# Space-Efficient Algorithms for Reachability in Surface-Embedded Graphs

Derrick Stolee* N. V. Vinodchandran†

May 4, 2012

**Abstract**

This work presents a log-space reduction which compresses a directed acyclic graph with $m$ sources embedded on a surface of genus $g$ to a graph on $O(m + g)$ vertices while preserving reachability between a given pair of vertices. Applying existing algorithms to this smaller graph gives improved space bounds as well as improved simultaneous time-space bounds for the reachability problem for a large class of directed acyclic graphs. Specifically, it significantly extends the class of surface-embedded graphs with log-space reachability algorithms: from planar graphs with $O(\log n)$ sources, to graphs with $2^{O(\sqrt{\log n})}$ sources embedded in a surface of genus $2^{O(\sqrt{\log n})}$. Additionally it also yields sublinear space ($n^{1-\epsilon}$ space) algorithms with polynomial running time for graphs with $n^{1-\epsilon}$ sources embedded on surfaces of genus $n^{1-\epsilon}$.

## 1 Introduction

Graph reachability problems are central to space-bounded computations. Different versions of this problem characterize several important space complexity classes. The problem of deciding whether there is a path from a given vertex $u$ to a vertex $v$ in a directed acyclic graph is the canonical complete problem for non-deterministic log-space ($\mathsf{NL}$). The recent breakthrough result of Reingold implies that the undirected reachability problem characterizes the complexity of deterministic log-space ($\mathsf{L}$) [13]. It is also known that certain restricted promise versions of the directed reachability problem characterize randomized log-space computations ($\mathsf{RL}$) [14]. Clearly, progress in space complexity studies is directly related to progress in understanding graph reachability problems. We refer the readers to a (two decades old, but excellent) survey by Avi Wigderson [19] and a recent update by Eric Allender [1] to further understand the significance of reachability problems in complexity theory.

In this paper we focus on designing *deterministic* algorithms for reachability with improved space complexity. For the general directed graph reachability problem the best known result remains the 40-year old $O(\log^2 n)$ space bound due to Savitch [16] (where $n$ is the number of vertices in the graph). Designing a deterministic algorithm for the directed graph reachability problem that asymptotically beats Savitch's bound is the most significant open questions in this topic. While this remains a difficult open problem, investigating classes of directed graphs for which we can design space efficient algorithms that beat Savitch's bound is an important research direction with some outstanding results, including Saks and Zhou's $O(\log^{3/2} n)$ bound for reachability problems

*Departments of Computer Science and Mathematics, University of Nebraska–Lincoln, dstolee@cse.unl.edu
†Department of Computer Science, University of Nebraska–Lincoln, vinod@cse.unl.edu

characterizing $\mathsf{RL}$ computations [15] and Reingold's log-space algorithm for the undirected reachability problem [13]. In this paper we consider the reachability problem over *directed acyclic graphs that are embedded on topological surfaces*. We present the best (to date) space complexity upper bounds for the reachability problem over this class of directed graphs.

### Prior Results

Jakoby, Liśkiewicz, and Reischuk [8] and Jakoby and Tantau [9] show that various reachability and optimization questions for *series-parallel* graphs admit deterministic log-space algorithms. Series-parallel graphs are a very restricted subclass of planar DAGs. In particular, such graphs have a single source and a single sink. Allender, Barrington, Chakraborty, Datta, and Roy [2] extended the result of Jakoby *et al.* to show that the reachability problem for Single-source Multiple-sink Planar DAGs (SMPDs) can be decided in logarithmic space. Building on the work of Allender *et al.* [2], in [17], the present authors show that reachability for planar DAGs with $O(\log n)$ sources can be decided in logarithmic space. Theorem 1 below is implicit in [17].

**Theorem 1** ([17])**.** *Let $\mathcal{G}(m)$ denote the class of planar DAGs with at most $m = m(n)$ sources, where $n$ is the number of vertices. The reachability problem over $\mathcal{G}(m)$ can be solved by a log-space nondeterministic machine using a one-way certificate of $O(m)$ bits. In particular, reachability over $\mathcal{G}(m)$ can be decided deterministically in* $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ *space.*

The $O(\log n + m)$ space bound is obtained by a brute-force search over all certificates of length $O(m)$. Setting $m = O(\log n)$ we get a deterministic log-space algorithm for reachability over planar graphs with $O(\log n)$ source nodes. The $O(\log n \cdot \log m)$ bound is obtained by first converting the nondeterministic algorithm to a layered graph with $m$ layers and $\mathrm{poly}(n)$ vertices in each layer, and then applying Savitch's algorithm on this layered graph. The second bound leads to a deterministic algorithm that beats Savitch's bound for reachability over DAGs with $2^{o(\log n)}$ sources (for example, setting $m = 2^{\log^{1-\epsilon} n}$, it gives a $\log^{2-\epsilon} n$ space algorithm for reachability over planar graphs with $2^{\log^{1-\epsilon} n}$ source nodes).

However, if we are aiming for deterministic algorithms with $O(\log n)$ space complexity, the above theorem could not handle asymptotically more than $\log n$ sources. In this paper we improve the upper bound from $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ to $O(\log n + \log^2 m)$. This yields a new deterministic log-space algorithm for reachability over planar DAGs with $m = 2^{O(\sqrt{\log n})}$ source nodes. We also extend our results to graphs embedded on higher genus surfaces. In addition, techniques of this paper also leads to new results on simultaneous time-space bounds for reachability which are not implied by [17].

The main technique of [17] (that leads to $O(\log n \cdot \log m)$ bound) can be viewed as a log-space reduction that takes $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m)$ and outputs $\langle G', u', v', \rangle$ 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) $G'$ is a layered graph with $m$ layers and $\mathsf{poly}(n)$ vertices per layer. This $\mathsf{poly}(n)$ factor in the size of $G'$ makes it useless for obtaining a logarithmic space bound. We get rid of this $\mathsf{poly}$(n) factor by avoiding the intermediate nondeterminism and giving a direct reduction to a new reachability instance. This requires a more careful analysis of the topological interaction of paths in surface-embedded graphs.

### New Results

Let $n$ be the number of vertices in the input graph. Let $\mathcal{G}(m, g)$ denote the class of 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 2.** *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.*

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

**Theorem 3.** *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in deterministic $O(\log n + \log^2(m + g))$ space.*

This improves the earlier-known space bound of $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ and also extends it to higher genus graphs. By setting $m = g = 2^{O(\sqrt{\log n})}$ we get a deterministic log-space algorithm for reachability over graphs in $\mathcal{G}(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})})$.

**Corollary 4.** *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.*

By setting $m$ and $g$ to be $n^{o(1)}$ we get $o(\log^2 n)$ bound. The following corollary as stated is implicit in [17]. However, the space bound we get for any specific function $n^{l(n)}$ where $l(n) \in o(1)$ is better than what is implied by the results of [17].

**Corollary 5.** *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 $o(\log^2 n)$.*

Theorem 2 leads to new simultaneous time-space bound for the reachability problem. 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 linear space, the goal here is to improve the space bound while maintaining a polynomial running time. The most significant result here is Nisan's $O(\log^2 n)$ space, $n^{O(1)}$ time bound for $\mathsf{RL}$ [12]. The best upper bound for general directed reachability is the 20-year old $O(n/2^{\sqrt{\log n}})$ space, $n^{O(1)}$ time algorithm due to Barnes, Buss, Ruzzo and Schieber [4]. Combining our reduction with a simple depth-first search gives better simultaneous time-space bound for reachability over a large class of graphs that beats the Barnes *et al.* bound.

**Theorem 6.** *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in polynomial time using $O(\log n + m + g)$ space.*

Note that Theorem 6 has a space bound which matches to $O(\log n + m)$ space bound of Theorem 1, except it guarantees polynomial time, where the previous bound gave $2^{O(m)}\,\mathrm{poly}(n)$ running time. For any $\epsilon < 1$, 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.

**Corollary 7.** *For any $\epsilon$ 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 6 can be slightly improved to $O\left((m+g)2^{-\sqrt{\log(m+g)}}\right)$ by using the Barnes *et al.* algorithm instead of depth-first search, which will give a $o(n^{\epsilon})$ space bound in the above corollary.

**Theorem 8.** *The reachability problem for graphs in $\mathcal{G}(m,g)$ can be decided in deterministic polynomial time using $O\left(\log n + \frac{m+g}{2^{\sqrt{\log(m+g)}}}\right)$ space.*

Before we go into further details, we note that throughout 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.

## 1.1 Outline

Theorem 2 is proven in several parts. We begin in Section 2 by reviewing some concepts of topological embeddings including log-space algorithms on embedded graphs. In Section 3, 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. We present log-space algorithms of Allender, Barrington, Chakraborty, Datta, and Roy [2] to decide reachability using local edges. In order to control how the global edges interact, we define the notion of *topological equivalence* among global edges in Section 4. We show that the number of possible equivalence classes is bounded by $O(m+g)$. Then, Section 5 describes a finite list of *patterns* that characterize how paths use edges in 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 6, we describe a graph on $O(m+g)$ vertices called the *pattern graph* whose vertices are described by patterns on equivalence classes. The edges in the pattern graph are defined by a very restricted reachability condition between equivalence classes. 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 throughout 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.

## 1.2 Notation

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

We assume that the given graph is acyclic. Lemma 9 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 9.** *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 $\{e_1, \dots, e_{|E(G)|}\}$. For each $i \in \{0, 1, \dots, |E(G)|\}$, let $G_i$ be the subgraph of $G$ where an edge $e_j$ is present in $G_i$ if $e_j \in F$ or $j \le i$. Iterate through all such $i$ and test if $A(G_i, \mathrm{Head}(e_{i+1}), \mathrm{Tail}(e_{i+1}))$ 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 $i \in \{1, \dots, |E(G)|\}$ if and only if $G$ is acyclic and all queries $A(G_i, \mathrm{Head}(e_{i+1}), \mathrm{Tail}(e_{i+1}))$ return False. □

## 2 Topological Embeddings and Algorithms

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 assume that such an embedding 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. Since computing or approximating a low-genus embedding of a non-planar graph is an NP-complete problem [5,18], we require the embedding to be given as part of the input and we consider reachability in $\mathcal{G}(m, g)$ to be a promise problem. In the case of genus zero, we can compute a planar embedding in log-space and the promise condition can be removed.

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 $\chi_S$ 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 [10]), so $\chi_S$ 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 $G$. A surface separating curve $C$ is called *contractible* if removal of the nodes in $C$ disconnects $G$ where at least one of the connected components has an induced embedding homeomorphic to a disc.

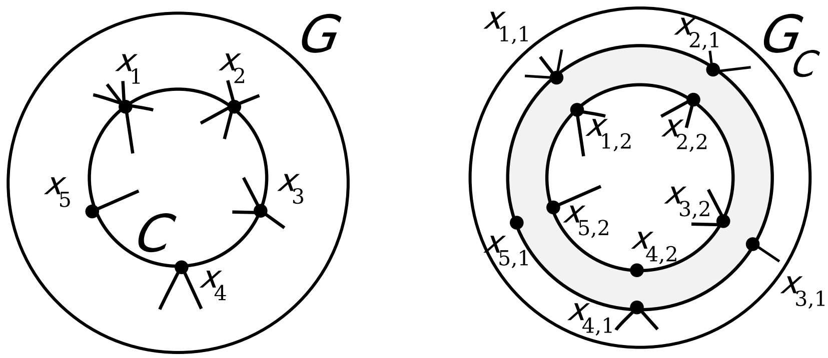
In order to perform log-space algorithms on curves in the graph, we must be able to represent these curves in log-space. A curve $C$ is *log-space walkable* if there is a log-space algorithm which outputs the edges of $C$ in order. Examples of such curves are given in the following section. Given a log-space walkable curve $C$, it is possible to detect the type (separating, contractible, or neither) of $C$ in log-space.

First, note that if $C$ is not orientable (i.e. there are an odd number of negatively-signed edges in $C$) then $C$ cannot be separating or contractible. By first checking the parity of such edges, we can assume that $C$ is orientable.

Given an orientable curve $C = x_1 x_2 \dots x_k$ (indices taken modulo $k$), we can create (in log-space) an auxiliary graph $G_C$ where each vertex $x_i$ is copied to two vertices $x_{i,1}, x_{i,2}$ with edges $x_i x_{i+1}$ copied to two edges $x_{i,1} x_{i+1,1}$ and $x_{i,2} x_{i+1,2}$. However, an edge from a vertex $y$ in $V(G) \setminus C$ to a vertex $x_i$ in $C$ maps to one of two edges:

1. $yx_i$ maps to $yx_{i,1}$ if $yx_i$ appears between $x_{i-1}x_i$ and $x_i x_{i+1}$ in the clockwise order about $x_i$.
2. $yx_i$ maps to $yx_{i,1}$ if $yx_i$ appears between $x_i x_{i+1}$ and $x_{i-1}x_i$ in the clockwise order about $x_i$.

There is a natural combinatorial embedding of $G_C$ induced from the embedding of $G$ by using the same cyclic relations for vertices $y \in V(G) \setminus C$ and for split vertices $x_{i,1}$ and $x_{i,2}$, use the orientation of $x_i$ but skip the edges which are not incident to the new vertex. See Figure 1 for an example of such a split. The following properties are simple to prove:




(a) A graph $G$ with curve $C$.

(b) The split graph $G_C$.

*Figure 1:* Splitting $G$ at a curve $C$.

1. $C$ is separable if and only if $G_C$ is disconnected. In this case, $G_C$ has two components.
2. $C$ is contractible if and only if $G_C$ is disconnected and at least one of the components is embedded with characteristic zero.

Moreover, using Reingold's undirected reachability algorithm we can detect that $C$ is separable. Given a vertex $y \notin C$, we can also detect which connected component of $G_C$ contains $y$. We shall exploit both of these properties in the following two sections as we partition the edge set using topological information.

## 3 Forest Decomposition

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 [2, 17].

Let $G$ be a directed acyclic graph and let $u, v$ be two vertices. Our goal is to decide whether there is a 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 10** (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 $G$.

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 will 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 11** (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 $xy$.

**Definition 12** (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 (a) $x$ and $y$ are on the same tree in $F$, (b) the closed tree curve at $xy$ is contractible (i.e. the curve cuts $S$ into a disk and another surface), and (c) 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. Otherwise, the edge $xy$ is *global.*

### 3.1 Paths within a single tree

**Definition 13** (Region of a 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 $\mathcal{R}[T]$ are together homeomorphic to 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 [2], 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]. We can also assign a clockwise or counter-clockwise direction to all local edges in a source tree region $\mathcal{R}[T_{s_j}]$.

**Definition 14** (Rotational Direction within $\mathcal{R}[T]$)**.** For a local edge $xy$, the closed tree curve at $xy$ is cyclicly oriented by the direction of $xy$. The edge $xy$ is considered clockwise (counter-clockwise) if this cyclic orientation is clockwise (counter-clockwise) with respect to the orientation of $\mathcal{R}[T]$.

**Definition 15** (Irreducible Path)**.** A path $P = x_1 x_2 \dots x_k$ in $G$ is *$F$-irreducible* if for each $i < j$ so that $x_i$ is an $F$-ancestor of $x_j$, then $x_i x_{i+1} \dots x_{j-1} x_j$ is the path in $F$ from $x_i$ to $x_j$. We say $P$ is *irreducible* when the forest decomposition $F$ is implied from context.

**Lemma 16.** *If there is a path from $x$ to $y$ in $G$, there is an $F$-irreducible path from $x$ to $y$.*

*Proof.* Replace the violating subpaths with the given tree paths. □

A very useful property of irreducible paths is that they travel in a single rotational direction within each source tree.

**Lemma 17.** *Let $P$ be an irreducible local path from $x$ to $y$ in a source tree $T$, where $y$ is on the boundary of $\mathcal{R}[T]$. There is a unique direction (clockwise or counter-clockwise) so that all non-tree edges of $P$ follow this direction.*

[1]This definition of *local* differs from the use in [2] and [17].

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

*Proof.* Let $e$ be the first local edge in $P$. Without loss of generality, we assume it takes a clockwise orientation. Assume for the sake of contradiction there exists a local edge in $P$ that takes a counterclockwise orientation. Let $f$ be the first such edge. Consider how $P$ travels from the head of $e$ to reach the tail of $f$. Note that all non-tree edges in this path have a clockwise orientation. This gives three cases:

**Case 1:** $P$ passes through the ancestor path of Head($f$) at a vertex $a$. In this case, $P$ is not irreducible, since $f$ is not a tree edge and an irreducible path would take the tree edges from $a$ to Head($f$).

**Case 2:** $P$ passes through the descendants of Head($f$) at a vertex $b$. In this case, following $P$ from $a$ to Head($f$) then the tree path from Head($f$) to $a$ creates a cycle, contradicting that $G$ is a DAG.

**Case 3:** $P$ travels around the descendants of Head($f$) using a local edge $e'$. Now, Head($f$) is properly contained within the tree cycle given by $e'$. In order for $P$ to reach $y$ on the boundary of $\mathcal{R}[T]$, $P$ must cross this curve. This must cross the descendants of Tail($e'$) or Head($e'$), creating a cycle, contradicting that $G$ is acyclic.

Therefore, such an $f$ does not exist and all edges take the same orientation. □

### 3.2 Reachability within a single tree

We now focus on the reachability problem within a single tree $T_{s_j}$. By the definition of local edges, we have the subgraph given by local edges within a single tree is a single-source multiple-sink planar DAG. Allender *et al.* [2] solved the reachability problem in this class of graphs. We review their method as well as adapt the method to test directional reachability.

**Definition 18** (Step and Jump Edges)**.** A local edge $e \notin F$ is a *jump* edge if the tree curve $C_e$ partitions $V(G) \setminus C_e$ into two non-trivial parts. Otherwise, $e$ is a *step* edge.

First, we discuss how to solve reachability when restricted to tree and step edges.

**Theorem 19** (Allender *et al.* [2])**.** *Let $s_j$ be a source in $G$. Reachability within $\mathcal{R}[T_{s_j}]$ using tree and step edges is log-space computable.*

*Proof.* Here, we consider the subgraph in $\mathcal{R}[T_{s_j}]$ given by the tree and step edges to be a planar graph with a single source. Since we have removed the jump edges in $\mathcal{R}[T_{s_j}]$, all sinks in this graph are on the boundary of $\mathcal{R}[T_{s_j}]$. By adding a new global sink $t$ to the outer face, the graph $\mathcal{R}[T_{s_j}]+t$ becomes a Single-source Single-sink Planar DAG (SSPD).

The cyclic orientation of edges at each vertex must have the outgoing edges and incoming edges in two consecutive blocks. If not, suppose that the edges $e_1, e_2, e_3, e_4$ appear in clockwise order at a vertex $x$, with $e_1, e_3$ are outgoing edges and $e_2, e_4$ are incoming edges. Since there is a single source $s_j$, there are paths $P_2$ and $P_4$ from $s_j$ to $x$ using the edges $e_2$ and $e_4$, respectively. Likewise, there are paths $P_1$ and $P_3$ from $x$ to $t$ starting with edges $e_1$, and $e_3$, respectively. This gives two closed curves $C_1$ (composed of $P_1$ and $P_3$) and $C_2$ (composed of $P_2$ and $P_4$) which cross at $x$. Thus, they must cross at another point $y$. By following $C_1$ from $x$ to $y$ and $C_2$ from $y$ to $x$, there is a cycle in $G$, a contradiction.

Given that the outgoing edges at any vertex $x$ are in a single block of the cyclic orientation, we can define the notion of *left-most* and *right-most* outgoing edges of $x$ as those appearing as the first and last (respectively) outgoing edges of the block with respect to the clockwise ordering. This defines a *left-most walk* and a *right-most walk* from a vertex $x$ by following the left-most and right-most edges, starting at $x$ and terminating at $t$. The left-most and right-most walks define a closed curve $C_x$ that includes $x$ and $t$.

A vertex $y$ is inside this curve $C_x$ if and only if it is reachable from $x$: if $y$ is within $C_x$, any path from $s_j$ to $y$ must cross the curve $C_x$, creating a path from $x$ to $y$, and if $y$ is reachable from $x$ via a path $P$, the edges of $P$ must appear between the left-most and right-most walks from $x$. Hence, by splitting $\mathcal{R}[T_{s_j}] + t$ along $C_x$ and computing if $y$ is within $C_x$, we can detect reachability. $\square$

Using the step-reachability algorithm as a subroutine, we now discuss directional reachability using all local edges.

**Theorem 20** (Allender *et al.* [2])**.** *Given vertices $x, y$ on the boundary of $\mathcal{R}[T_{s_j}]$ and a direction $d$ (left or right), reachability from $x$ to $y$ in $\mathcal{R}[T_{s_j}]$ using local edges using an irreducible path in direction $d$ is log-space computable.*

*Proof.* We shall define a log-space data structure called an *explored region* which in turn defines a set of vertices in $\mathcal{R}[T]$. The crucial property of these vertices is that all jump edges with tail in the set and head outside the set are reachable from $x$. We will then use these edges to modify the explored region while maintaining this property. When complete, the explored region will contain $y$ if and only if $y$ is reachable from $x$ via an irreducible path with rotational direction $d$, with respect to the orientation of the source $s_j$.

We shall assume that the direction $d$ is Right (clockwise). The other direction follows by symmetry.

Given a vertex $w$ in $T_{s_j}$, define $\mathrm{ReachStep}(w)$ to be the vertices in $T_{s_j}$, reachable from $w$ by tree and step edges. Define functions $\mathrm{StepLeft}(w)$ and $\mathrm{StepRight}(w)$ to be the vertices within $\mathrm{ReachStep}(w)$ which appear most counter-clockwise and clockwise, respectively, breaking ties by selecting vertices closer to the source $s_j$ along $T$.

We shall define two log-size variables ReachLeft and ReachRight and initialize them as $\mathrm{StepLeft}(x)$ and $\mathrm{StepRight}(x)$. These two variables store enough information for the explored region. The vertex set $\mathrm{Between}(\mathrm{ReachLeft}, \mathrm{ReachRight})$ is defined as the vertices which are strictly between ReachLeft and ReachRight in the clockwise order of $T_{s_j}$ and the descendants of ReachLeft and ReachRight. Note that this does *not* include the ancestors of ReachLeft and ReachRight.

Of particular interest to the explored region are jump edges with tail in the explored region $\mathrm{Between}(\mathrm{ReachLeft}, \mathrm{ReachRight})$ and head *not* in the explored region. We call these edges *exiting* edges. Note that a jump edge $e$ is exiting if and only if the tree curve at $e$ contains ReachRight.

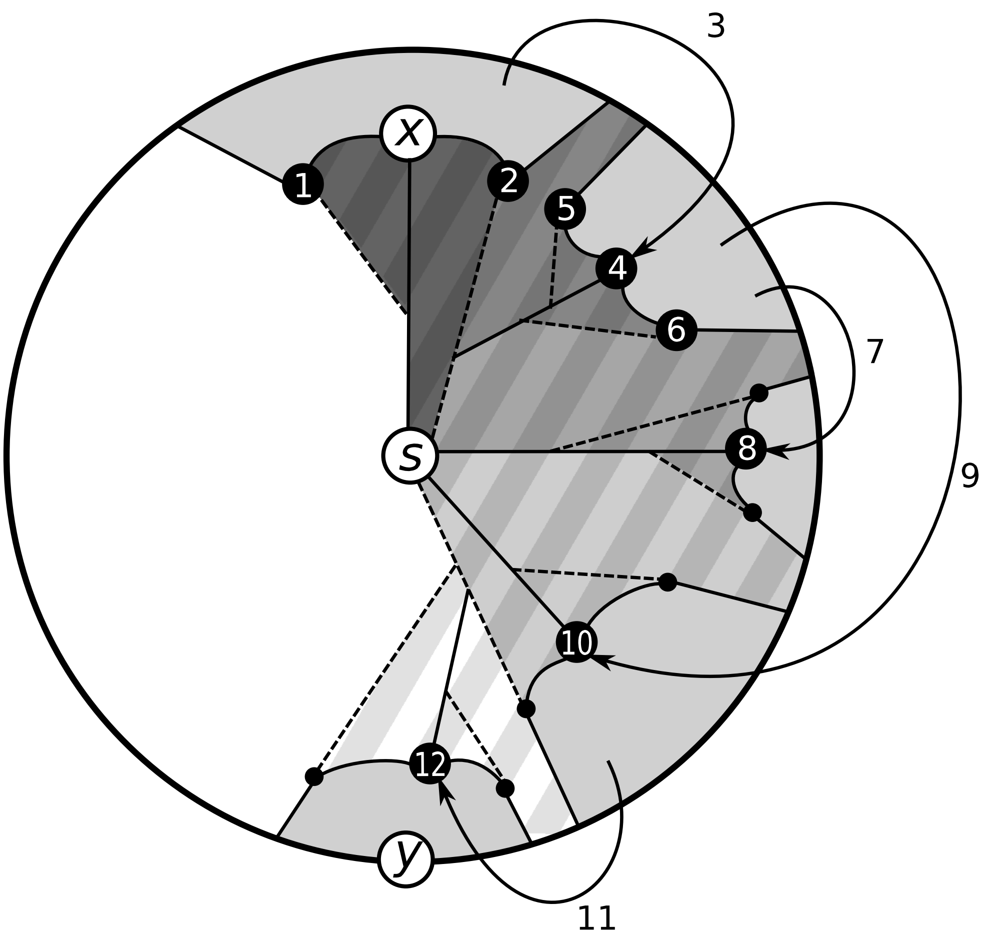
Since each $d$-directional exiting edge contains ReachRight, the exiting edges form a linear order $e_1, e_2, \ldots, e_r$ where $e_i$ is contained within the tree curve on $e_j$ if and only if $i < j$. We shall extend the explored region by using the minimal exiting edge, denoted $e_{\text{jump}}$, and setting ReachRight to $\mathrm{StepRight}(\mathrm{Head}(e_{\text{jump}}))$.

Proceed to extend the explored region until one of two situations arise: if the vertex $y$ is within $\mathrm{ReachStep}(\mathrm{Head}(e_{\text{jump}}))$, we return $\mathsf{True}$; if there are no exiting edges, stop and return $\mathsf{False}$. This process is detailed in Algorithm 1.

The correctness of $\mathrm{ReachLocal}(x, y, d)$ requires the following claim regarding the explored region.

**Claim 21.** *At every stage of Algorithm 1, every exiting edge $e$ has* $\mathrm{Tail}(e)$ *reachable from $x$ using a $d$-directional irreducible path.*

*Proof of Claim.* Without loss of generality, we assume $d = \mathrm{R}$. We proceed by induction on the number of iterations in the execution of $\mathrm{ReachLocal}(x, y, d)$. When ReachLeft and ReachRight are




1. StepLeft($x$)
2. StepRight($x$)
3. $e_{\text{jump}}^{(1)}$
4. Head($e_{\text{jump}}^{(1)}$)
5. StepLeft(Head($e_{\text{jump}}^{(1)}$))
6. StepRight(Head($e_{\text{jump}}^{(1)}$))
7. $e_{\text{jump}}^{(2)}$
8. Head($e_{\text{jump}}^{(2)}$)
9. $e_{\text{jump}}^{(3)}$
10. Head($e_{\text{jump}}^{(3)}$)
11. $e_{\text{jump}}^{(4)}$
12. Head($e_{\text{jump}}^{(4)}$)

The shaded region is the explored region. The flat gray areas are reachable while the striped areas are not. The striped area is darker depending on how many iterations that region was in the explored region.

*Figure 2:* An example execution of ReachLocal($x, y, \mathrm{R}$).

**Algorithm 1** ReachLocal$(x, y, d)$ — Returns $\mathsf{True}$ if and only if $y$ reachable from $x$

```
ReachLeft ← StepLeft(x)
ReachRight ← StepRight(x)
i ← 1
while there exists a d-directional exiting edge do
  e_jump^(i) ← the minimal d-directional exiting edge
  if y ∈ ReachStep(Head(e_jump^(i))) then
    return True
  else if d = Right then
    ReachRight ← StepRight(Head(e_jump^(i)))
  else if d = Left then
    ReachLeft ← StepLeft(Head(e_jump^(i)))
  end if
  i ← i + 1
end while
return False
```

initialized, the explored region consists of vertices within ReachStep$(x)$ and vertices strictly within the curve given by concatenating the following paths:

$$s_j \xrightarrow{T} \text{ReachLeft} \xrightarrow{\text{(local)}} x \xrightarrow{\text{(local)}} \text{ReachRight} \xrightarrow{T} s_j.$$

If a jump edge $e$ has tail within the explored region, then either (1) it is within ReachStep$(x)$ and is reachable, or (2) it is bound by the curve and must not be an exiting edge. Thus, the claim holds for the first iteration.

Assume the claim holds for the $k$th iteration. Consider the next iteration's selection of $e_{\text{jump}}$ and let $e$ be a jump edge with tail within the new explored region. If the tail of $e$ is in the previous explored region, the induction step shows the claim holds. Otherwise, there are only two cases. First, the tail of $e$ is within ReachStep(Head$(e_{\text{jump}})$) and $e$ is reachable since $e_{\text{jump}}$ was reachable by induction. Second, the tail of $e$ is strictly within the curve given by concatenating the following paths:

$$s_j \xrightarrow{T} \text{Tail}(e_{\text{jump}}) \xrightarrow{e_{\text{jump}}} \text{Head}(e_{\text{jump}}) \xrightarrow{(local)} \text{ReachRight} \xrightarrow{T} s_j,$$

and hence the edge $e$ is not exiting. This proves the claim. □

Given the above claim, observe that when ReachLocal$(x, y, d)$ returns $\mathsf{True}$ it is correct, as there is some subset of the $e_{\text{jump}}$ edges which can be combined with local paths to create a path from $x$ to $y$.

To finish, we must prove that if there is a $d$-directional irreducible path from $x$ to $y$ in $\mathcal{R}[T_{s_j}]$, then ReachLocal$(x, y, d)$ returns $\mathsf{True}$. Fix a path from $x$ to $y$ that uses the minimum number jump edges and consider the sequence $e_1, \dots, e_t$ of jump edges within this path. The minimum number of jump edges guarantees that Tail$(e_i) \in$ ReachStep(Head$(e_{i-1})$) and Tail$(e_{i+1}) \notin$ ReachStep(Head$(e_{i-1})$) for all suitable $i \in \{2, \dots, t-1\}$. The first jump edge $e_1$ is an exiting edge for the first explored region.

We claim that at each iteration where $y$ is not in ReachStep(Head$(e_{\text{jump}})$), there is an edge $e_i$ of the path that is an exiting edge. This is given by the choice of $e_{\text{jump}}$ as the mimimal $d$-directional exiting edge. In the previous iteration, there was some $e_i$ that was exiting. If $e_i$ was selected as $e_{\text{jump}}$, then $\text{Tail}(e_{i+1})$ is within $\text{ReachStep}(e_{\text{jump}})$ and $\text{Head}(e_{i+1})$ is not. Since all jump edges are $d$-directional, the edge $e_{i+1}$ is an exiting edge and the claim holds for another iteration.

Suppose that $e_{\text{jump}}$ was not selected to be $e_i$. Then, the tree curve at $e_{\text{jump}}$ is contained within the tree curve at $e_i$. This provides two cases: (1) $\text{Head}(e_i) \notin \text{ReachStep}(\text{Head}(e_{\text{jump}}))$ and $e_i$ is still an exiting edge, or (2) $\text{Head}(e_i) \in \text{ReachStep}(\text{Head}(e_{\text{jump}}))$ and hence $\text{Tail}(e_{i+1}) \in \text{ReachStep}(\text{Head}(e_{\text{jump}}))$. In the latter case it is not immediate that $e_{i+1}$ is an exiting edge, but some edge $e_{i'}$ with $i' > i$ will be an exiting edge, since $y$ is not in $\text{ReachStep}(\text{Head}(e_{\text{jump}}))$. □

## 4 Topological Equivalence

The following notion of topological equivalence plays a central role in our algorithms. It was originally presented in [17] for planar graphs, but we extend it to arbitrary surfaces.

**Definition 22** (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: (a) They span the same source trees in $F$ (assume $x$ and $w$ are on the same tree), (b) 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 of the definition is immediate. Transitivity is implied by the following lemma, which is immediate from the definitions.

**Lemma 23.** *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 $S$, as given in the following corollary.

**Corollary 24.** *Given an equivalence class $E$ of global edges, let $S_E = \bigcup_{e_1, e_2 \in E} D(e_1, e_2)$. The surface $S_E$ is a disk.*

*Proof.* Lemma 23, implies that for any triple $e_1, e_2, e_3 \in E$ and any pair of the disks $D(e_1, e_2)$, $D(e_1, e_3)$, and $D(e_2, e_3)$ are either adjacent or have a containment relationship. There is an ordering $e_1, \ldots, e_k$ of the edges of $E$ so that the disks $D(e_i, e_{i+1})$ pairwise intersect only at boundaries. Gluing the disks $D(e_{i-1}, e_i)$ and $D(e_i, e_{i+1})$ along $e_i$ constructs $S_E$ as a disk. □

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 $i$th equivalence class in this ordering. We shall use this notation to label the equivalence classes.

**Definition 25** (The Region of an Equivalence Class)**.** Let $E_i$ be an equivalence class of global edges. Define the *region enclosed by* $E_i$ as $\mathcal{R}[E_i] = \bigcup_{e_1,e_2\in E_i} D(e_1, e_2)$.

The region $\mathcal{R}[E_i]$ has some properties which are quickly identified. 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 $\mathcal{R}[E_i]$. The *boundary* of $\mathcal{R}[E_i]$ is given by these two edges and their ancestor paths in $F$ on all four endpoints. All vertices in a source tree $T$ are contained in the region $\mathcal{R}[T]$. 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 within the boundary of $\mathcal{R}[E_i]$ are within $\mathcal{R}[T_A]$ and $\mathcal{R}[T_B]$. 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 the trees $T_A$ and $T_B$ are 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. Note that the endpoints of edges in $E_i$ lie on the boundary of the regions $\mathcal{R}[T_A]$ and $\mathcal{R}[T_B]$. 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. 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.

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 26.** *Let $G$ be a graph embedded on a surface $S$ with Euler characteristic $\chi_S$ 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

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

□

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). Hence, the regions $\mathcal{R}[T_{s_j}]$ and $\mathcal{R}[E_i]$ form a ribbon graph, which encodes the entire surface but has only $m$ vertices and $O(m+g)$ edges.

Consider an equivalence class $E_i$ between source trees $T_A$ and $T_B$, a rotational direction $d$ (clockwise or counterclockwise), and a vertex $x$ in $T_A$ outside the region $\mathcal{R}[E_i]$. We say that the vertex $x$ *fully reaches* $E_i$ in the direction $d$ if there is an irreducible $d$-directional local path from $x$ to an endpoint of each edge in $E_i$. If $x$ does not fully reach $E_i$ in direction $d$, but there is a local path from $x$ to an endpoint of some edge of $E_i$, then we say $x$ *partially reaches* $E_i$ in this direction. If such a path is irreducible, then the path follows a clockwise or counter-clockwise direction within $T_A$ and we say $x$ fully (or partially) reaches $E_i$ *using a clockwise (or counter-clockwise) rotation.*

**Lemma 27.** *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*

1. *$x$ fully reaches each $E_{i_j}$ for $j \in \{1, \ldots, \ell - 1\}$.*
2. *$x$ either fully or partially reaches $E_{i_0}$ and $E_{i_\ell}$.*
3. *If $x$ is not in the interior of $\mathcal{R}[E_{i_0}]$, $x$ fully reaches $E_{i_0}$.*

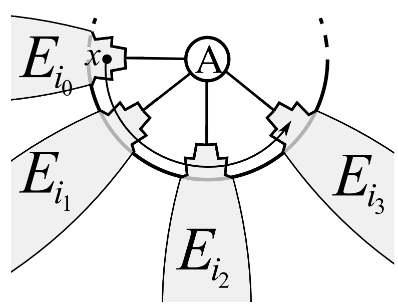


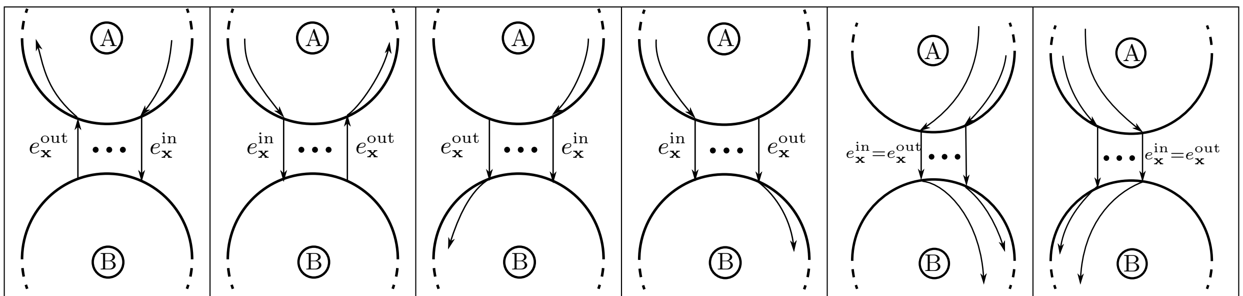

*Figure 3:* A vertex $x$ with three counter-clockwise reachable classes, $E_{i_1}$,, $E_{i_2}$, and $E_{i_3}$, as in Lemma 27.

*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 $E_{i_\ell}$ 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. □

## 5 Global Edges and Patterns

At this point, we take a very different approach than [17]. The algorithm described in [17] focused on reachability within the regions $\mathcal{R}[T]$ on the source trees $T$. Here, we focus on reachability within and between equivalence classes $E_i$. We create a constant number of vertices derived from each equivalence class. This constant is given by the number of distinct ways a path can enter the region $\mathcal{R}[E_i]$, use edges in $E_i$, then leave the region $\mathcal{R}[E_i]$. We call these *patterns*.

**Definition 28** (The Pattern Set)**.** Let $E_i$ be an equivalence 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 $\langle abc \rangle$ with $a, c \in \{\mathrm{L}, \mathrm{R}\}$, $b \in \{\mathrm{S}, \mathrm{X}\}$ where $a$ is the clockwise (R) or counter-clockwise (L) direction the path takes as it enters $\mathcal{R}[E_i]$, $c$ is the direction the path takes as it leaves $\mathcal{R}[E_i]$, and if $b = \mathrm{S}$, the path enters and leaves $\mathcal{R}[E_i]$ on the same end and if $b = \mathrm{X}$, the path enters and leaves $\mathcal{R}[E_i]$ on opposite ends [3]. Define the *pattern set*, $\mathcal{P} = \{\langle \mathrm{RSR} \rangle, \langle \mathrm{LSL} \rangle, \langle \mathrm{RXR} \rangle, \langle \mathrm{RXL} \rangle, \langle \mathrm{LXR} \rangle, \langle \mathrm{LXL} \rangle\}$.




| $\langle$RSR$\rangle$ | $\langle$LSL$\rangle$ | $\langle$RXL$\rangle$ | $\langle$LXR$\rangle$ | $\langle$RXR$\rangle$ | $\langle$LXL$\rangle$ |
|---|---|---|---|---|---|
| *Full Patterns* | | | | *Nesting Patterns* | |

*Table 1:* Different patterns using an edge class $E_i$, entering from the $A$-end of $\mathcal{R}[E_i]$.

Let $E_i$ be an edge class and $\mathcal{R}[E_i]$ be the enclosed region. Let $t$ be an end of $\mathcal{R}[E_i]$ (either $A$ or $B$) and fix an orientation on that end and a pattern $p$ that involves $E_i$. Then the *entrance* (*exit*) of the pattern at the $t$-end is the ancestor path on the boundary of $\mathcal{R}[E_i]$ on the $t$-end that a path must cross *before* (respectively, *after*) using the edges in $E_i$ that induce the pattern $p$ with the given orientation. (See Figure 4 for a visual representation of the entrance and exit of a pattern.)

We can now define *pattern descriptions* which are the vertices of the pattern graph that we will define in the next section.

**Definition 29** (Pattern Descriptions)**.** Let $k$ be the number of topological equivalence classes of edges of $G$. A *pattern description* is a tuple $\mathbf{x} = (i, t, o, p)$ where $i \in \{1, \ldots, k\}$, $t \in \{A, B\}$, $o \in \{+1, -1\}$, and $p \in \mathcal{P}$. Here $i$ represents the equivalence class $E_i$, $t$ represents the end of $\mathcal{R}[E_i]$ that contains the entrance, $o \in \{+1, -1\}$ specifies if the orientation of the path is in agreement with (or opposite to, respectively) the local orientation of the tree on the $t$-side of $E_i$, and $p \in \mathcal{P}$ represents the pattern used in $E_i$. The set $\{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P}$ of all pattern descriptions is denoted by $V_{\mathrm{P}}$.

For example, the description $(i, B, +1, \langle \mathrm{RXL} \rangle)$ is an element in $V_{\mathrm{P}}$ corresponding to a $\langle \mathrm{RXL} \rangle$ pattern, using at least one edge of the class $E_i$ starting at the $B$-side and leaving the $A$-side, oriented

[3] The interested reader will find the notation for patterns derived from move sequences in the Coin Crawl Game from [17].

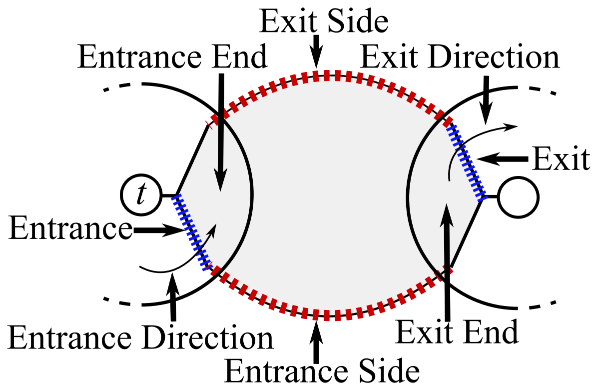



*Figure 4:* Terminology for the entrance and exit of a pattern and the modifiers of *direction*, *end*, and *side*. This example is an ⟨LXR⟩ pattern.

to agree with the $B$-side. Lemma 26 implies the number of descriptions is $O(m + g_S)$ where $m$ is the number of sources and $g$ the genus of the surface. A pattern description can be represented with $\lceil \log k \rceil + 5 = O(\log(m + g_S))$ bits[4].

We now investigate some properties of paths that induce these pattern descriptions. We focus on a path which uses local edges and global edges in a single equivalence class 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 30** (Canonical Edge Pair)**.** Let $\mathbf{x} = (i, t, o, p)$ be a pattern description centered at the edge class $E_i$. There are two edges (*incoming* and *outgoing*) in $E_i$, called the *canonical edge pair* for $\mathbf{x}$. The *outgoing edge*, $e^{\text{out}}_{\mathbf{x}}$, is the edge $e \in E_i$ with head on the exit end that is farthest from the exit side so that there exists a local path from $\text{Head}(e)$ to the exit of $\mathcal{R}[E_i]$. The *incoming edge*, $e^{\text{in}}_{\mathbf{x}}$, is the edge $e \in E_i$ with the tail on the entrance end that is closest to the entrance side so that either $e = e^{\text{out}}_{\mathbf{x}}$ or $\text{Tail}(e^{\text{out}}_{\mathbf{x}})$ is reachable from $\text{Head}(e)$ using local paths and edges in $E_i$.

## 5.1 Full Patterns

Full patterns are named so because a path which induces a full pattern intersects the ancestor path of at least one endpoint of every edge in the class. Hence, every edge is reachable. This leads to the property that if an irreducible path induces such a pattern, then the path might as well use the canonical edges in the corresponding equivalence class.

**Lemma 31.** *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 $\mathbf{x}$. Then 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 $\mathbf{x}$.*

[4]This bland fact is in fact very important for the later use of Savitch's Theorem.

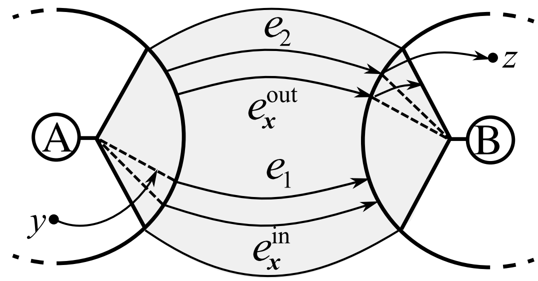



*Figure 5:* The edges used in the proof of Lemma 31 in an ⟨LXR⟩ pattern.

*Proof.* Note that if the tail of $e_{\mathbf{x}}^{\text{in}}$ is reachable from $y$ using a local path in the entrance direction, and $z$ is reachable from the head of $e_{\mathbf{x}}^{\text{out}}$ 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 $e_{\mathbf{x}}^{\text{in}}$ and $e_{\mathbf{x}}^{\text{out}}$ 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 $(e_{\mathbf{x}}^{\text{in}}, e_{\mathbf{x}}^{\text{out}})$ in the ordering of the edges in $E_i$. An example of the edges $e_1$ and $e_2$ are shown in Figure 5.

If $e_1$ is closer to the entrance side of $E_i$ compared to $e_{\mathbf{x}}^{\text{in}}$, then (by the definition of $e_{\mathbf{x}}^{\text{in}}$) there is no path from the head of $e_1$ to the tail of $e_{\mathbf{x}}^{\text{out}}$ 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 $e_{\mathbf{x}}^{\text{out}}$, 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. □

**Lemma 32.** *Let $\mathbf{x}$ be a pattern description of full type. The canonical edge pair $(e_{\mathbf{x}}^{\text{in}}, e_{\mathbf{x}}^{\text{out}})$ is log-space computable.*

*Proof.* The outgoing edge, $e_{\mathbf{x}}^{\text{out}}$, 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 $e_1 = e_{\mathbf{x}}^{\text{out}}$ and set $e_2$ to be null.

Proceed by iterating through the edges in $E_i$ starting at $e_{\mathbf{x}}^{\text{out}}$ 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 $e_{\mathbf{x}}^{\text{out}}$ 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 $e_{\mathbf{x}}^{\text{out}}$ using a local path. If so, then update $e_2$ to this edge.

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

### 5.2 Nesting Patterns

Nesting patterns are named so because irreducible paths which induce such patterns use exactly one edge of this class, 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 properties of nesting patterns.

**Lemma 33.** *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 34.** *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. □

**Lemma 35.** *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* $\text{Head}\left(e_{\mathbf{x}}^{\text{out}}\right)$.

While it would be useful to have a property similar to Lemma 31 for nesting patterns, there may exist a vertex $w$ from which there are paths that induce a nesting pattern without reaching the canonical incoming edge. We can define a new edge in the class that is similarly canonical, except with respect to the vertex $w$.

**Definition 36** (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 (a) there is a local path from $w$ to $\text{Tail}(e)$ in the entrance direction, and (b) there is a local path from $\text{Head}(e)$ to the exit boundary of $\mathcal{R}[E_i]$.

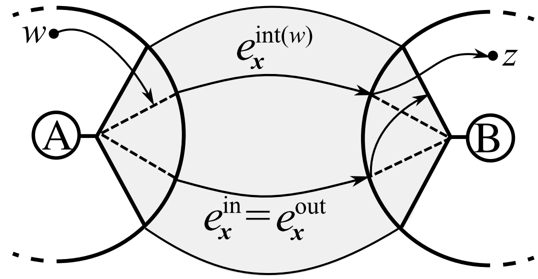



*Figure 6:* The most-interior edge from a vertex $w$ in a pattern description $\mathbf{x}$ with an $\langle$RXR$\rangle$ pattern.

**Lemma 37.** *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,* $e_{\mathbf{x}}^{\mathrm{int}(w)}$, *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* $\mathrm{Head}\left(e_{\mathbf{x}}^{\mathrm{int}(w)}\right)$ *to* $z$ *in the exit direction of* $\mathbf{x}$. *If* $w$ *fully reaches* $E_i$, *then* $e_{\mathbf{x}}^{\mathrm{int}\,w} = e_{\mathbf{x}}^{\mathrm{out}}$.

*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 $e_{\mathbf{x}}^{\mathrm{int}(w)}$ 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 33, 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 $e_{\mathbf{x}}^{\mathrm{int}(w)}$ or else either $w$ does not reach $\mathrm{Tail}(e)$ or $\mathrm{Head}(e)$ does not reach the exit of $\mathcal{R}[E_i]$. The path that exits the class $E_i$ from the head of $e_{\mathbf{x}}^{\mathrm{int}(w)}$ 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 $e_{\mathbf{x}}^{\mathrm{int}(w)}$ and so is anything reachable from the head of $e$, including $z$.

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

## 6 The Pattern Graph

We now describe a graph on $O(m + g_S)$ vertices that preserves $uv$-reachability.

**Definition 38** (The Pattern Graph)**.** Given $G$ and $F$ as above, the *pattern graph*, denoted $\mathrm{P}(G, F) = (V'_{\mathrm{P}}, E'_{\mathrm{P}})$ is a directed graph defined as follows. The vertex set $V'_{\mathrm{P}} = \{u', v'\} \cup V_{\mathrm{P}} = \{u', v'\} \cup (\{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P})$. For two pattern descriptions $\mathbf{x}, \mathbf{y} \in V_{\mathrm{P}}$, an edge $\mathbf{x} \to \mathbf{y}$ is in $E'_{\mathrm{P}}$ 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 $\mathrm{Head}(e_{\mathbf{x}}^{\mathrm{out}})$ to $\mathrm{Tail}(e_{\mathbf{y}}^{\mathrm{in}})$ which induces the sequence $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ of nesting pattern descriptions.
2. For each $j \in \{1, \ldots, \ell\}$, $\mathrm{Tail}(e_{\mathbf{z}_j}^{\mathrm{in}})$ is not reachable from $\mathrm{Head}(e_{\mathbf{x}}^{\mathrm{out}})$ using irreducible paths that induce the pattern descriptions $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$.

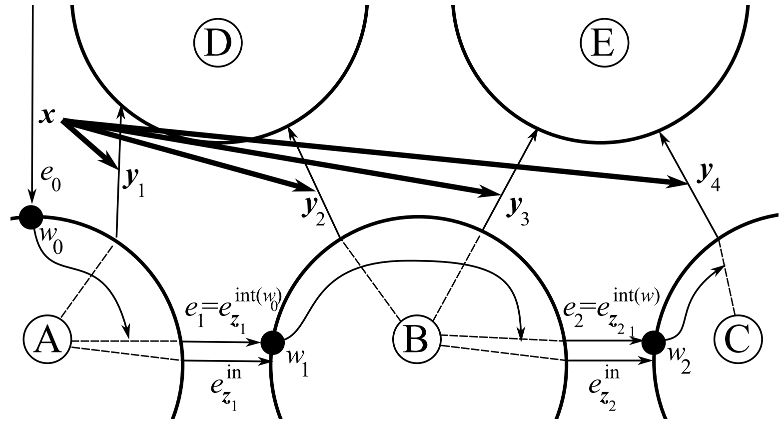



*Figure 7:* The nesting patterns $\mathbf{z}_1$ and $\mathbf{z}_2$ satisfy the adjacency conditions in Definition 38 from $\mathbf{x}$ to each $\mathbf{y}_j$. The pattern adjacencies are enumerated during the algorithm of Lemma 40 where $e$ is assigned to $e_0$, $e_1$, and $e_2$, sequentially. Note that $e_0 = e_{\mathbf{x}}^{\mathrm{out}}$, $e_1 = e_{\mathbf{z}_1}^{\mathrm{int}(\mathrm{Head}(e_0))}$, and $e_2 = e_{\mathbf{z}_2}^{\mathrm{int}(\mathrm{Head}(e_1))}$. 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 40 terminates at $e_2$, since $e_2$ does not give a partially-reachable class.

In addition, for a description $\mathbf{x} = (i, t, o, p)$ there is an edge $u' \to \mathbf{x}$ in $E'_{\mathrm{P}}$ if and only if $\mathbf{x}$ has the $t$-end in the tree $T_u$. Also, for a pattern description $\mathbf{x} = (i, t, o, p)$ there is an edge $\mathbf{x} \to v'$ in $E'_{\mathrm{P}}$, 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 39.** *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.* ($\Rightarrow$) Let $P$ be an irreducible path from $u$ to $v$ in $G$. $P$ 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 31 implies that we may assume the first edge in the global edge class of $\mathbf{x}_i$ used by $P$ is $e_{\mathbf{x}_i}^{\mathrm{in}}$ and the last such edge is $e_{\mathbf{x}_i}^{\mathrm{out}}$.

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 31, $e_{\mathbf{x}_j}^{\mathrm{in}}$ is reachable from $e_{\mathbf{x}_i}^{\mathrm{out}}$ 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 $\mathrm{Tail}(e_{\mathbf{z}_{i'}}^{\mathrm{in}})$ is reachable from $\mathrm{Head}(e_{\mathbf{x}_i}^{\mathrm{out}})$ by a path using the nesting patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{i'-1}$ as the adjacency certificate. By Lemma 37, $\mathrm{Tail}(e_{\mathbf{x}_j}^{\mathrm{in}})$ is reachable from $\mathrm{Head}(e_{\mathbf{z}_{i'}}^{\mathrm{out}})$ 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 $\mathrm{P}(G, F)$.

($\Leftarrow$) 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 $e^{\mathrm{out}}_{\mathbf{x}_1}$. 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 31 and 37 imply there is an irreducible path $P_i$ in $G$ from the head of $e^{\mathrm{out}}_{\mathbf{x}_j}$ to the tail of $e^{\mathrm{in}}_{\mathbf{x}_{j+1}}$ that is either a local path or induces a list of nesting pattern descriptions which form an adjacency certificate. Also, by Definition 30, there exist (possibly empty) paths $Q_j$ from $e^{\mathrm{in}}_{\mathbf{x}_j}$ to $e^{\mathrm{out}}_{\mathbf{x}_j}$ using local paths and edges of the class $E_{i_j}$. These paths concatenate to a path $uP_0e^{\mathrm{out}}_{\mathbf{x}_1}P_1e^{\mathrm{in}}_{\mathbf{x}_2}Q_2e^{\mathrm{out}}_{\mathbf{x}_2}P_2e^{\mathrm{in}}_{\mathbf{x}_3}\ldots e^{\mathrm{out}}_{\mathbf{x}_{\ell-1}}P_{\ell-1}e^{\mathrm{in}}_{\mathbf{x}_\ell}v$ from $u$ to $v$ in $G$. □

**Lemma 40.** *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 $\mathrm{P}(G,F)$. It is simple to find the pattern descriptions $\mathbf{x}, \mathbf{y}$ so that $u \to \mathbf{x}$ and $\mathbf{y} \to v$.

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 27, 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 $e^{\mathrm{in}}_{\mathbf{y}}$ 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 $E_{i_\ell}$ can be checked if $e^{\mathrm{in}}_{\mathbf{y}}$ is reachable from $e$. The only pattern that could be used without having $e^{\mathrm{in}}_{\mathbf{y}}$ reachable is a nesting pattern.

To enumerate all neighbors of $\mathbf{x}$ in $\mathrm{P}(G,F)$, perform the above subroutine on $e^{\mathrm{out}}_{\mathbf{x}}$, adding edges from $\mathbf{x}$ to each reachable pattern description $\mathbf{y}$. If the nesting pattern $\mathbf{z}$ on $E_{i_\ell}$ is not fully reachable (i.e. there is no local path from $e$ to $e^{\mathrm{in}}_{\mathbf{z}}$ in the proper direction) then compute the most-interior edge $e^{\mathrm{int}(\mathrm{Head}(e))}_{\mathbf{z}}$. Repeat the subroutine on this edge, continuing until the class $E_{i_\ell}$ is fully reachable (or the list is empty). In the $j$th iteration, let $w_{j-1} = \mathrm{Head}(e)$ and $\mathbf{z}_j = \mathbf{z}$. See Figure 7 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. Each class $\mathbf{z}_j$ has the edge $e^{\mathrm{in}}_{\mathbf{z}_j}$ *not* reachable from $\mathbf{x}$ using the patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$. This means that the pattern $\mathbf{z}_j$ is centered at the class $E_{i_\ell}$ computed by the iteration of the subroutine on the edge $e^{\mathrm{int}(w_{j-1})}_{\mathbf{z}_{j-1}}$. 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. □

**Theorem 41** (Main Theorem)**.** *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 $\mathrm{P}(G,F)$. Theorem 39 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 40 gives that $G'$ is log-space computable. By Lemma 26, 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. $\square$

## 7 Discussion

We have succeeded in enlarging the class of surface-embedded DAGs which admit deterministic log-space algorithms for reachability. By extending the concept of topological equivalence from [17], we have shown that this is a useful algorithmic construct. Perhaps the structures built in this paper have application to other problems. Placing planar DAG reachability within $\mathsf{L}$ will likely require significant new ideas since the source-to-genus tradeoff hints that an algorithm for $m$-source planar DAGs will also apply to $m$-genus DAGs.

Further, the algorithms developed in this work improve upper bounds for the class $\mathcal{G}(m,g)$ for sub-polynomial values of $m$ and $g$. 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.


## Acknowledgments

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.

The first author is supported in part by the NSF grants CCF-0916525 and DMS-0914815. The second author is supported in part by the NSF grants CCF-0830730 and CCF-0916525.


## References


[1] E. Allender. Reachability problems: An update. *Computation and Logic in the Real World*, pages 25–27, 2007.

[2] E. Allender, D. A. M. Barrington, T. Chakraborty, S. Datta, and S. Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

[3] E. Allender and K.-J. 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).

[4] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. 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.

[5] J. Chen, S. Kanchi, and A. Kanevsky. A note on approximating graph genus. *Information processing letters*, 61(6):317–322, 1997.


[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 reach-poly graphs is decidable by a $\mathsf{LogDCFL}$ machine.

| Ealier known graph class | Space bound $s$ | New graph class given by Theorem 3 |
|---|---|---|
| Undirected Graphs [13]<br>SMPD[4] [2]<br>LMPD[5] [17] | $O(\log n)$ | $\mathcal{G}\left(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})}\right)$ |
| Poly-mixing time [14,15] | $O\left(\log^{\frac{3}{2}} n\right)$ | $\mathcal{G}\left(2^{O(\log^{\frac{3}{4}} n)}, 2^{O(\log^{\frac{3}{4}} n)}\right)$ |
| Reach-poly graphs [3,7] | $O\left(\frac{\log^2 n}{\log\log n}\right)$ | $\mathcal{G}\left(2^{O\left(\frac{\log n}{\sqrt{\log\log n}}\right)}, 2^{O\left(\frac{\log n}{\sqrt{\log\log n}}\right)}\right)$ |
| | $o(\log^2 n)$ | $\mathcal{G}(n^{o(1)}, n^{o(1)})$ |
| All directed graphs [16] | $O(\log^2 n)$ | |

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

| Earlier known graph class | Space bound $s$ with poly-time | New graph class given by Theorem 8 |
|---|---|---|
| Poly-mixing time [12,14]<br>Reach-poly graphs[6] [6,11] | $O(\log^2 n)$ | |
| | $2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}$ | $\mathcal{G}\left(2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}, 2^{O\left(\log^{\frac{1}{2}+\varepsilon} n\right)}\right)$ |
| | $o(n^{\varepsilon})$ | $\mathcal{G}(O(n^{\varepsilon}), O(n^{\varepsilon}))$. |
| All directed graphs [4] | $O\left(\frac{n}{2^{\sqrt{\log n}}}\right)$ | |

*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 $s$.

[6] S. 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.

[7] B. Garvin, D. Stolee, R. Tewari, and N. V. Vinodchandran. Reachfewl = reachul. *17th Annual International Computing and Combinatorics Conference*, 2011. to appear.

[8] A. Jakoby, M. Liśkiewicz, and R. Reischuk. Space efficient algorithms for directed series-parallel graphs. *Journal of Algorithms*, 60(2):85–114, 2006.

[9] A. Jakoby and T. 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.

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

[11] K.-J. 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.

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

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

[14] O. Reingold, L. Trevisan, and S. 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.

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

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

[17] D. Stolee, C. 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.

[18] C. Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

[19] A. Wigderson. The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132, 1992.