Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 148 (2005)

The Directed Planar Reachability Problem

Eric Allender[1], Samir Datta[2], and Sambuddha Roy[3]

[1] Department of Computer Science, Rutgers University, Piscataway, NJ 08855, allender@cs.rutgers.edu
[2] Chennai Mathematical Institute, Chennai, TN 600 017, India sdatta@cmi.ac.in
[3] Department of Computer Science, Rutgers University, Piscataway, NJ 08855, samroy@paul.rutgers.edu

Abstract. We investigate the s-t-connectivity problem for directed planar graphs, which is hard for L and is contained in NL but is not known to be complete. We show that this problem is logspace-reducible to its complement, and we show that the problem of searching graphs of genus 1 reduces to the planar case.
We also consider a previously-studied subclass of planar graphs known as *grid graphs*. We show that the directed planar s-t-connectivity problem reduces to the reachability problem for directed grid graphs.
A special case of the grid-graph reachability problem where no edges are directed from right to left is known as the "layered grid graph reachability problem". We show that this problem lies in the complexity class UL.

1 Introduction

Graph reachability problems play a central role in the study and understanding of subclasses of P. The s-t-connectivity problem for directed graphs (STCONN) is complete for nondeterministic logspace (NL); the restriction of this problem to undirected graphs, called USTCONN, has recently been shown to be complete for logspace (L) [Rei05]; thus this problem has the same complexity as the s-t-connectivity problem for graphs of outdegree 1 (and even for graphs of indegree and outdegree at most 1 [Ete97]).

Grid graphs are an important restricted class of graphs for which the reachability problem has significant connections to complexity classes. (The vertices in a grid graph are a subset of $\mathbb{N} \times \mathbb{N}$, and all edges are of the form $(i, j) \rightarrow (i + b, j)$ or $(i, j) \rightarrow (i, j + b)$, where $b \in \{1, -1\}$.) In most settings (and in particular in all of the results we will present in this paper) it is sufficient to restrict attention to grid graphs where the start vertex s lies in the first column, and the terminal vertex t lies in the final column. In [BLMS98], Barrington et al showed that the reachability problem in (directed or undirected) grid graphs of width k captures the complexity of depth k AC^0. Barrington also considered general grid graphs without the width restriction, calling this the Grid Graph Reachability problem (GGR) [Bar02]. The construction of [BLMS98, Lemma 13] shows that GGR reduces to its complement via uniform projections. (The problems STCONN and USTCONN also reduce to their complements via uniform projections, as a consequence of [Imm88,Sze88,Rei05,NTS95].) Reachability problems for grid graphs have proved easier to work with than the corresponding problems for general graphs. For instance, the reachability problem for *undirected* grid graphs was shown to lie in L

ISSN 1433-8092

in the 1970's [BK78], although more than a quarter-century would pass before Reingold proved the corresponding theorem for general undirected graphs.

Barrington also defined what we will refer to as the *layered* grid graph reachability problem LGGR, in which no edges are directed from right to left in the grid. (That is, there is no edge of the form $(i, j) \rightarrow (i, j-1)$; we use the convention that vertex (i, j) refers to the point in row i and column j.) Barrington originally called these graphs "acyclic" grid graphs, because this is a simple syntactic condition guaranteeing that a grid graph will have no cycles. (However, this terminology was confusing because there are grid graphs without cycles that do not meet this syntactic criterion.) In personal communication, Barrington suggested the name "layered", with the following justification. It is shown in [Bar02] that this problem is equivalent to the special case where all edges are directed from left to right or from top to bottom. Thus without loss of generality, the start node is in the top left corner. If such a grid graph is rotated 45 degrees counterclockwise, one obtains a graph whose "columns" correspond to the diagonals of the original graph, where s is the only node in the first "column", and all edges in one column are directed "northeast" or "southeast" to their neighbors in the following column. This is consistent with the usual usage of the word "layered" in graph theory.

Barrington showed that GGR and LGGR are hard for NC^1 under uniform projections [Bar02], but the best upper bound that was identified by Barrington for these problems is NL.

Our focus in this paper is the restriction of STCONN to planar (directed) graphs PLANAR.STCONN. This problem is hard for L under uniform projections, as a consequence of [Ete97], and it lies in NL. Nothing else has been published regarding its computational complexity. Thus the class of problems $\leq_m^{\log}$-reducible to PLANAR.STCONN can be viewed as a complexity class lying between L and NL. We show that this class is closed under complement, by presenting a $\leq_m^{\log}$ reduction of PLANAR.STCONN to its complement; we do not know if this reduction can be accomplished by uniform projections or even by NC^1 reductions; in contrast to the case for STCONN, USTCONN, and GGR. We also show that this class contains the s-t-connectivity problem for graphs of genus 1; the generalization for graphs of higher genus remains open.

We have two separate proofs of closure under complement. We present a direct proof in Section 3. In Section 4 we present a reduction showing PLANAR.STCONN $\leq_m^{\log}$ GGR. This also proves closure under complement, because [BLMS98, Lemma 13] shows that GGR reduces to its complement.

Our final technical contribution is to show that LGGR lies in the complexity class UL. This must be viewed as a slight improvement, since it is shown in [ARZ99] that NL $=$ UL if there is any problem in DSPACE(n) that requires circuits of exponential size, and it is shown in [RA00] that NL/poly = UL/poly (unconditionally). We actually show that LGGR lies in UL$\cap$coUL, although (in contrast to all of the other reachability problems we consider) it remains open if LGGR reduces to its complement. (Note also that it remains open if UL $=$ coUL.) Some other examples of reachability problems in UL were presented by Lange [Lan97]; these problems are obviously in UL (in the sense that the positive instances consist of certain graphs that contain only one path from s to t), and the main contribution of [Lan97] is to present a *completeness* result for a natural

subclass of UL. In contrast, positive instances of LGGR can have many paths from s to t. We know of no reductions (in either direction) between LGGR and the problems considered in [Lan97].

Series-parallel graphs are an important and well-studied subclass of planar directed graphs. Jakoby, Liskiewicz, and Reischuk showed in [JLR01] that s-t-connectivity in series-parallel graphs can be computed in logspace (and in fact is complete for L). They also show the much stronger result that *counting* the number of paths between s and t can be computed in logspace for series-parallel graphs. Very recently, in joint work with David Mix Barrington and Tanmoy Chakraborty, we have identified some even larger classes of planar directed graphs for which s-t-connectivity can be solved in logspace; these results will be described in a subsequent paper.

2 Reduction to a Special Case

In this section we present a reduction showing that it is sufficient to consider the special case where the vertices s and t both lie on the external face of the planar graph. This was useful in our direct proof of closure under complement, but it is also useful in presenting our reduction to grid-graph reachability.

Let G be a directed graph. Testing if G is planar reduces to the undirected s-t-connectivity problem [AM04] and thus can be done in logarithmic space [Rei05]. Furthermore, if a graph is planar then a planar combinatorial embedding (i.e., a cyclic ordering of the edges adjacent to each vertex) can be computed in logarithmic space [AM04]. Given a combinatorial embedding, it is easy to check if two vertices lie on the same face. (The vertices on each face adjacent to a vertex v can be enumerated by starting at some (undirected) edge adjacent to v and starting a walk from v along that edge; each time a new vertex w is entered along some edge e the walk continues along the edge that succeeds e in the cyclic ordering of edges around w.) Thus in logspace we can check if s and t lie on the same face. If so, then the graph G is already in the desired form, since we can consider any face to be the "external" face in the embedding.

If s and t do not lie on the same face, then by use of the undirected connectivity algorithm we can determine if there is an undirected path from s to t. If there is no such path, then clearly there is no directed path, either. Otherwise (as observed in [AM04]) we can find a simple undirected path $\mathcal{P} = (s, v_1, v_2, \ldots, v_m, t)$ in logspace. First, we construct a new face with s and t on it, by "cutting" along the path $\mathcal{P}$. (That is, we replace each vertex v_i on $\mathcal{P}$ by vertices $v_{i,a}$ and $v_{i,b}$. For any vertex v_i on $\mathcal{P}$, let u and x be the vertices appearing before and after v_i on $\mathcal{P}$; that is, $u \in \{s, v_{i-1}\}$ and $x \in \{t, v_{i+1}\}$. Let $e_1, \ldots, e_{d_a}$ be the edges embedded "above" the edges connecting v_i to u and x in the cyclic ordering around v_i, and let $e'_1, \ldots, e'_{d_b}$ be the edges embedded "below" the edges between v_i and u and x. That is, if the undirected path from s to t moves from left to right, edges $e_1, \ldots, e_{d_a}$ appear on the left side of this path, and edges $e'_1, \ldots, e'_{d_b}$ appear on the right side. Let $\mathcal{L}$ be the set of all edges adjacent to $\mathcal{P}$ embedded on the left side, and let $\mathcal{R}$ be the set of all edges adjacent to $\mathcal{P}$ embedded on the right side. In the new graph, the edges in $\mathcal{L}$ that were connected to v_i are connected to $v_{i,a}$ and those in $\mathcal{R}$ are connected to $v_{i,b}$. Edges between v_i and $\{v_{i+1}, v_{i-1}$ are duplicated, with edges between $v_{i,c}$ and $\{v_{i+1,c}, v_{i-1,c}\}$ for $c \in \{a, b\}$. Similarly, edges

between s and v_1 (and t and v_m) are duplicated, with edges between s and $v_{1,a}$ and $v_{1,b}$ (and edges between t and $v_{m,a}$ and $v_{m,b}$, respectively). This is illustrated in Figure 1.)

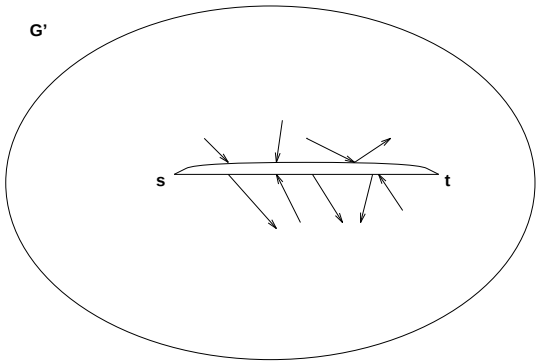

Fig. 1. Cutting along a st path

This new graph G' is planar, and has vertices s and t on the same face (the only new face created). Since we can embed any planar graph such that any specific face is the outer face, we re-embed our graph G' such that s and t are now on the outer face. From now on we assume G' has this embedding.

In the process of going from G to G' we have changed the connectivity of the graph; s and t might have been connected in G but they might not be connected in G'. In particular, any directed path in G from s to t that uses edges from *both* $\mathcal{L}$ and $\mathcal{R}$ is not replicated in G'. We solve this problem by pasting together copies of the graph G', as follows. The outer face of G' consists of two undirected paths from s to t: $s, v_{1,a}, v_{2,a}, \ldots, v_{m,a}, t$ and $s, v_{1,b}, v_{2,b}, \ldots, v_{m,b}, t$. The operation of "pasting" two copies of G' together consists of identifying the vertices $v_{1,a}, v_{2,a}, \ldots, v_{m,a}$ in one copy with the vertices $v_{1,b}, v_{2,b}, \ldots, v_{m,b}$ in the other copy. (Note that this amounts to "sewing together" two copies of the path that were "cut apart" in creating G' from G.) The graph G'' consists of $2n+1$ copies of G' pasted together in this way: the "original copy" in the middle, and n copies pasted in sequence to the top boundary of the outer face, and n copies pasted in sequence to the bottom boundary.

G'' has (the original copies of) s and t on the outer face. A simple inductive argument shows that there is a directed path from s to t in G if and only there is a directed path from (the original copy of) s to one of the copies of t in G''. A pathological example showing that many copies of G' are needed is shown in Figure 2. To complete the reduction, we construct a graph H that consists of G'' along with a new vertex t'' with directed edges from each copy of t to t''. The vertices s and t'' appear on the external face of H, and there is a directed path from s to t in G if and only if there is a directed path from s to t'' in H.

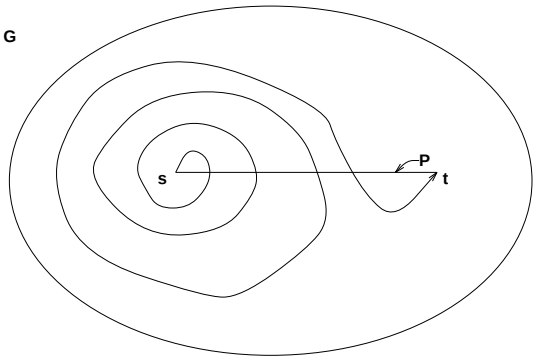

Fig. 2. A pathological case

3 Closure Under Complement

.

In this subsection we reduce the s-t-connectivity problem for directed planar graphs with s and t on the outer face to the complement of the s-t-connectivity problem for directed planar graphs. It has recently come to our attention that a very similar theorem and proof appear in [HRS93][Theorem 3].

First, however, we define some terms more carefully. As observed in the preceding subsection, it is easy in logspace to enumerate all of the vertices that appear on a given face of a planar combinatorial embedding. Note however that the edges that connect these vertices around the face might *not* constitute a simple (undirected) cycle. However, for any *bounded* face F of a planar embedding, there is always a simple undirected cycle called the *boundary* of F. Observe that the edges appearing on the boundary of F can also be enumerated in logspace (because they are the only edges that are traversed in only one direction while enumerating all of the vertices that appear on the face F). With this observation in hand, we proceed with our discussion of the special case.

We are given a directed graph G embedded in the plane with the source and sink vertices s and t on the outer face. The basic intuition is that if there is no directed path between s and t in G then there is a proof of this fact, consisting of a geometric curve $\mathcal{C}$ slicing across G with s on one side of $\mathcal{C}$ and t on the other side, where all of the edges crossing $\mathcal{C}$ are directed away from t; thus no path can start at s and reach t. We give a formal definition of such a curve (which we call a *separating cut*), and observe that there is a separating cut if and only if there is not a path from s to t in G. Then we show how to construct a directed planar graph H such that there is a separating cut for G if and only if there is a directed path from s' to t' in H.

Definition 3.1. *A* separating cut *for G is a simple closed curve $\mathcal{C}$ in the plane, with s embedded in the interior of $\mathcal{C}$, and t embedded in the exterior of $\mathcal{C}$, where $\mathcal{C}$ intersects no vertex of G, and all edges of G that intersect $\mathcal{C}$ are directed from the exterior to the interior of $\mathcal{C}$.*

Lemma 3.2. *Given a planar directed graph G with vertices s and t on the outer face, there is a separating cut if and only if if there is no directed path from s to t in G.*

Proof. It is obvious that if there is a path from s to t then there is no separating cut. We will show that the converse holds.

We are assuming that the directed graph G does not have a path from s to t. Since s and t are on the external face, the external face consists of two (undirected) paths from s to t. Call one of these paths the *upper boundary*, and call the other path the *lower boundary* of the graph. (We will consider s to be embedded to the left of t, so traversing the upper boundary from s to t is in a clockwise direction, and traversing the lower boundary from s to t proceeds counterclockwise. Note also that certain edges might appear on *both* the top and bottom boundaries.) Let u and v be two adjacent vertices on the upper boundary, such that u is reachable from s via a directed path (possibly *not* using edges from the upper boundary) and none of the vertices between v and t are reachable from s. (Such a pair must exist, since t is not reachable, and s is.) See Figure 3.

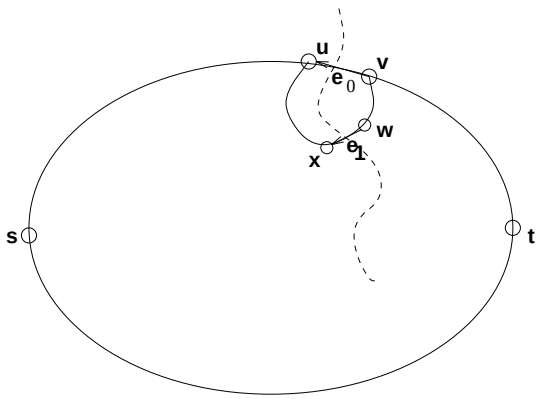

Fig. 3. Graph G with u, v

Clearly the edge e_0 between u and v is directed from v to u. Except for the trivial case where e_0 lies on both the upper and lower boundaries, the edge e_0 forms part of the boundary of some bounded face F_0, and is directed in a counterclockwise direction around F_0. Since u and v both lie on the boundary of F_0, and u is reachable from s and v is not, there is some pair of adjacent vertices w and x on the boundary of F_0 such that x is not reachable from s, and all of the vertices between w and u are reachable from s. Clearly the edge e_1 between w and x is directed from x to w and thus is directed clockwise around the boundary of F_0.

We now define a separating cut $\mathcal{C}$ that starts in the external face and crosses over e_0 into F_0 and leaves F_0 over edge e_1. We now find ourselves either in the external face or in some other bounded face F_1, where e_1 is oriented counterclockwise around the boundary of F_1. This is illustrated by the dotted line in Figure 3.

We will establish some invariants regarding the curve $\mathcal{C}$, and for this it will be necessary to talk about the *right side* and the *left side* of $\mathcal{C}$. Just as a river has a right bank and a left bank, so also $\mathcal{C}$ passes vertices to its right and left as it proceeds away from the

upper boundary. Thus u and w are on the right side of $\mathcal{C}$ and v and x are on the left side. Note that for each edge e_i that $\mathcal{C}$ crosses, the vertex on the right side of $\mathcal{C}$ is reachable from s and the vertex on the left side is not. Also, for each bounded face F_i that $\mathcal{C}$ enters, the edge e_i is oriented counterclockwise around F_i, and e_{i+1} is oriented clockwise around F_i, and all vertices to the right of $\mathcal{C}$ on the boundary of F_i are reachable from s.

Using these invariants, it is clear how to define e_2. We continue curve $\mathcal{C}$ over F_1 to cross over e_2, where we now find ourselves either in the external face or in some bounded face F_2. In this way, we define e_3, F_3, etc., and extend $\mathcal{C}$ across them. We claim that $\mathcal{C}$ never crosses itself, and that in a finite number of steps the curve $\mathcal{C}$ reaches the external face by crossing an edge on the lower boundary. When we do, we continue through the external face around s and close the curve where we started above the upper boundary. Thus the proof will be complete once we establish this claim.

In order to prove that that $\mathcal{C}$ never crosses itself, consider the *first* time $\mathcal{C}$ revisits some face F_i. (That is, consider any i and the least $j > i$ such that $F_i = F_j$.) Assume also that up to this point $\mathcal{C}$ has not crossed the same edge twice, nor crossed itself. We will show that $\mathcal{C}$ can continue on without crossing itself or crossing the same edge twice. This is illustrated in Figures 4 and 5. The previous time that $\mathcal{C}$ crossed F_i it entered across edge e_i and exited across edge e_{i+1}. Our invariants tell us that all of the vertices on the boundary of F_i in the region running counterclockwise between e_i and e_{i+1} are reachable from s, and also that, when curve $\mathcal{C}$ comes back to $F_i = F_j$ by crossing edge e_j, one of the endpoints of e_j is not reachable from s. Thus e_j must appear in the region running *clockwise* between e_i and e_{i+1} (and hence the situation illustrated in Figure 4 cannot arise). The edge e_{i+1} is not equal to e_j (because they are oriented in opposite directions) and by induction $e_i \neq e_j$. Edge e_{j+1} is found by proceeding counterclockwise around F_j from e_j and stopping when a vertex is encountered that is not reachable from s. Thus e_{j+1} will be found in the region strictly between e_j and e_i, and thus $\mathcal{C}$ can cross from e_j to e_{j+1} without intersecting its earlier traversal from e_i to e_{i+1}.

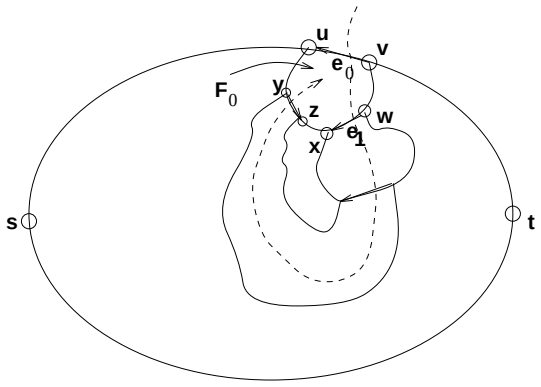

Fig. 4. $\mathcal{C}$ crossing itself from left.

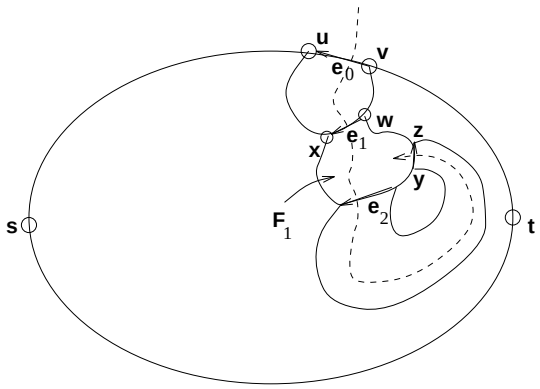

Fig. 5. $\mathcal{C}$ crossing itself from right.

$\mathcal{C}$ might return yet again to $F_i = F_j = F_k$, entering F_i along some edge e_k. Retaining our inductive hypothesis, the same argument as above shows that e_k cannot appear between e_i and e_{i+1} or between e_j and e_{j+1}, and it cannot appear between e_{i+1} and e_j because this would require $\mathcal{C}$ to cross itself, contrary to our inductive hypothesis. Thus e_k must appear in the segment of the boundary of F_i running counterclockwise between e_{j+1} and e_i. The same argument as with e_{j+1} shows that e_{k+1} must appear in the segment of the boundary of F_i running counterclockwise between e_k and e_i, and thus $\mathcal{C}$ can enter F_i across e_k and exit over e_{k+1} without crossing itself. An inductive argument shows that no matter how many times $\mathcal{C}$ returns to face F_i, the edges it uses to enter and exit F_i form a sequence running counterclockwise around the boundary.

We have now established that the curve $\mathcal{C}$ never crosses itself, and it never crosses the same edge twice. Since the graph contains only a finite number of edges, it is clear that $\mathcal{C}$ will enter the external face after a finite number of steps. In order to establish our claim and complete the proof, it remains only to show that when $\mathcal{C}$ does enter the external face, it must happen on the lower boundary.

If $\mathcal{C}$ were to enter the external face along the upper boundary, then it cannot happen between t and e_0, because none of the vertices in that segment of the upper boundary are reachable from s, whereas each edge that $\mathcal{C}$ is adjacent to a vertex that is reachable from s. On the other hand, if $\mathcal{C}$ were to enter the external face along the upper boundary between s and e_0, then $\mathcal{C}$ could be extended to return to its starting point, with the property that for each directed edge (u, v) that crosses $\mathcal{C}$, the vertex v on the left side of $\mathcal{C}$ (the exterior) is not reachable from s, but the vertex u on the right side (the interior) of $\mathcal{C}$ is reachable from s. This is a contradiction, since every path from s to the interior of $\mathcal{C}$ would have to pass through a vertex that is not reachable from s. (See Figure 6.) This completes the proof of the claim, and hence the lemma is proved.

We can now present the details of our reduction.

We are given a directed graph G embedded in the plane with the source and sink vertices s and t on the outer face. In logspace, we can check if there is an edge that appears on both the upper and lower boundaries of G that is directed away from t and toward s on these boundaries. If this is the case, then clearly there is no path from s to

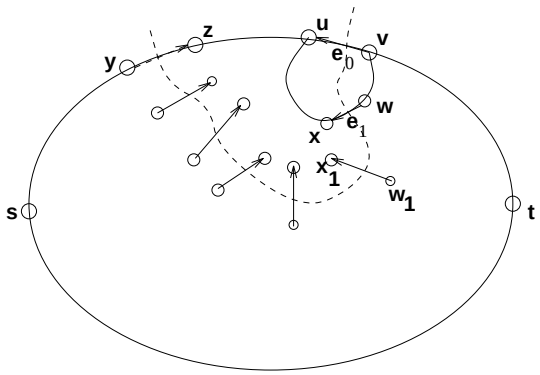

Fig. 6. Curve $\mathcal{C}$ returning to the upper boundary

t. If this is not the case, then we construct the graph H with vertices $\{s', t'\} \cup \{F : F$ is a bounded face of the planar embedding of $G\}$. There are directed edges of the form (s', F) for all vertices F such that F contains an edge on the upper boundary of G that is directed away from t. There are directed edges of the form (F, t') for all vertices F such that F contains an edge on the lower boundary of G that is directed away from t. There are directed edges (F, F') if there is an edge e that is directed clockwise around the boundary of F and counterclockwise around the boundary of F'.

It is clear that H is a planar graph, because (except for s' and t') it is a subgraph of the dual of G.

The proof is completed by showing

- Given G, an encoding of H can be constructed in logarithmic space.
- There is a path from s' to t' in H if and only if there is no path from s to t in G.

In order to construct an encoding for H, it is necessary to have a scheme for assigning names to faces of G. Given any edge e in G, there can be at most two bounded faces F_1 and F_2 of G adjacent to e. There must also be some vertex v that is on the boundary of F_1 and not of F_2 (and vice-versa). Thus any such pair (e, v) can serve as a "name" for F. We let the lexicographically least such pair serve as the name of F. Given any vertex v of G, it is easy in logspace to enumerate all of the names of the faces that are adjacent to v.

We have already observed that it is easy to enumerate the upper and lower boundaries of G. Thus, it is easy to enumerate all of the edges adjacent to s' and t' in H. In order to determine if there is an edge (F, F') in H, it is necessary to determine the orientation of each edge around each face of G. For faces adjacent to the upper and lower boundaries of G, this is easy to compute in logspace. We can build an (undirected) bipartite graph with vertices of the form (e, F, c) where $c \in \{$clockwise, counterclockwise$\}$. If e appears on the boundary of F and F', then there are undirected edges:

$((e, F,$clockwise$),(e, F',$counterclockwise$))$ and

$((e, F',$clockwise$),(e, F,$counterclockwise$))$.

Since we are assuming that no edge appears on both the upper and lower boundaries of G, knowing the orientation of the edges on the upper and lower boundaries completely

determines the orientation of all other edges of G. Thus, to determine which way e is oriented around face F, it suffices to apply the logspace algorithm for undirected s-t connectivity to determine the (unique) value of c such that (e, F, c) is in the same connected component as some triple (e', F', c') giving the correct information about some edge e' on the upper boundary.

We have now shown that an encoding of H can be constructed in logspace.

If there is no path from s to t in G, then there is a separating cut. As shown in the proof of Lemma 3.2, this cut starts above the upper boundary (in the position of s') and enters a sequence of faces $F_0, F_1, \ldots$ entering each F_i across an edge that is oriented counterclockwise around F_i, and departing across an edge that is oriented clockwise around F_i, before finally crossing the lower boundary. It is clear that this yields a directed path $s', F_0, \ldots, t'$ in H.

Conversely, assume that there is a directed path from s' to t' in H. Thus there is a simple path $s', F_0, F_1, \ldots, F_r, t'$ where each F_i is distinct. We can describe a curve starting above the upper boundary of G and crossing into F_0, and from there into F_1 etc., eventually reaching the lower boundary. We then continue the curve around s and back to its starting point above the upper boundary. Clearly this is a simple curve, and all edges that cross the curve are directed from the exterior to the interior; thus it is a separating cut, demonstrating that there is no edge from s to t in G.

4 Grid Graphs

In this section, we present a $\leq_m^{\log}$ reduction of PLANAR.STCONN to GGR.

Using the reduction of Section 2, we may assume that we are given a planar graph G with s and t on the external face. By applying the reachability algorithm on undirected graphs, we can merge all vertices that are joined by bidirected edges, and thus we can assume that all edges are unidirectional; note that the graph remains planar after this transformation. We may also assume without loss of generality that G has no vertex of degree (indegree + outdegree) greater than 3, and that s has degree two. (To see this, observe that if v is a vertex of degree $d > 3$, then we may replace v with d vertices arranged in a directed cycle, with each one adjacent to one of the d edges that were connected to v. In order to compute this transformation it is important to note that we can compute the planar embedding in logspace. If the vertex s has degree three, then an additional vertex of degree two can be inserted into this cycle, and re-named s.)

We can compute an (undirected) spanning tree T of G in logspace. The vertex s is a vertex of T, and we can consider it to be the root of T; without loss of generality s has two children in T. By our assumptions on G, the tree T is a binary tree; the planar embedding of G imposes an ordering on the children of each node in T. As observed in [AM04], we can compute the height $h(v)$ of each node v in T in logspace (by counting the number of vertices that are ancestors of v). For notational convenience, define the height of the root s to be 1, and if v has child u then $h(u) = h(v) + 1$.

At this point, we are ready to assign each vertex of G to a grid point. Our grid graph will consist of a "fine grid" and a "coarse grid". The coarse grid consists of points placed at the corners of large squares (of size $(4n + 1) \times (4n + 1)$) of the fine grid. (The fine grid will be used to route non-tree edges between vertices placed on the coarse grid.)

For any node x, define $w(x)$ to be the number of leaves of T that appear strictly to the left of x; $w(x)$ can be computed easily in logspace by traversing T. Each vertex x is assigned to the point $(h(x), w(x)+1)$ in the coarse grid; note that the root s is placed at the top left corner $(1,1)$. If node x is at position (i,j) in the coarse grid, then the tree edge from x to its left child is embedded as a vertical path to point $(i+1,j)$ in the coarse grid. If x also has a right child y, then this edge is embedded as a horizontal path to location $(i, w(y)+1)$ followed by a vertical path to location $(i+1, w(y)+1)$ in the coarse grid. This is illustrated in Figure 7.

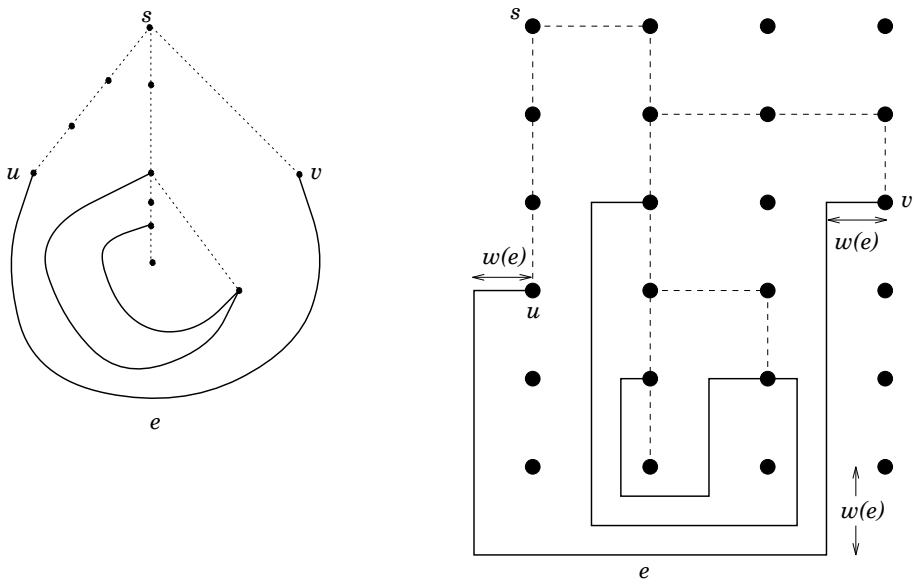

Fig. 7. Embedding a graph on the grid. Edges used in the spanning tree are shown as dashed lines; non-tree edges are solid.

For every non-tree edge e in the tree we can find the number $w(e)$ of non-tree edges enclosed by the unique cycle formed by adding e to the tree. (For edge $e=(u,v)$, $w(e)$ can be computed by finding the least common ancestor y of u and v and determining for each non-tree edge connected to a descendant of y if it is embedded to the right or left of the paths between y and u and v.) For any non-tree edge $e=(u,v)$, note that u and v have degree at most two in the tree T, and thus there is no tree edge attached horizontally adjacent to u or v. The embedding determines if the path representing e should be attached to the east or west sides of u and v. If the embedding goes around a leaf z of the tree T, then the path is routed horizontally from u to a location $w(e)$ fine grid points to the east or west of the column containing z, and vertically down to a point $w(e)$ fine grid points below the level of the leaf of maximum height, and from there horizontally to a point $w(e)$ fine grid points east or west of the column containing v, then vertically to the level of v, and then horizontally to attach to v. If the embedding

does not go around a leaf, then a simpler path can be drawn: horizontally to a point $w(e)$ fine grid points east or west of v, then vertically to the level of v, and then horizontally to connect to v. It is easy to verify that no paths collide in this way. See Figure 7 for an example.

Thus we have the following theorem.

Theorem 4.1. PLANAR.STCONN$\leq_m^{\log}$GGR

Observe that this result, combined with [BLMS98, Lemma 13], provides an alternate construction of a $\leq_m^{\log}$ reduction of PLANAR.STCONN to its complement.

5 More Closure Properties

Different types of logspace reductions were introduced and studied by Ladner and Lynch [LL76], who showed that logspace Turing and truth-table reducibilities coincided ($A\leq_T^{\log} B$ iff $A\leq_{tt}^{\log} B$). They also introduced a more restrictive version of logspace-computable truth-table reducibility, known as logspace Boolean formula reducibility $\leq_{bf-tt}^{\log}$. $A\leq_{bf-tt}^{\log} B$ if there is a logspace computable function f such that $f(x) = (q_1, q_2, \ldots, q_r, \phi)$ where each q_i is a query and ϕ is a Boolean formula with r variables $y_1, \ldots, y_r$, such that $x \in A$ if and only if ϕ evaluates to 1 when the variables y_i are assigned the truth value of the statement "$q_i \in B$". Additional results about this type of reducibility can be found in [BST93,BH91].

Corollary 5.1. $A\leq_m^{\log}$PLANAR.STCONN *if and only if* $A\leq_{bf-tt}^{\log}$PLANAR.STCONN.

Proof. One direction is trivial; thus assume that $A\leq_{bf-tt}^{\log}$PLANAR.STCONN. For a given input x, let $f(x) = (q_1, q_2, \ldots, q_r, \phi)$ be the result of applying the reduction to x. Without loss of generality, the formula ϕ has negation operations only at the leaves (since it is easy in logspace to apply DeMorgan's laws to rewrite a formula). Using closure under complementation, we can even assume that there are no negation operations at all in the formula. By the results of Section 2, we can assume that each graph q_i is a planar graph with s and t on the external face. Given two such graphs G_1, G_2, note that both G_1 and G_2 are in PLANAR.STCONN if and only if the graph with the terminal vertex of G_1 connected to the start vertex of G_2 is in PLANAR.STCONN, and thus it is easy to simulate an AND gate. Similarly, an OR gate can be simulated by building a new graph with start vertex s connected to the start vertices of both G_1 and G_2, and with edges from the terminal vertices of G_1 and G_2 to a new vertex t. These constructions maintain planarity, and they also maintain the property that s and t are on the external face. Simulating each gate in turn involves only a constant number of additional vertices and edges, and it is easy to see that this gives rise to a $\leq_m^{\log}$ reduction.

6 Higher Genus

In this section we prove that the s-t-connectivity problem for graphs of genus 1 reduces to the planar case. Throughout this section, we will assume that we are given an embedding Π of a graph G onto a surface of genus 1. (Unlike the planar case, it does

not appear to be known if testing if a graph has genus $g > 0$ can be accomplished in logspace, even for $g = 1$ [MV00].) Given such an embedding, using [AM04], we can check in logspace if the minimal genus of the graph is 1.

We introduce here some terminology and definitions relating to graphs on surfaces. It will be sufficient to give informal definitions of various notions; the interested reader can refer to [MT01] for more rigorous definitions.

A *closed orientable* surface is one that can be obtained by adding handles to a sphere in 3-space. The genus of the resulting surface is equal to the number of handles added; see also the text [GT87]. Given a graph G, the genus of the graph is the genus of the (closed orientable) surface of least genus on which the graph can be embedded.

Given a graph G embedded on a closed orientable surface, and a cycle of the graph embedded on the surface, there are two (possibly intersecting) subgraphs, called the two *sides* of the cycle with respect to the embedding. Informally, a side of a cycle is the set of vertices of the graph that are path-connected (via a path in the graph, each edge of the graph being considered regardless of direction) to some vertex on the cycle, such that this path does not cross the cycle itself. (In the considerations below, we are concerned only with genus 1 graphs for which this notion of path-connectivity suffices.) A cycle thereby has two sides, which are called the *left* and the *right* sides. If the left and right sides of a cycle have nonempty intersection, then we call the cycle a *surface-nonseparating cycle*. Note that a graph embedded on a sphere (i.e., a planar graph) does not have any surface-nonseparating cycles. Also, it is easy to see that a *facial* cycle (one that forms the boundary of a face in the embedding of the graph on the surface) cannot be surface-nonseparating. Given a cycle C in an embedded graph, it is easy to check in logspace, if C is surface-nonseparating or not: merely check if there is a vertex $v \in G$, such that v is path-connected to both sides of C (on the embedding).

Lemma 6.1. *Let G be a graph of genus $g > 0$, and let T be a spanning tree of G. Then there is an edge $e \in E(G)$ such that $T \cup \{e\}$ contains a surface-nonseparating cycle.*

Proof. The proof follows ideas from [Tho90] which introduces the "3-path condition":

Definition 6.2. *Let $\mathcal{K}$ be a family of cycles of G as follows. We say that $\mathcal{K}$ satisfies the 3-path condition if it has the following property. If x, y are vertices of G and P_1, P_2, P_3 are internally disjoint paths joining x and y, and if two of the three cycles $C_{i,j} = P_i \cup P_j$ ($1 \leq i < j \leq 3$) are not in $\mathcal{K}$, then also the third cycle is not in $\mathcal{K}$.*

We quote the following from [MT01].

Proposition 6.3. *(Proposition 4.3.1 of [MT01]) The family of Π-surface-nonseparating cycles satisfies the 3-path condition.*

Suppose, that $\forall e$, $(T \cup \{e\})$ does not have a surface-nonseparating cycle. We will prove that no cycle C in the graph G can be surface-nonseparating, by induction on the number k of non-tree edges in C. This contradicts the fact that every non-planar graph has a surface-nonseparating cycle ([MT01, Lemma 4.2.4 and the following discussion]) and thus suffices to prove the claim.

The basis ($k = 1$) follows from the above supposition.

For the inductive step from $k - 1$ to k, let a cycle C be given with k edges not in T.

Take any non-tree edge $e = (x, y)$ on C. Consider the tree path P between x and y. If P never leaves the cycle C, then C is a fundamental cycle and we are done by the assumption for $k = 1$. Otherwise, we can consider a maximal segment S of P not in C. Let S lie between vertices u and v of C. Now, we have three paths between u and v : the two paths between u and v on C (call these C_1, C_2), and path S. Note that both $S \cup C_1$ and $S \cup C_2$ have less than k non-tree edges. Hence they are not surface-nonseparating cycles by the induction assumption. So, by the 3-path condition, neither is $C = C_1 \cup C_2$.

This completes the induction, and the proof.

At this point we are able to describe how to reduce the s-t-connectivity problem for graphs of genus 1 to the planar case.

Given a graph G of genus 1 and an embedding Π of G onto the torus, construct an (undirected) spanning tree T of G. (It follows from [NTS95,Rei05] that spanning trees can be constructed in logspace.) For each edge e of G that is not in T, determine if the unique cycle C_e in $T \cup \{e\}$ is surface-nonseparating, as follows.

Let $C_e = \{v_1, v_2, \cdots, v_r\}$. Let G_e be the graph obtained from G by *cutting along* the cycle C_e (as described in [MT01, p. 105]). (For the purposes of visualization, it is useful to imagine cycles as embedded on an inner tube. Cutting along a surface-separating cycle amounts to cutting a hole in the inner tube (resulting in two pieces). In contrast, if C_e is surface-nonseparating, then it is embedded either like a ribbon tied around the tube, or like a whitewall painted on the inner tube. In the former case, cutting along C_e turns the inner tube into a bent cylinder with a copy of C_e on each end; in the latter case cutting along C_e results in a flat ring with one copy of C_e around the inside and one around the outside. In this latter case, the graph is again topologically equivalent to a cylinder with a copy of C_e on each side.) More formally, the graph G_e has two copies of each of the vertices $\{v_1, v_2, \cdots, v_r\}$, which we denote by $\{v_{1,1}, v_{2,1}, \cdots, v_{r,1}\}$, and $\{v_{1,2}, v_{2,2}, \cdots, v_{r,2}\}$. For every edge (u, v_j) (or (v_j, u)) on the right side of C_e (according to Π), G_e has the edge $(u, v_{j,1})$ ($(v_{j,1}, u)$, respectively), and for every edge (u, v_j) ((v_j, u),respectively) on the left side of C_e we have the edge $(u, v_{j,2})$ (or $(v_{j,2}, u)$) in G_e. The graph G_e also has two copies of the cycle C_e, which we denote by $C_{e,1}$ and $C_{e,2}$. That is, we have edges between $v_{j,b}$ and $v_{j+1,b}$ for each $b \in \{1, 2\}$ and each $1 \leq j \leq r$, directed as in C_e. An important property of cutting along the cycle C_e is that if C_e was surface-nonseparating, then the resulting graph G_e is planar, and the the cycles $C_{e,1}$ and $C_{e,2}$ are *facial* cycles ([MT01, p. 106,Lemma 4.2.4]). (Otherwise, G_e will not be planar.) Thus in logspace we can determine if C_e is surface-nonseparating.

By Lemma 6.1, we are guaranteed to find a surface-nonseparating cycle by testing each edge e that is not in T. The graph G_e does not have the same connectivity properties as G; s and t might have been connected in G but not in G_e. In particular, any directed path in G from s to t that uses edges from *both* the right and left sides of C_e is not replicated in G_e. As in Section 2, we solve this problem by pasting together copies of the graph G_e, as follows. The operation of "pasting" two copies of G_e together consists of identifying the vertices $v_{1,1}, v_{2,1}, \ldots, v_{r,1}$ in one copy with the vertices $v_{1,2}, v_{2,2}, \ldots, v_{m,2}$ in the other copy. (Note that this amounts to "sewing together" two copies of the path that were "cut apart" in creating G_e from G.)

Now construct the graph G' consisting of $2n+1$ copies of G_e pasted together in this way: the "original copy" in the middle, and n copies along each side, forming one long cylinder. Since this cylinder has genus zero, it is easy to see that G' is planar.

As in Section 2, a simple inductive argument shows that there is a directed path from s to t in G if and only there is a directed path from (the original copy of) s to one of the copies of t in G'. Thus we have presented a logspace-computable disjunctive truth-table reduction to the planar directed s-t-connectivity problem. We obtain a many-one reduction by appeal to Corollary 5.1 Thus we have proved the following theorem.

Theorem 6.4. *The s-t-connectivity problem for graphs of genus one is $\leq_m^{\log}$ reducible to the planar directed s-t-connectivity problem.*

7 Layered Grid Graphs

Theorem 7.1. LGGR $\in$ UL.

Proof. Let G be a layered $n \times n$ grid graph, with vertex s in column 1 and vertex t in column n. We define a weight function w on the edges of G as follows. If e is directed vertically (that is, from (i,j) to (i',j) for $i' \in \{i+1, i-1\}$), then e has weight zero. Otherwise, e is directed horizontally and is of the form $(i,j) \rightarrow (i,j+1)$. In this case, the weight of e is i. This weight function induces a natural weight function on paths; the weight of a path is the sum of the weights of its edges. (It is a helpful approximation to think of the weight of a path as the number of boxes of the grid that lie above the path.)

The minimal-weight simple path from s to any vertex v is unique. This is because if there are two paths P_1 and P_2 from s to v that have the same weight, there must be some column in which P_1 is higher than P_2 and another column in which P_2 is higher than P_1. Since G is a layered grid graph, this means that there is some point in between these two columns in which the two paths intersect. The path from s to v that follows the two paths until they diverge, and then follows the path closer to the top of the grid until they next intersect, and continues in this way until v is reached, will have smaller weight than either P_1 or P_2, and thus they cannot have had minimal weight.

At this point, we are able to mimic the argument of [RA00].

Let C_k be the set of all vertices in column k that are reachable from s. Let $c_k = |C_k|$. Let Σ_k be the sum, over all $v \in C_k$ of the minimal weight path from s to v. Exactly as in [RA00], there is a UL algorithm that, given (G, k, c_k, Σ_k, v), can determine if there is a path from s to v *or not*. (We emphasize the words "or not"; if there is no path, the UL machine will determine this fact; the algorithm presented in [RA00] has this property.) Furthermore, this algorithm has the property that, if v is reachable from s, then the UL machine can compute the weight of the minimal-weight path from s to v. (Informally, the machine tries each vertex x in column k in turn, keeping a running tally of the number of vertices that have been found to be reachable, and the total weight of the guessed paths. For each vertex x, the machine guesses whether there is a path from s to x; if it guesses there is a path, then it tries to guess the path, and increments its running totals. If $x = v$, then it remembers the weight of the path that was guessed. At the end, if the running totals do not equal c_k and Σ_k, then the machine concludes that

it did not make good guesses and aborts. By the properties of the weight function, there will be exactly one path that makes the correct guesses and does not abort.)

It suffices now to show that a UL machine can compute the values c_k and Σ_k. Observe first of all that c_1 is easy to compute (by simply walking up and down column 1 from s and counting how many vertices are reachable), and $\Sigma_1 = 0$.

Assuming that the values c_k and Σ_k are available, the numbers c_{k+1} and Σ_{k+1} can be computed as follows. Initialize c_{k+1} and Σ_{k+1} to zero. For each vertex v in column $k+1$, for each edge of the form $x \to y$ to a vertex y in column $k+1$ such that there is a path in column $k+1$ from y to v, if $x \in C_k$ via a minimal-weight path of weight w_x, then compute the weight w'_x of the path to v through x. Let w_v be the minimum of all such x. Increment c_{k+1} by one (to indicate that v is reachable) and increase Σ_{k+1} by w_v. (This algorithm is actually more general than necessary; it is easy to show that the minimal-weight path to v will always be given by the "topmost" vertex $x \in C_k$ for which there is an edge $x \to y$ to a vertex y that can reach v in column $k+1$.)

This completes the proof.

We observe that we have shown that a UL algorithm can also determine if there is *not* a path from s to t, and thus LGGR is in UL $\cap$ coUL.

Acknowledgments

We acknowledge many people for sharing with us their thoughts about what was already known about this problem, including David Mix Barrington, Til Tantau, Omer Reingold, Paul Beame, Pierre McKenzie, Jeff Edmunds, Anna Gal, Vladimir Trifonov, K.V. Subrahmanyam, Meena Mahajan, and Tanmoy Chakraborty. The first and third authors acknowledge the support of NSF Grant CCF-0514155. We also acknowledge the helpful comments provided to us by the program committee.

References

[AM04] Eric Allender and Meena Mahajan. The complexity of planarity testing. *Information and Computation*, 189:117–134, 2004.

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59(2):164–181, 1999.

[Bar02] David A. Mix Barrington. Grid graph reachability problems. Talk presented at Dagstuhl Seminar on Complexity of Boolean Functions, Seminar number 02121, 2002.

[BH91] Samuel R. Buss and Louise Hay. On truth-table reducibility to SAT. *Inf. Comput.*, 91(1):86–102, 1991.

[BK78] Manuel Blum and Dexter Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 132–142, 1978.

[BLMS98] David A. Mix Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. Searching constant width mazes captures the AC^0 hierarchy. In *15th International Symposium on Theoretical Aspects of Computer Science (STACS)*, number 1373 in Lecture Notes in Computer Science, pages 73–83. Springer, 1998.

[BST93] Harry Buhrman, Edith Spaan, and Leen Torenvliet. The relative power of logspace and polynomial time reductions. *Computational Complexity*, 3:231–244, 1993.

[Ete97] Kousha Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54(3):400–411, Jun 1997.

[GT87] Jonathan Gross and Thomas Tucker. *Topological Graph Theory*. John Wiley and Sons, New York, 1 edition, 1987.

[HRS93] Magnús Halldórsson, Jaikumar Radhakrishnan, and K. V. Subrahmanyam. Directed vs. undirected monotone contact networks for threshold functions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 604–613, 1993.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[JLR01] A. Jakoby, M. Liskiewicz, and R. Reischuk. Space efficient algorithms for series-paralle graphs. In *18th International Symposium on Theoretical Aspects of Computer Science (STACS)*, number 2010 in Lecture Notes in Computer Science, pages 339–352. Springer, 2001. To appear in J. Algorithms.

[Lan97] Klaus-Jörn Lange. An unambiguous class possessing a complete set. In *14th International Symposium on Theoretical Aspects of Computer Science (STACS)*, number 1200 in Lecture Notes in Computer Science, pages 339–350. Springer, 1997.

[LL76] R. Ladner and N. Lynch. Relativization of questions about log space reducibility. *Mathematical Systems Theory*, 10:19–32, 1976.

[MT01] Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. John Hopkins University Press, Maryland, 1 edition, 2001.

[MV00] Meena Mahajan and Kasturi R. Varadarajan. A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 351–357, 2000.

[NTS95] N. Nisan and A. Ta-Shma. Symmetric Logspace is closed under complement. *Chicago Journal of Theoretical Computer Science*, 1995.

[RA00] K. Reinhardt and E. Allender. Making nondeterminism unambiguous. *SIAM Journal of Computing*, 29:1118–1131, 2000.

[Rei05] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings 37th Symposium on Foundations of Computer Science*, pages 376–385. IEEE Computer Society Press, 2005.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Tho90] C. Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Comb. Theory Ser. B*, 48(2):155–177, 1990.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'