts where only local paths and equivalent global edges are used. Each time global edges in a class E_i are used, we focus on two disks $\mathcal{R}[T_A]$ and $\mathcal{R}[T_B]$ connected by the strip $\mathcal{R}[E_i]$.

3 Patterns in Equivalence Classes

If a directed path P from u to v exists in G, then P leaves the tree rooted at u and travels through the other source trees and global edge equivalence classes before taking a global edge to v. The crucial observation to this work is the following: If we focus on certain "nice" paths, then there are a finite number of ways such paths can enter and exit the region of an equivalence class. Thus for each equivalence class E_i, there are a finite number of "patterns" we need to consider. In this section we formalize these notions.

First we define what we mean by a "nice" path.

Definition 3.1 (Irreducible Paths)**.** Let G be a DAG and F be a forest decomposition of G. Let $P = x_1, \ldots, x_k$ be a directed path in G. P is said to be *irreducible* if whenever a vertex x_i appears before x_j in P and x_j is a descendant of x_i in some tree of F, then P follows the edges in F from x_i to x_j.

Note that if a path exists between two vertices, an irreducible path also exists, by swapping violating subpaths with the appropriate tree paths.

Irreducible paths that use only local edges travel clockwise or counter-clockwise, depending on the orientation of the tree. We associate *right* and *left* with these rotational directions, respectively.

Definition 3.2. Consider an equivalence class E_i between source trees T_A and T_B and a vertex x in T_A, outside the region $\mathcal{R}[E_i]$.

- The vertex x *fully reaches* E_i if there is a local path from x to the corresponding endpoint of each edge in E_i.
- If x does not fully reach E_i, but there is a local path from x to the corresponding endpoint of some edge of E_i, then x *partially reaches* E_i.
- If a path given above is irreducible, then the path follows a clockwise or counter-clockwise direction within T_A. Then, x fully (partially) reaches E_i *using a clockwise (counter-clockwise) rotation.*

Lemma 3.3. *Let x be a vertex in a source tree T_A. For each rotational direction (clockwise or counter-clockwise), there is an ordering $E_{i_0}, E_{i_1}, \ldots, E_{i_\ell}$ of the edge classes reachable via irreducible paths in that direction so that x fully reaches each E_{i_j} for $j \in \{1, \ldots, \ell-1\}$, x either fully or partially reaches E_{i_0} and E_{i_ℓ}, and if x is not in the interior of $\mathcal{R}[E_{i_0}]$, x fully reaches E_{i_0}.*

Proof. Construct the list using all reachable classes in the given rotational direction and order by their appearance. The irreducible path P from x to the class E_{i_ℓ} must intersect the tree paths from the source to the edges in each class E_{i_j} for all $j < \ell$, with $x \notin \mathcal{R}[E_{i_j}]$, since the edges in P lie in $\mathcal{R}[T]$, but the endpoints of the edges in E_{i_j} are on the boundary of $\mathcal{R}[T]$. Hence, x fully reaches these classes. □

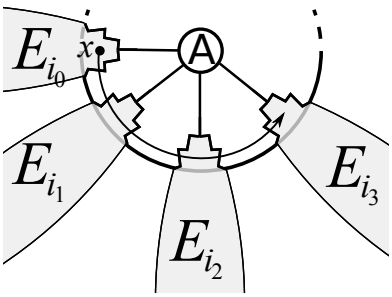


Figure 2: A vertex x with three counter-clockwise reachable classes, E_{i_1}, E_{i_2}, and E_{i_3}, as in Lemma 3.3.

Definition 3.4 (Induced Class List)**.** Each irreducible path P between two vertices x and y induces a list of edge classes $E_{i_1}, \ldots, E_{i_\ell}$ for some ℓ where the global edges of P visit each class E_{i_j} in order of increasing j, and $E_{i_j} \neq E_{i_{j+1}}$ for each $j \in \{1, \ldots, \ell - 1\}$. This list is the *induced class list* for the path P.

Once the induced class list is known, the path P must take local paths between edges in E_i and edges in E_{i+1}. This direction is determined by the orientation of the path, which is inherited from x as if a normal vector were being pushed along the surface of S on the curve of P. Since P is irreducible, these local paths have a clockwise or counterclockwise direction, determined by the orientation of the path (which may agree or disagree with the orientation of the current tree). While this local path traverses from E_i to E_{i+1}, it must cross the boundary of $\mathcal{R}[E_i]$ and the boundary of $\mathcal{R}[E_{i+1}]$ at an ancestor path of a boundary edge. There are only two possible ends (A or B) these local paths can start and end, only two possible rotational directions (R and L for *right* and *left*), and two possible orientations (+ or − with respect to the current tree). Note that since P is irreducible, P does not leave $\mathcal{R}[E_i]$ at the same boundary that it entered $\mathcal{R}[E_i]$.

For each edge class E_i in the induced class list of P, there are *induced patterns* which are given by the entrance end of $\mathcal{R}[E_i]$, the orientation of P with respect to the source tree on the entrance end, the rotational direction of the local path at the entrance, the exit end of $\mathcal{R}[E_i]$, and the rotational direction of the local path at the exit. These items together form a *pattern description*. Specifically, the entrance direction, the exit end, and the exit direction combine into a *pattern* which has the most important information about the behavior of P within E_i.

Definition 3.5. Let E_i be a class of global edges. An irreducible path P that involves an edge of the class E_i *induces* a pattern on E_i defined by the following cases[3]:

Full Patterns:

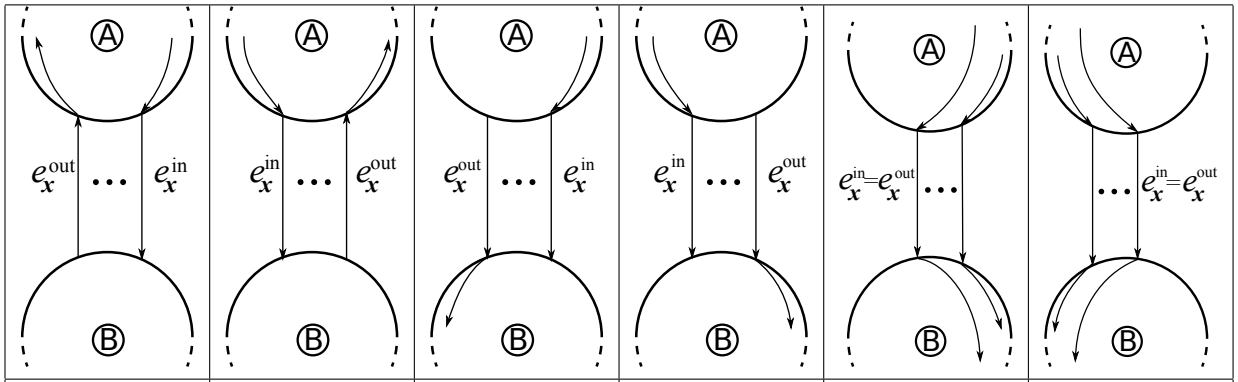
- ⟨RR⟩ enters $\mathcal{R}[E_i]$ via a clockwise path and exits via a clockwise path on the same end.
- ⟨LL⟩ enters $\mathcal{R}[E_i]$ via a counter-clockwise path and exits via a counter-clockwise path on the same end.
- ⟨RXL⟩ enters $\mathcal{R}[E_i]$ via a clockwise path and exits via a counter-clockwise path on the opposite end.
- ⟨LXR⟩ enters $\mathcal{R}[E_i]$ via a counter-clockwise path and exits via a clockwise path on the opposite end.

[3]The interested reader will find the notation for patterns derived from move sequences in the Coin Crawl Game from [SBV10].

Nesting Patterns:

- ⟨RXR⟩ enters $\mathcal{R}[E_i]$ via a clockwise path and exits via a clockwise path on the opposite end.
- ⟨LXL⟩ enters $\mathcal{R}[E_i]$ via a counter-clockwise path and exits via a counter-clockwise path on the opposite end.

The names *full* and *nesting* refer to specific properties — properties which are investigated in following sections — that are revealed when paths attempt to induce these patterns. See Table 1 for figures of these patterns.



⟨RR⟩	⟨LL⟩	⟨RXL⟩	⟨LXR⟩	⟨RXR⟩	⟨LXL⟩
Full Patterns				*Nesting Patterns*	

Table 1: Different patterns using an edge class E_i, entering from the A-end of $\mathcal{R}[E_i]$. The notation $e_{\mathbf{x}}^{\text{in}}$ and $e_{\mathbf{x}}^{\text{out}}$ refers to Definition 3.7.

For future reference, define the *pattern set*, $\mathcal{P}$, as

$$\mathcal{P} = \{\langle \text{RR}\rangle, \langle \text{LL}\rangle, \langle \text{RXR}\rangle, \langle \text{RXL}\rangle, \langle \text{LXR}\rangle, \langle \text{LXL}\rangle\}.$$

Definition 3.6 (Entrance and Exit)**.** Given an edge class E_i the region $\mathcal{R}[E_i]$ has a boundary given by ancestor paths of the outer edges $e_a, e_b \in E_i$. Let t be an end of $\mathcal{R}[E_i]$ (either A or B) and fix an orientation on that end and a pattern p.

- the *entrance* of the pattern is the ancestor path on the boundary of $\mathcal{R}[E_i]$ on the t-end that a path must cross before using the edges in E_i that induce the pattern p with the given orientation.
- the *exit* of the pattern is the ancestor path on the boundary of $\mathcal{R}[E_i]$ on the t-end that a path must cross after using the edges in E_i that induce the pattern p with the given orientation.

See Figure 3 for a visual representation of the entrance and exit of a pattern, along with some modifiers to help describe different parts of the region $\mathcal{R}[E_i]$.

The main difference between full and nesting patterns is that nesting patterns have the entrance and exit on the same side of the region, while full patterns involve both sides of the region. A path through the region from one side to the other will force all edges in the class to be reachable.

Combining Definition 3.4 and Definition 3.5, we see that each irreducible path P from u to v induces a list of edge classes $E_{i_1}, E_{i_2}, \ldots, E_{i_\ell}$ paired with the patterns P takes through the classes.

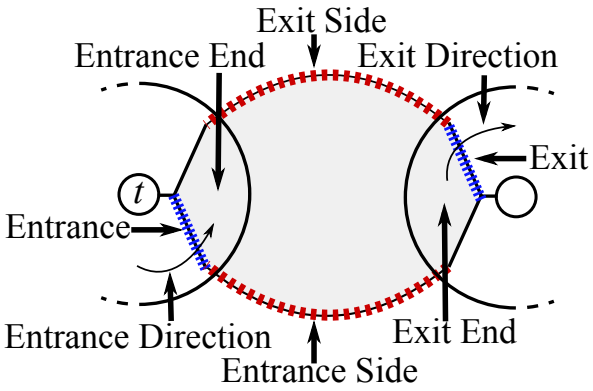


Figure 3: Terminology for the entrance and exit of a pattern and the modifiers of *direction*, *end*, and *side*. This example is an ⟨LXR⟩ pattern.

The reduction in the next section takes the graph G and the forest decomposition F and outputs a graph $\mathrm{P}(G, F)$ with vertices u', v', and vertices corresponding to patterns on global edge classes. These vertices come from the set

$$V_{\mathrm{P}} = \{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P},$$

where k is the number of equivalence classes in G. A vertex $(i, t, o, p) \in V_{\mathrm{P}}$ is a *pattern description* and has an interpretation as:

1. $i \in \{1, \ldots, k\}$ gives the equivalence class E_i.
2. $t \in \{A, B\}$ gives the end of $\mathcal{R}[E_i]$ that contains the entrance.
3. $o \in \{+1, -1\}$ specifies if the orientation of the path is in agreement with (or opposite to) the local orientation of the tree on the t-side of E_i. The rotation directions of the pattern depend on this orientation of the path.
4. $p \in \mathcal{P}$ gives the pattern used in E_i.

For example, the vertex $(i, B, +1, \langle\mathrm{RXL}\rangle)$ is the vertex in V_{P} corresponding to a ⟨RXL⟩ pattern crossing the class E_i starting at the B-side and leaving the A-side, oriented to agree with the B-side. Note that it requires $O(\log(m + g_S))$ bits to store the label of a pattern description.

First, we must investigate some properties of paths that induce these patterns. We focus on a path which uses local paths and edges in a single equivalence class of global edges and induces a single pattern on that class. These single-pattern paths will be concatenated to make larger paths once the structure of the shorter paths is understood.

An important property of these patterns is that if the pattern is of full type or the equivalence class is fully reachable, we can assume without loss of generality that the path used two special edges, which we call the *canonical edge pair*.

Definition 3.7 (Canonical Edge Pairs)**.** Let $\mathbf{x} = (i, t, o, p) \in V_{\mathrm{P}}$ be a pattern description centered at the edge class E_i with pattern p. There are two edges (*incoming* and *outgoing*) in E_i, called the *canonical edge pair* for $\mathbf{x}$, defined as

- The *outgoing edge*, $e^{\mathrm{out}}_{\mathbf{x}}$, is the edge $e \in E_i$ that is farthest from the exit side so that there exists a local path from $\mathrm{Head}(e)$ to the exit of $\mathcal{R}[E_i]$.

- The *incoming edge*, $e^{\text{in}}_{\mathbf{x}}$, is the edge $e \in E_i$ that is closest to the entrance side so that $\text{Tail}(e^{\text{out}}_{\mathbf{x}})$ is reachable from $\text{Head}(e)$ using local paths and edges in E_i.

The Structure of Full Patterns

Full patterns are named because they can only be used if every single edge in the equivalence class is eventually reachable. This gives us complete knowledge on when these patterns can be induced, based on reachability using the canonical edge pair. The following lemmas describe these properties.

Lemma 3.8. *Let* $\mathbf{x}$ *be a pattern description of full type. The canonical edge pair* $(e^{\text{in}}_{\mathbf{x}}, e^{\text{out}}_{\mathbf{x}})$ *is log-space computable.*

Proof. The outgoing edge, $e^{\text{out}}_{\mathbf{x}}$, is computed by enumerating the set of edges in the class E_i with head on the exit end of $\mathcal{R}[E_i]$ which reach the boundary of the region $\mathcal{R}[E_i]$ using local edges in the exit direction of the pattern.

The incoming edge is computed by an iterative procedure. Store two edge pointers, e_1 and e_2. These edges will always be in the class E_i or null. The edge e_1 will have tail in the entrance end of $\mathcal{R}[E_i]$ and e_2 with have tail in the exit end of $\mathcal{R}[E_i]$. Initialize $e_1 = e^{\text{out}}_{\mathbf{x}}$ and set e_2 to be null.

Proceed by iterating through the edges in E_i starting at $e^{\text{out}}_{\mathbf{x}}$ to the last edge in E_i on the entrance side of $\mathcal{R}[E_i]$. Each edge is a candidate to update e_1 and e_2.

If the tail is in the entrance side of $\mathcal{R}[E_i]$, check if the head reaches the tail of e_2 or $e^{\text{out}}_{\mathbf{x}}$ using a local path. If so, then update e_1 to this edge.

If the tail is in the exit side of $\mathcal{R}[E_i]$, check if the head reaches the tail of e_1 or $e^{\text{out}}_{\mathbf{x}}$ using a local path. If so, then update e_2 to this edge.

After all edges have been tested, set $e^{\text{in}}_{\mathbf{x}} = e_1$. There is a path from e_1 to $e^{\text{out}}_{\mathbf{x}}$ using local paths and edges in E_i by considering the reverse sequence of e_1 and e_2 updates that allowed $\text{Tail}(e^{\text{out}}_{\mathbf{x}})$ to be reachable from $\text{Head}(e_1)$. Further, no edge beyond e_1 in the proper direction can reach $e^{\text{out}}_{\mathbf{x}}$ because it must cross the ancestor paths from e_1 to the sources on each endpoint. □

Lemma 3.9. *Let* $\mathbf{x}$ *be a pattern description of full type centered at an edge class* E_i*. Let* $y, z \in V(G)$ *be vertices not inside* $\mathcal{R}[E_i]$*, where* y *is in the source tree on the entrance end of* $\mathbf{x}$ *and* z *is in the source tree on the exit end of* $\mathbf{x}$*.*

There is a path from y *to* z *in* G *using only local paths and edges of the class* E_i *that induces the pattern* $\mathbf{x}$ *if and only if* $\text{Tail}(e^{\text{in}}_{\mathbf{x}})$ *is reachable from* y *using a local path in the entrance direction of* $\mathbf{x}$ *and* z *is reachable from* $\text{Head}(e^{\text{out}}_{\mathbf{x}})$ *using a local path in the exit direction of* $\mathbf{x}$*.*

Proof. Note that if the tail of $e^{\text{in}}_{\mathbf{x}}$ is reachable from y using a local path in the entrance direction, and z is reachable from the head of $e^{\text{out}}_{\mathbf{x}}$ using a local path in the exit direction, then there is a path from y to z that induces the pattern $\mathbf{x}$ using the path between $e^{\text{in}}_{\mathbf{x}}$ and $e^{\text{out}}_{\mathbf{x}}$ given by the definition of the canonical pair.

If a path exists from y to z that induces the pattern $\mathbf{x}$, then there is at least one edge of the class E_i in the path. Let e_1 be the first edge of class E_i used in the path and e_2 be the last. Consider where e_1 and e_2 are in comparison to the canonical pair $(e^{\text{in}}_{\mathbf{x}}, e^{\text{out}}_{\mathbf{x}})$ in the ordering of the edges in E_i. An example of the edges e_1 and e_2 are shown in Figure 4.

If e_1 is closer to the entrance side of E_i compared to $e^{\text{in}}_{\mathbf{x}}$, then (by the definition of $e^{\text{in}}_{\mathbf{x}}$) there is no path from the head of e_1 to the tail of $e^{\text{out}}_{\mathbf{x}}$ using local paths and edges in E_i. Hence, a path from e_1 that leaves $\mathcal{R}[E_i]$ in the exit direction can not cross the ancestor path of the tail of $e^{\text{out}}_{\mathbf{x}}$,

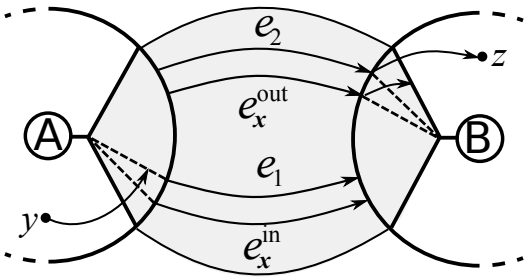


Figure 4: The edges used in the proof of Lemma 3.9 in an ⟨LXR⟩ pattern.

so it must cross the ancestor path of the head of $e_{\mathbf{x}}^{\text{out}}$. This implies there is an edge e in E_i in the direction of $e_{\mathbf{x}}^{\text{out}}$ that is farther from the exit direction and whose head reaches the head of $e_{\mathbf{x}}^{\text{out}}$. This contradicts the definition of $e_{\mathbf{x}}^{\text{out}}$, since there is now a local path from the head of e_1 that reaches the boundary of $\mathcal{R}[E_i]$ in the exit direction.

Therefore, the edge e_1 appears after $e_{\mathbf{x}}^{\text{in}}$ in the order on E_i starting from the entrance side. This implies that y has a local path that crosses the ancestor path from the tail of $e_{\mathbf{x}}^{\text{in}}$ and hence reaches the tail of $e_{\mathbf{x}}^{\text{in}}$. If $e_{\mathbf{x}}^{\text{out}}$ is on the exit side of E_i compared to e_2, then by the definition of $e_{\mathbf{x}}^{\text{out}}$, there is no local path from the head of e_2 that reaches the boundary of $\mathcal{R}[E_i]$ in the exit direction. So, e_2 is on the exit side of E_i compared to $e_{\mathbf{x}}^{\text{out}}$. The local path that reaches the boundary of $\mathcal{R}[E_i]$ from the head of $e_{\mathbf{x}}^{\text{out}}$ crosses the ancestor path to the head of e_2, so z is reachable from the head of $e_{\mathbf{x}}^{\text{out}}$ using a local path. □

The Structure of Nesting Patterns

Nesting patterns are named because exactly one edge is used in an irreducible path that induces a nesting pattern, and we may assume that the edge used is the one farthest from the entrance that is reachable (and that a local path exists from its head to the exit). The following lemmas describe these properties.

Lemma 3.10. *If an irreducible path using local paths and edges in a global edge class E_i induces a nesting pattern, then the path uses exactly one edge in the class E_i.*

Proof. Let x and y be vertices outside E_i with a path from x to y that induces a nesting pattern on E_i. Let e_1 be the first edge in E_i used and e_2 be the second. Note that e_2 cannot be closer to the entrance direction than e_1, or else the head of e_2 is a descendant of the local path from x to the tail of e_1, contradicting irreducibility. Also, e_2 cannot be farther from the entrance direction than e_1 or else the path from the head of e_2 to y must cross the ancestor path at the head of e_1, creating a cycle, contradicting that the graph is acyclic. □

Lemma 3.11. *Let* $\mathbf{x}$ *be a pattern description of nesting type centered at a global edge class E_i. Then, $e_{\mathbf{x}}^{\text{in}} = e_{\mathbf{x}}^{\text{out}}$, and $e_{\mathbf{x}}^{\text{out}}$ is log-space computable.*

Proof. By the definition of $e_{\mathbf{x}}^{\text{out}}$, there is a local path P from the head of $e_{\mathbf{x}}^{\text{out}}$ to the boundary of $\mathcal{R}[E_i]$ in the exit direction (which is also the entrance direction). All edges in E_i closer to the boundary in the entrance direction from $e_{\mathbf{x}}^{\text{out}}$ have at least one endpoint reachable from P. If any of these edges could reach $e_{\mathbf{x}}^{\text{out}}$, then there would be a cycle. Therefore, $e_{\mathbf{x}}^{\text{in}} = e_{\mathbf{x}}^{\text{out}}$.

Iterate through the edges in E_i starting on the exit side. Then, $e_{\mathbf{x}}^{\text{out}}$ is the last edge in this order with a local path from the head to the boundary of $\mathcal{R}[E_i]$ in the exit direction. □

While it would be useful to have a property similar to Lemma 3.9 for nesting patterns, it is possible to induce a nesting pattern without reaching the incoming edge $e_{\mathbf{x}}^{\text{in}}$ of a nesting pattern description $\mathbf{x}$. This causes a small challenge, which is overcome in Definition 4.1. For now, we do have one half of the reachability properties in Lemma 3.9, that the outgoing edge $e_{\mathbf{x}}^{\text{out}}$ reaches everything that a path using the nesting pattern can reach.

Lemma 3.12. *Let $\mathbf{x}$ be a nesting pattern centered at an edge class E_i. Let y and z be vertices not inside $\mathcal{R}[E_i]$. If there exists an irreducible path from y to z using local paths and edges in the global edge class E_i which induces $\mathbf{x}$, then z is reachable from* $\text{Head}\left(e_{\mathbf{x}}^{\text{out}}\right)$.

Proof. Let e be the edge in E_i used in the path from y to z that induces the nesting pattern. If $e = e_{\mathbf{x}}^{\text{out}}$, then the result holds.

Otherwise, since the head of e reaches z in the exit direction using local paths, e is closer to the exit side of E_i than $e_{\mathbf{x}}^{\text{out}}$. Note that the path from the head of $e_{\mathbf{x}}^{\text{out}}$ that reaches the boundary of $\mathcal{R}[E_i]$ in the exit direction must cross the ancestor path at the head of e, so z is reachable from the head of $e_{\mathbf{x}}^{\text{out}}$ using a local path. □

The reachability property of nesting patterns given by Lemma 3.12 is not as strong as those for full patterns given by Lemma 3.9. This is because there may exist vertices w that have paths inducing the nesting pattern without reaching the incoming edge $e_{\mathbf{x}}^{\text{in}}$. While the global incoming edge $e_{\mathbf{x}}^{\text{in}}$ does not need to be reached in order to induce a nesting pattern, each starting vertex w determines a special edge that gives a strong reachability characterization.

Definition 3.13 (Most-Interior Edge)**.** Let $\mathbf{x} = (i, t, o, p)$ be a pattern description of nesting type and w be a vertex not in the interior of $\mathcal{R}[E_i]$. The *most-interior* edge of $\mathbf{x}$ reachable from w, denoted $e_{\mathbf{x}}^{\text{int}(w)}$, is the edge e in the class E_i that is farthest from the entrance side of $\mathcal{R}[E_i]$ so that there is a local path from w to $\text{Tail}(e)$ in the entrance direction, and there is a local path from $\text{Head}(e)$ to the exit boundary of $\mathcal{R}[E_i]$.

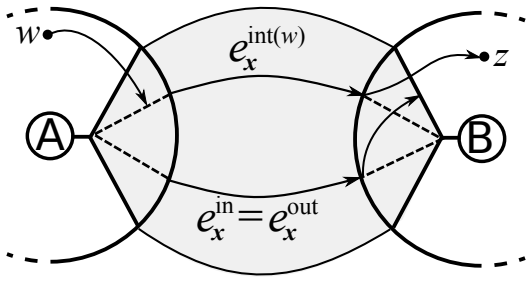


Figure 5: The most-interior edge from a vertex w in a pattern description $\mathbf{x}$ with an $\langle$RXR$\rangle$ pattern.

Lemma 3.14. *Let $\mathbf{x}$ be a pattern description of nesting type and w a vertex not in the interior of $\mathcal{R}[E_i]$. The most-interior edge, $e_{\mathbf{x}}^{\text{int}(w)}$, is log-space computable. For any vertex z not in $\mathcal{R}[E_i]$, there is a path from w to z that induces the pattern $\mathbf{x}$ if and only if there is an irreducible local path from* $\text{Head}\left(e_{\mathbf{x}}^{\text{int}(w)}\right)$ *to z in the exit direction of $\mathbf{x}$.*

Proof. The edges in the class E_i have an order using the rotation given by the entrance direction of the pattern description $\mathbf{x}$, where two edges in E_i can be compared using this order in log-space. Let $e_{\mathbf{x}}^{\text{int}(w)}$ be the edge e of class E_i farthest from the entrance side of $\mathcal{R}[E_i]$ with tail reachable from w and the head has a local path reaching the exit boundary of $\mathcal{R}[E_i]$ in the exit direction of $\mathbf{x}$. Note that this edge is computable in log-space using the SMPD algorithm and pairwise comparison of the rotational order of edges.

Consider an irreducible path P from w that induces the pattern description $\mathbf{x}$ to reach a vertex z outside $\mathcal{R}[E_i]$. By Lemma 3.12, the path P uses exactly one edge e of the class E_i. The edge cannot farther from the entrance side of $\mathcal{R}[E_i]$ than $e_{\mathbf{x}}^{\text{int}(w)}$ or else either w does not reach Tail(e) or Head(e) does not reach the exit of $\mathcal{R}[E_i]$. The path that exits the class E_i from the head of $e_{\mathbf{x}}^{\text{int}(w)}$ must pass through the tree path from the source to the head of e. Therefore, the head of e is reachable from the head of $e_{\mathbf{x}}^{\text{int}(w)}$ and so is anything reachable from the head of e, including z.

Since Tail($e_{\mathbf{x}}^{\text{int}(w)}$) is reachable from w using a local path in the entrance direction, anything reachable from Head($e_{\mathbf{x}}^{\text{int}(w)}$) using a local path in the exit direction is reachable from w using a path that induces the pattern description $\mathbf{x}$. □

Armed with these patterns and reachability patterns, we describe a graph of order $O(m + g_S)$ that preserves uv-reachability.

4 The Pattern Graph

Definition 4.1 (The Pattern Graph)**.** Given G and F as above, the *pattern graph*, denoted $\mathrm{P}(G, F)$, is the graph on vertex set

$$V'_{\mathrm{P}} = \{u', v'\} \cup V_{\mathrm{P}} = \{u', v'\} \cup \big[\{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P}\big],$$

where two pattern descriptions $\mathbf{x}, \mathbf{y} \in V_{\mathrm{P}}$ have an adjacency $\mathbf{x} \to \mathbf{y}$ if and only if there exists a (possibly empty) list of nesting pattern descriptions $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ (called an *adjacency certificate*), so that the following two conditions hold:

1. There is an irreducible path from Head($e_{\mathbf{x}}^{\text{out}}$) to Tail($e_{\mathbf{y}}^{\text{in}}$) which induces the sequence $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ of nesting pattern descriptions.
2. For each $j \in \{1, \ldots, \ell\}$, Tail($e_{\mathbf{z}_j}^{\text{in}}$) is not reachable from Head($e_{\mathbf{x}}^{\text{out}}$) using irreducible paths that induce the pattern descriptions $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$.

The two vertices u' and v', which are not pattern descriptions, have adjacencies as well. The vertex u' has an edge to all patterns on edge classes in G with the t-end in the tree T_u. A pattern description $\mathbf{x} = (i, t, o, p)$ has an edge to v' if and only if the class E_i is incident to v, t is the other end of the class, and $p \in \{\langle \mathrm{RXL} \rangle, \langle \mathrm{LXR} \rangle\}$.

Theorem 4.2. *There exists a path from u to v in G if and only if there exists a path from u' to v' in* $\mathrm{P}(G, F)$.

Proof. Let P be an irreducible path from u to v in G. We wish to show that there is a path from u' to v' in $\mathrm{P}(G, F)$.

P induces a sequence of pattern descriptions $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$. Note that $\mathbf{x}_1$ is centered at an edge class that is incident to T_u and the entrance end is on T_u. Note also that $\mathbf{x}_\ell$ is centered at an edge class where the edges have head v. Thus, in $\mathrm{P}(G, F)$, $u' \to \mathbf{x}_1$ and $\mathbf{x}_\ell \to v'$ are edges.

For full pattern descriptions $\mathbf{x}_i$, Lemma 3.9 implies that we may assume the first edge in the global edge class of $\mathbf{x}_i$ used by P is $e^{\text{in}}_{\mathbf{x}_i}$ and the last such edge is $e^{\text{out}}_{\mathbf{x}_i}$.

Fix $i \in \{1, \ldots, \ell - 1\}$ and let $\mathbf{x}_j$ be the next full pattern induced after $\mathbf{x}_i$. If $j = i + 1$, then the path P takes a local path between the edges that induce the patterns $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. By Lemma 3.9, $e^{\text{in}}_{\mathbf{x}_j}$ is reachable from $e^{\text{out}}_{\mathbf{x}_i}$ by a local path and an adjacency exists from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ in $\mathrm{P}(G, F)$, using an empty list of nesting patterns as the adjacency certificate.

Otherwise, $j > i+1$ and there are $j-i$ nested patterns between $\mathbf{x}_i$ and $\mathbf{x}_j$. Rename the nesting patterns between $\mathbf{x}_i$ and $\mathbf{x}_j$ as $\mathbf{z}_1, \ldots, \mathbf{z}_{j-i}$ where $\mathbf{z}_{i'} = \mathbf{x}_{i+i'}$. If $\mathbf{z}_1, \ldots, \mathbf{z}_{j-i}$ compose an adjacency certificate for $\mathbf{x}_i \to \mathbf{x}_j$, then this edge exists in $\mathrm{P}(G, F)$. Otherwise, there exists such a k that violates the adjacency condition between $\mathbf{x}_i$ and $\mathbf{x}_j$, then let i' be the smallest such index. There is an edge in $\mathrm{P}(G, F)$ from $\mathbf{x}_i$ to the nesting pattern description $\mathbf{z}_{i'}$, since $\mathrm{Tail}(e^{\text{in}}_{\mathbf{z}_{i'}})$ is reachable from $\mathrm{Head}(e^{\text{out}}_{\mathbf{x}_i})$ by a path using the nesting patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{i'-1}$ as the adjacency certificate. By Lemma 3.12, $\mathrm{Tail}(e^{\text{in}}_{\mathbf{x}_j})$ is reachable from $\mathrm{Head}(e^{\text{out}}_{\mathbf{z}_{i'}})$ using an irreducible path which induces the patterns $\mathbf{z}_{i'+1}, \ldots, \mathbf{z}_{j-i}$. By iteration, there is a path from $\mathbf{z}_{i'}$ to $\mathbf{x}_j$ in $\mathrm{P}(G, F)$, and hence a path from $\mathbf{x}_i$ to $\mathbf{x}_j$ in $\mathrm{P}(G, F)$. Connecting all of the edges between the full patterns in $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ gives a path from u' to v' in $\mathrm{P}(G, F)$.

It remains to show that any path in $\mathrm{P}(G, F)$ from u' to v' produces an irreducible path in G from u to v which induces those adjacencies.

Given a path $P = u', \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell, v'$ in $\mathrm{P}(G, F)$, let $\mathbf{x}_j = (i_j, t_j, o_j, p_j)$ for each $j \in \{1, \ldots, \ell\}$. Since $u' \to \mathbf{x}_1$ in $P(G)$, E_{i_1} is a class incident to T_u and all edges are reachable from u. Specifically, there is a tree path P_0 from u to $e^{\text{out}}_{\mathbf{x}_1}$. Similarly, since $\mathbf{x}_\ell \to v'$ in $\mathrm{P}(G, F)$, E_{i_k} is a class incident to T_v and all edges have v as a head. For each $j \in \{1, \ldots, \ell - 1\}$, Lemmas 3.9 and 3.12 imply there is an irreducible path P_i in G from the head of $e^{\text{out}}_{\mathbf{x}_j}$ to the tail of $e^{\text{in}}_{\mathbf{x}_{j+1}}$ that is either a local path or induces a list of nesting pattern descriptions which form an adjacency certificate. Also, by Definition 3.7, there exist (possibly empty) paths Q_j from $e^{\text{in}}_{\mathbf{x}_j}$ to $e^{\text{out}}_{\mathbf{x}_j}$ using local paths and edges of the class E_{i_j}. These paths concatenate to a path

$$uP_0e^{\text{out}}_{\mathbf{x}_1}P_1e^{\text{in}}_{\mathbf{x}_2}Q_2e^{\text{out}}_{\mathbf{x}_2}P_2e^{\text{in}}_{\mathbf{x}_3}\ldots e^{\text{out}}_{\mathbf{x}_{\ell-1}}P_{\ell-1}e^{\text{in}}_{\mathbf{x}_\ell}v$$

from u to v in G. □

Lemma 4.3. *The pattern graph* $\mathrm{P}(G, F)$ *is log-space computable.*

Proof. Given a pattern description $\mathbf{x}$, we describe a log-space algorithm for enumerating the pattern descriptions reachable by an edge in $\mathrm{P}(G, F)$.

A necessary subroutine takes a global edge e and enumerates all pattern descriptions reachable from $\mathrm{Head}(e)$ using local paths in the exit direction of $\mathbf{x}$. By Lemma 3.3, there is an ordered list of topological equivalence classes $E_{i_0}, E_{i_1}, \ldots, E_{i_\ell}$ reachable by local paths from the head of e. E_{i_0} is the class containing e, so e is in $\mathcal{R}[E_{i_0}]$. All other classes E_{i_j} (for $j \geq 1$, except possibly $j = \ell$) are fully reachable. Hence, each pattern description $\mathbf{y}$ centered at a class E_{i_j} with $j \in \{1, \ldots, \ell - 1\}$ (where the entrance direction of $\mathbf{y}$, orientation, and end all match the exit direction of $\mathbf{x}$) has $e^{\text{in}}_{\mathbf{y}}$ reachable from $\mathrm{Head}(e)$ using a local path. Each pattern description $\mathbf{y}$ with entering direction the same as the exit direction of $\mathbf{x}$ and centered at E_{i_ℓ} can be checked if $e^{\text{in}}_{\mathbf{y}}$ is reachable from e. The only pattern that could be used without having $e^{\text{in}}_{\mathbf{y}}$ reachable is a nesting pattern.

To enumerate all neighbors of $\mathbf{x}$ in $\mathrm{P}(G, F)$, perform the above subroutine on $e^{\text{out}}_{\mathbf{x}}$, adding edges from $\mathbf{x}$ to each reachable pattern description $\mathbf{y}$. If the nesting pattern $\mathbf{z}$ on E_{i_ℓ} is not fully reachable (i.e. there is no local path from e to $e^{\text{in}}_{\mathbf{z}}$ in the proper direction) then compute the most-interior edge

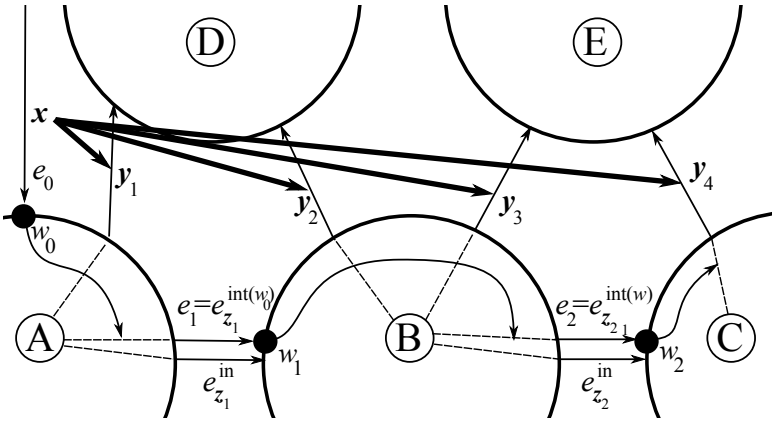


Figure 6: The nesting patterns $\mathbf{z}_1$ and $\mathbf{z}_2$ satisfy the adjacency conditions in Definition 4.1 from $\mathbf{x}$ to each $\mathbf{y}_j$. The pattern adjacencies are enumerated during the algorithm of Lemma 4.3 where e is assigned to e_0, e_1, and e_2, sequentially. Note that $e_0 = e_{\mathbf{x}}^{\text{out}}$, $e_1 = e_{\mathbf{z}_1}^{\text{int}(\text{Head}(e_0))}$, and $e_2 = e_{\mathbf{z}_2}^{\text{int}(\text{Head}(e_1))}$. The pattern $\mathbf{y}_1$ is reachable from w_0 with no internal nesting patterns. The patterns $\mathbf{y}_2$ and $\mathbf{y}_3$ are reachable from w_0 using the nesting pattern $\mathbf{z}_1$. The pattern $\mathbf{y}_4$ is reachable from w_0 using the nesting patterns $\mathbf{z}_1$ and $\mathbf{z}_2$. The algorithm from Lemma 4.3 terminates at e_2, since e_2 does not give a partially-reachable class.

$e_{\mathbf{z}}^{\text{int}(\text{Head}(e))}$. Repeat the subroutine on this edge, continuing until the class E_{i_ℓ} is fully reachable (or the list is empty). See Figure 6 for an example of this iterative procedure.

It is clear this algorithm takes log-space. It enumerates all neighbors of $\mathbf{x}$ in $\mathrm{P}(G, F)$, since a neighbor $\mathbf{y}$ requires a list of nesting classes $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ so that there is an irreducible path from $\mathbf{x}$ to $\mathbf{y}$ inducing these classes. Moreover, each class $\mathbf{z}_j$ has the edge $e_{\mathbf{z}_j}^{\text{in}}$ *not* reachable from $\mathbf{x}$ using the patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$. This means that the patterns $\mathbf{z}_j$ are centered at the class E_{i_ℓ} computed by the iteration of the subroutine on the edge $e_{\mathbf{z}_{j-1}}^{\text{int}(w_{j-1})}$. Moreover, $\mathbf{y}$ appears as a reachable class from the most-interior edge computed at $\mathbf{z}_\ell$, so $\mathbf{y}$ is enumerated. Finally, any pattern enumerated by this procedure can reconstruct the list of $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ by using the nesting patterns used in the subroutine iterations. □

We have all the necessary tools to prove the main theorem.

The Main Theorem

Theorem 1.1. *There is a log-space reduction that given an instance $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m, g)$ and u, v vertices of G, outputs an instance $\langle G', u', v'\rangle$ where G is a directed graph and u', v' vertices of G', so that*

(a) there is a directed path from u to v in G if and only if there is a directed path from u' to v' in G',
(b) G' has $O(m + g)$ vertices.

Proof. Fix a forest decomposition F and let G' be the pattern graph $\mathrm{P}(G, F)$. Lemma 4.2 shows that there is a path from u to v in G if and only if there is a path from u' to v' in $\mathrm{P}(G, F)$ if and only if there is a path from u' to v' in $\mathrm{P}(G, F)$. Lemma 4.3 gives that G' is log-space computable. By Lemma 2.9, there are at most $O(m+g)$ equivalence classes in G (with respect to F), and there is a constant multiple of pattern descriptions per equivalence class, so G' has $O(m+g)$ vertices. □

5 Comparison with other space-bounded reachability algorithms

See Table 2 for a list of space bounds of different algorithms for reachability in certain classes of graphs. Table 3 describes which results give which space bounds with simultaneous polynomial-time algorithms.

Ealier known graph class	Space bound s	New graph class given by Theorem 1.2
Undirected Graphs [Rei08] SMPD[4] [ABC+09] LMPD[5] [SBV10]	$O(\log n)$	$\mathcal{G}\left(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})}\right)$
Poly-mixing time [RTV06, SZ99]	$O\left(\log^{\frac{3}{2}} n\right)$	$\mathcal{G}\left(2^{O(\log^{\frac{3}{4}} n)}, 2^{O(\log^{\frac{3}{4}} n)}\right)$
Mangroves [AL98]	$O\left(\frac{\log^2 n}{\log\log n}\right)$	$\mathcal{G}\left(2^{O\left(\frac{\log n}{\sqrt{\log\log n}}\right)}, 2^{O\left(\frac{\log n}{\sqrt{\log\log n}}\right)}\right)$
	$o(\log^2 n)$	$\mathcal{G}(n^{o(1)}, n^{o(1)})$
All directed graphs [Sav70]	$O(\log^2 n)$	

Table 2: A table of graph classes (old and new) for which reachability can be solved using space s, for various interesting values of s.

6 Discussion

This work significantly extends the concepts introduced in [SBV10] in multiple directions. The notion of topological equivalence has been extended to arbitrary surfaces. A careful analysis of the interactions between the equivalent classes of edges was used to create a graph of size linear in the number of equivalence classes that preserves uv-reachability. It appears that improving the bounds proved in this paper may require significantly new ideas.

One potential attack could be to design an explicit algorithm A for selecting the forest decomposition F_A. Our arbitrary selection has been handled in the *worst case*, as if an adversary chose the worst possible decomposition for our algorithm. The goal of such a decomposition algorithm would be to reduce the minimum number of patterns induced by a path from u to v. Note that the *best case* has the tree T_u given as a DFS or BFS tree centered at u, where all reachable vertices from u are in the tree. Since the bound on the number of equivalence classes we prove is based on the number of sources and the genus of S, it may not be possible to find a forest decomposition with asymptotically fewer equivalence classes. However, if the length of the shortest path from u' to v' in the pattern graph could be bound to sub-polynomial in m and g, a standard Savitch-like divide and conquer technique will give a $o(\log^2(m+g))$ space bound. This might be possible by

[4]SMPD: Single-source Multiple-sink Planar DAG

[5]LMPD: Log-source Multiple-sink Planar DAG

[6]It is a quick observation that reachability in mangroves is decidable by a LogDCFL machine.

Earlier known graph class	Space bound s with poly-time	New graph class given by Theorem 1.7
Poly-mixing time [RTV06, Nis95] Mangroves[6] [Lan97, Coo79]	$O(\log^2 n)$	
	$2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}$	$\mathcal{G}\left(2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}, 2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}\right)$
	$o(n^{\varepsilon})$	$\mathcal{G}(O(n^{\varepsilon}), O(n^{\varepsilon}))$.
All directed graphs [BBRS92]	$O\left(\frac{n}{2^{\sqrt{\log n}}}\right)$	

Table 3: A table of graph classes (old and new) with simultaneous time-space bound $(n^{O(1)}, s)$ for reachability for various values of s.

relaxing the space requirement on A from log-space to $o(\log^2(m+g))$. It might be possible to find a balance between the space requirement on A and the improvement it makes to the resulting pattern graph. Another approach might be to carefully study the properties of the pattern graph $\mathrm{P}(G, F)$ and exploited them to solve reachability in $\mathrm{P}(G, F)$ while improving the space bound.

The fact that the space bounds we get is for the case when both m and g are asymptotically equal can somewhat be explained by the fact that sources can be removed by increasing the genus of the surface. Thus an alternate approach to get the space bounds that we prove here is to reduce the number of sources by increasing the genus (by adding a new vertex and connecting this to all other sources through additional handles). However, this approach will not lead to any simplification in our analysis and hence we prefer to keep these parameters separate.

Acknowledgements

We thank Jeff Erickson for sharing his knowledge on topological embeddings of graphs. We also thank Jonathan F. Buss for discussions on simultaneous time-space bounds for reachability at the 2010 Conference on Computational Complexity.

References

[ABC+09] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

[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).

[BBRS92] Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. In *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, pages 27–33, 1992.

[Coo79] S.A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 338–345. ACM, 1979.

[EJT10] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *FOCS '10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[JLR06] Andreas Jakoby, Maciej Liśkiewicz, and Rüdiger Reischuk. Space efficient algorithms for directed series-parallel graphs. *Journal of Algorithms*, 60(2):85–114, 2006.

[JT07] Andreas Jakoby and Till Tantau. Logspace algorithms for computing shortest and longest paths in series-parallel graphs. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, pages 216–227, 2007.

[KV10] Jan Kynčl and Tomáš Vyskočil. Logspace reduction of directed reachability for bounded genus graphs to the planar case. *ACM Transactions on Computation Theory*, 1(3):1–11, 2010.

[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.

[Nis95] Noam Nisan. RL $\subseteq$ SC. In *In Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing*, pages 619–623, 1995.

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

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *STOC '06: Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*, pages 457–466, New York, NY, USA, 2006. ACM.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[SBV10] Derrick Stolee, Chris Bourke, and N. V. Vinodchandran. A log-space algorithm for reachability in planar acyclic digraphs with few sources. *25th Annual IEEE Conference on Computational Complexity*, pages 131–138, 2010.

[SZ99] Michael Saks and Shiyu Zhou. $\mathrm{BP}_H\mathrm{SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de