Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 108 (2004)

Topology inside NC^1

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

Samir Datta†
Chennai Mathematical Institute
Chennai, TN 603 103, India
sdatta@cmi.ac.in

Arsenii Karnaukhov
Independent Researcher
senixka@gmail.com

Sambuddha Roy ‡
Qualtrics
Seattle, WA 98101, USA
shombuddho@gmail.com

Alexander Shekhovstov
Columbia University
New York, NY 10027, USA
alex.v.shekhovtsov@gmail.com

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 [16], 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 constant-width circuits of polynomial size and thickness two already suffice to capture all of NC^1.

1. Introduction

Circuit complexity provides a vocabulary for classifying and understanding the complexity of Boolean functions. The complexity class NC^1 plays a central role in the study of circuit complexity; NC^1 can be defined either as the class of Boolean functions that can be represented by Boolean formulae of size polynomial in n (where n is the number of input variables), or as the class of Boolean functions that can be computed by circuits of depth $O(\log n)$, consisting of gates of fan-in $O(1)$. Our focus in this paper is on NC^1 and its subclasses.

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 [6] 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 [9, 7, 21]. 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 [24, 23], but for m composite, it remains unknown if EXP is contained in (nonuniform[1]) $\mathrm{AC}^0[m]$. Indeed, it was viewed as a major advance when it was shown that NEXP is not contained in (nonuniform) ACC^0 [27]; subsequently, Murray and Williams improved this, to show that nondeterministic quasipolynomial time is not in ACC^0 [22]. Other lower bounds for ACC^0 circuits building on [27, 22] can be found in [11, 12, 13, 14, 26].

*Supported in part by NSF grants CCF-0832787 and CCF-1064785.

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

‡Work done while this author was a graduate student at Rutgers University, NJ, USA

[1]Briefly, a circuit family $\{C_n : n \in \mathbb{N}\}$ is said to be *uniform* if there is an efficient algorithm to construct C_n. Our focus in this paper is on nonuniform circuit families. For more background, consult the text by Vollmer [25].

ISSN 1433-8092

In 2006, Hansen [16] gave a very surprising characterization of ACC^0, in terms of constant-width circuits. Barrington's theorem [6] 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 [19] 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. Subsequently, it was shown that even the more restrictive model of planar nondeterministic branching programs suffices to recognize all languages in ACC^0 [18].

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 that would be necessary in order to represent the graph, where each transparency is planar. (For more complete definitions, please consult a graph theory text, such as [15].) Planar graphs have crossing number 0, genus 0, and thickness 1. For any graph G, $\text{thickness}(G) - 1 \leq \text{genus}(G) \leq \text{crossing.number}(G)$.

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.

If C is a circuit whose graph has genus k, we will say that C has genus k.

A preliminary version of this work appeared as [4, 3]. However, the proof of the main theorem presented in the preliminary version is incorrect, as discussed in [2], where the task of finding a correct proof was listed as an open question. The third and fifth authors of the current article found a quite simple proof, with some assistance from ChatGPT. That is the proof that is presented in Section 3.

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*) where 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 $V_0, \ldots, V_r$ is max$\{|V_i| : 0 \leq i \leq r\}$.*

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.

Hansen's characterization of ACC^0 in terms of planar constant-width circuits is the starting point for our investigation:

Theorem 1 *[16] ACC^0 is equal to the set of languages accepted by constant-width planar circuits of polynomial size.*

Definition 3 *If C is a layered circuit with layers $V_0, \ldots V_r$, then $C[a, b]$ denotes the subcircuit of C spanned by layers $V_a \ldots, V_b$.*

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. The reader need have no detailed understanding of the topological notion of genus; an informal grasp of the topic is sufficient. Informally, a graph has genus k if it can be embedded with no edge crossings on a plane with k "handles", where a "handle" is a bent cylinder that is attached to the plane at each end.

Definition 4 *The genus of G, denoted $\gamma(G)$ is the least number k such that G can be embedded in a surface of genus k. Thus a planar graph has genus 0.*

The preliminary version of this work [4, 3] contained a detailed discussion of certain properties that an embedding of a circuit onto a surface of genus k could be assumed to have, summarized as [4, Theorem 1]. Although it is possible that these properties could be useful for future work, they are not needed for the simplified proof of our main theorem, and thus this material has not been included in this revision.

Instead the main fact that we need about genus is summarized in the following theorem from [10, Corollary 2]:

Theorem 2 *Suppose the graph G consists of connected components $G_1, \ldots, G_k$. Then,*

$$\gamma(G) = \gamma(G_1) + \ldots + \gamma(G_k).$$

3. Small Genus Characterizes ACC^0

Theorem 3 *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 [16] where the genus is even required to be zero.

For the other direction, we first require the following technical lemma:

Lemma 4 *Let C be a layered circuit of genus g, with layers $V_0, \ldots, V_r$. Then, there is a number $q \leq 3g + 10$ and indices $0 = t_1 < t_2 < \ldots < t_q = r$, such that for any $i = 1 \ldots q - 1$,*

- *either $C[t_i, t_{i+1}]$ is planar, or*
- *$t_{i+1} = t_i + 1$.*

Proof: We construct the sequence t_i greedily. For $i = 2, 3, \ldots$ define $t_i > t_{i-1}$ to be the largest number such that $C[t_{i-1}, t_i]$ is planar. If even $C[t_{i-1}, t_{i-1} + 1]$ is not a planar graph, then set $t_i = t_{i-1} + 1$. We proceed until step $i = q$ when we reach $t_q = r$.

We claim that $q \leq 3g + 10$. To see this assume, $q > 3g + 10$, and we will derive a contradiction. By definition, $C[t_i, t_{i+1} + 1]$ is not planar for any $i < q - 1$. For $3i + 1 < q$, the graphs $C[t_{3i}, t_{3i+1} + 1]$ are disjoint and are not planar. Therefore, by Theorem 2 we have genus of C, $\gamma(C) \geq \sum_i \gamma(C[t_{3i}, t_{3i+1} + 1]) > g$ (since removing edges does not increase genus, and since each $C[t_{3i}, t_{3i+1} + 1]$ has genus at least 1). This is the desired contradiction. ∎

We now continue with the proof of the main theorem. Let $\{C_n : n \in \mathbb{N}\}$ be a family of circuits of polynomial size with width w and genus $g = O(\log^c n)$. Consider a given circuit C_n. Let $V_0, \ldots, V_r$ be the layers of C_n. Let $0 = t_1 < t_2 < \ldots < t_q = r$ be given by Lemma 4. For any $i < q$, $C[t_i, t_{i+1}]$ can be computed by ACC^0 circuits. This is because either $C[t_i, t_{i+1}]$ is planar (in which case this claim follows from Theorem 1) or $t_{i+1} = t_i + 1$ holds and $C[t_i, t_{i+1}]$ can even be computed by AC^0. For every i such that $C[t_i, t_{i+1}]$ is planar, we use Hansen's theorem to convert it into a constant-depth AND, OR, MOD$_m$ circuit C_i' (not necessarily planar). It is important to note that the modulus m depends only on the width of the circuits in the circuit family, and thus the same modulus m can be used for all of the subcircuits of C_n, for every n. This is implicit in [16], but it is appropriate to say a bit more about this point here. In [16], Hansen shows that simulating planar circuits of width w can be reduced to simulating *cylindrical* circuits of width no more than w, and then appealing

to a theorem of [19], in which it is shown that cylindrical circuits of width w' compute only functions in ACC^0 (where the proof of this inclusion uses modular gates of modulus m, where the modulus depends only on the width w'). Since there are $O(1)$ different choices of $w' \leq w$, this possibly gives rise to $O(1)$ different moduli $m_1, \ldots m_c$, but (as observed in [24]), computation mod m_i can be simulated with a MOD_m gate, for $m = \prod_{i=1}^{c} m_i$.

Thus there is a function $f : \{0,1\}^n \times \{0,1\}^w \times [q-1] \rightarrow \{0,1\}^w$ 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. This is because, using AC^0 circuitry, we can use the string i to select the ACC^0 circuit for segment i, and then use x and v to compute the desired output z.

Thus, given x, in ACC^0 we can build a layered graph with width $2^w = O(1)$ and polylogarithmically many levels, such that there is an edge from node v in level j to node z in level $j+1$ if and only if $f(x, v, j) = z$. More precisely, when we say that we can "build" such a graph, we mean that there is a set of predetermined vertices and for each pair of vertices (u, v) we compute the value of a Boolean variable $b_{(v,u)}$ that represents existence of the edge $u \rightarrow v$. Furthermore, input x is accepted if and only if there is a path in this graph from the string v in the first level encoding the initial gate values in the circuit, to a string z in the final level where the output gate has value 1. Finding paths in such graphs can be done in AC^0.

Let us describe how to check if there exists a path between two vertices in such a graph G using AC^0 circuitry. If two vertices v and u are l layers apart, then there are at most 2^{wl} possible paths between them $v = v_1, \ldots, v_l = u$. We can build a disjunctive normal form formula $\vee_{v=v_1,\ldots,v_l=u}(b_{(v_1,v_2)} \wedge b_{(v_2,v_3)} \wedge \ldots \wedge b_{(v_{l-1},v_l)})$ that checks if there exists a path between v and u. Set $l = \Theta(\log n)$, then this formula has depth 2 and polynomial size. Denote by $V_0, V_1, \ldots, V_q$ the layers of G, and wlog assume that q is divisible by l. For every i and $u \in V_{il}, v \in V_{(i+1)l}$ we compute if there is a path between them. Consider a modified graph G' obtained from G by leaving only layers $V_0, V_l, V_{2l}, \ldots, V_q$ and connecting two vertices in adjacent layers if and only if there is a path between them in G. Note that the number of layers in G' is $q/l + 1$. We have reduced our problem to checking if there is a path between two vertices in G'. let us recursively check the existence of this path. Since the initial number of layers is $O(\log^c n)$ and $l = \Theta(\log n)$, the recursion depth is $O(c)$. On each level of recursion the number of pairs of vertices is polynomial and the DNF for computing existence of a path between is polynomial as well. Therefore, the final circuit has polynomial size and constant depth.

∎

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 of circuit with thickness two suffices for this task. First recall Barrington's characterization of NC^1:

Theorem 5 ([6]) NC^1 *is the class of languages accepted by constant-width polynomial-size circuits with AND, OR, NOT gates.*

Consider Figure 1, 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 no wire goes through more than one page and the subgraph on any one half-plane is *upward planar*. (A layered circuit is *upward planar* if, when the layers are arranged in columns with layer 0 on the left and the output layer on the right, all edges from any layer i to $i+1$ can be embedded as a straight line segment, with no edge crossings. The more general planar circuits that characterize ACC^0 allow edges embedded as curves that loop around the initial and final layers. It is known that constant-width upward planar circuits of polynomial size characterize AC^0 [8].) It is easy to see that if only two pages are used, then the entire graph is upward planar, and by [8] such circuits can compute only languages in AC^0. Let us show that any graph that can be embedded on three pages in this way has thickness two.

Lemma 6 *Any graph embedded in three pages, such that every edge is within only one page, has thickness two.*

Proof: Consider a graph embedded on three pages. Let us rotate the third page around the spine until it coincides with the first page. By doing small perturbations we can assume that no two different vertices in the third page and first page coincide.

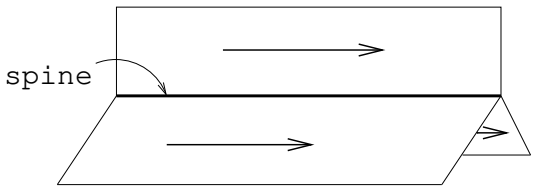

Figure 1. A circuit on three pages

Now, let us color the edges on the first and second page in red, while the edges on the third page in black. By definition, the red edges do not cross and the black edges do not cross. Therefore, the thickness of the graph is at most 2. ∎

Now the question arises as to what happens when we allow more than two pages in our circuit. Define k*Pages* to be the class of languages captured by constant-width polynomial-size circuits on k pages. Recall that we allow a gate to be an OR, an AND, or an input literal, and there can be multiple occurrences of the same input literal. Theorem 5 implies k*Pages* $\subseteq$ NC^1 for every k.

We show that three pages suffice:

Theorem 7 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 5 *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:*

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 2 which illustrates stacks(3,3,2). Our main simulation using this model (in Lemma 10) involves permuting five values stored in the stacks as displayed in Figure 2. Manipulating these values will be accomplished using stack heights (3,3,2).

Let us show 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. The stack is placed on the page so that its top element is closer to the spine than its bottom element. Notice that popping a bit corresponds to copying it toward the spine while pushing is the reverse (note that this can be accomplished by using single-variable AND gate). 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. After an operation is performed we copy the contents of the stacks one layer down and proceed with the next operations. See the illustration Figure 3 for details.

Therefore we have the following proposition:

Proposition 8 stacks$(O(1), O(1), O(1)) \subseteq 3$*Pages.*

We have already observed that one inclusion is trivial:

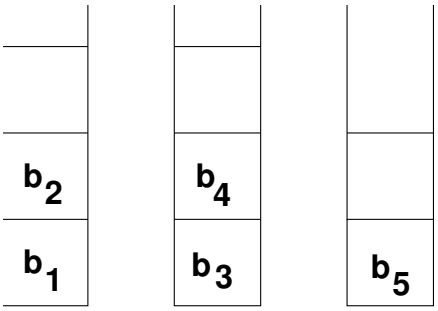

Figure 2. Canonical placement of bits in a stacks(3,3,2) computation

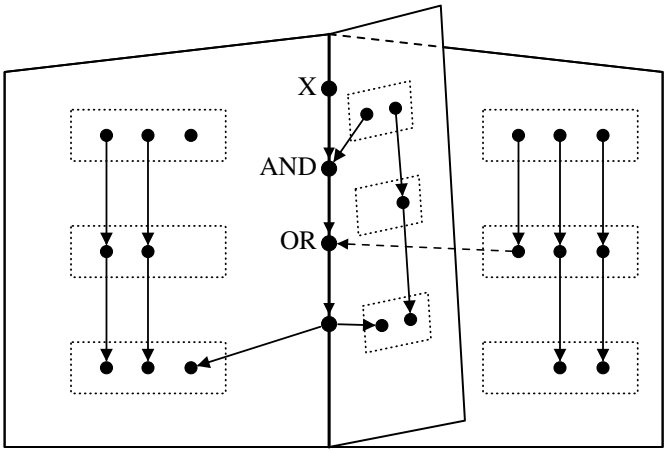

Figure 3. An illustration of how stacks$(O(1), O(1), O(1))$ are realized on 3 pages.

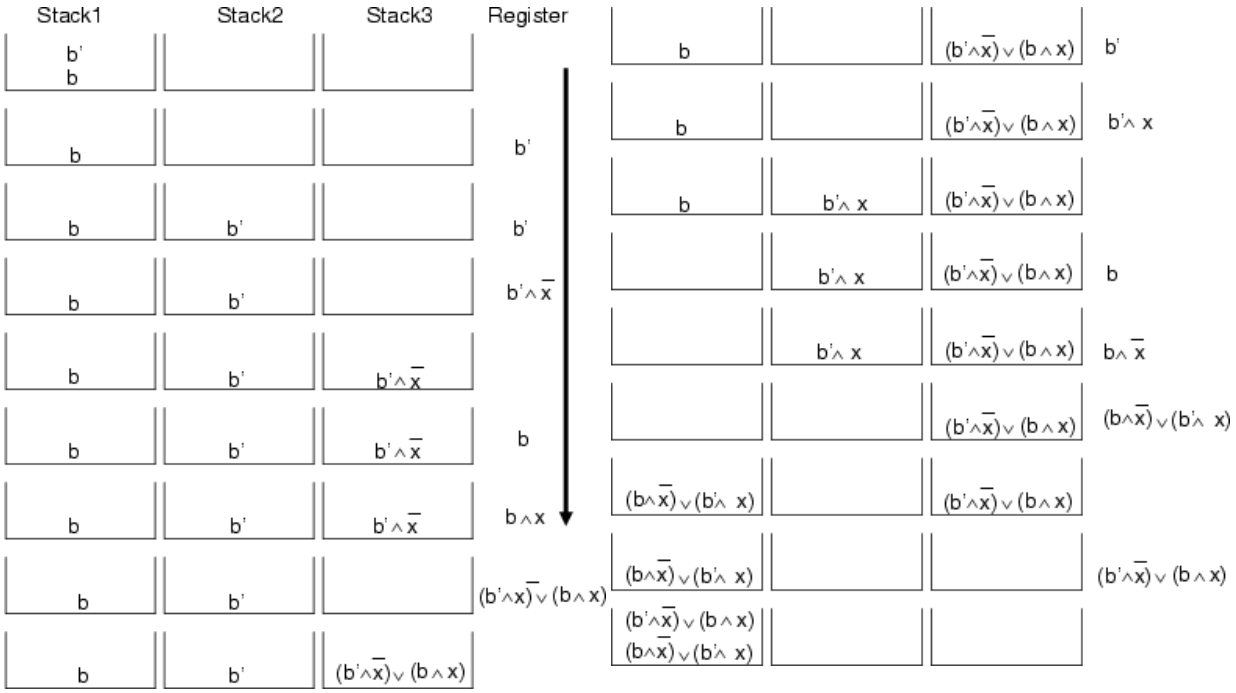

Figure 4. A transposition (if input bit x = 1) and identity transformation (if x = 0)

Fact 9 3Pages $\subseteq$ NC^1.

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

Lemma 10 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 [6]. 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 2. 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 2, 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 11 *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.

Proof: Figure 4 illustrates the proof of Lemma 11.2. 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 11.1 can be accomplished by starting with the third step in Figure 4 (and changing the last step, so that it moves the contents of the register to Stack 2). ∎

Lemma 10 now follows immediately from Lemma 11, 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 [6], [9]; 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 [5] 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 [5]. This characterization

has been extended, to give a characterization of arithmetic NC^1 in terms of a class of log-width planar branching programs [20].

We find it somewhat intriguing that the graph-theoretic notion of genus is linked (via Theorem 3 and [9]) to the algebraic notion of solvability. It would be interesting to know if there are deeper reasons for this linkage.

In parting, we should also mention a result of Hansen [17] where he proves that *quasipolynomial* size constant width nondeterministic branching programs exactly capture *quasipolynomial* size ACC^0. Hansen's result immediately implies that quasipolynomial size ACC^0 may either be characterized by quasipolynomial size polylog genus constant width circuits or by polylog genus constant width (nondeterministic) branching programs. All of these characterizations hold in the nonuniform setting. A number of obstacles would need to be overcome, before a corresponding result can be proven for uniform circuit complexity.

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. A preliminary version of this research (with an erroneous proof of Theorem 3) was presented at the 20th Annual IEEE Conference on Computational Complexity (CCC), in 2005.

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] E. Allender. Guest column: Parting thoughts and parting shots (read on for details on how to win valuable prizes!). *SIGACT News*, 54(1):63–81, 2023.

[3] E. Allender, S. Datta, and S. Roy. Topology inside NC^1. *Electronic Colloquium on Computational Complexity (ECCC)*, (108), 2004.

[4] E. Allender, S. Datta, and S. Roy. Topology inside NC^1. In *IEEE Conference on Computational Complexity*, pages 298–307, 2005.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] J. Battle, F. Harary, Y. Kodama, and J. W. T. Youngs. Additivity of the genus of a graph. *Bulletin of the American Mathematical Society*, 68(6):565–568, 1962.

[11] L. Chen. New lower bounds and derandomization for ACC, and a derandomization-centric view on the algorithmic method. In Y. T. Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[12] L. Chen. Nondeterministic quasi-polynomial time is average-case hard for ACC circuits. *SIAM J. Comput.*, 54(4):S19–332, 2025.

[13] L. Chen, X. Lyu, and R. R. Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In S. Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 1–12. IEEE, 2020.

[14] S. Chen and P. A. Papakonstantinou. Depth reduction for composites. *SIAM J. Comput.*, 48(2):668–686, 2019.

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

[16] K. A. Hansen. Constant width planar computation characterizes ACC^0. *Theory of Computing Systems*, 39(1):79–92, 2006.

[17] K. A. Hansen. Constant width planar branching programs characterize ACC^0 in quasipolynomial size. In *23rd IEEE Conference on Computational Complexity (CCC)*, pages 92–99, 2008.

[18] K. A. Hansen and M. Koucký. A new characterization of ACC^0 and probabilistic CC^0. *Computational Complexity*, 19(2):211–234, 2010.

[19] K. A. Hansen, P. B. Miltersen, and V. Vinay. Circuits on cylinders. *Comput. Complex.*, 15(1):62–81, 2006.

[20] M. Mahajan and B. V. R. Rao. Arithmetic circuits, syntactic multilinearity, and the limitations of skew formulae. In *Proc. MFCS*, number 5162 in Lecture Notes in Computer Science, pages 455–466. Springer, 2008.

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

[22] C. D. Murray and R. R. Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020.

[23] 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.

[24] 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.

[25] H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.

[26] F. Wang. NEXP does not have non-uniform quasipolynomial-size ACC circuits of $o(\log\log n)$ depth. In *Proc. TAMC*, number 6648 in Lecture Notes in Computer Science, pages 164–170. Springer, 2011.

[27] R. Williams. Non-uniform ACC circuit lower bounds. In *IEEE Conference on Computational Complexity*, pages 115–125, 2011.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il