Circuits on Cylinders

Kristoffer Arnsfelt Hansen
Peter Bro Miltersen
V. Vinay

BRICS Report Series RS-02-50

ISSN 0909-0878 2002

Circuits on Cylinders

Kristoffer Arnsfelt Hansen[*] Peter Bro Miltersen[*] V Vinay[†]

December 2002

Abstract

We consider the computational power of constant width polynomial size cylindrical circuits and nondeterministic branching programs. We show that every function computed by a $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ circuit can also be computed by a constant width polynomial size cylindrical nondeterministic branching program (or cylindrical circuit) and that every function computed by a constant width polynomial size cylindrical circuit belongs to $\mathbf{ACC}^0$.

1 Introduction

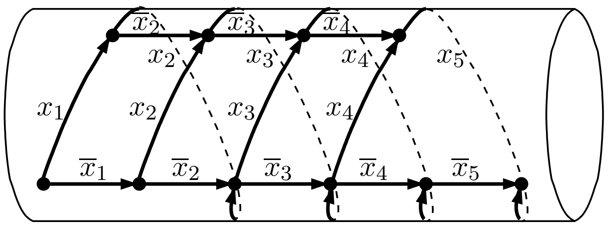

Figure 1: A cylindrical branching program of width 2 computing PARITY.

In this paper we consider the computational power of constant width, polynomial size *cylindrical* branching programs and circuits.

It is well known that there is a rough similarity between the computational power of *width* restricted circuits and *depth* restricted circuits, but that this

[*]Department of Computer Science, University of Aarhus and BRICS, Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation. Email: {`arnsfelt,bromille`}`@daimi.au.dk`
[†]Indian Institute of Science, Bangalore, India. Email: `vinay@csa.iisc.ernet.in`

similarity is not a complete equivalence. For instance, the class of functions computed by a family of circuits of *quasi*-polynomial size and polylogarithmic depth is equal to the class of functions computed by a family of circuits of quasi-polynomial size and polylogarithmic width. On the other hand, the class of functions computed by a family of circuits of *polynomial* size and polylogarithmic width (non-uniform **SC**) is, in general, conjectured to be different from the class of functions computed by a family of circuits of polynomial size and polylogarithmic depth (non-uniform **NC**). For the case of *constant* depth and width, there is a provable difference in computational power; the class of functions computable by constant depth circuits of polynomial size, i.e, $\mathbf{AC}^0$, is a proper subset of the functions computable by constant width circuits (or branching programs) of polynomial size, the latter being, by Barrington's Theorem [1], the bigger class $\mathbf{NC}^1$. On the other hand, Vinay [7] and Barrington et al [2, 3] showed that by putting a *geometric* restriction on the computation, the difference disappears: The class of functions computable by *plane*, constant width, polynomial size circuits (or nondeterministic branching programs) is exactly $\mathbf{AC}^0$. Thus, both $\mathbf{AC}^0$ and $\mathbf{NC}^1$ can be captured by a constant width as well as by a constant depth circuit model. It is then natural to ask if one can similarly capture classes between $\mathbf{AC}^0$ and $\mathbf{NC}^1$ defined by various constant depth circuit models, such as $\mathbf{ACC}^0$ and $\mathbf{TC}^0$, by some natural constant width circuit or branching program model.

In this paper we make some progress towards answering this question by considering a slightly more relaxed geometric restriction than planarity: We consider the functions computed by *cylindrical* polynomial size, constant width circuits (or nondeterministic branching programs). Informally (for formal definitions, see the next section), a layered circuit (branching program) is cylindrical if it can be embedded on the surface of a cylinder in such a way that each layer is embedded on a cross section of the cylinder (disjoint from the cross sections of the other layers), no wires intersect and all wires between two layers are embedded on the part of the cylinder between the two corresponding cross sections (see Figure 1).

It is immediate that constant width polynomial size cylindrical branching programs have more computational power than constant width polynomial size plane branching programs: The latter compute only functions in $\mathbf{AC}^0$ [2] while the former may compute PARITY (see Figure 1). We ask what their exact computational power is and show that their power does not extend much beyond computing functions such as PARITY. Indeed, they can only compute functions in $\mathbf{ACC}^0$. To be precise, the first main result of this paper is the following *lower* bound on the power of cylindrical computation.

Theorem 1 *Every Boolean function computed by a polynomial size* $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ *circuit is also computed by a constant width, polynomial size cylindrical nondeterministic branching program.*

By a $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ circuit we mean a polynomial sized circuit with an AND gate at the output, a layer of OR gates feeding the AND gate, a layer of MOD_m gates (perhaps for many different constant-bounded values of m) feeding the OR gates and a (multi-output) $\mathbf{AC}^0$ circuit feeding the MOD gates. It is not known if the inclusion is proper. We prove Theorem 1 by a direct construction, generalising and extending the simple idea of Figure 1.

Our second main result is the following *upper* bound on the power of cylindrical computation.

Theorem 2 *Every Boolean function computed by a constant width, polynomial size cylindrical circuit is in* $\mathbf{ACC}^0$.

The proof of Theorem 2 is the most technical part of this paper. The simulation is done (as were many previous results about constant width computation) by using the theory of finite monoids and the results of Barrington and Therien [4]. Thus, we show the inclusion by relating the computation of cylindrical circuits to solving the word problem of a certain finite monoid and then show that this monoid is solvable.

A standard simulation shows that every Boolean function computed by a constant width, polynomial size cylindrical nondeterministic branching program is also computed by a constant width, polynomial size cylindrical circuit. For completeness, we describe this simulation in Proposition 3. Thus, one can exchange "cylindrical nondeterministic branching program" with "cylindrical circuit" and vice versa in our two main results.

Organisation of Paper

In section 2, we formally define the notions of cylindrical branching program and circuits. We also give an overview of the algebraic tools we use. In section 3, we show Theorem 1. In section 5, we show Theorem 2. As this proof is quite technical, we warm up by showing, in section 4 by a somewhat easier proof that cylindrical branching programs (rather than circuits) compute only functions in $\mathbf{ACC}^0$. We conclude with some discussions and open problems in section 6.

2 Preliminaries

Bounded depth circuits

Let $A \subset \{0, \ldots, m-1\}$. Using the notation of Grolmusz and Tardos [5], a MOD_m^A gate takes n boolean inputs $x_1, \ldots, x_n$ and outputs 1 if $\sum_{i=1}^n x_i \in A$ (mod m) and 0 otherwise. We let $\mathbf{MOD}$ denote the family of MOD_m^A gates for all constant bounded m and all A. Similarly will $\mathbf{AND}$ and $\mathbf{OR}$ denote the family of unbounded fanin AND and OR gates.

If G is a family of boolean gates and $\mathcal{C}$ is a family of circuits we let $G \circ \mathcal{C}$ denote the class of polynomial size circuit families consisting of a G gate taking circuits from $\mathcal{C}$ as inputs.

$\mathbf{AC}^0$ is the class of functions computed by polynomial size bounded depth circuits consisting of NOT gates and unbounded fanin AND and OR gates. $\mathbf{ACC}^0$ is the class of functions computed when we also allow unbounded fanin MOD gates computing MOD_k for constants k. We will also use $\mathbf{AC}^0$ and $\mathbf{ACC}^0$ to denote the class of circuits computing the languages in the respective classes.

Cylindrical branching programs and circuits

A digraph $D = (V, A)$ is called *layered* if there is a partition $V = V_0 \cup V_1 \cup \cdots \cup V_h$ such that all arcs of A goes from *layer* V_i to the *next layer* V_{i+1} for some i. We call h the *depth* of D, $|V_i|$ the width of layer i and $k = \max |V_i|$ the width of D.

Let $[k]$ denote the integers $\{1, \ldots, k\}$. For $a, b \in [k]$ where $a \not\equiv b + 1 \pmod{k}$ we define the (cyclic) interval $[a, b]$ to be the set $\{a, \ldots, b\}$ if $a \leq b$ and $\{a, \ldots, k\} \cup \{1, \ldots, b\}$ if $a > b$. Furthermore let $(a, b) = [a, b] \setminus \{a, b\}$, and let $(a, b) = [k] \setminus \{a, b\}$ if $a \equiv b + 1 \pmod{k}$.

Let D be a layered digraph in which all layers have width k. We will assume the nodes in each layer numbered $1, \ldots, k$, and refer to nodes by these numbers. Then, D is called a *cylindrical* if the following property is satisfied: For every pair of arcs going from layer l to layer $l + 1$ connecting node a to node c and node b to node d the following must hold: Nodes in the interval (a, b) of layer l can only connect to nodes in the interval $[c, d]$ of layer $l + 1$ and nodes in the interval (b, a) of layer l can only connect to nodes in the interval $[d, c]$ of layer $l + 1$.

Notice this is equivalent of saying that nodes in the interval (c, d) of layer $l + 1$ can only connect to nodes in the interval $[a, b]$ of layer l and nodes in the interval (d, c) of layer $l + 1$ can only connect to nodes in the interval $[b, a]$ of layer l.

A *nondeterministic branching program*[1] is a acyclic digraph where all arcs are labelled by either a literal, i.e. a variable or a negated variable, or a boolean constant, and an initial and a terminal node. An input is accepted if and only if there is a path from the initial node to the terminal node in the graph that results from substituting constants for the literals according to the input and then deleting arcs labelled by 0.

[1]Our definition deviates slightly from the usual definition where nodes rather than edges are labelled by literals and unlabelled nodes serve as special nondeterministic "choice"-nodes, but it is easily seen to be polynomially equivalent - also in the cylindrical case - and it is more convenient for us.

We will only consider branching programs in *layered form*, that is, viewed as a digraph it is layered. We can assume that the initial node is in the first layer and the terminal node in the last layer, and furthermore that these are the only nodes incident to arcs in these layers. We can also assume that all layers have the same number of nodes, by the addition of dummy nodes.

By a *cylindrical branching program* we will then mean a bounded-width nondeterministic branching program in layered form, which is cylindrical when viewed as a digraph.

A *cylindrical circuit* is a circuit consisting of fanin 2 AND and OR gates and fanin 1 COPY gates, which when viewed as a digraph is a cylindrical digraph. Inputs nodes can be literals or boolean constants. The output gate is in the last layer. We can assume that all layers have the same number of nodes by adding dummy input nodes to the first layer and dummy COPY gates to the other layers.

A standard simulation of nondeterministic branching programs by circuits extends to cylindrical branching programs and cylindrical circuits. We give the details for completeness.

Proposition 3 *Every function computed by a width k, depth d cylindrical branching program is also computed by a width $O(k)$, depth $O(d \log k)$ cylindrical circuit*

Proof Replace every node in the branching program by an OR-gate. Replace each arc, going from, say, node u to node v and labelled with the literal x, with a new AND-gate taking two inputs, gate u and the literal x and with the output of the AND-gate feeding gate v.

This transformation clearly preserves the cylindricality of the graph. Also, the width of the circuit is linear in the width of the branching program. The resulting OR-gates may have fan-in bigger than two. We replace each such gate with a tree of fan-in two OR-gates, preserving the width and blowing up the depth by at most a factor of $O(\log k)$. $\square$

Monoids and groups

Let x and y be elements of a group G. The *commutator* of x and y is the element $x^{-1}y^{-1}xy$. The subgroup $G^{(1)}$ of G generated by all of the commutators in G is called the *commutator subgroup* of G. In general, let $G^{(i+1)}$ denote the commutator subgroup of $G^{(i)}$. G is *solvable* if $G^{(n)}$ is the trivial group for some n. It follows that an Abelian group, and in particular a cyclic group, is solvable.

A *monoid* is a set M with an associative binary operation and a two sided identity. A subset G of M is a *group in M* if it is a group with respect to the operation of M. Note that a group G in M is not necessarily a submonoid of M as the identity element of G may not be equal to the identity element of M. M is called *solvable* if every group in M is solvable. The *word problem* for a finite monoid M is the computation of the product $x_1 x_2 \dots x_n$ given $x_1, x_2, \dots, x_n$ as input. A theorem by Barrington and Therien [4] states that the word problem for a solvable finite monoid is in $\mathbf{ACC}^0$.

3 Simulation of bounded depth circuits by cylindrical branching programs

In this section, we prove Theorem 1. As a starting point, we shall use the "only if" part of the following correspondence established by Vinay [7] and Barrington et al [2]. We include here a proof of the "only if" part for completeness.

Theorem 4 *A language is in* $\mathbf{AC}^0$ *if and only if it is accepted by a polynomial size, constant width plane branching program.*

Here a *plane* branching program is a layered branching program satisfying, that for every pair of arcs going from layer l to layer $l+1$ connecting node a to node c and node b to node d, if $a < b$ then $c \leq d$.

We need some simple observations. First observe that if we can simulate a class of circuits $\mathcal{C}$ with plane (cylindrical) branching programs, then we can also simulate $\mathbf{AND} \circ \mathcal{C}$ by plane (cylindrical) branching programs by simply concatenating the appropriate branching programs.

Another way to combine branching programs is by *substitution* where we simply substitute a branching program for the edges corresponding to a particular literal. The effect of this is captured in the following lemma.

Lemma 5 *If* $f(x_1, \dots, x_n)$ *is computed by a plane (cylindrical) branching program of size* s_1 *and width* w_1 *and* $g_1, \dots, g_n$ *and* $\overline{g}_1, \dots, \overline{g}_n$ *are computed by plane branching programs, each of size* s_2 *and width* w_2 *then* $f(g_1, \dots, g_n)$ *is computed by a plane (cylindrical) branching program of size* $O(s_1 w_1 s_2)$ *and width* $O(w_1^2 w_2)$.

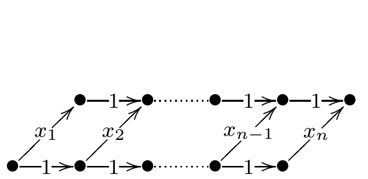

Figure 2: A planar branching program computing OR.

Combining the above observations with the construction in Figure 2, simulating an OR gate, we have established the "only if" part of Theorem 4.

Simulation of a MOD_m^A gate can be done as shown in Figure 3 if one disregards the top nodes in the first and last layers and modifies the connections between the second-to-last layer to take the set A into account. Thus, combining this construction with Lemma 5, the "only if" part of Theorem 4 and the closure of cylindrical branching programs under polynomial fan-in AND, we have established that we can simulate $\mathbf{AND} \circ \mathbf{MOD} \circ \mathbf{AC}^0$ circuits by bounded width polynomial size cylindrical circuits.

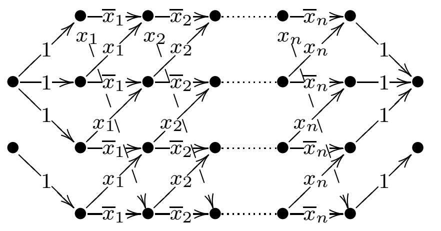

Figure 3: A cylindrical branching program fragment for MOD_4.

The construction as shown in Figure 3 has actually more use, by seeing it as computing elements of M_2, where M_2 is the monoid of binary relations on [2]. The general construction of a branching program fragment for MOD_m^A taking n inputs is as follows: Without loss of generality we can assume that $|A| = 1$ and in fact $A = \{0\}$ since we aim for simulating $\mathbf{OR} \circ \mathbf{MOD}$. The branching program fragment will have $n + 3$ layers. The first and last layer of width 2 and the middle layers of width m. The top node in the first layer has arcs to all nodes but node 1 and the bottom node has an arc to node 1. The top node in the last layer has arcs from all nodes but the one in A and the bottom node has an arc from this node. The nodes in the middle layers represent the sum of a prefix of the input modulo m in the obvious way. Consider now the elements of M_2 shown in Figure 4. The branching program fragment just described corresponds to (a) and (b) for $m = 2$ and $m > 2$ respectively, when the simulated MOD gate evaluates to 0. In both cases, the fragment correspond to (c) when the simulated MOD gate evaluates to 0.

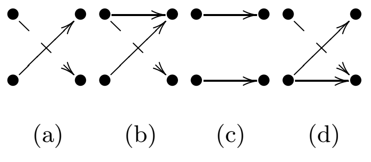

Figure 4: Some elements of M_2.

We can now describe our construction for simulating **OR** $\circ$ **MOD** circuits. The construction interleaves branching program fragments for (d) between the branching program fragments for the MOD gates. This can be seen as a way of "short circuiting" the branching program in the case that one of the MOD gates evaluate to 1. Finally we add layers at both ends picking out the appropriate nodes for the simulation. The entire construction is shown in Figure 5. The correctness can easily be verified.

The simulation of **OR** $\circ$ **MOD** circuits, the "only if" part of Theorem 4, Lemma 5, and the closure of cylindrical branching programs under polynomial fan-in AND, together completes the proof of Theorem 1.

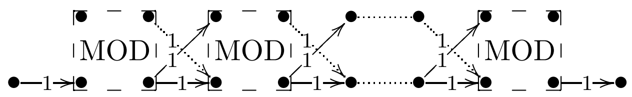

Figure 5: A cylindrical branching program computing MOD $\vee \cdots \vee$ MOD.

4 Simulation of cylindrical branching programs by bounded depth circuits

In this section, we warm up to the proof of Theorem 2 to be presented in the next section, by giving a simpler (but similar) proof of the weaker result that bounded width polynomial size cylindrical nondeterministic branching programs compute only functions in $\mathbf{ACC}^0$.

In fact, we shall prove that for fixed k the following "branching program value problem" BPV_k is in $\mathbf{ACC}^0$: Given a width k cylindrical branching program *and* a truth assignment to its variables, decide if the program accepts. As any function computed by width k cylindrical polynomial size branching program clearly is a Skyum-Valiant projection [6] of BPV_k, we will be done.

We shall prove that BPV_k is in $\mathbf{ACC}^0$ by showing that it reduces, by an $\mathbf{AC}^0$ reduction, to the word problem of the monoid M_k we define next. Then, we show that the monoid M_k is solvable, and since this implies, by the result of Barrington and Therien [4] that the word problem for M_k is in $\mathbf{ACC}^0$, our proof will be complete.

We define M_k to be the monoid of *binary relations* on $[k]$ which capture the calculation of width k branching programs embedded on a cylinder in the following sense: M_k is the monoid generated by all the relations which express how arcs can travel between two adjacent layers in an width k cylindrical digraph. The monoid operation is the usual composition operation of binary relations, i.e., if $A, B \in M_k$ and $x, y \in [k]$, $xABy \Leftrightarrow \exists z : xAz \wedge zBy$.

BPV_k reduces to the word problem for M_k by the following $\mathbf{AC}^0$ reduction:

Substitute constants for the literals in the branching program according to the truth assignment. Consider now the cylindrical digraph D consisting only of arcs which have the constant 1 associated. Then, the branching program accepts the input given if and only if there is a path from the initial node in the first layer to the terminal node in the last layer of D. We can decide this by simply decomposing D into a sequence $A_1, A_2, \ldots, A_h$ of elements from M_k, computing the product $A = A_1 A_2 \cdots A_h$ and checking whether this is different from the zero element of M_k.

Thus, we just need to show that M_k is solvable. Our proof is finished by the following much stronger statement.

Proposition 6 *All groups in M_k are cyclic.*

Proof Let $G \subseteq M_k$ be a group with identity E. Let $A \in G$ and let R be the set of all x such that xEx. As will be shown next it will be enough to consider elements of R to capture the structure of A.

Let $x \in R$. Since $AA^{-1} = E$ there exists z such that xAz and $zA^{-1}x$. Since $A^{-1}A = E$ it follows zEz, that is, $z \in R$. Hence there exists a function $\pi_A : R \to R$ such that

$$\forall x : xA\pi_A(x) \wedge \pi_A(x)A^{-1}x$$

To see that A is completely described by by π_A, we define a relation $\hat{A}$ on $[k]$ such that $x\hat{A}y \Leftrightarrow \pi_A(x) = y$. That is, $\hat{A}$ is just π_A viewed as a relation. Since $\hat{A} \subseteq A$ it follows $E\hat{A}E \subseteq EAE = A$. Conversely let xAy. Since $E^k A = A$ there exists $z \in R$ such that xEz and zAy. Since $\pi_A(z)A^{-1}z$ we get $\pi_A(z)Ey$. That is xEz, $z\hat{A}\pi_A(z)$ and $\pi_A(z)Ey$. Thus $xE\hat{A}Ey$. Hence we obtain that $A = E\hat{A}E$.

We would like to have both that π_A is a permutation and that $\{\pi_A | A \in G\}$ is a group. This is in general not true, since E can be any transitive relation in M_k.

To obtain this we will first simplify the structure of the elements of G using the following equivalence relation on [k] defined by

$$x \sim y \Leftrightarrow (xEy \wedge yEx) \vee x = y.$$

Let $A \in G$. If $x \sim x'$ and $y \sim y'$ then $xAy \Leftrightarrow x'Ay'$, since $EAE = A$. Thus A gives rise to a relation $\tilde{A}$ on $[k]/\!\sim$ where $xAy \Leftrightarrow [k]_x \tilde{A} [k]_y$ and it will follow that $\{\tilde{A} | A \in G\}$ is an isomorphic group of G.

For this we need to show that $\widetilde{AB} = \tilde{A}\tilde{B}$. This follows since $[k]_x \widetilde{AB} [k]_z \Leftrightarrow xABz \Leftrightarrow \exists y : xAy \wedge yBz \Leftrightarrow \exists y : [k]_x \tilde{A} [k]_y \wedge [k]_y \tilde{B} [k]_z \Leftrightarrow [k]_x \tilde{A}\tilde{B} [k]_z$

We can find an isomorphic copy of this group in M_k as follows. Choose for each equivalence class $[k]_x$ a representative $r([k]_x)$ in $[k]_x$. Define a relation C on $[k]$ such that $xCy \Leftrightarrow x = y = r([k]_x)$. Thus $\forall x : r([k]_x) C r([k]_x)$. Let

$\sigma : G \to M_k$ be given by $\sigma(A) = CAC$. Then $\sigma(G)$ is the desired isomorphic copy of G. We can thus assume that the equivalence classes with respect to $\sim$ are of size 1.

We now return to the study of π_A. The following property, that for $x, y \in R$ it holds that $xEy \Leftrightarrow \pi_A(x)E\pi_A(y)$, is satisfied:

If xEy then $\pi_A(x)A^{-1}y$ since $A^{-1}E = A^{-1}$. As $A^{-1}A = E$ it follows that $\pi_A(x)E\pi_A(y)$.

Conversely if $\pi_A(x)E\pi_A(y)$ then $xA\pi_A(y)$ since $xA\pi_A(x)$ and $AE = A$. As $\pi_A(y)A^{-1}y$ and $AA^{-1} = E$ it then follows that xEy.

We can now conclude that π_A is a permutation on R: If $\pi_A(x) = \pi_A(y)$ then $\pi_A(x) \sim \pi_A(y)$ so $x \sim y$, that is, $x = y$. Also π_A is uniquely defined : Assume $\hat{\pi}_A : R \to R$ satisfies

$$\forall x : xA\hat{\pi}_A(x) \wedge \hat{\pi}_A(x)A^{-1}x$$

Let $x \in R$. We then obtain $\pi_A(x) \sim \hat{\pi}_A(x)$ so $\pi_A(x) = \hat{\pi}_A(x)$. Hence $\pi_A = \hat{\pi}_A$.

Now we can conclude that $\{\pi_A | A \in G\}$ is a permutation group which is isomorphic to G. For this we need to show that $\pi_{AB} = \pi_B \circ \pi_A$.

Let $x \in R$. Since $xA\pi_A(x)$ and $\pi_A(x)B\pi_B \circ \pi_A(x)$ it follows $xAB\pi_B \circ \pi_A(x)$.

Since $\pi_B \circ \pi_A(x)B^{-1}\pi_A(x)$ and $\pi_A(x)A^{-1}x$ it follows $\pi_B \circ \pi_A(x)B^{-1}A^{-1}x$, i.e. $\pi_B \circ \pi_A(x)(AB)^{-1}x$

Since π_{AB} is uniquely defined the result follows.

To show that $\{\pi_A | A \in G\}$ is cyclic we need the following fact, which easily follows from the definition of cylindricality

Fact: Let A be a relation which can be directly embedded on a cylinder. Let $p_1 < p_2 < \dots p_m$ and $q_1 < q_2 < \dots < q_m$ and π a permutation on $[m]$ such that $\forall i : p_i A q_{\pi(i)}$. Then π is in the cyclic group of permutations on $[m]$ generated by the cycle $(1\ 2 \dots m)$.

Now let $r_1 < r_2 < \dots < r_m$ be the elements of R. Write $A \in G$ as $A = A_1 A_2 \dots A_h$ where the A_i's can be directly embedded on the cylinder. Since $r_i A \pi_A(r_i)$ we have for all i, elements of $[k]$, $r_i = q_i^0, q_i^1, \dots, q_i^h = \pi_A(r_i)$ such that $q_i^j A_{j+1} q_i^{j+1}$. For fixed j all the q_i^j's are distinct. If not we would have i_1 and i_2 such that $r_{i_1} A \pi_A(r_{i_2})$ and $r_{i_2} A \pi_A(r_{i_1})$. But then since $\pi_A(r_{i_1})A^{-1}r_{i_1}$ and $\pi_A(r_{i_2})A^{-1}r_{i_2}$ we then get $r_{i_1} E r_{i_2}$ and $r_{i_2} E r_{i_1}$. That is $r_{i_1} \sim r_{i_2}$ which implies $r_{i_1} = r_{i_2}$. Now by the fact and induction on h we have a permutation π in the cyclic group generated by the cycle $(1\ 2 \dots m)$ such that $r_{\pi(i)} = \pi_A(r_i)$. Thus π_A is in the cyclic group generated by the cycle $(r_1\ r_2 \dots r_m)$ and we can conclude that G is cyclic.

$\square$

5 Simulation of cylindrical circuits by bounded depth circuits

In this section we prove Theorem 2. Following the outline of last section, we consider for fixed k the following "circuit value problem" CV_k: Given a width k cylindrical circuit *and* a truth assignment to its input variables, decide if the circuit evaluates to 1. We shall reduce CV_k, by an $\mathbf{AC}^0$ reduction, to the word problem of the monoid $\hat{N}_k$ defined next, which will be proved to be solvable. By the result of Barrington and Therien [4] it then follows that CV_k is in $\mathbf{ACC}^0$.

Consider a width k cylindrical circuit C with k variable input nodes, all placed in the first layer. We can view this as computing a function mapping $\{0,1\}^k$ to $\{0,1\}^k$ by reading off the values of the nodes in the last layer.

We let N_k be the monoid of functions mapping $\{0,1\}^k$ to $\{0,1\}^k$ which are computed by such circuits with constant input nodes disallowed.

We let $\hat{N}_k$ be the monoid of functions mapping $\{0,1\}^k$ to $\{0,1\}^k$ where we in addition to the above also allow nodes in the last layer to be constant input nodes.

We will call C a N_k circuit and $\hat{N}_k$ circuit, respectively. That $\hat{N}_k$ is in fact a monoid follows from the following lemma

Lemma 7 *Even if we allow constants in all but the first layer in $\hat{N}_k$ circuits, only functions in $\hat{N}_k$ are computed.*

Proof Consider a cylindrical circuit of depth d with possible constant input nodes in all layers. By induction the depth $d-1$ subcircuit of the first layers d layers can be computed by a cylindrical circuit such that all constant input nodes are at layer d. Consider now a node in layer $d+1$ which has an arc from a constant input node. If it always evaluates to a constant we can simply replace it by a constant input node. Otherwise the node is an OR or an AND node which can be replaced by a COPY node, copying the non-constant input node of the old gate. Now all constant input nodes in layer d can simply be replaced by dummy COPY nodes, since they have no outgoing arcs anymore. $\square$

We are now able to describe the $\mathbf{AC}^0$ reduction of CV_k to the word problem for $\hat{N}_k$: Substitute constants for the variable input nodes according to the truth assignment. Each layer of the circuit except the first can now be viewed as a depth 1 $\hat{N}_k$ circuit by preceeding it by a layer of k variable input nodes. Let $C_1, C_2, \ldots, C_h$ be the circuits obtained this way, and compute the corresponding elements $f_1, f_2, \ldots, f_h$ of $\hat{N}_k$ (represented e.g. by tabulation). Now compute the product $f = f_h \circ \cdots \circ f_2 \circ f_1$ and evaluate it on the constants appearing in the first layer of the circuit. The output of the circuit can then be read off in the entry of this result corresponding to the output node of the circuit.

Now we just need show that $\hat{N}_k$ is solvable. We will in fact, as in the previous section obtain the stronger result that all its groups are cyclic. First we show that it is sufficient to consider N_k.

Proposition 8 *Every group in $\hat{N}_k$ is isomorphic to a group in N_k.*

Proof Suppose $G \subseteq \hat{N}_k$ is a group with identity e.

We construct an injective (monoid) homomorphism $\phi : G \to N_k$. This proves the result. First we need to prove the following claim.

Claim: Let f and g be elements of G computed by $\hat{N}_k$ circuits C_1 and C_2. The output nodes of C_1 and C_2 that are constant are the same and their output values are the same in both circuits.

Proof of Claim: Let C_1' be a $\hat{N}_k$ circuit computing f^{-1}. Note that $C_1 \circ C_1' \circ C_2$ also computes g. Hence, if C_1 has some constant output node of value 1, then C_2 must have the same output node be constant 1 (as seen by feeding the input 0^k through C_2). Analogously, if C_1 has some constant output node of value 0, then C_2 must have the same output node be constant 0 (as seen by feeding the input 1^k through C_2). By repeating the argument with the roles of C_1 and C_2 reversed, we have proved the claim.

By the claim, a fixed set of output nodes S are constant for all $\hat{N}_k$ circuits computing functions in G. Construct the homomorphism ϕ as follows: Let $f \in G$ be computed by an $\hat{N}_k$ circuit C. Then $\phi(f)$ is the function computed by C, modified as follows: Add an extra layer of input nodes at the bottom of the circuit. The old input nodes not in S are replaced by COPY nodes copying the corresponding input node from the new layer. The old input nodes in S are replaced by constant input nodes corresponding to the constant output nodes. Then an extra layer of COPY nodes is added on the top of the circuit. For nodes not in S we just copy the previous value. For a node $a \in S$ we pick a node $b \notin S$ to copy, where $a \equiv b + i \pmod{k}$ and $i > 0$ is minimal. The resulting circuit is then reduced as in Lemma 7.

It is easy to see that ϕ is a homomorphism from G to $\hat{N}_k$. We claim that the constructed circuit is actually a N_k circuit.

To show this, we have to argue that the above construction does not introduce new constant output nodes. This can be seen as follows: Let C and C' be $\hat{N}_k$ circuits computing f and $f^{-1} \in G$. Then if a new constant output node is introduced in the construction for C, the same node is already a constant output node in the circuit $C \circ C'$, computing e, as C' is feeding the same constants to C in $C \circ C'$ as are fed to C in the above construction. This, however, contradicts the claim above.

Thus ϕ is a homomorphism from G to N_k. We now just have to argue that it is injective and we are done. Let C and C_e be $\hat{N}_k$ circuits computing $f \in G$

and e, where $f \neq e$. We then have that for some x, $C(x) \neq C_e(x)$. Since the output nodes in S have the same value there is an output node not in S where the values differ. By noting that the above construction retains the values of the output nodes not in S, we obtain that $\phi(f)(x) \neq \phi(e)(x)$. $\square$

Now we can turn to the study of the monoid N_k. Note that it is generated by the depth 1 N_k circuits. This will be useful in some of the following properties we will prove, since it allows us, by induction, to just consider depth 1 circuits.

As in [3] it will be convenient to identify input vectors in $\{0,1\}^k$ with its set of *maximal 1-intervals*, only here we consider *cyclic* intervals. For example is the vector 1010011011 identified with the set of intervals $\{[3,3],[6,7],[9,1]\}$.

We will only do this identification for inputs which contain at least one interval, that is, we disregard the vectors 0^k and 1^k.

Lemma 9 *Let $x \in \{0,1\}^k$ contain m interval and $f \in N_k$. Then $f(x)$ contains at most m intervals. If $f(x)$ in fact contains m intervals and $m \geq 1$ then $f(x) = \{f(I) | I \in x\}$.*

Proof Clearly $f(0^k) = 0^k$ and $f(1^k) = 1^k$ so we can assume $m \geq 1$. Assume that C is a depth 1 N_k circuit computing f. Let $c, d \in [k]$ be nodes which evaluate to 1 on input x. Each of c and d must have an arc from an interval of x. Suppose c and d have arcs from the same interval I in x, from a to c and b to d, say. Either $[a,b] \subseteq I$ or $[b,a] \subseteq I$. If $[a,b] \subseteq I$ then all nodes in the interval (c,d) evaluate to 1 since they must take arcs from $[a,b] \subseteq I$. Similarly if $[b,a] \subseteq I$ then all nodes in the interval (d,c) evaluate to 1.

Thus nodes which have arcs to the same interval in x are in the same interval of $C(x)$ if any, and the first part follows.

Now assume that there also is m intervals in $C(x)$. To show the second part we have to rule out that an interval in $C(x)$ can take arcs from more than one interval in x. But if that were case there would also different intervals with arcs from the same interval in x, contradicting the above. $\square$

Any subset of $\{0,1\}^k$ becomes a poset by lifting the order $0 < 1$ pointwise. The functions in N_k are monotone with respect to this order. When considering only intervals this order coincides with the subset inclusion order and we will use these orders interchangeably.

Lemma 10 *Let $\{[a_i,b_i] \mid i = 1,\ldots,m\}$ and $\{[c_i,d_i] \mid i = 1,\ldots,m\}$ be antichains of intervals in $\{0,1\}^k$ such that $a_1 < \cdots < a_m$ and $c_1 < \cdots < c_m$. Let $f \in N_k$ and assume π is a permutation on $[m]$ such that $f([a_i,b_i]) = [c_{\pi(i)}, d_{\pi(i)}]$. Then π is in the cyclic group of permutations on $[m]$ generated by the cycle $(1\ 2 \cdots m)$.*

Proof Assume C is a depth 1 N_k circuit computing f. In the following all index arithmetic is done modulo m. There must be an arc from $[a_i, b_i] \setminus [a_{i+1}, b_{i+1}]$ to $[c_{\pi(i)}, d_{\pi(i)}] \setminus [c_{\pi(i)+1}, d_{\pi(i)+1}]$ for all i since otherwise $[c_{\pi(i)}, d_{\pi(i)}] \subseteq [c_{\pi(i)+1}, d_{\pi(i)+1}]$. From this the result follows. $\square$

Say that an interval $[a_0, b_0] \subseteq [a_1, b_1] \cap [a_2, b_2]$ is *in between* intervals $[a_1, b_1]$ and $[a_2, b_2]$ if there is $a_3 \in [a_1, b_1] \setminus [a_2, b_2]$ and $b_3 \in [a_2, b_2] \setminus [a_1, b_1]$ such that $[a_0, b_0] \subseteq (a_3, b_3)$.

If $\{[a_1, b_1], [a_2, b_2]\}$ form an antichain this is equivalent of requiring that $[a_2, b_0] \subseteq [a_1, b_0]$.

Lemma 11 *Suppose* $[c_0, d_0] \subseteq [c_1, d_1] \cap [c_2, d_2]$ *is in between* $[c_1, d_1]$ *and* $[c_2, d_2]$. *Suppose* $f \in N_k$ *maps intervals* $[a_i, b_i]$ *to* $[c_i, d_i]$ *for* $i = 1, 2, 3$. *If* $[a_0, b_0] \subseteq [a_1, b_1] \cap [a_2, b_2]$ *then* $[a_0, b_0]$ *is in between* $[a_1, b_1]$ *and* $[a_2, b_2]$.

Proof Assume f is computed by a depth 1 N_k circuit C.

By assumption there is $c_3 \in [c_1, d_1] \setminus [c_2, d_2]$ and $d_3 \in [c_2, d_2] \setminus [c_1, d_1]$ such that $[c_0, d_0] \subseteq (c_3, d_3)$. There must exist $a_3 \in [a_1, b_1] \setminus [a_2, b_2]$ such that there is an arc from a_3 to c_3 in C, since otherwise we would have $c_3 \in [c_2, d_2]$. Similarly there exists $b_3 \in [a_2, b_2] \setminus [a_1, b_1]$ such that there is an arc from b_3 to d_3 in C. Since $[c_0, d_0] \subseteq (c_3, d_3)$ all arcs to $[c_0, d_0]$ must thus go from $[a_3, b_3]$. It follows $[a_0, b_0] \subseteq [a_3, b_3]$, in fact $[a_0, b_0] \subseteq (a_3, b_3)$ since $[a_0, b_0] \subseteq [a_1, b_1] \cap [a_2, b_2]$. $\square$

In the following let G be a group in N_k with identity e. Since $e \circ e = e$ we get that e is the identity mapping on the image of e, $\operatorname{Im} e$. Thus any $f \in G$ is a permutation of $\operatorname{Im} e$, since $f \circ f^{-1} = f^{-1} \circ f = e$ and $e \circ f = f$.

Lemma 12 *Let* $f \in G$ *and* $\mathcal{S} \subseteq \operatorname{Im} e$. *If* f *is a permutation on* $\mathcal{S}$ *then* f *a permutation on the set of minimal elements in* $\mathcal{S}$, $\min(\mathcal{S})$, *with respect to* $<$.

Proof Assume on the contrary that $x \in \min(\mathcal{S})$ such that $f(x) \notin \min(\mathcal{S})$. Then there exists $y \in \min(\mathcal{S})$ such that $y < x$. But then $f^{-1}(y) < x$, which contradicts $x \in \min(\mathcal{S})$. $\square$

Now we can finally complete the proof of Theorem 2 by the following proposition.

Proposition 13 *All groups in* N_k *are cyclic*

Proof Let G be a group in N_k with identity e and let $\mathcal{I}$ be the set of intervals in $\operatorname{Im} e$.

Let $f \in G$. Since $f \circ e = f$ it follows that f is completely described by its restriction to $\operatorname{Im} e$. By Lemma 9 f is furthermore completely described by its restriction to $\mathcal{I}$.

We decompose $\mathcal{I}$ into antichains, $\mathcal{I}_j = \min\left(\mathcal{I} \setminus \bigcup_{i<j} \mathcal{I}_i\right)$.

By Lemma 12 and induction, f is a permutation on $\mathcal{I}_j$ and furthermore a *cyclic shift* by Lemma 10. Thus G acts as a cyclic group on $\mathcal{I}_j$. It follows that G is Abelian, which is clearly more than enough to prove Theorem 2. To obtain that G is in fact cyclic, we will argue that f is in fact completely described by its restriction to $\mathcal{I}_1$.

For an interval I let $\mathcal{I}_j(I)$ denote the set $\{J \in \mathcal{I}_j \mid I \subseteq J\}$. Pick an interval $I \in \mathcal{I}_1$ such that $\mathcal{I}_j(I) \neq \emptyset$. Observe that $\mathcal{I}_j(f(I)) = \{f(J) \in \mathcal{I}_j \mid I \subseteq J\}$. The intervals of $\mathcal{I}_j(I)$ are linearly ordered by having I in between, and likewise are the intervals of $\mathcal{I}_j(f(I))$ linearly ordered by having $f(I)$ in between. By Lemma 11 (applied to f^{-1}) these orderings are respected by f. In particular is the first interval in $\mathcal{I}_j(I)$ mapped into the first interval in $\mathcal{I}_j(f(I))$ by f, which is enough to describe how f acts on $\mathcal{I}_j$. $\square$

6 Conclusion and open problems

We have located the class of functions computed by small constant width cylindrical circuits (or nondeterministic branching programs) between $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ and $\mathbf{ACC}^0$. It would be very interesting to get an exact characterisation of the power of cylindrical circuits and branching programs in terms of bounded depth circuits. It is not known whether $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ is different from $\mathbf{ACC}^0$ and this seems a difficult problem to resolve, so we cannot hope for an unconditional separation of the power of cylindrical circuits from $\mathbf{ACC}^0$. On the other hand, it seems difficult to generalise the simulation of $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ by cylindrical branching programs to handle more than one layer of MOD gates and we tend to believe that such a simulation is in general not possible. Thus, one could hope that by better understanding the structure of the monoids we have considered in this paper, it would be possible to prove an upper bound seemingly better than $\mathbf{ACC}^0$, such as for instance $\mathbf{AC}^0 \circ \mathbf{MOD} \circ \mathbf{AC}^0$.

It would also be interesting to separate the power of branching programs from the power of circuits. As circuits can be trivially negated while preserving cylindricality, we immediately have that not only $\mathbf{\Pi}_2 \circ \mathbf{MOD} \circ \mathbf{AC}^0$ but also $\mathbf{\Sigma_2} \circ \mathbf{MOD} \circ \mathbf{AC}^0$ can be simulated by small constant width cylindrical circuits. On the other hand, we don't know if $\mathbf{\Sigma_2} \circ \mathbf{MOD} \circ \mathbf{AC}^0$ can be simulated by small constant width cylindrical branching programs. Note that in the plane case, both models capture $\mathbf{AC}^0$ and in the geometrically unrestricted case, both models capture $\mathbf{NC}^1$, so it is not clear if one should *a priori* conjecture the cylindrical models to have different power. Note that if the models have identical power then they can simulate $\mathbf{AC}^0 \circ \mathbf{MOD} \circ \mathbf{AC}^0$. This follows from the fact that the branching program model is closed under polynomial fan-in AND while the circuit model is closed under negation.

An interesting problems concerns the blowup of width to depth when going from a cylindrical circuit or branching program to an $\mathbf{ACC}^0$ circuit. Our proof does not yield anything better than a doubly exponential blowup. Again, by better understanding the structure of the monoids we have considered, one could hope for a better upper bound.

References

[1] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in $\mathbf{NC}^1$. *J. Comput. System Sci.*, 38(1):150–164, 1989.

[2] D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. Searching constant width mazes captures the $\mathbf{AC}^0$ hierarchy. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, pages 73–83, 1998.

[3] D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. On monotone planar circuits. In *14th Annual IEEE Conference on Computational Complexity*, pages 24–31. IEEE Computer Society Press, 1999.

[4] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of $\mathbf{NC}^1$. *Journal of the ACM (JACM)*, 35(4):941–952, 1988.

[5] V. Grolmusz and G. Tardos. Lower bounds for $(\mathrm{mod}p - \mathrm{mod}m)$ circuits. *SIAM Journal on Computing*, 29(4):1209–1222, Aug. 2000.

[6] S. Skyum and L. G. Valiant. A complexity theory based on boolean algebra. *Journal of the ACM (JACM)*, 32(2):484–502, 1985.

[7] V. Vinay. Hierarchies of circuit classes that are closed under complement. In *11th Annual IEEE Conference on Computational Complexity (CCC-96)*, pages 108–117, Los Alamitos, May 24–27 1996. IEEE Computer Society.