Electronic Colloquium on Computational Complexity, Report No. 130 (2006)

One-input-face MPCVP is Hard for L, but in LogDCFL

Tanmoy Chakraborty[1] and Samir Datta[2]

[1] Dept. Of Compute and Information Science, Univ. of Pennsylvania, USA
tanmoy@seas.upenn.edu
[2] Chennai Mathematical Institute, India sdatta@cmi.ac.in

Abstract. A monotone planar circuit (MPC) is a Boolean circuit that can be embedded in a plane, and that has only AND and OR gates. Yang showed that the one-input-face monotone planar circuit value problem (MPCVP) is in NC^2, and Limaye et. al. improved the bound to LogCFL. Barrington et. al. showed that evaluating monotone upward stratified circuits, a restricted version of the one-input-face MPCVP, is in $\mathsf{LogDCFL}$. In this paper, we prove that the unrestricted one-input-face MPCVP is also in $\mathsf{LogDCFL}$. We also show this problem to be L-hard under quantifier free projections.

Key Words: L, LogDCFL, monotone planar circuits.

1 Introduction

The problem of evaluating Boolean circuits is a widely studied problem in complexity theory. In [12], the problem of evaluating a Boolean circuit (CVP) was shown to be P-complete under logspace many-one reductions. Special cases of CVP, namely, the *monotone* CVP and the *planar* CVP, have also been shown to be P-complete in [9]. However, a special case of both these versions, the *planar monotone* CVP (MPCVP), is known to be in NC.

It was shown in [10] that upward stratified MPCVP, a special case of MPCVP (see Section 2 for definitions), is in NC^2. The upper bound for this problem was subsequently improved to LogCFL in [7], and quite recently to $\mathsf{LogDCFL}$ in [5].

A less restrictive case, the layered upward MPCVP, was shown to be in NC^3 in [11]. Independently and in parallel, it was shown in [15] and [6] that general MPCVP is in NC^4 and NC^3 respectively.

In [15], it was shown that one-input-face MPCVP, a less restricted case than upward stratified, is in NC^2. Recently, it was shown in [13] that one-input-face MPCVP is in $\mathsf{L}(\mathsf{PDLP} \oplus \mathsf{LogDCFL}) \subseteq \mathsf{LogCFL}$. ($\mathsf{PDLP}$ is the problem of finding the longest path in a planar DAG. Its best known upper and lower bounds are NL and L, respectively.) The upper bound for general MPCVP was also improved to $\mathsf{AC}^1(\mathsf{LogCFL}) = \mathsf{SAC}^2$ in [13].

ISSN 1433-8092

Cylindrical and toroidal circuits were also discussed in [13]. Stratified monotone cylindrical circuits were shown to be in $\mathsf{LogDCFL}$, one-input-face monotone cylindrical circuits in $\mathsf{L}(\mathsf{PDLP} \oplus \mathsf{LogDCFL})$, and monotone cylindrical circuits(in full generality) in $\mathsf{AC}^1(\mathsf{LogDCFL})$. Toroidal circuits were shown to be in SAC^2.

The main result of this paper is that **one-input-face MPCVP is in LogDCFL**. Our method is inspired chiefly from the insights about grid graphs and single source planar graphs developed in [2] and [1]. Our result has been mentioned as personal communication in [13], and we are grateful to its authors for valuable discussion.

We also show that L is a lower bound for one-input-face MPCVP, *i.e.* one-input-face MPCVP is L-hard under quantifier free projections. As corollary to the main result of this paper, we infer that one-input-face cylindrical circuits, can be evaluated in $\mathsf{LogDCFL}$.

The rest of the paper is organized as follows: Section 2 gives some necessary definitions and basic or known facts, Section 3 gives an overview of the proof of our main result, while Section 4 presents its details. In Section 5, we show the L-hardness of one-input-face MPCVP under quantifier free projections, and in Section 6, we summarize the results. Section 7 provides our conclusion.

2 Definitions and Facts

A *Boolean* circuit is a circuit with AND, OR and NOT gates, apart from the input gates. The gates (as vertices) and wires (as edges directed towards the gate for which it is an input wire) of the circuit form a directed acyclic graph (DAG). We shall consider Boolean circuits which also have COPY gates of fan-in one: a COPY gate outputs 1 if and only if its input is 1. Note that the behaviour of a COPY gate is the same as an AND or OR gate with fan-in one.

Circuit value problem (CVP) is the problem of evaluating a circuit when values of the input gates are specified. A circuit is called *monotone* if it does not have any NOT gate. A circuit is called *planar* if its underlying DAG has a planar embedding. MPCVP refers to the restriction of CVP in which the circuit is monotone as well as planar.

A planar circuit is said to be *one-input-face* if it has a planar embedding such that all the input gates are on a single face. The planar embedding need not be given as part of the input, as the following lemma shows.

Lemma 1. *An appropriate planar embedding for a one-input-face circuit can be found in logspace.*

Proof. Consider the planar DAG G corresponding to the circuit C. Add a source vertex s in G, and add edges from s to all the input gates, to obtain a graph G'. Since C is one-input-face, G' is also planar. Find a planar embedding of G', and delete s to get the required embedding for G. A planar embedding can be computed by a logspace transducer, since it was shown to be in $\mathsf{FL}^{\mathsf{SL}}$ in [3], and it was proved that $\mathsf{SL} = \mathsf{L}$ in [14]. $\square$

A planar embedding can be specified by listing the edges incident on each vertex, in cyclic order around the vertex. Such a specification is called a *combinatorial embedding*. A planar embedding is said to be *bimodal* if all the incoming edges at every vertex appear consecutively in the cyclic ordering. For a bimodal planar embedding, we can define the *clockwise-most* and *anticlockwise-most* incoming and outgoing edges at every vertex v without any ambiguity. We can, infact, order all the incoming edges and all the outgoing edges, according to their cyclic ordering, clockwise or anticlockwise.

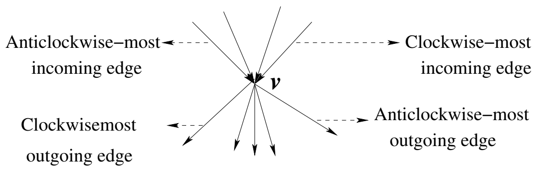

Fig. 1. Bimodality at a vertex v

A planar DAG is called an SSPD if it has a single source (vertex with indegree zero), and a single sink (vertex with outdegree zero). It is well known (*e.g.*, see [1],[15]) that any planar embedding of an SSPD is bimodal. A planar DAG is called an SMPD if it has a single source, but can have multiple sinks.

Similar to planar circuits, one may also consider cylindrical circuits (i.e. embeddable on the surface of a cylinder), and toroidal circuits (i.e. embeddable on the surface of a torus). Please see [13] for definition and properties of such embeddings.

A circuit is said to be *layered* if there is a partition of the vertex set $V = V_0 \cup V_1 \cup V_2 \dots V_k$, such that all the edges go from V_i to V_{i+1} for some i. Each subset of the partition is called a *layer*. A layered circuit is said to be stratified if there is such a partition, in which all the input gates (vertices) are in V_0. For layered circuits, it is important that the input provides the layering information; all the previous results critically use this fact. Finding a layering for general circuits that can be layered is not known to be in LogDCFL.

A circuit (graph) is said to be *upward planar* if there is a planar embedding in which every edge is monotonically increasing in the upward, or any particular, direction. A circuit (graph) is said to be *upward layered (stratified)* if it is layered (stratified), and the layers give an upward planar embedding. Clearly, an upward stratified circuit is also a one-input-face circuit.

LogCFL and LogDCFL are the classes of languages that are logspace many-one reducible to non-deterministic and deterministic context-free languages, respectively. LogDCFL can be alternately described as the class of languages decidable by a logspace Turing machine that is also provided with a stack, which runs in polynomial time. The following facts are known:

- $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{LogCFL}$,

– $\mathsf{L} \subseteq \mathsf{LogDCFL} \subseteq \mathsf{LogCFL}$, and
– $\mathsf{LogCFL} = \mathsf{SAC}^1 \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2$.

Grid graphs are planar graphs whose vertices are a subset of the integral points of a finite two-dimensional grid (called *grid points*), and whose edges are either from (i, j) to $(i + b, j)$ (horizontal edge), or from (i, j) to $(i, j + b)$ (vertical edge), where $b \in \{-1, 1\}$. A grid graph has the naturally defined directions up, down, left and right, which are synonymous with north, south, west and east, respectively. We follow the convention that the first coordinate increases rightward, and call it the *rightward/eastward coordinate*, while the second coordinate increases downward, and we call it the *downward/southward coordinate*. [1] and [2] are good references for terminology and facts associated with grid graphs.

A grid graph is said to be *1-forbidden* if it has edges only in three of the four directions. A grid graph is said to be *2-forbidden* or *layered* if it has either rightward or leftward edges, and has either upward or downward edges. Note that a layered grid graph is upward layered (view the grid graph diagonally). Note that a layered grid graph, viewed diagonally, is also an upward layered graph. Each layer consists of all the vertices that lie on a line parallel to the diagonal, and the ordering of the layers can be deduced easily in logspace.

The problem ORD is defined as reachability from a vertex s to another vertex t in a directed graph, consisting of n vertices $v_1, v_2 \ldots v_n$ and $(n - 1)$ edges (given in the input as ordered pairs of vertices), such that the graph is a directed path. Every vertex v has a unique successor $S(v)$. An equivalent definition of the problem in terms of total orders is given in [8].

It was shown in [8] that ORD is L-complete under quantifier free projections (qfp's). For details on these extremely low level reductions please see [8].

3 Overview

In [13], one-input-face MPCVP was reduced to upward stratified MPCVP, by making oracle calls to the PDLP problem, which finds the longest path in a planar DAG, and then the $\mathsf{LogDCFL}$ algorithm given in [5] was used to solve the one-input-face MPCVP in $\mathsf{L}(\mathsf{PDLP} \oplus \mathsf{LogDCFL})$.

We prove that the one-input-face MPCVP is in LogDCFL, by finding a logspace reduction from one-input-face MPCVP to the upward stratified MPCVP. This result would have followed trivially from the algorithm in [13] if PDLP were in $\mathsf{LogDCFL}$, but such a result has not yet been proved, and, for all we know, PDLP can be NL-hard. We take a completely different approach to bypass the PDLP problem and obtain a logspace reduction.

3.1 Graph-Circuit Conversion

In this paper, we shall often store a circuit as a DAG G, with vertices corresponding to gates and edges corresponding to wires. For interpreting G as a circuit, it is required that every vertex carries exactly one of the labels $0, 1$, AND, OR, COPY and SRC. The label SRC shall indicate the dummy vertices, that are not present in C. The other labels shall indicate the type of the gate corresponding to the vertex. Further, one of the vertices carries a second label of OUTPUT, which will correspond to the output gate of the circuit. Note that it is possible for a DAG to have a labelling that cannot be interpreted as a meaningful Boolean circuit.

We shall use the following *conversion algorithm*, which, given a DAG G and a labelling of its vertices, decides if the labelling *valid*, *i.e.* whether it can be interpreted as a meaningful circuit, and also produces the unique circuit corresponding to G, if it is meaningful:

1. If some vertex labelled COPY does not have indegree one, report that the labelling is not valid.
2. Delete all vertices (and edges incident on them) that should not be there in the circuit. These include vertices labelled SRC, and also those vertices v labelled COPY, such that there is a path from another vertex u, labelled SRC, all whose internal vertices are labelled COPY.
3. Replace the remaining vertices by gates according to the labelling, and the edges by wires. The gate corresponding to the vertex labelled OUTPUT is marked as the output gate of the circuit produced.

Since the hardest step in the conversion algorithm involves checking reachability in graphs by *simple* paths (paths whose internal vertices have total degree 2 in the graph), the algorithm can be implemented in logspace.

We shall refer to the circuit obtained by the conversion algorithm as the circuit corresponding to the graph. For any vertex that is not deleted by the algorithm, the gate corresponding to it will have a value in the evaluation of the circuit, which we shall refer to as the *value at the vertex*.

Note that, given a circuit C, it is trivial to construct a graph G, such that the conversion algorithm applied on G yields C.

3.2 Steps of the reduction

Given a one-input-face MPC C, consider its underlying single-source planar DAG, with vertices labelled accordingly. We add a source vertex s to the graph, with edges to all the vertices labelled 0 or 1, and label it as SRC. Let this graph, which is an SMPD, be G.

The reduction then proceeds sequentially in 5 major steps. Each step takes the output of the previous step as its input, and uses it to produce some output, in logspace. Step 1 takes G(with its labelling) as input. Each of steps 1-4 output a planar DAG (that has certain useful properties) with a valid labelling. We shall ensure that the value of the circuit corresponding to the output of each step is the same as that of the input circuit C. Step 5 produces an upward stratified circuit, hence completing the reduction.

The chief properties of the output of the each step is listed below:

1. An SSPD G_1.

2. An SSPD G_2 whose total degree at each vertex is bounded by 3, and the indegree and outdegree by 2.
3. A 1-forbidden grid graph G_4, that is also an SMPD.
4. A layered grid graph G_5, that is also an SMPD.
5. An upward stratified circuit C''.

The upward stratified circuit C'' obtained at the end of step 5 can then be evaluated in LogDCFL, as described in [5].

4 Details of the Reduction

In this section, we provide the necessary details about how to implement the steps, outlined in the overview, in logspace, and also show that the circuit value is preserved.

4.1 Step 1

Suppose that a vertex u does not have a path to t, the vertex labelled OUTPUT. Then the value at t is independent of the value at u. So, deleting u does not affect the circuit value. If we delete all such vertices, then it is easy to see that the resulting graph has a single source s and a single sink t, *i.e.* the resulting graph G_1 is an SSPD, and the circuit value remains unchanged. It was shown in [1] that reachability in single-source planar DAGs is in L, so G_1 can be constructed from G by a logspace transducer.

4.2 Step 2

We compute a planar embedding (combinatorial) of G_1. This can be done in logspace, by Lemma 1. Note that since G_1 is an SSPD, the embedding is bimodal.

To reduce the degrees of the vertices as required, we replace each vertex v of G_1 by a gadget, to obtain the graph G_2. It comprises two directed binary trees, one with its root as its source, and the other with its root as its sink. We shall refer to the former as *outgoing tree*, and to the latter as *incoming tree* (see Figure 2). There is also an edge from the root of the incoming tree to that of the outgoing tree. Both the trees have depth at most $\lceil \log |V| \rceil$, where $|V|$ is the number of vertices in G_1. The number of leaves in the incoming tree (*incoming leaves*) is equal to the indegree of v, and the number of leaves in the outgoing tree (*outgoing leaves*) is equal to the outdegree of v. All the vertices of the outgoing tree are labelled COPY. If v were labelled AND, OR or SRC, all the vertices of the incoming tree are labelled AND, OR or SRC, respectively. If v were labelled 0 or 1, then the vertices of the incoming tree, except its root, are labelled COPY, while its root is labelled 0 or 1. Note that the gadget corresponding to s will not have an incoming tree, and the gadget corresponding to t will not have an outgoing tree. For s, the root of the outgoing tree is labelled SRC, and for t, the root of the incoming tree is labelled OUTPUT.

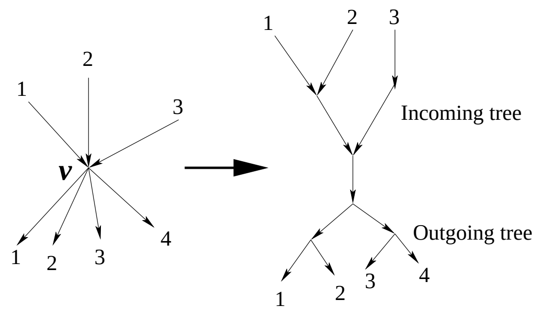

Fig. 2. Gadget to replace any vertex v with indegree 3 and outdegree 4

Note that the incoming leaves and the outgoing leaves are arranged in a bimodal fashion, *i.e.* the incoming leaves appear consecutively in a cyclic ordering. Now, for every edge $e = (u, v)$ in G_1, which is the i^{th} outgoing edge of u and the j^{th} incoming edge of v(unambiguously defined, due to bimodality in G_1), we put an edge in G_2 from the i^{th} outgoing leaf of the gadget for u to the j^{th} incoming leaf of v. Because of bimodality in G_1, G_2 is planar. Also, G_2 is an SSPD, satisfying the degree constraints. It is easy to see that the value of the circuit for G_2 is the same as that for G_1.

Since the gadget for each vertex is dependent only on its indegree and outdegree, they can be constructed by a logspace transducer. The other edges of G_2 can also be added by the same transducer.

4.3 Step 3

This is the most involved step in our reduction. We first convert G_2 into an SMPD G_3 with certain advantageous features, that has the same circuit value as G_2, and then embed G_3 as a *1-forbidden* grid graph G_4, by only subdividing some of the edges (*i.e.* replacing edges by simple paths) of G_3. We shall label the new vertices created due to the subdividing as COPY, and it is easy to see that the circuit value will remain unchanged. Note that the degree constraints achieved in Step 2 are also not violated.

The process of embedding in the grid is similar in spirit to the process given in [2], where it was shown how to embed a planar graph in a grid using only logspace, preserving reachability. Here, we have an SSPD to embed instead of a general planar graph, while we additionally require that the grid graph produced should be monotone along one axis (we shall ensure that G_4 has no westward edge), and also want to preserve circuit value. This is significant, because reachability is precisely evaluation of circuits with only OR gates, and hence possibly easier to preserve than values of circuits with both AND and OR gates.

Using the mentioned embedding of G_2, we construct a subgraph H, by deleting all incoming edges except the clockwise-most one at every vertex of G_2 except the source and sink (the clockwise-most edge is unambiguously defined, due to bimodality). Delete all but one (arbitrarily chosen) of the edges incoming to the sink t. It is easy to see that H is a directed tree spanning all vertices, with s as its root.

We can now classify the edges of G_2 as tree edges (those present in H) and non-tree edges. The non-tree edges can be further classified as *forward edges* (from a vertex to its descendant in H), and *cross edges* (between different subtrees). Since G_2 is a DAG, there is no *back* edge (from a vertex to its ancestor). Due to the bounded degree of G_2, H is a binary tree. We perform an Euler traversal (same as a *dfs* traversal for a tree) of H starting at s, choosing the anticlockwise-most unexplored edge at every stage (we consider the embedding of H derived from G_2). In the beginning, at s, we make an arbitrary choice of the edge to explore first. We write down the *discovery time* $d[v]$ and the *finishing time* $f[v]$ of every vertex v using a logspace transducer.

Before describing the reduction any further, we need the following lemmas.

Lemma 2. *Suppose H is drawn as the dfs-tree, mentioned above, in standard fashion, with the child explored first drawn as the left child at every vertex (see Figure 3). The combinatorial embedding of H thus obtained is the same as that derived from G_2.*

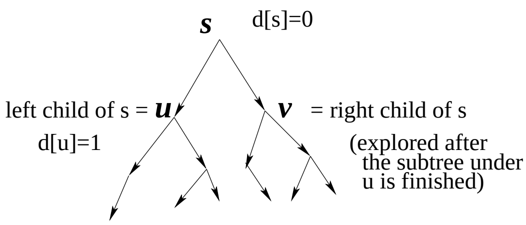

Fig. 3. Standard drawing of a dfstree

Proof (Lemma 2). In the tree embedding, the edge to the left child is the anticlockwise-most outgoing edge, which is the edge explored first during the dfs. □

Hence, it is possible to add and embed the non-tree edges to the dfs-tree in a planar way such that the combinatorial embedding is the same as that of G_2 at the end of the previous step. The dfs-tree helps us define the left and right of every vertex that is not the source or a leaf in H (see Figure 4). There cannot be any non-tree edge incoming to or outgoing from a vertex u between its left and right child, due to bimodality and degree constraint, respectively. Hence, every non-tree edge is incoming to and outgoing from every vertex from either its left or its right. For leaves, there is no distinction between left and right, and we shall take the liberty of either.

Lemma 3. *Any non-tree edge (u, v), is incoming to v from the left of v.*

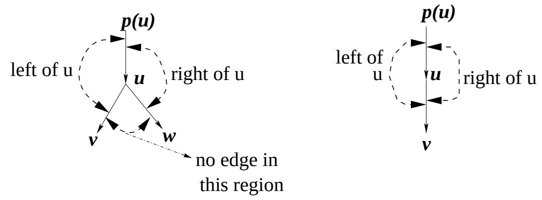

Fig. 4. Left and right of a vertex v with i) two children, and ii) one child

Proof (Lemma 3). If v is a leaf, the statement trivially holds. Suppose v is not a leaf, and (u, v) is a non-tree edge incoming to v from the right. Then, if $p(u)$ is the parent of u in H, then the edge $(p(u), u)$ is not the clockwise-most incoming edge at u, which contradicts our method of construction of H. $\square$

For any two vertices u and v that do not share an ancestor-descendant relationship, we say that u is to the left of v if the discovery and finishing times of u is less than that of v, and vice versa otherwise. We say that a cross edge (u, v) is *leftward* or *rightward*, depending on whether u is to the right or left of v, respectively. We say that a forward edge (u, v) is leftward or rightward, depending on whether the edge is outgoing from the right or left of u, respectively (see Figure 5).

Notice that Lemma 3 does not imply that there are no leftward edges, since its quite possible that the origin u of an edge (u, v) is to the left of the terminal v. It just says that even such edges approach v from the left (see Figure 5).

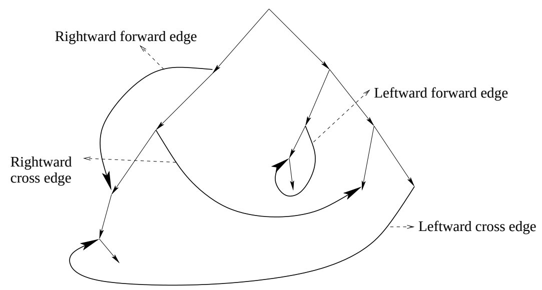

Fig. 5. Possible types of non-tree edges

If we neglect the direction of the edges, every non-tree edge (u, v) added to H produces a unique cycle, consisting of the undirected tree-path between u and v, and the edge (u, v) itself. We call the curve formed by the edges of this cycle as the *characteristic closed curve* of (u, v).

Lemma 4. *Suppose (u, v) is a rightward non-tree edge. Then t cannot be strictly inside characteristic closed curve of (u, v).*

Proof (Lemma 4). Suppose t lies strictly inside the curve. Since G_2 is an SSPD, there must be a path from v to t. Suppose the first edge of this path has its end-point inside the curve. If v has an outgoing edge outside the curve, this contradicts bimodality. Otherwise, the tree edge to v is not the clockwise-most incoming edge at v in G_2, thus contradicting the construction criterion of H. Hence, the first edge of the path must go outside the curve.

Consider the first vertex w on the path from v to t that intersects the curve. This vertex must either be an ancestor of u or v(or both). In both cases, the existence of a directed cycle follows, since there is a path from every ancestor of u and v to v, hence contradicting that G_2 is a DAG. □

Lemma 5. *Suppose (u, v) is a leftward non-tree edge. Then t cannot be strictly outside the characteristic closed curve of (u, v).*

Proof (Lemma 5). Suppose t lies strictly outside the curve. Then, there must be a path from v to t. Again, there cannot be an edge outgoing from v whose end-point is outside the curve, due to bimodality and the construction criterion of H. So, if t is not inside the curve, there must be a vertex at which the path from v to t intersects the curve. Since v is reachable from all ancestors of u and v, this contradicts that G_2 is a DAG. □

We shall now construct a graph G_3 with an analogous tree H', such that there is no leftward non-tree edge, and the circuit value remains unchanged. Suppose there are k leftward non-tree edges. To construct G_3, we make $k+1$ disjoint copies of G_2 without the leftward edges, and label the copies $1, 2 \ldots k, k+1$. For every leftward edge (u, v) in G_2 and $\forall i,\ 1 \leq i \leq k$, add an edge between u of the i^{th} copy and v of the $(i+1)^{th}$ copy of G_2. Finally, we add a new source s', and add edges from s' to all the $k+1$ copies of s (see Figure 6). Expectedly, we label s' as SRC. H' consists of the copies of H, plus s' and its outgoing edges.

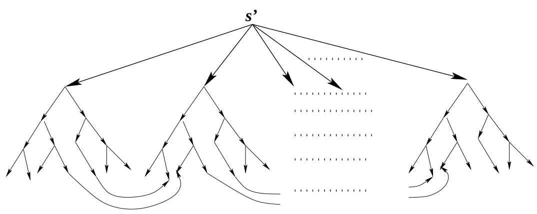

Fig. 6. Construction of G_3

We claim that G_3 is planar. To show this, we observe that from lemma 5 and planarity, it follows that the leftward edges in G_2 are *nested*, *i.e.* if e_1 and e_2 are two leftward edges, either E_1 is contained in the characteristic closed curve of e_2, or vice versa. Thus the cross edges between the copies do not intersect, and, infact, those between any two consecutive copies are also nested. Further, no such edge is nested within a rightward edge, for that would contradict lemma 4. So, these edges between copies do not intersect any other edge.

Note that G_3 is no longer an SSPD, but an SMPD, hence there is no clear choice for the output gate (vertex) of the circuit (graph). For every vertex v of G_2, we say that the i^{th} copy of v in G_3 gets the *correct* value if its circuit value in G_3 is the same as that of v in G_2, otherwise we say that it gets the *wrong* value. We claim that the $(k+1)^{th}$ copy of t has the correct value, and hence we shall label it as the output gate in G_3.

Suppose the k leftward edges in G_2 are $(u_1, v_1), (u_2, v_2) \dots (u_k, v_k)$, with (u_2, v_2) nested inside (u_1, v_1), (u_3, v_3) nested inside (u_2, v_2), and so on, (u_k, v_k) being the innermost leftward non-tree edge. Note that, due to degree constraints, $v_1, v_2 \dots v_k$ are all distinct vertices. To prove our claim, we shall use the following lemmas:

Lemma 6. *A vertex in G_3, that is not in the first copy of G_2, can get the wrong value only if at least one of the vertices, whose values are fed into it, get the wrong value.*

Proof (Lemma 6). Easy. □

Lemma 7. *There is no path from v_i to u_j or from v_i to v_j, if $i \neq j$, in G_2, $\forall 1 \leq j \leq i \leq k$.*

Proof (Lemma 7). Suppose a path from v_i to u_j exists. The path must move rightwards, and end up to the right of u_i. From Lemma 4, it follows that the path cannot pass u_i from below. Hence the path must go through an ancestor of u_i, contradicting that G_2 is a DAG. Similar reasons along with lemma 5 show that there cannot be a path from v_i to v_j either. □

We say that the i^{th} copy of a vertex v has *primitive error* if it gets the wrong value, but all the vertices in the i^{th} copy, that have an edge to it in G_3, get the correct value.

Lemma 8.
1. *If the i^{th} copy of a vertex v gets the wrong value, it must be reachable from a vertex of the i^{th} copy that has primitive error.*
2. *Also, no vertex, other than $v_1, v_2 \dots v_k$, can have primitive error in any of its copy.*
3. *A vertex v_j can have a primitive error in the i^{th} copy only if u_j gets a wrong value in the $(i-1)^{th}$ copy.*

Proof (Lemma 8). Easy. □

We shall prove the following statement using induction:

Lemma 9. *In the i^{th} copy, u_j and v_j get the correct value, and hence do not have a primitive error, $\forall 0 < j < i \leq (k+1)$.*

Proof (Lemma 9). For $i = 1$, the statement is trivially true. Suppose we have proved the statement for the i^{th} copy, $i \leq k$. Then, from Lemma 8 (2), it follows that the only vertices that might have primitive error, in the i^{th} copy, are $v_i, v_{i+1} \dots v_k$. $u_1, u_2 \dots u_i$ are reachable from none of them, by lemma 7, and so gets the correct value in the i^{th} copy, by lemma 8 (1). By lemma 8 (3), $v_1, v_2 \dots v_i$ do not have primitive error in the $(i+1)^t h$ copy. □

Putting $i = k + 1$ in the Lemma 9, we get that the $(k+1)^{th}$ copy does not have any vertex with primitive error, and hence, by lemma 8 (1), no vertex in the $(k+1)^{th}$ gets the wrong value. Hence our claim is proved.

Note that our construction has ensured that G_3 consists of the tree edges of H' and rightward non-tree edges only. Also note that a rightward non-tree edge can start from the left of a vertex, go leftwards, and then go rightwards to end at a vertex to the right of its starting vertex. But, for embedding in a grid in a 1-forbidden fashion, we demand that every cross edge should always be rightwards in direction. In precise terms, we demand the following:

– Every non-tree edge should be a cross edge.
– Every such cross-edge should be rightward.
– Every such cross-edge should start from the right of a vertex and ends at the left of a vertex.

By Lemma 3, our construction has ensured that every non-tree edge ends at the left of its end-point. For every non-tree edge (u, v) that starts from the left of u, we divide (u, v) into two edges, (u, w) and (w, v), and add (u, w) to H', so that w becomes the left child of u. The non-tree edge (w, v) starts from a leaf, and so trivially satisfies the condition. Clearly, the degree constraints are not violated. Since the forward edges present in G_3 must be rightward, and hence start from the left of a vertex, this process gets rid of all forward edges (see Figure 7).

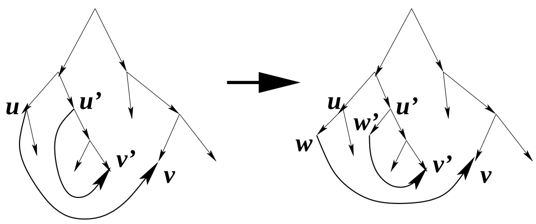

Fig. 7. Subdividing non-tree edges that start from the left of a vertex

For simplicity, we continue to call the modified graph as G_3, and the tree as H'. The new vertices generated due to the subdivisions are labelled as COPY, clearly the circuit value remains unchanged.

To complete the step, we now embed G_3 in a grid, by only subdividing its edges. The vertices formed due to subdividing are labelled COPY, and, clearly, the circuit value is preserved. This part of the step is almost identical to the process of embedding any planar graph in a grid, given in [2].

In the embedding, each edge of G_3 corresponds to a grid path in the grid graph G_4 thus produced, and every vertex of G_3 correspond to a grid point in G_4. If we view these grid paths as the curved edges of G_3 drawn on the plane, the embedding process ensures that the combinatorial embedding of G_3 remains unchanged. This fact, coupled with the carefully chosen parameters in the process, ensure that no two grid paths, that represent two edges of G_3, intersect. The nature of the non-tree edges of G_3 ensures that G_4 is 1-forbidden.

4.4 Step 4

Barrington ([4]) gave a logspace conversion from a 1-forbidden grid graph to a layered grid graph, preserving reachability. We present the procedure for the sake of completeness.

We first embed the tree edges(*i.e.* edges of H'). Let $h(v)$ be the height of a vertex v in the tree H', defined as the number of proper ancestors of v. Let $w(v)$ denote the number of vertices in H' that lie to the left of v. (We, of course, perform yet another Euler traversal of the modified H'.) We assign vertex v to the grid point $(h(v), w(v))$. Thus, s is assigned to $(0, 0)$, the top left corner of the grid. If v has a left child x (assigned to $(h(v) + 1, w(v))$, since $w(v) = w(x)$), then we embed the edge (v, x) as a southward path between these two grid points. If v has a right child y (assigned to $(h(v) + 1, w(y))$, where $w(y) > w(v)$), then we embed the edge (v, y) as an eastward path from $(h(v), w(v))$ to $(h(v), w(y))$, followed by a southward path from $(h(v), w(y))$ to $(h(v) + 1, w(y))$. It can be observed that no two grid paths, that represent two edges of H', intersect.

To embed the cross edges we make the grid finer, so that there is a $2m \times 2m$ subgrid of the finer grid in every 1×1 square of the course grid, where m is the number of cross edges in G_3. Thus, a course grid point (i, j) corresponds to the fine grid point $2mi, 2mj$.

Let h' be the maximum height of any vertex in H'. Let $e = (u, v)$ be any cross edge. Let x be the rightmost descendant of u, and let y be the leftmost descendant of v, in H'. Let $l(e)$ be the number of cross edges enclosed by the characteristic closed curve of e. All these values can be computed in logspace. We embed e as an eastward grid path from the grid point corresponding to u till the eastward coordinate $2m\, w(x) + l(e) + 1$, followed by a southward path till the southward coordinate $2mh' + l(e) + 1$, followed again by an eastward path till the eastward coordinate $2m(w(y) - 1) + l(e) + 1$, followed by a northward path till the southward coordinate $2mh(v)$, finally followed by an eastward path to the grid point corresponding to v (see Figure 8).

Observe that the reduction, with an easy-to-compute labelling, preserves circuit value as well.

4.5 Step 5

We apply the conversion algorithm on G_5 to obtain a circuit C'. Since G_5 is a layered grid graph, C' is upward layered (since G_5 had only northeast and southeast edges, each layer consists of the vertices on a particular north-south grid line). Moreover, since G_5 is an SMPD, C' is a one-input-face circuit, with the inputs appearing on the external face.

We convert C' into an upward stratified circuit C''(thus completing the reduction), as follows: For each input gate that is on a layer V_i for some $i > 0$,

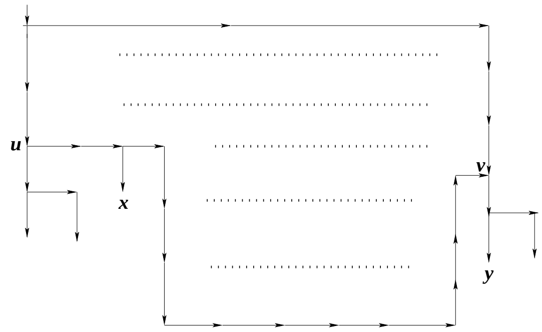

Fig. 8. Embedding of a typical cross-edge (u, v), after embedding the tree edges

add a copy of it to V_0, label the original gate as COPY, introduce a COPY gate at all intermediate layers $V_1, V_2 \ldots V_{i-1}$, and connect the new gate to the original gate through all these new gates. Again, this operation can be performed in logspace. Since the entire reduction consisted of a constant number of steps, each of which is in logspace, so the entire reduction is in logspace.

5 L-Hardness

In this section, we show that one-input-face MPCVP is L-hard under qfp's, by reducing ORD to it via qfp's.

Given an instance of ORD, we map it to the following instance of one-input-face MPCVP: there is an OR gate for every vertex v_i, which takes as input the gate corresponding to the vertex $S(v_i)$ and a constant gate (the single vertex that has no successor has only a constant gate as input). The constant input is 1 for t and 0 for all other vertices. The gate corresponding to s is made the output gate. Notice that the circuit thus constructed outputs 1 if and only if s precedes t in the ordering induced by S, *i.e.* the problem instance belongs to ORD. Clearly, the circuit is planar, and all constant gates(inputs) appear on the external face.

6 Results

Hence, we have proved that

Theorem 1. *One-input-face MPCVP is in* $\mathsf{LogDCFL}$*, but is* L*-hard under quantifier free projections.*

It was shown in [13] that monotone stratified cylindrical circuits can be evaluated in $\mathsf{LogDCFL}$, by reducing it to monotone upward stratified circuits in logspace, and then using the algorithm given in [5]. It was also shown in [13] that monotone one-input-face cylindrical circuits are in $\mathsf{L}(\mathsf{PDLP} \oplus \mathsf{LogDCFL})$, by reducing it to monotone stratified cylindrical circuits using oracle calls to

PDLP. Since one-input-face cylindrical circuits also have a one-input-face planar embedding (see [13]), so Theorem 1 trivially implies that both these problems are in LogDCFL.

Corollary 1. *One-input-face monotone cylindrical circuits (and hence monotone stratified cylindrical circuits) can be evaluated in* LogDCFL.

7 Conclusion

A close inspection of the logspace reduction, that we have described in Sections 3 and 4, reveals that it does not use the fact that the circuit is monotone, not even the fact that the circuit is Boolean. In other words, given any one-input-face planar circuit (need not be Boolean, *i.e.* gates and wires can take more than two values) with any kind of gates, we can produce an equivalent upward stratified circuit in logspace, provided we are allowed to use COPY gates. Hence, the reduction in this paper can be applied to much more general situations.

The exact complexity of one-input-face MPCVP remains open. In other words, is the problem solvable in L? Or is it hard for LogDCFL? General MPCVP has a larger gap between its lower and upper bounds. It is known to be hard only for L, while the best known upper bound is LogCFL.

Acknowledgement

We thank Nutan Limaye and Meena Mahajan for valuable discussions on the subject. We also thank Dave Barrington and Eric Allender for their comments on a preliminary version of this paper.

References

1. E. Allender, D. A. M. Barrington, T. Chakraborty, S. Datta and S. Roy. Grid Graph Reachability Problems. In *Electronic Colloquium on Computational Complexity, Technical Report TR05-149*, 2005.
2. E. Allender, S. Datta and S. Roy. The Directed Planar Reachability Problem. In *Proc. 25th annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LNCS vol. 3821*, pages 238–249, 2005.
3. E. Allender and M. Mahajan. The Complexity of Planarity Testing. In *Information and Computation, vol. 189(1)*, pages 117-134, 2004.
4. D. A. M. Barrington. Grid Graph Reachability Problems. In *Talk presented at Dagstuhl Seminar on Complexity of Boolean Functions, Seminar number 02121*, 2002.
5. D. A. M. Barrington, C.-J. Lu, P. B. Miltersen and S. Skyum. Searching Constant Width Mazes Captures the AC^0 Hierarchy. In *Proceedings of 15th International Symposium on Theoretical Aspects of Computer Science (STACS), LNCS vol. 1373*, pages 73–83, 1998.
6. A. L. Delcher and S. R. Kosaraju. An NC Algorithm for Evaluating Monotone Planar Circuits. In *SIAM Journal of Computing, vol. 24(2)*, pages 369-375, 1995.
7. P. W. Dymond and S. A. Cook. Complexity Theory of Parallel Time and Hardware. In *Information and Computation, vol. 80(3)*, pages 205–226, 1989.

8. K. Etessami. Counting quantifiers, successor relations, and logarithmic space. In *Journal of Computer and System Sciences , vol. 54(3)*, page 400-411, 1997.
9. L. M. Goldschlager. The Monotone and Planar Circuit Value Problems are Logspace Complete for P. In *SIGACT News, vol. 9(2)*, pages 25–29, 1977.
10. L. M. Goldschlager. A Space-Efficient Algorithm for the Monotone Planar Circuit Value Problem. In *Information Processing Letters, vol. 10(1)*, pages 25–27, 1980.
11. S. R. Kosaraju. On the Parallel Evaluation of Classes of Circuits. In *Proc. 10th annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LNCS vol. 472*, pages 232–237, 1990.
12. R. E. Ladner. The Circuit Value Problem is Logspace Complete for P. In *SIGACT News*, pages 18-29, 1975.
13. N. Limaye, M. Mahajan and J. Sarma M. N. Evaluating monotone circuits on cylinders, planes, and tori. In *Electronic Colloquium on Computational Complexity, Technical Report TR06-009*, 2006.
14. O. Reingold. Undirected st-Connectivity in Log-Space. In *Proceedings of 37th ACM Symposium on Theory of Computing (STOC), IEEE Computer Society Press*, pages 376–385, 2005.
15. H. Yang. An NC Algorithm for the General Planar Monotone Circuit Value Problem. In *SPDP: 3rd IEEE Symposium on Parallel and Distributed Processing.* ACM Special Interest Group on Computer Architecture (SIGARCH), and IEEE Computer Society, 1991.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/