Topology inside NC^1

Eric Allender
Rutgers University
New Brunswick, NJ 08855, USA
allender@cs.rutgers.edu

Samir Datta
Chennai Mathematical Institute
Chennai, TN 600 017, India *
sdatta@winlab.rutgers.edu

Sambuddha Roy
Rutgers University
New Brunswick, NJ 08855, USA
samroy@paul.rutgers.edu

Abstract

We show that ACC^0 is precisely what can be computed with constant-width circuits of polynomial size and polylogarithmic genus. This extends a characterization given by Hansen, showing that planar constant-width circuits also characterize ACC^0. Thus polylogarithmic genus provides no additional computational power in this model. We consider other generalizations of planarity, including crossing number and thickness. We show that thickness two already suffices to capture all of NC^1.

1. Introduction

The complexity class ACC^0 is one of the most important subclasses of NC^1. Barrington's characterization of NC^1 in terms of constant-width branching programs [3] highlighted the importance of algebraic considerations in studying small circuit complexity classes, and initiated a productive line of research reinforcing the connections between circuit complexity and formal language theory [6, 4, 11]. In this framework, computation over *non-solvable* monoids gives complete problems for NC^1, while computation over *solvable* monoids yields problems in ACC^0.

The class ACC^0 also attracts attention, because it lies at the frontier of current lower bound techniques. ACC^0 is the union of the classes $\mathrm{AC}^0[m]$ of problems computed by constant-depth polynomial-size circuits of AND, OR, and MOD_m gates. If m is prime, then $\mathrm{AC}^0[m]$ is known to be a proper subclass of ACC^0 [14, 13], but for m composite, it remains unknown if NEXP is contained in (nonuniform) $\mathrm{AC}^0[m]$.

*This research was done while this author was a postdoctoral associate at WINLAB, Rutgers University.

Last year, Hansen [9] proved a very surprising new characterization of ACC^0, in terms of constant-width circuits. Barrington's theorem [3] yields as a corollary a characterization of NC^1 as precisely the problems solvable by constant-width circuits of polynomial size. If NOT gates are allowed, then these circuits can be made to be planar, but if NOT gates are allowed only at the leaves (i.e., at the inputs), then Hansen is able to build on earlier work [10] to show that ACC^0 is precisely the class of languages accepted by polynomial-size constant-width *planar* circuits. This is a beautiful and unexpected characterization, making no blatant reference to counting mod m or to the algebraic considerations that have been central to all previous work on ACC^0.

TC^0 is an important complexity class lying between ACC^0 and NC^1. Since the papers of Hansen and Barrington combine to give characterizations of ACC^0 and NC^1 in terms of constant-width circuits, it was natural to pick up the question of whether TC^0 also corresponds to a class of constant-width circuits. Since we wanted to characterize a complexity class that is intermediate between ACC^0 and NC^1, we focused on graphs that are "intermediate" in some sense between planar and unrestricted. In this paper, we consider some of the most important graph-theoretic notions that generalize the concept of planarity:

- Crossing Number
- Genus
- Thickness

It will suffice for the reader to have an informal grasp of these notions; we provide a few additional definitions later on where they are needed. The *crossing number* of a graph is the least number of "edge intersections" required in any embedding of the graph in the plane. The *genus* of a graph

is the least number of "handles" (or "doughnut halves") that need to be attached to the plane in order to provide a surface on which the graph can be embedded with no edge crossings. The *thickness* of a graph is the smallest number of blocks needed in a partition of the edge set, so that the vertices can be embedded in a plane so that none of the edges in any block of the partition cross each other; intuitively this is the number of transparencies needed to represent the graph, where each transparency is planar. (For more complete definitions, please consult a graph theory text, such as [7].) Planar graphs have crossing number 0, genus 0, and thickness 1. For any graph G, thickness$(G) - 1 \leq$ genus$(G) \leq$ crossing.number(G).

Our main theorem is that constant-width polynomial size circuits of polylogarithmic genus compute exactly the problems in ACC^0. As a corollary, the same is true for circuits with polylogarithmic crossing number. In contrast, constant-width circuits of thickness two already suffice to compute all problems in NC^1. We can view this as a positive result, because it yields additional information about ACC^0 and NC^1. However, it is also in some sense a negative result, in that it removes the most prominent candidates for a possible characterization of TC^0 in terms of constant-width circuits. We leave the task of finding such a characterization as our main open problem.

2. Definitions and Preliminaries

We first define a layered digraph :

Definition 1 *We call a digraph* layered *if there is a partition of the vertex set into sets $V_0, V_1, \cdots, V_l$ (and we call them* layers *or* levels*) and every (directed) edge in the graph is from some layer V_i to V_{i+1}.*

Definition 2 *The* width *of a layered digraph with layers $V_0, \ldots, V_l$ is max$\{|V_i| : 0 \leq i \leq l\}$.*

A circuit of width w is a layered digraph of width w where each vertex is labeled either as an AND gate, an OR gate, an input variable x_i, or a negated input variable $\neg x_i$. It is important to note that inputs can appear on any level, and inputs can appear more than once.

A circuit is planar if it can be embedded in the plane with no two edges crossing. More generally, a circuit has genus $\leq k$ if it can embedded on a surface of genus k with no edges crossing. Because of Theorem 1 below, most of this paper can be read and understood without any detailed understanding of the topological notion of genus. (The detailed definitions that we do require are encapsulated within the proof of Theorem 1.) Theorem 1 allows us to restrict attention to a particular class of genus k surfaces, consisting of a plane with k "handles". (Informally, a "handle" is a bent cylinder that is attached to the plane at each end.) The

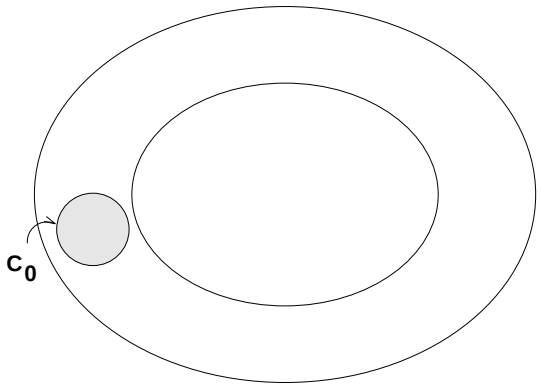

Figure 1. A facial cycle on a torus

two circles on the plane where the handle is attached are called the "handle connections". For any handle h, arbitrarily label one of its handle connections the "east" connection h_e and the other one the "west" connection h_w. When we embed a graph into a plane with k handles, we will consider only embeddings where each vertex is embedded in the plane.

Given a graph embedded on a plane with k handles $h_1, \ldots, h_k$, for any directed edge $e = (u, v)$ in the graph there is a word w_e over the alphabet $\{p\} \cup \{h_i, h_{i,e}, h_{i,w} : 1 \leq i \leq k\}$ recording the regions of the surface that are encountered while traversing the edge from u to v. Note that w_e begins and ends with p because all vertices are embedded in the plane. A *traversal* of handle h_i is a subword of the form $ph_{i,e}xh_{i,w}p$ or $ph_{i,w}xh_{i,e}p$, where $x \in \{h_{i,e}, h_{i,w}, h_i\}^*$. (That is, e traverses handle h_i if it enters at one end and exits at the other end.)

The following theorem can be viewed as presenting a "normal form" for genus k graphs, that will be convenient for us to work with.

Theorem 1 *Given a graph $G = (V, E)$ with genus k, there is an embedding of G into a plane with k handles such that*

- *Every vertex is embedded in the plane.*
- *$E = E_P \cup E_H$ where each edge in E_H traverses at least one handle, and each edge in E_P traverses no handle (and thus without loss of generality is embedded entirely in the plane, since one can slide any "partial traversal" out of the handle).*
- *Each handle connection lies in a face of the planar graph (V, E_P).*
- *For each edge $e \in E_H$ and each handle h, e traverses h at most twice.*

Proof: This proof was suggested to us by Carsten Thomassen; we thank him for allowing us to present it here.

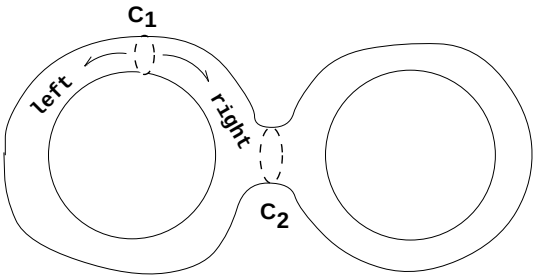

Figure 2. A surface-nonseparating cycle

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 [12] for more rigorous definitions.

A *closed orientable* surface is one which can be obtained by adding some handles to a sphere in 3-space (the genus of the resulting surface being equal to the number of handles added; see also the text [8]). 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 the cycle is the set of vertices of the graph that are path-connected to some vertex on the cycle, such that this path does not cross the cycle itself. A cycle thereby has two sides, which are called the *left* and the *right* sides. If the left and the right sides of a specific cycle have nonempty intersection, then we call the cycle a *surface-nonseparating cycle*. Note for instance 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. In Figure 1, C_0 is a facial cycle, hence surface separating. The shaded region is one side of C_0 and the rest of the torus the other side. In Figure 2, C_1 is surface-nonseparating while C_2 is not.

Given a graph G of genus k, consider an embedding of G onto a surface of genus k. If $k > 0$ then there will be a surface-nonseparating cycle [12, Lemma 4.2.4 and the following discussion]. Choose one such surface-nonseparating cycle C_1 in our graph G and cut along it ([12, p. 105]) – let $C_1 = \{v_{1,1}, v_{1,2}, \cdots, v_{1,r_1}\}$ be the cycle and let G_1 be the graph obtained from G by *cutting along* the cycle C. The graph G_1 has two copies of each of the vertices $\{v_{1,1}, v_{1,2}, \cdots, v_{1,r_1}\}$, which we denote by $\{v_{1,1,1}, v_{1,2,1}, \cdots, v_{1,r_1,1}\}$, and $\{v_{1,1,2}, v_{1,2,2}, \cdots, v_{1,r_1,2}\}$. For every undirected edge $(u, v_{1,j})$ on the right side of the cycle C_1 we have the edge $(u, v_{1,j,1})$ in G_1, and for every undirected edge $(u, v_{1,j})$ on the left side of the cycle C_1 we have the edge $(u, v_{1,j,2})$ in G_1. The graph G_1 also has two copies of the cycle C_1, which we denote by $C_{1,1}$ and $C_{1,2}$. That is, we have edges between $v_{1,j,b}$ and $v_{1,j+1,b}$ for each $b \in \{1, 2\}$ and each $1 \leq j \leq r_1$. An important property of cutting along the cycle C_1 is that in the resulting graph G_1, the copies $C_{1,1}$ and $C_{1,2}$ are *facial* cycles ([12, p. 106,Lemma 4.2.4]). Label the face corresponding to $C_{1,b}$ with the name "$C_{1,b}$"; since $C_{1,b}$ is facial it cannot be surface-nonseparating and hence it will never be selected as the cycle C_j in subsequent stages (although individual vertices on $C_{1,b}$ might appear on such a cycle C_j). That is, we will maintain the property that in all of the graphs G_j that are constructed in subsequent stages, there will be a face labeled $C_{1,b}$.

It is important to observe that the orientation of the vertices is reversed in $C_{1,1}$ and $C_{1,2}$; equivalently, if we were to connect a handle to the faces that have boundaries $C_{1,1}$ and $C_{1,2}$, then we could embed edges connecting $v_{1,j,1}$ and $v_{1,j,2}$ through the handle without introducing any edge crossings. We emphasize that G_1 contains exactly r_1 more edges than G, corresponding to the duplication of cycle C_1.

By Lemma 4.2.4 of [12], the genus of the graph G_1 is less than that of G.

If the genus of G_1 is still greater than zero, we can choose a surface-nonseparating cycle $C_2 = \{v_{2,1}, v_{2,2}, \cdots, v_{2,r_2}\}$ in G_1 and cut it along C_2 to obtain graph G_2, which has smaller genus than G_1 and which contains two facial cycles labeled $C_{2,1}$ and $C_{2,2}$. After k steps we obtain a graph G_k of genus zero, which we embed in the plane.

The graph G_k has faces labeled $C_{j,b}$ for $1 \leq j \leq k$ and $1 \leq b \leq 2$. Create a handle h_j with connections in the faces $C_{j,1}$ and $C_{j,2}$.

A single vertex v in G may correspond to many different vertices in G_k if copies of it were made in the various steps of cutting along the cycles C_j. For each v, we will create a tree T_v that connects all of these copies, as follows. For each pair of cycles $C_{j,1}$ and $C_{j,2}$ in G_k, add "temporary" edges through handle h_j connecting the vertices $v_{j,i,1}$ and $v_{j,i,2}$. The "temporary" edges that are added in this way connect all of the copies of each original vertex v with each other, but it will not in general be a tree. For each vertex v of the original graph, select one representative copy of v and create a rooted tree T_v consisting of "temporary" edges that connect v to each of its copies.

Now consider the graph H that results by taking graph G_k and performing the following steps:

1. Delete all edges that occur on any cycle $C_{j,b}$.
2. For each vertex v in turn, contract the "temporary" edges of T_v, and pull the copies of v to the root of T_v across the handles, bringing along the edges that are adjacent to the vertices of T_v.

This graph H has the same number of vertices as G. Any two vertices that are adjacent in H are adjacent in G. No

edge of H crosses any bridge more than once. H is embedded in a plane with k handles.

However, H is a proper subgraph of G. The only edges of G that are not present in H are the edges that correspond to edges of some cycle $C_{j,b}$ of G_k that were deleted in the first step of our construction of H. We need to embed those edges.

Consider any edge (v, u) of G that is absent in H; (v, u) corresponds to some edge (v', u') on a cycle $C_{j,b}$ in G_k. We embed an edge from v to u by following a path through the handles from v to the spot on the plane where v' was embedded (corresponding to a path in the spanning tree T_v), and continuing on to the spot on the plane where u' was embedded, and then through the handles (corresponding to a path in the spanning tree T_u) toward vertex u. The path from v to v' uses each handle at most once, and the same is true for the path from u' to u. Thus no edge traverses any handle more than twice. ∎

Theorem 1 leaves open the possibility that a single edge will traverse several handles. When discussing circuits, however, this complication can be avoided, as the following lemma demonstrates.

Lemma 2 *Given a layered circuit C of width w, genus k, and size s, there is an equivalent layered circuit C' of width $O(w^2)$, genus k, and size $O(skw^2)$ that can be embedded onto a plane with k handles satisfying the conditions of Theorem 1 with the additional restriction that no edge of C' traverses more than one handle.*

Proof: Consider an embedding of C into a plane with k handles, as guaranteed by Theorem 1. For any edge that traverses more than one handle, or that traverses some handle more than once, insert a new vertex between any two handle traversals. At most $2k - 1$ new vertices are added per edge. Since there are at most w^2 edges between any two layers of C, this adds at most $O(kw^2)$ new vertices. The modified graph is no longer layered. For each two adjacent levels $l, l + 1$ of C (that might now be separated by paths of length $2k - 1$), insert additional "dummy" gates (i.e., OR gates with one input and one output) to create the layered circuit C'. The new graph has width at most w^2 because at most w^2 "dummy" gates appear on any level of the resulting graph. ∎

The embedding of circuit C' guaranteed by Lemma 2 might have several handle connections attached to any given face of the planar part of the circuit. We find it convenient to modify the graph by adding additional non-functional edges to subdivide faces, so that no face contains more than one handle connection. This transformation might cause the width of the graph to increase, but because the new edges are purely an augmentation to the embedding and do not contain functional circuit edges, it will not cause problems for us.

Theorem 3 *Given a layered graph $G = (V, E_P \cup E_H)$ embedded in a plane with k handles satisfying the conclusions of Lemma 2, there is a layered graph $G' = (V \cup V', E_P \cup E'_P \cup E_H)$ whose embedding extends the embedding of G such that the graph $G'' = (V \cup V', E_P \cup E'_P)$ is embedded in the plane and no face of G'' has more than one handle connection inside it.*

Proof: Consider any face of the embedding of the planar graph (V, E_P). We will partition this face into a finite number of regions, assigning a color to each region. Let there be d handle connections inside this face, $h_1, \ldots, h_d$. Assign color c_i to connection h_i. For each edge e that enters (or exits) connection h_i, color e with color c_i on that portion of the edge that lies between the boundary of the face and the point at which it touches the connection h_i. (No segment receives two colors in this way.) If there are l edge segments adjacent to a handle connection h_i, then this gives rise to a partition of the face into l segments and l regions arranged around the handle connection like slices of pie. Some of these regions might contain other handle connections; those that do *not* contain other handle connections receive color c_i. No region receives more than one color in this way; regions that do not receive a color are said to be *white*. Any vertex on the boundary of the face that is adjacent only to regions of one color c_i receives color c_i; any vertex on the boundary of the face that is adjacent to regions of two or more different colors (one of which must be white) is colored white. If there is more than one handle connection in the face, then every handle connection is adjacent to some white region, and every white region is adjacent to some white vertex on the boundary.

Consider any white region that is adjacent to some handle connection h_i. The border of this white region includes some arc of the handle connection h_i (and it includes the entire handle connection if only one edge segment connects h_i to the border of the face). The ends of this arc are connected to edge segments that attach to some white vertices u and v on the border of the face (and note that $u = v$ in the degenerate case mentioned above). We can now embed a new edge in the white region, attaching u to v and creating a new face, which we now color with color c_i, thereby decreasing by one the number of white regions adjacent to the handle connection.

Repeat this process, until each handle connection h_i is completely surrounded by regions colored c_i. The boundary of the region colored c_i is now a planar face that contains exactly one handle connection. The graph might now no longer be layered, but it is straightforward to insert new vertices in the middle of the new edges, to obtain a layered graph. ∎

Cylindrical graphs play an important role in our analysis, just as they do in Hansen's work [9]. Cylindrical graphs

are a subclass of layered planar graphs, consisting of those graphs that can be embedded on the surface of a cylinder, where each layer of vertices is placed on a ring around the cylinder between its neighbor layers, and all edges lie on the surface of the cylinder going from one layer to the next, with no crossings.

We quote the theorem from [9]:

Theorem 4 *A layered digraph G is layered cylindrical if and only if it is the subgraph of an acyclic planar layered digraph with a unique source and sink.*

We will need some additional information about the transformation that Hansen uses to prove Theorem 4. Consider a marked face of the acyclic planar layered digraph G that is a subgraph of an acyclic planar layered digraph with a unique source and sink. In Hansen's transformation, this face corresponds to a marked face on the embedding of G on the cylinder. (That is, if a face on an acyclic planar layered digraph has vertices $\{v_1, v_2, \cdots v_r\}$, then there is a face on the cylinder with the same vertices in the same order.)

When we have a genus k graph $G = (V, E_P \cup E_H)$ embedded on a plane with k handles and additionally the planar graph (V, E_P) is cylindrical, then we obtain an embedding of G on a cylinder with k handles, where each face of the planar graph (V, E_P) corresponds to a face of the cylindrical embedding, and the handle connections are similarly attached to faces on the cylinder. We summarize this discussion in the following theorem.

Theorem 5 *Let $G = (V, E_P \cup E_H)$ be a layered digraph embedded in a plane with k handles satisfying the conclusions of Lemma 2, where the graph (V, E_P) is the subgraph of an acyclic planar digraph with a unique source and sink. Then there is a layered graph $G' = (V \cup V', E_P \cup E'_P \cup E_H)$ embedded into a cylinder with k handles such that the embedding of the subgraph $G'' = (V \cup V', E_P \cup E'_P)$ into the cylinder has the property that no face of G'' contains more than one handle connection.*

3. Small Genus Characterizes ACC^0

Theorem 6 *Let A be a language. A is in ACC^0 if and only if A is accepted by a family of constant-width circuits of polynomial size and polylogarithmic genus.*

Proof: One direction follows immediately from Hansen's characterization [9] where the genus is even required to be zero. For the other direction, we follow Hansen's basic strategy and prove the theorem by induction on the width w of the circuit family accepting A. More precisely, we will prove the following :

Claim 7 $\forall w \forall k \forall l \exists c \exists d$ *Circuits of width w and genus $\log^l n$ and size n^k can be simulated by ACC^0 circuits of depth d and size n^c.*

The basis, when $w = 1$ is trivially true.

For the inductive step, consider a circuit family $\{C_n\}$ of width $w + 1$, size n^k and genus $\log^l n$. Let $G' = (V \cup V', E_P \cup E'_P \cup E_H)$ be the graph guaranteed by Theorem 3, such that $G = (V, E_P \cup E_H)$ is the graph of the constant-width circuit C_n, where G is embedded into a plane with $\log^l n$ handles and no face of the planar graph $(V \cup V', E_P \cup E'_P)$ contains more than one handle connection.

Without loss of generality, there is a vertex s in layer 1 of G that is connected by a path to some vertex t in the rightmost layer of G. Let $G_{\rm cyl}$ be the subgraph of G consisting of all edges of G that lie on some path from s to t in G, and let $G_{\rm rest}$ be the remainder of G; the vertices of $G_{\rm cyl}$ and $G_{\rm rest}$ partition the vertices of G. Note that $G_{\rm rest}$ has width at most w because $G_{\rm cyl}$ contains at least one vertex from every level. Also note that by Theorem 4, $G_{\rm cyl}$ can be embedded on a cylinder with $\log^l n$ handles, because the planar part of $G_{\rm cyl}$ has a unique source and sink. If we consider $G_{\rm rest}$ as a circuit note that there are some gates whose inputs lie in $G_{\rm cyl}$; for each such gate h that is connected to a gate g of $G_{\rm cyl}$, create an input node and label it with variable y_g. Similarly, for each gate g of $G_{\rm cyl}$ that receives input from a gate h of $G_{\rm rest}$, create an input node and label it with variable z_h.

For each input variable z_h of $G_{\rm cyl}$ the subcircuit of $G_{\rm rest}$ on which gate h depends computes a function entirely of the original input variables (because if it relied on a variable y_g this would constitute a path from s to g to h to t and thus h would be part of $G_{\rm cyl}$). By induction hypothesis, the value of each such gate h can be computed by ACC^0 circuits of some fixed depth and size.

We show in Lemma 8 below that the value of each gate of $G_{\rm cyl}$ can be computed in ACC^0 by circuits that take the values z_h as input, along with the original input values. Combining these circuits with the ACC^0 circuits computing the values of the corresponding gates h of $G_{\rm rest}$ yields ACC^0 circuits that compute the values that each gate g of $G_{\rm cyl}$ takes on in the original circuit C_n.

Thus we can compute the correct values y_g that we can provide as input to the remaining parts of $G_{\rm rest}$, in order to compute the values of the output gates of C_n. The proof is now complete, once we have established Lemma 8. ∎

Lemma 8 $\forall w \forall k \forall l \exists c \exists d$ *such that circuits of width w and size n^k that are embedded on a cylinder with $\log^l n$ handles can be simulated by ACC^0 circuits of depth d and size n^c.*

Proof: Let $G = (V, E_P \cup E_H)$ be the graph of a circuit of width w and size n^k embedded on a cylinder with $\log^l n$ handles. Let $G' = (V \cup V', E_P \cup E'_P \cup E_H)$ be the graph

embedded on the same surface, such that no face of the cylindrical part of $G^{'}$ contains more than one handle connection, as guaranteed by Theorem 5. Let the levels of the layered graph $G^{'}$ be numbered $1, \ldots, p(n)$. Let the handle attachments of $G^{'}$ be $h_1, \ldots, h_s$ (for $s \leq 2\log^l n$).

Our first step will be to chop the cylinder into a polylogarithmic number of segments, corresponding to levels where handles start and stop. For a handle attachment h_i, define *start*(h_i) to be the least element of the set $\{j$: there is an edge (u, v) that traverses the handle attached at h_i such that u is on level $j\}$. Similarly, define *end*(h_i) to be the largest element of the set $\{j$: there is an edge $(u, v) \in E_H$ that traverses the handle attached at h_i such that v is on level $j\}$. Let $a_1 < a_2 < \ldots < a_r$ be numbers such that $\{a_j : 1 \leq j \leq r\}$ = {*start*(h_i), *start*$(h_i) + 1$, *end*(h_i), *end*$(h_i) - 1 : 1 \leq i \leq s\}$. Note that r is polylogarithmic in n.

Slice the cylinder into $r + 1$ segments, where segment 1 consists of layers $1 \ldots a_1$, segment 2 consists of layers $a_1 \ldots, a_2$, ...and segment $r + 1$ consists of layers $a_r \ldots, a_{p(n)}$.

We argue below that there is a function f computable in ACC^0 where f takes as input a triple (x, v, i) and outputs a string z such that v and z are bit strings of length w, having the property that if the w gates at the start of segment i have the values given by the vector v and the string x is used to provide values to the input gates appearing in segment i, then the gates at the output level of segment i will take on the values given by the vector z.

Assume for the moment that we have such a function f computable in ACC^0. Then we can build a graph with width 2^w and polylogarithmically many levels, such that there is an edge from node v to node z in level j if and only if $f(x, v, j) = z$. Finding paths in such graphs can be done in AC^0. (For instance, we can build a DNF of polynomial size that computes paths of length $\log n$, and in depth 4 we can compute paths of length $\log^2 n$, etc.) Thus the proof will be complete if we can show that f is computable in ACC^0.

Consider a segment that starts in layer a_j and ends in layer a_{j+1}. If $a_j + 1 = a_{j+1}$ then the circuitry in this segment is computable in NC^0 and thus certainly the desired function f can be computed in ACC^0.

Otherwise, $a_j + 1 < a_{j+1}$, so that the jth segment has length more than one. Let us say that handle connection h_i is *active* if *start*$(h_i) \leq a_j$ and *end*$(h_i) \geq a_{j+1}$. Because of the way the sequence of slice points $a_1, \ldots, a_r$ was defined and because we are dealing with the case where $a_j + 1 < a_{j+1}$, it follows that *start*$(h_i) < a_j$ and *end*$(h_i) > a_{j+1}$; that is, no handle connection is actually starting or ending at the start or end of this segment. It is important to note that, although G has width bounded by the constant w, there might be polylogarithmically many handles that are active in this segment, and the cylindrical part $(V \cup V^{'}, E_P \cup E_P^{'})$ of the graph $G^{'}$ (which has the property that at most one handle attachment is in any face) might also have polylogarithmic width. Let $h_1, h_2, \ldots, h_d$ be the handles that are active in segment j.

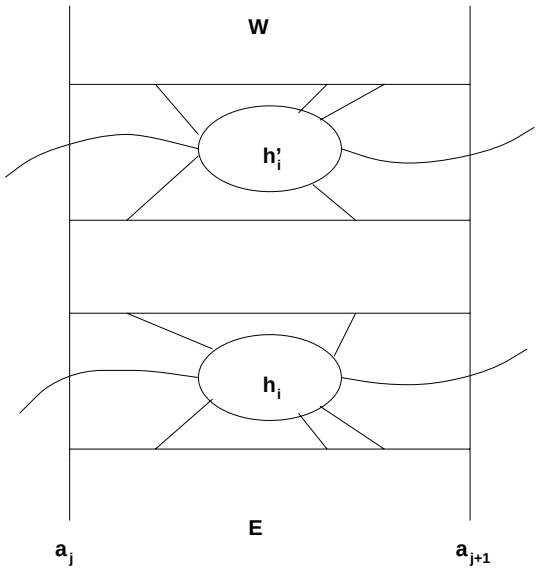

Figure 3. Handle attachment

Consider any handle attachment h_i that is active in segment j. Consider the face of the cylindrical part of $G^{'}$ to which h_i is attached. As illustrated in Figure 3, there are edges from levels before a_j and after a_{j+1} that enter or exit h_i. Although we draw our constant-width circuits with the outputs on the right end (so that computation proceeds from left to right), it will be convenient to use the compass points to refer to directions on the cylinder using the convention that the output level is North, so that the computation proceeds from South to North. Thus as depicted in the figure, East is to the bottom of the figure, and West is at the top. Using this convention, we can speak of the face around h_i as having an East side and a West side. The faces that surround $h_1, \ldots, h_d$ all start before the start of segment j and end after segment j. Thus we can view them as being stripes arranged along the sides of the cylinder. In this way, each handle connection h_i has an East neighbor and a West neighbor (where the East neighbor of h_i is the handle connection whose face is encountered first when moving East from the face of h_i around the cylinder). The handle that is connected to h_i on one end is connected to some other handle connection $h_{i'}$ on the other end. (This handle connection $h_{i'}$ need not be an East or West neighbor of h_i.) Because of the edges that connect h_i and $h_{i'}$ to levels outside segment j, it is clear that edges in E_H that traverse the handle between h_i and $h_{i'}$ must connect the East side of one face with the West side of the other face; any attempt to em-

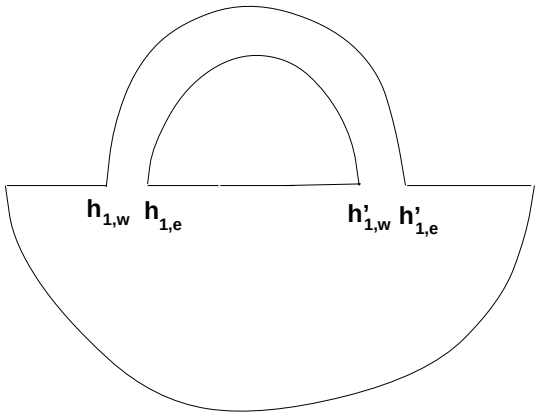

Figure 4. Cross-section of a cylinder with one handle in a segment

bed an edge between the West sides of the two faces would necessarily cross the edges that extend beyond segment j.

We claim that all of the edges in segment j are embedded onto at most d disjoint cylinders. This is illustrated in Figure 4 (with $d = 2$), which presents a cross-section of a cylinder (with East to the right and West to the left). The d handle connections $h_1, \ldots, h_d$ are arranged around the cylinder, each attached to a face of the cylindrical part of G'. We can diagram this cross-section by building a graph with vertices for the East and West sides of the faces around each handle connection h_i. The East side of each face is connected to its West neighbor; edges from the circuit can be embedded along this surface. The East side of each face is also connected to the West side of the face to which its handle jumps. Note that there are no edges from the East side of a face to the West side of the same face; no circuit edges are embedded in this region. The graph that is constructed in this way is 2-regular, and thus it is the union of disjoint cycles. If we create a 2-dimensional surface connecting these cycles at each end of segment j, we create a cylinder (in Figure 5, this process leads to exactly one cylinder). Every circuit edge appearing in segment j lies on one of these cylinders. For each cylinder, the circuit that is embedded on it is planar and has width at most w. Thus by [9] the function it computes lies in ACC^0.

∎

4. A new characterization of NC^1

Genus is just one of several possible generalizations of planarity. In this section we consider *thickness*, and we show that all problems in NC^1 can be solved by constant-width polynomial-size circuits of thickness two. We actually prove a stronger result showing that a very limited type

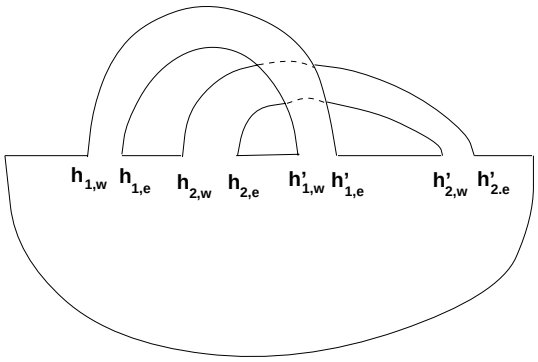

Figure 5. Cross-section of a cylinder with multiple handles in a segment

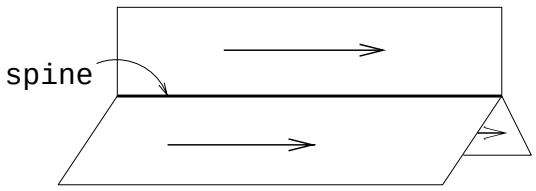

Figure 6. A circuit on three pages

of circuit with thickness two suffices for this task.

Consider Figure 6, showing three half-planes joined at a common intersecting line called the *spine*. This is the type of surface on which we will embed our constant-width circuits, with the restriction that the subgraph on any one half-plane is *upward planar*. (Equivalently, in each page, all edges between adjacent layers can be represented by straight line segments.) It is clear that any graph that can be embedded on three pages in this way has thickness two. It is also clear that if only two pages are used, then the entire graph is upward planar, and by [5] such circuits can compute only languages in AC^0.

Now the question arises as to what happens when we allow more pages in our circuit. Defining k*Pages* to be the class of languages captured by computation on k pages, each of $O(1)$ width, we prove that

Theorem 9 $\mathrm{NC}^1 = 3$Pages.

In order to present our characterization of NC^1 in terms of constant-width circuits on three pages, it is useful to define a simple nonuniform model of computation:

Definition 3 *Define* stacks(a, b, c) *to be the class of languages accepted by machines with three pushdown stores (with heights bounded by* a*,* b*, and* c*, respectively) and a computation register. Only binary values can be stored on the stacks. The program for the machine consists of a sequence of instructions (one instruction for each time step), from the following:*

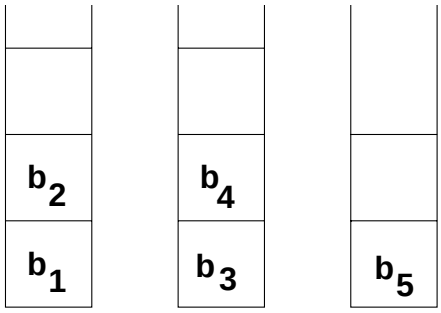

Figure 7. Canonical placement of bits in a stacks(3,3,2) computation

1. *push the register value into a stack;*
2. *pop the topmost entry from a stack into the register;*
3. *copy the topmost entry from a stack into the register (without removing it from the stack);*
4. *discard the topmost entry of a stack;*
5. *compute the $\vee$ or $\wedge$ of the topmost entries of two stacks and store it in the register; or*
6. *compute the $\vee$ or $\wedge$ of the topmost entry of a stack with an input literal and store it in the register.*

Notice that the machine is oblivious in that the stack(s) and literal corresponding to an instruction are independent of the input. The output of the machine is the final value that is stored in the register, and we restrict the running time to be polynomial in the length of the input.

See Figure 7 which illustrates stacks(3,3,2). Our main simulation using this model (in Lemma 12) involves permuting five values stored in the stacks as displayed in Figure 7. Manipulating these values will be accomplished using stack heights (3,3,2).

It is easy to see that oblivious computation with 3 stacks can be simulated by computation with 3 pages. The height of the stacks corresponds to the width of the pages with the spine serving as the register. Notice that popping a bit corresponds to copying all the bits on the corresponding page toward the spine while pushing is the reverse. An operation such as the $\vee$ of the topmost bits of two stacks can be simulated by bringing the corresponding bits toward the spine where the $\vee$ operation is performed. Therefore we have the following proposition:

Proposition 10 stacks$(O(1), O(1), O(1)) \subseteq$ 3Pages.

Since 3Pages consists of problems computable by polynomial length constant width circuits, we have:

Fact 11 3Pages $\subseteq$ NC^1.

Thus the following lemma will show that NC^1, 3Pages and stacks(3,3,2) all coincide.

Lemma 12 NC$^1 \subseteq$ stacks$(3, 3, 2)$.

Proof: Consider Barrington's proof that languages in NC^1 can be computed by permutation branching programs over S_5 [3]. The state of the branching program at some stage of the computation can be represented as a sequence of five bits $(b_1, b_2, b_3, b_4, b_5)$, where $b_j = 1$ if the execution of the branching program has led from the start state to state j; since the branching program in Barrington's simulation is deterministic, exactly one of the bits b_j will be set to 1 at any stage. We represent the initial state by (1,0,0,0,0). A permutation branching program consists of a sequence of instructions $(x_i, \theta_0, \theta_1)$, with the interpretation that if x_i is equal to 0, then the representation of the state of the program after the next step is obtained by permuting the five bits $(b_1, b_2, b_3, b_4, b_5)$ according to permutation θ_0, whereas permutation θ_1 is used if $x_i = 1$. Without loss of generality, we may assume that one of $\{\theta_0, \theta_1\}$ is the identity permutation ι (because instruction $(x_i, \theta_0, \theta_1)$ can be simulated by $(x_i, \theta_0, \iota)(x_i, \iota, \theta_1)$. We may further assume that each θ_b is a transposition, since each permutation can be represented as a product of transpositions. To determine if an input word is accepted, it suffices to check if $b_1 = 1$ after the last instruction is executed.

The idea of the proof is that we put the five bits in the three stacks using heights $2, 2, 1$ respectively in a canonical way, as illustrated in Figure 7. Each instruction is of the form (x_i, τ, π) where one of $\{\tau, \pi\}$ is a transposition and the other is ι. We focus attention on two types of instructions (x_i, τ, π):

Type 1 The transposition swaps the bits that appear on the tops of two different stacks.

Type 2 The transposition swaps the bits that appear in one stack.

Note that every instruction can be simulated by a sequence of instructions of Type 1 and Type 2. (For instance, in Figure 7, to swap bits b_1 and b_3, we first use Type 2 instructions to invert stacks 1 and 2, then use a Type 1 instruction on stacks 1 and 2, and then again invert stacks 1 and 2.)

Lemma 13 *1. A Type 1 instruction on two stacks can be accomplished by using exactly one extra place in the other stack.*

2. *A Type 2 instruction on one stack can be accomplished by using exactly one more place in each of the other two stacks.*

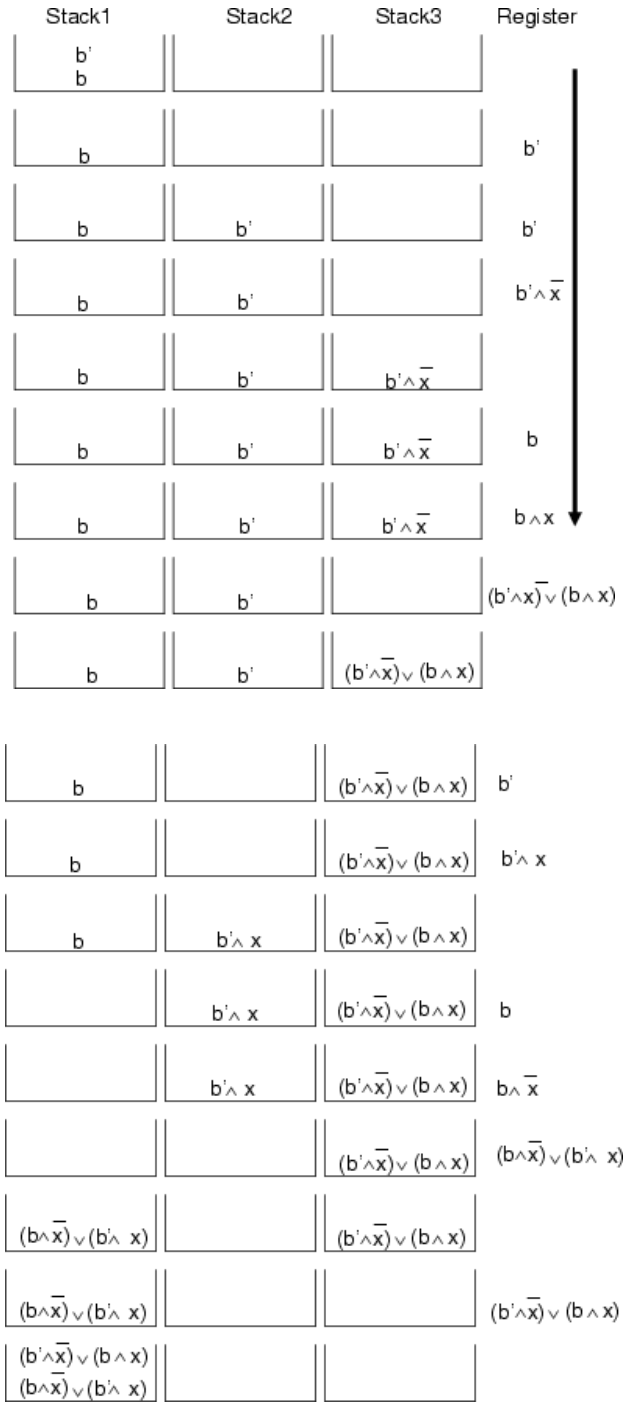

Figure 8. A transposition (if input bit x = 1) and identity transformation (if x = 0)

Proof: Figure 8 illustrates the proof of Lemma 13.2. A Type 2 instruction decides whether to interchange two bits b, b' or leave them alone, depending on the value of the literal x. This corresponds to replacing the sequence (b', b) on one stack, with the sequence $((\bar{x} \wedge b) \vee (x \wedge b'), (\bar{x} \wedge b') \vee (x \wedge b))$. This is implemented by computing each of the two expressions in succession and (during the computation of the second) removing the bits b, b'.

The proof of Lemma 13.1 can be accomplished by starting with the third step in Figure 8. ∎

Lemma 12 now follows immediately from Lemma13, since an operation of Type 2 on the first stack requires heights $2, 3, 2$, and an operation of Type 2 on the second stack requires heights $3, 2, 2$, for a maximum of $3, 3, 2$. Any other operation requires less overhead. ∎

5. Discussion

Our initial goal of finding a characterization of TC^0 in terms of constant width circuits remains a challenge for future work. It is worth mentioning some approaches that seem not to work. The results in this paper seem to rule out characterizations in terms of crossing number, genus, or thickness. Algebraic approaches to circuit complexity tend to mimic the structure of regular sets, and it is known that any regular set that is not complete for NC^1 lies inside ACC^0 [3], [6]; thus this avenue does not seem promising when searching for a characterization of TC^0. One might attempt to follow the approach of [1] by considering arithmetizations of Boolean circuits. However, a width-two planar branching program is presented in [2] for which the problem of counting the number of accepting paths is hard for NC^1 under ACC^0 reductions; this rules out many approaches that one might try in searching for a characterization of TC^0. On the positive side, a characterization of $\#AC^0$ (and hence of TC^0) in terms of counting paths in a restricted class of branching programs is presented in [2], but this does not seem to lead to an appealing characterization in terms of constant-width arithmetic circuits.

We find it somewhat intriguing that the graph-theoretic notion of genus is linked (via Theorem 6 and [6]) to the algebraic notion of solvability. It would be interesting to know if there are deeper reasons for this linkage.

6. Acknowledgments

We thank Robin Thomas, Dan Archdeacon, Bojan Mohar, Bill Steiger, Meena Mahajan, Kasturi Varadarajan, and Carsten Thomassen for their generous help in answering our questions about graph genus.

References

[1] M. Agrawal, E. Allender, and S. Datta. On TC^0, AC^0, and arithmetic circuits. *Journal of Computer and System Sciences*, 60:395–421, 2000.

[2] A. Ambainis, E. Allender, D. A. M. Barrington, S. Datta, and H. LêThanh. Bounded depth arithmetic circuits: Counting and closure. In *Proc. ICALP*, number 1644 in Lecture Notes in Computer Science, pages 149–158. Springer, 1999.

[3] D. A. Barrington. Bounded-width polynomial size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[4] D. A. M. Barrington, K. J. Compton, H. Straubing, and D. Thérien. Regular languages in NC^1. *Journal of Computer and System Sciences*, 44:478–499, 1992.

[5] D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. On monotone planar circuits. In *IEEE Conference on Computational Complexity*, pages 24–, 1999.

[6] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35:941–952, 1988.

[7] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[8] J. Gross and T. Tucker. *Topological Graph Theory*. John Wiley and Sons, New York, 1 edition, 1987.

[9] K. A. Hansen. Constant width planar computation characterizes ACC^0. In *21th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 44–55, 2004.

[10] K. A. Hansen, P. B. Miltersen, and V. Vinay. Circuits on cylinders. In *Fundamentals of Computation Theory, 14th International Symposium, FCT 2003, Malmö, Sweden, August 12-15, 2003, Proceedings*, pages 171–182, 2003.

[11] P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata-theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

[12] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, Maryland, 1 edition, 2001.

[13] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Matem. Zametki*, 41(4):598–607, 1987.

[14] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 77–82, 1987.