SUPERLINEAR LOWER BOUNDS FOR BOUNDED-WIDTH BRANCHING PROGRAMS

David A. Mix Barrington†
COINS Department
University of Massachusetts
Amherst, Massachusetts 01003
USA

Howard Straubing‡
Department of Computer Science
Boston College
Chestnut Hill, Massachusetts 02167
USA

Abstract

We use algebraic techniques to obtain superlinear lower bounds on the size of bounded-width branching programs to solve a number of problems. In particular, we show that any bounded-width branching program computing a nonconstant threshold function has length $\Omega(n \log\log n)$, improving on the previous lower bounds known to apply to all such threshold functions. We also show that any program over a finite solvable monoid computing products in a nonsolvable group has length $\Omega(n \log\log n)$. This result is a step toward proving the conjecture that the circuit complexity class ACC^0 is properly contained in NC^1.

A preliminary version of this paper appeared in the Proceedings of the 1991 Structure in Complexity Theory Symposium.

1. The Main Results

In this paper we describe a general algebraic technique for obtaining superlinear lower bounds on the length of bounded-width branching programs to solve certain problems. Our method is based on the interpretation, due to Barrington and Thérien [5] of these programs as generalizations of ordinary finite automata.

We are thus able to apply the well-established connection between finite automata and finite monoids to branching programs. More precisely, we start from the simple algebraic fact that a homomorphism from a free monoid into a finite monoid N cannot simulate

† Research supported by NSF Grant CCR-8714714.
‡ Research supported by NSF Grant CCR-8902369.

iterated multiplication in another finite monoid M, unless M is a homomorphic image of a submonoid of N. We find, through a sequence of Ramsey-theoretic arguments, that an analogous result holds for branching programs. This is the source of all our lower bounds.

A *branching program* with n inputs is a directed acyclic graph in which there is a single source node, each node is labelled by an element of $\{1,\ldots,n\}$, each non-sink node has exactly two exiting edges, labelled 0 and 1, and each sink node is designated as either an accepting or a rejecting node. A computation of the program on input $a_1 \cdots a_n \in \{0,1\}^*$ begins at the source node. At each step the label i of the present node is examined, and the program branches to one of the two successor nodes, depending on whether $a_i = 0$ or $a_i = 1$. The computation continues in this manner until a sink node is reached. The input is accepted or rejected depending on whether this sink node is an accepting or a rejecting node. Thus the program accepts a subset of $\{0,1\}^n$. Usually we are interested in families of branching programs, consisting of one program for each input length. Such a family then recognizes a subset of $\{0,1\}^*$.

Bounded-width branching programs were introduced in [6] and [8]. A bounded-width branching program of width w consists of k levels, each with w nodes, except for the first level which has a single node. Each directed edge goes from a node on one level to a node on the next level. The *size* of the program is k. By a family of bounded-width branching programs we mean such a family in which the width is a constant independent of the length of the input. At the expense of increasing the size by a constant factor, we may assume that the program is *input oblivious,* that is, that every node on a given level has the same label.

Barrington [3] and Barrington and Thérien [5] gave the following algebraic interpretation of such programs: Each level is an instruction to read the i^{th} bit of the input string (where i is the label of the nodes on this level) and then emit one of two maps $g_0, g_1 : \{1,\ldots,w\} \to \{1,\ldots,w\}$, depending on the value of the bit consulted. The maps emitted at each level are composed, and the input is accepted if and only if the composite maps 1 to an accepting value in $\{1,\ldots,w\}$. The set of maps from $\{1,\ldots,w\}$ into itself forms a finite *monoid* (*i.e.,* a set with an associative multiplication and an identity element) with composition of maps as multiplication. This leads to the formulation of a new model: Let A be a finite alphabet. A *program over a finite monoid* M with input alphabet A consists of a sequence of *instructions*

$$(i_1, f_1), \ldots, (i_r, f_r)$$

—where for each $j, i_j \in \{1,\ldots,n\}$, and f_j is a map from A into M—together with a subset X of M. On input $a_1 \cdots a_n \in A^*$, the program emits the value

$$f_1(a_{i_1}) \cdots f_r(a_{i_r}) \in M,$$

and accepts the input if and only if this value belongs to X. Barrington and Thérien established that every bounded-width branching program of length r and width w is equivalent to a program of length r over a submonoid of T_w, the monoid of all maps on a w-element set with left-to-right composition as the multiplication. Conversely, every program of length r over a finite monoid M is equivalent to a branching program of width $|M|$ and length r.

The value of this new model, and the reason for its introduction, was the discovery that the algebraic structure of M is related to the kinds of circuits that can recognize the same language that the program recognizes. Thus Barrington [3] showed that if M contains a group that is not solvable, then every language in NC^1 is recognized by a polynomial-length family of programs over M. Furthermore, Barrington and Thérien [5] showed that a language is in the circuit complexity class AC^0 if and only if it is recognized by a polynomial-length family of programs over a finite monoid that contains no nontrivial groups, and is in the circuit complexity class ACC^0 if and only if it is recognized by a polynomial-length family of programs over a finite monoid that contains only solvable groups.*

It is conjectured that ACC^0 is strictly contained in NC^1. According to the above remarks this conjecture would be proved by finding superpolynomial lower bounds on the length of programs over monoids containing solvable groups that recognize some language in NC^1. One of the goals of our current research is to find such algebraic proofs of circuit lower bounds.

This new model also gives us a highly structured, algebraic way of looking at branching programs, which we use to much advantage in this paper. For example, our Theorem 1.3 below makes no explicit reference to finite monoids, although we use these ideas throughout its proof.

Here are our main results:

1.1 Theorem. Let N be a finite monoid in which every group has order dividing $q > 0$. Let $p > 0$ be an integer not dividing q. Then any family of programs over N recognizing the language

$$L_p = \{a_1 \cdots a_n \in \{0,1\}^* : \sum_{i=1}^{n} a_i \equiv 0 \pmod{p}\}$$

has length $\Omega(n \log\log n)$.

1.2 Theorem. Let G be a finite group. Any family of programs over G recognizing the language $1^* \subseteq \{0,1\}^*$ (*i.e.,* computing the AND of its input bits) has length $\Omega(n \log\log n)$.

1.3 Theorem.*(a)* Any family of width t branching programs recognizing the language

$$\{a_1 \cdots a_n \in \{0,1\}^* : \sum_{i=1}^{n} a_i > t\}$$

has length $\Omega(n \log\log n)$.

(b) Let $g : \mathbf{N} \to \mathbf{N}$ be a function such that

* ACC^0 consists of those languages recognized by constant-depth polynomial-size families of circuits containing unbounded fan-in AND, OR and $MODq$ gates for some fixed q. This class is called ACC in earlier papers, but we prefer to use this notation to underscore the analogy with the AC^i hierarchy.

$$\lim_{n\to\infty} g(n) = \lim_{n\to\infty} n - g(n) = +\infty.$$

Then for any family of bounded-width branching programs recognizing

$$L_{T(g)} = \{a_1 \cdots a_n \in \{0,1\}^* : \sum_{i=1}^{n} a_i \geq g(n)\}$$

there is a constant $c > 0$ such that for infinitely many values of n, the n^{th} program of the family has length at least $cn \log\log n$.

Let G be a finite group. We can view G as a finite alphabet, so that each string w in G^* also has a value $\pi(w)$ in G, obtained by multiplying together all the letters in the string. The language L_G consists of all strings $w \in G^*$ such that $\pi(w) = 1$.

1.4 Theorem. Let N be a finite monoid in which every group is solvable, and let G be a finite nonsolvable group. Then any family of programs over N recognizing L_G has length $\Omega(n \log\log n)$.

Let us say something about the significance of these results and where they stand in relation to similar work. Barrington and Thérien [5] have shown that a program over a finite monoid M in which every group is solvable can be simulated by an ACC^0 circuit in which every modular gate has a period dividing the order of a group in M, with the size of the circuit bounded by a polynomial in the length of the program. Smolensky [14] showed that ACC^0 circuits in which every modular gate has period q^k, for q prime, require exponential size to recognize L_p unless p is a power of q. Since every group of prime power order is solvable, it follows that Theorem 1.1 holds with an *exponential* lower bound on program size when q is a prime power. On the other hand, if the assumption that q is a prime power is dropped, then there may be a polynomial upper bound, as it follows from results of Barrington [3] that one can add bits modulo 7 with a polynomial-length program over a nonsolvable group of order 60. Our theorem appears to be the only nontrivial lower bound result known to apply in all cases.

Theorem 1.2 is due to Cai and Lipton [7]; we include it here because it is a direct corollary of our main theorem. In fact, the present work owes a lot to the methods that Cai and Lipton used to establish this theorem. It is conjectured (see [4]) that if G is solvable, then exponential-length programs are required to compute the AND function, whereas there is a polynomial upper bound for nonsolvable groups.

Theorem 1.3(a) improves on a

$$\Omega(n \log\log n/(\log\log\log n))$$

lower bound due to Pudlàk [13]. Alon and Maass [1] establish $\Omega(n \log n)$ lower bounds on program length for $L_{T(g)}$, where $g(n)$ is of the form n^δ, and Babai, *et. al.*, [2] find $\Omega(n \log n)$ lower bounds for asymptotically almost all threshold functions. Our result in 1.3(b), while smaller than the bounds established in [1] and [2], is the best one known to apply to all nonconstant threshold functions (that is, threshold $g(n)$, where neither $g(n)$ nor $n - g(n)$ is bounded above by a constant).

Theorem 1.4 is an important step in our program to prove that ACC^0 is strictly contained in NC^1. Of course, what we want is a superpolynomial lower bound, rather than the superlinear bound given here. We can interpret this theorem in such a manner as to give a new circuit lower bound. Let us define an ACC^0 *formula* to be an ACC^0 circuit that is a tree, rather than an arbitrary directed acyclic graph. The size of such a formula is the number of nodes. We will show:

1.5 Theorem. Any family of ACC^0 formulas recognizing L_G, where G is a nonsolvable finite group, has size $\Omega(n \log\log n)$.

We think that the real significance of this work lies in the very general character of the arguments. Indeed, all the theorems stated above are direct corollaries of a single algebraic result which states, in essence, that programs of linear or nearly linear length over finite monoids behave a great deal like homomorphisms into finite monoids. The precise statement of this result (Theorem 2.2) will be given in Section 2, after some algebraic preliminaries. Sections 3 and 4 are devoted to the proof of the theorem: In Section 3 we show how to treat programs that make a constant number of scans, either left-to-right or right-to-left, over their input. In Section 4 we reduce arbitrary linear size (or slightly superlinear size) programs to this case. The proof in Section 3 requires a Ramsey-theoretic argument that goes back to Erdös and Szekeres; Section 4 uses a different Ramsey-theoretic result, due to Cai and Lipton [7]. Theorems 1.1-1.5 will be established in Section 5; the reader who is only interested in how these theorems follow from our general algebraic result can skip Sections 3 and 4.

2. Algebraic Preliminaries and the Fundamental Theorem

We first give a brief rundown of some basic properties of finite monoids. For more detailed information on the matters discussed here, see [9] (especially Chapter III), [11] or [12]. If M is a finite monoid, then for each $m \in M$ the set $\{m^r : r \geq 0\}$ is finite, and thus $m^{t+q} = m^t$ for some $t \geq 0, q > 0$. If we let T be the maximum of these t and Q the least common multiple of the q over all $m \in M$, then $m^{T+Q} = m^T$ for all $m \in M$. An element e of a monoid is said to be *idempotent* if $e^2 = e$. With T and Q as above, choose R such that $T + R$ is divisible by Q. Then m^{T+R} is idempotent; in particular every element of M has an idempotent power.

A *homomorphism* $\phi : M_1 \to M_2$ of finite monoids is a map satisfying $\phi(mm') = \phi(m)\phi(m')$ for all $m, m' \in M_1$. If A is a finite alphabet, then A^* forms a (infinite) monoid under concatenation of strings, and every map $\phi : A \to M$ extends to a unique homomorphism from A^* into M. (That is, A^* is the *free monoid* on A.) We say that a monoid M_1 *divides* a monoid M_2, and write $M_1 \prec M_2$, if there is a submonoid N of M_2 and a homomorphism ϕ from N onto M_1. There is an important characterization of division in terms of homomorphisms from free monoids, which we shall frequently use: Let $\phi : A^* \to N, \psi : A^* \to M$, be homomorphisms, with ψ surjective. If M does not divide N, then there exist $u, v \in A^*$ such that $\psi(u) \neq \psi(v)$ and $\phi(u) = \phi(v)$ (otherwise there would be a homomorphism $\eta : \phi(A^*) \to M$ such that $\psi = \eta \circ \phi$, implying $M \prec N$). If a group G divides a finite monoid M, then G is the homomorphic image of a group in M.

The *direct product* $M_1 \times M_2$ of monoids is the cartesian product together with componentwise multiplication, which makes $M_1 \times M_2$ a monoid. The *reversal* of a monoid M, denoted M^R, is the monoid with the same underlying set as M, and with multiplication $*$ given by

$$s * t = t \cdot s,$$

where the right-hand side denotes the product in M. If N is a finite monoid, then the *reversal closure* of N, denoted $rcl(N)$, is the family of all finite monoids M such that M divides a direct product

$$(N \times N^R)^k = \underbrace{N \times N^R \times \cdots \times N \times N^R}_{k \text{ times}}$$

for some $k > 0$.

2.1 Lemma. Let A be a finite alphabet, M and N finite monoids, and $\psi : A^* \to M$ a surjective homomorphism. Suppose $M \notin rcl(N)$. Then there exist $m_1 \neq m_2 \in M$ such that for every homomorphism $\phi : A^* \to K$, where $K \in rcl(N)$, there exist $u, v \in A^*$ with $\psi(u) = m_1, \psi(v) = m_2$, and $\phi(u) = \phi(v)$.

Proof. Observe that there are only finitely many homomorphisms

$$\phi_1, \ldots, \phi_k : A^* \to N \times N^R.$$

Let

$$\Phi : A^* \to (N \times N^R)^k$$

be the homomorphism whose i^{th} component is ϕ_i. It is easy to show that any homomorphism $\phi : A^* \to K \in rcl(N)$ factors through Φ, and the result follows immediately from the previous remarks concerning division. ∎

When the conclusions of Lemma 2.1 hold, we say that $m_1, m_2 \in M$ are *N-separated* by ψ.

We are now ready to state the main theorem.

2.2 Theorem. Let M and N be finite monoids, where $M \notin rcl(N)$. Let A be a finite alphabet, and $\psi : A^* \to M$ a surjective homomorphism such that $\psi(a) = 1$ for some $a \in A$. Consider a family of programs over N such that the length of the n^{th} program is $n \cdot k(n)$, where

$$\lim_{n \to \infty} n^{1/[(3|N|)^{2|A| \cdot k(n)}]} = +\infty.$$

Let $m_1, m_2 \in M$ be N-separated by ψ. Then for all sufficiently large n there exist $w, w' \in A^n$ such that $\psi(w) = m_1$, $\psi(w') = m_2$, and such that the n^{th} program of the family gives the same value on w and w'.

The proof of this theorem will be given in Sections 3 and 4.

3. Sweeping Programs

Let $k > 0$. A *k-sweeping* program over a finite monoid N is a program that factors into k segments, where in each segment all n letters of the input are read, either from left to right or from right to left. More precisely, let the sequence of instructions of the program be

$$(i_j, f_j), j = 1, \ldots, r,$$

as in Section 1. Then $r = kn$, and for all $0 \leq j < k$, either $i_{jn+p} = p$, for all $p = 1, \ldots, n$, or $i_{jn+p} = n - p + 1$, for all $p = 1, \ldots, n$.

In this section we shall establish a version of Theorem 2.2 for sweeping programs. Before proceeding to the proof, let us show how the argument works by treating a special case. How do we prove that a linear-size family of programs over a finite monoid M that contains no nontrivial groups cannot recognize L_2? If the program family is a homomorphism $\phi : A^* \to M$ we proceed as follows: Since M contains no nontrivial groups there is an integer $t > 0$ such that for all $m \in M$, $m^t = m^{t+1}$. Thus $\phi(1^{t+1}0^t) = \phi(1^t0^{t+1})$, so the homomorphism cannot distinguish between strings in L_2 and strings outside L_2. Now let us consider a family of programs that make a single sweep of the input from left to right. The foregoing argument does not generalize directly to this case, because the value emitted by the program when reading a particular bit depends on the position of the bit as well as the value of the bit. The way we accomplish the generalization is by considering very long (length s) input strings. For $1 \leq i < j \leq s$ we color the segment (i, j) by the pair $(m_1, m_2) \in M \times M$, where m_1 is the value emitted by the program upon reading 10^{j-i-2} in positions $i, \ldots, j-1$, and m_2 is the value emitted upon reading 0^{j-i-1} in the same positions. If N is very large then by Ramsey's theorem there is a sequence $i_0 < \cdots < i_{2t+1}$ such that the segments $(i_0, i_1), \ldots, (i_{2t}, i_{2t+1})$ all have the same color. We can now apply the program to the strings

$$xu_1 \cdots u_{t+1}v_{t+2} \cdots v_{2t+1}y$$

and

$$xu_1 \cdots u_t v_{t+1} \cdots v_{2t+1}y,$$

where

$$u_r = 10^{i_r - i_{r-1} - 2}, v_r = 0^{i_r - i_{r-1} - 1},$$

and x and y are arbitrary strings of length $i_0 - 1$ and $s - i_{2t+1} + 1$, respectively. The two strings have opposite parity, but the value emitted by the program on the first string is $m_1^{t+1}m_2^t = m_1^t m_2^{t+1}$, which is the value emitted on the second string, thus the program family cannot recognize parity.

This is the essence of our proof. Lemma 3.1 is the precise Ramsey-theoretic result we use. (Ramsey's theorem itself gives much smaller lower bounds.) Lemma 3.2 gives the above argument about programs in a more general setting, so that it applies to programs making several sweeps rather than a single sweep.

3.1 Lemma. Let $r, n, k > 0$, and consider a k-coloring of the set $\{(i,j) : 1 \leq i < j \leq n\}$. If $n > r^k$ then there is a sequence

$$1 \leq i_0 < i_1 < \cdots < i_r \leq n$$

such that all (i_j, i_{j+1}) have the same color.

Proof. Suppose that no such sequence exists. To each $j \in \{1, \ldots, n\}$ we associate $\tau_j \in \{0, \ldots, r-1\}^k$, where the p^{th} component of τ_j is the length q of the longest sequence

$$i_0 < i_1 < \cdots < i_q$$

such that $i_q = j$ and all (i_t, i_{t+1}) are colored p. If $n > r^k$ then there exist $j_1 < j_2$ such that $\tau_{j_1} = \tau_{j_2}$. Consider now the color p assigned to (j_1, j_2). If the p^{th} component of τ_{j_1} is q then the p^{th} component of τ_{j_2} is at least $q+1$, a contradiction.∎

This lemma generalizes the well-known result of Erdös and Szekeres [10] that if $n > r^2$ then every sequence $a_1 \cdots a_n$ of numbers contains a monotone subsequence of length r. In this case an edge (i,j) is colored UP or DOWN depending on whether $a_i < a_j$ or $a_i \geq a_j$.

3.2 Lemma. Suppose we are given a family of $k(n)$-sweeping programs over a finite monoid N, with input alphabet A, and for each $i > 0$ a map $\alpha_i : A \to A^i$. Let $r > 0$. If

$$n > r^{|N|^{|A| \cdot k(n)}}$$

then there exist a sequence

$$1 \leq i_0 < i_1 < \cdots i_r \leq n$$

and a homomorphism

$$\Phi : A^* \to N_1 \times \cdots \times N_{k(n)},$$

where $N_c = N$, or $N_c = N^R$, depending on whether the c^{th} sweep of the program is left to right or right to left, with the following property: Let $u \in A^{i_0 - 1}$, $v \in A^{n - i_r}$. Then there exist $s_0, \ldots, s_{k(n)} \in N$ such that for any $w = a_1 \cdots a_r \in A^r$, the value of the n^{th} program on

$$u \cdot \alpha_{i_1 - i_0}(a_1) \cdots \alpha_{i_r - i_{r-1}}(a_r) \cdot v$$

is

$$s_0 \cdot \phi_1(w) \cdot s_1 \cdots s_{k(n)-1} \cdot \phi_{k(n)}(w) \cdot s_{k(n)}.$$

(Here ϕ_i denotes the i^{th} component of the homomorphism Φ. The product of the $2 \cdot k(n) + 1$ factors is computed in N.)

Proof. Let $A = \{a_1, \ldots, a_s\}$, and let the program instructions be

$$(i_j, f_j), 1 \leq j \leq n \cdot k(n), 1 \leq i_j \leq n.$$

We color $\{(i,j) : 1 \leq i < j \leq n\}$ by elements of $(N^{|A|})^{k(n)}$. Let $1 \leq p \leq s$, $1 \leq c \leq k(n)$. The p^{th} component of the c^{th} component of the color assigned to (i,j) is

$$f_{(c-1)n+i}(b_1) \cdots f_{(c-1)n+j-1}(b_{j-i})$$

or

$$f_{cn-(j-i)}(b_{j-i}) \cdots f_{cn-1}(b_1),$$

where $\alpha_{j-i}(a_p) = b_1 \cdots b_{j-i}$. The first alternative holds if the c^{th} sweep of the program is left to right, and the second if the c^{th} sweep is right to left. By Lemma 3.1, there exists a sequence

$$1 \leq i_0 < i_1 < \cdots < i_r \leq n$$

such that all (i_t, i_{t+1}) are assigned the same color γ. Let $\phi_i(a_p) = m_{i,p}$, where $m_{i,p}$ is the p^{th} component of the i^{th} component of γ. Then $\Phi = (\phi_1, \ldots, \phi_{k(n)})$ extends to a unique homomorphism

$$\Phi : A^* \to N_1 \times \cdots \times N_{k(n)}.$$

The result now follows directly. Observe that s_0 is the product of the first $|u|$ values emitted by the program, if the first sweep is left to right, and the product of the first $|v|$ values if the first sweep is right to left. Similarly $s_{k(n)}$ is the product of the last $|u|$ or $|v|$ values, depending on the direction of the last sweep. For $1 < c < k(n)$, s_c is the product of the values emitted by the $(cn - k_1 + 1)^{th}$ through the $(cn + k_2)^{th}$ instructions of the program, where k_1 and k_2 are either $|u|$ or $|v|$, depending upon the directions of the c^{th} and $(c+1)^{th}$ sweeps.∎

The foregoing lemma allows us, in a sense, to view programs as homomorphisms. In order to apply it we shall require the following strengthened version of Lemma 2.1—this tells us that under certain conditions we can assume that the strings u and v have the same length:

3.3 Lemma. Let M and N be finite monoids, and let A be a finite alphabet. Suppose $M \notin rcl(N)$, and $\phi : A^* \to N$, $\psi : A^* \to M$ are homomorphisms, where ψ is surjective and $\psi(a) = 1$ for some $a \in A$. Let $m_1, m_2 \in M$ be N-separated by ψ. Then for all sufficiently large r there exist $u, v \in A^r$ such that $\psi(u) = m_1, \psi(v) = m_2$ and $\phi(u) = \phi(v)$.

Proof. There exists $p > 0$ such that $\phi(a^p) = e$ is idempotent. Let $S = \{xa^p : x \in A^p\}$. Then S generates a free submonoid of A^*. It follows from Lemma 2.1 that there exist $u', v' \in S^*$ such that $\psi(u') = m_1, \psi(v') = m_2$, and $\phi(u') = \phi(v')$. If $|u'| = |v'|$ we are done. Otherwise, we can suppose without loss of generality that $|u'| < |v'|$. The lengths of u' and v' differ by a multiple of p, and thus there exists $k > 0$ such that $|u'a^{kp}| = |v'|$. But $\psi(u'a^{kp}) = \psi(u') = m_1$, and $\phi(u'a^{kp}) = \phi(u')e^k = \phi(u')$, because $\phi(S^*) \subseteq Ne$. The desired conclusion follows with $u = u'a^{kp}, v = v'$. ∎

Observe that the length r of the string required in Lemma 3.3 can be bounded by a function of $|M|, |N|$ and $|A|$.

We can now prove a version of our fundamental theorem for sweeping programs.

3.4 Proposition. Let M, N be finite monoids, where $M \notin rcl(N)$. Let A be a finite alphabet, and $\psi : A^* \to M$ a surjective homomorphism, where $\psi(a) = 1$ for some $a \in A$. Consider a family of $k(n)$-sweeping programs over N, where

$$\lim_{n\to\infty} n^{\frac{1}{|N|^{|A|k(n)}}} = +\infty.$$

Let $m_1, m_2 \in M$ be N-separated by ψ. Then for all sufficiently large n, there exist $w, w' \in A^n$ such that $\psi(w) = m_1, \psi(w') = m_2$, and such that the n^{th} program of the family gives the same value on w and w'.

Proof. For all $i > 0$ define $\alpha_i : A \to A^i$ by $\alpha_i(b) = ba^{i-1}$, for $b \in A$. Observe $\psi(\alpha_i(b)) = \psi(b)$. Choose r large enough for the conclusions of Lemma 3.3 to hold. If n is large enough, then

$$n > r^{|N|^{|A|\cdot k(n)}}.$$

By Lemma 3.2 there exist

$$1 \le i_0 < i_1 < \cdots < i_r \le n,$$

a homomorphism

$$\Phi : A^* \to N_1 \times \cdots \times N_{k(n)}$$

(where $N_j = N$ or $N_j = N^R$ according to the direction of the j^{th} sweep) and $s_0, \ldots, s_{k(n)} \in M$ such that the value of the n^{th} program on

$$a^{i_0-1}\alpha_{i_1-i_0}(b_1)\cdots\alpha_{i_r-i_{r-1}}(b_r)a^{n-i_r}$$

is

$$s_0\phi(b_1\cdots b_r)s_1\cdots s_{k(n)-1}\phi_{k(n)}(b_1\cdots b_r)s_{k(n)}.$$

By Lemma 3.3, there exist $u = b_1\cdots b_r$, $v = b'_1\cdots b'_r$ such that $\Phi(u) = \Phi(v)$, $\psi(u) = m_1$, and $\psi(v) = m_2$. Let

$$w = a^{i_0-1}\alpha_{i_1-i_0}(b_1)\cdots\alpha_{i_r-i_{r-1}}(b_r)a^{n-i_r},$$

$$w' = a^{i_0-1}\alpha_{i_1-i_0}(b'_1)\cdots\alpha_{i_r-i_{r-1}}(b'_r)a^{n-i_r}.$$

Then $\phi_i(u) = \phi_i(v)$ for all i, so the n^{th} program of the family has the same value on both w and w'. On the other hand, $\psi(w) = \psi(u) = m_1$, and $\psi(w') = \psi(v) = m_2$. ∎

4. Reduction of Linear-Size Programs to Sweeping Programs

In this section we complete the proof of Theorem 2.2 by showing how to reduce the general case to that of sweeping programs. The reduction is accomplished by means of a combinatorial lemma, due to Cai and Lipton, which we now state as Lemma 4.1. The proof, which uses Ramsey-theoretic techniques similar to those of [1], is in [7]. Let σ be a sequence of elements from $\{1,\ldots,n\}$ in which each element of $\{1,\ldots,n\}$ appears exactly k times. If $I \subseteq \{1,\ldots,n\}$ let σ_I denote the subsequence of σ consisting of all occurrences of the elements of I.

4.1 Lemma. Let σ, n, k be as above. Then there exists $I \subseteq \{1,\ldots,n\}$ such that $|I| > n^{\frac{1}{3^k}}$, and such that σ_I factors into j segments

$$\tau_1 \tau_2 \cdots \tau_j,$$

where $j \leq k$ and each τ_i is monotone increasing or monotone decreasing. ∎

Now consider a program with n inputs over a finite monoid N. Let $I \subseteq \{1,\ldots,n\}$. A *restriction* of the program to I is a program over N with $|I|$ inputs obtained as follows: We fix the input letters in the positions $\{1,\ldots,n\}\backslash I$. This fixes the values in N emitted by the instructions of the program that query these positions. Let $I = \{i_1 < \cdots < i_q\}$. If the letters in these positions are set to $a_1,\ldots,a_q$ then the value emitted by the program is

$$m_0 \cdot g_1(a_{j_1}) \cdot m_1 \cdots g_p(a_{j_p}) \cdot m_p,$$

for some $m_0,\ldots,m_p \in N$, and maps $g_1,\ldots,g_p : A \to N$. The g_j give the values emitted by the by the instructions that query the letters positions in I; and the m_j are the products of the values emitted by the instructions between successive queries of the letters in positions in I—these depend on the fixed input letters in $\{1,\ldots,n\}\backslash I$, but not on the input letters a_i. Our new program has p instructions. On input $a_1 \cdots a_q$ the first instruction emits $m_0 \cdot g_1(a_{j_1}) \cdot m_1$, and the i^{th} instruction, for $i > 1$, emits $g_i(a_{j_i}) \cdot m_i$. In particular, the assignment of values to the letters in positions $\{1,\ldots,n\}\backslash I$ defines a map $\eta : A^q \to A^n$ such that the value of the restricted program on $u \in A^q$ is equal to the value of the original program on $\eta(u)$.

Proof of Theorem 2.2. Since the n^{th} program has length $n \cdot k(n)$, there exists a set $J \subseteq \{1,\ldots,n\}$ such that $|J| = \frac{n}{2}$ and such that each position in J is consulted no more than $2 \cdot k(n)$ times. We define a restriction of the original program to J by setting all the letters in positions $\{1,\ldots,n\}\backslash J$ to a, where $\psi(a) = 1$. It thus suffices to establish the theorem for the restricted program. We can add instructions that always emit the identity element to insure that each input letter is consulted exactly $2 \cdot k(n)$ times. By Lemma 2.1 there exists a subset I of J such that

$$|I| \geq \left(\frac{n}{2}\right)^{\frac{1}{3^{2 \cdot k(n)}}}$$

and any restriction to I is a j-sweeping program, where $j \leq 2 \cdot k(n)$. In particular, we can consider the restriction obtained by setting the letters in positions $J\backslash I$ to a, where $\psi(a) = 1$. It thus suffices to establish the conclusion for this j-sweeping program on $(\frac{n}{2})^{\frac{1}{3^{2 \cdot k(n)}}}$ inputs. But

$$\left(\left(\frac{n}{2}\right)^{\frac{1}{3^{2\cdot k(n)}}}\right)^{\frac{1}{|N|^{|A|\cdot j}}} \geq \left(\frac{n}{2}\right)^{\frac{1}{(3|N|^{|A|})^{2\cdot k(n)}}},$$

which, according to our hypothesis, grows without bound, so the desired result follows from Proposition 2.4. ∎

5. Proofs of Theorems 1.1-1.4

We first make a brief remark about the asymptotics implicit in Theorem 2.2: If we choose

$$k(n) = c \cdot \log\log n,$$

where

$$c = \frac{1}{4|A| \cdot \log(3|N|)},$$

then

$$\log(n^{1/(3|N|)^{2|A|\cdot k(n)}}) = (\log n)^{\frac{1}{2}},$$

so the hypotheses of the theorem are satisfied. This is the source of the $\Omega(n \log\log n)$ lower bounds in these four theorems.

Proof of Theorem 1.1. Let N be a finite monoid in which every group has order dividing q. It is easy to show that every group in the direct product of two monoids M_1 and M_2 is contained in the direct product of a group in M_1 with a group in M_2. It follows that every cyclic group in $rcl(N)$ has order dividing q. In particular, let M_p be the cyclic group $\{1 = g^p, g, \ldots, g^{p-1}\}$; then $M_p \notin rcl(N)$. Define a homomorphism $\psi : \{0,1\}^* \to M_p$ by $\psi(0) = 1$, $\psi(1) = g$. Then ψ satisfies the hypotheses of Theorem 2.2. Observe that if $m_1, m_2 \in M_p$ are N-separated by ψ, then so are $m_1 h$ and $m_2 h$ for any $h \in M_p$. (Use strings uw and vw, where $\phi(w) = h$.) In particular, we may suppose $m_1 = 1$. We now apply Theorem 2.2: Suppose we have a family of programs over N of length $c \cdot n \log\log n$, where c is chosen as described at the beginning of this section. Then if n is sufficiently large, there exist $w, w' \in A^n$ such that $\psi(w) = m_1 = 1, \psi(w') = m_2$, and such that the n^{th} program gives the same value on w and w'. But $w \in L_p$ and $w' \notin L_p$, so the family of programs does not recognize L_p. ∎

Proof of Theorem 1.2. Let M be the monoid $\{0,1\}$, with multiplication

$$0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0, 1 \cdot 1 = 1.$$

It is straightforward to show $M \notin rcl(G)$ for any finite group G. (Indeed, the reversal of a group G is isomorphic to G, and any divisor of a direct product of finite groups must itself be a finite group.) We define a homomorphism $\psi : \{0,1\}^* \to M$ by $\psi(0) = 0, \psi(1) = 1$. Then ψ satisfies the hypotheses of Theorem 2.2, and for $w \in \{0,1\}^*, \psi(w) = 1$ if and only if $w \in 1^*$. By Theorem 2.2, if there is a family of programs over G of size $c \cdot n \log\log n$, with c chosen as above, then there exist $n > 0$, $w \in 1^n$, $w' \in \{0,1\}^n \backslash \{1^n\}$, such that the

n^{th} program gives the same value on w and w'. Thus the family of programs does not recognize 1^*. ∎

To prove Theorem 1.3, we need the following lemma.

5.1 Lemma. Let N be a finite monoid such that for some $t \geq 0, q > 0$, $m^{t+q} = m^t$ for all $m \in N$. Then any family of programs over N accepting the set

$$L(t+1,n) = \{a_1 \cdots a_n \in \{0,1\}^n : \sum_{i=1}^{n} a_i > t\}$$

for all sufficiently large n has length $\Omega(n \log\log n)$.

Proof. Consider the finite monoid

$$M = \{1, s, s^2, \ldots, s^t, s^{t+1} = s^{t+2}\}.$$

Define a homomorphism $\psi : \{0,1\}^* \to M$ by $\psi(0) = 1$, $\psi(1) = s$. It is easy to verify that every monoid in $rcl(N)$ satisfies the identity $x^{t+q} = x^t$, and thus $M \notin rcl(N)$. Let $m_1 = s^k, m_2 = s^l$, with $k < l$, be N-separated by ψ. Then $\psi(u) = m_1, \psi(v) = m_2$ for some $u, v \in \{0,1\}^*$. Thus $\psi(u \cdot 1^{t+1-l}) = s^{t+1+k-l}$, and $\psi(v \cdot 1^{t+1-l}) = s^{t+1}$ are N-separated by ψ, so in applying Theorem 2.2 we may assume $m_2 = s^{t+1}$. The result now follows from Theorem 2.2 by noting that $\psi(w') = s^{t+1}$ if and only if $w' \in L(t+1, |w'|)$.

Proof of Theorem 1.3. *(a)* Consider the monoid T_t of all functions from $\{1, \ldots, t\}$ into itself, with composition as the operation. (Strictly speaking, it is necessary to specify whether the composition of two functions is written with the first function to be applied on the right or the left. The two monoids that result from these choices are nonisomorphic, but reversals of one another.) Let m be any element of this monoid, and let $Im(m)$ denote the image set of the function m. It follows that $Im(m^{r+1}) \subseteq Im(m^r)$ for all r, and thus $Im(m^t) = Im(m^{t+1})$. Thus $m^{t+q} = m^t$ for some $q \geq 1$. The result now follows at once from Lemma 5.1 and Theorem 2.2.

(b) First observe that if there is a family of programs over N that recognizes $L_{T(g)}$, then there is a family of programs of the same length that recognizes $L_{T(h)}$, where $h(n) = n - g(n)$. We may thus suppose that there are infinitely many values of n such that $g(n^2) \leq n^2 - n$, otherwise, we can simply reverse the roles of $g(n)$ and $h(n)$ to obtain this. Now let t, q be such that $m^{t+q} = m^t$ for all $m \in N$. Then for infinitely many values of r we have

$$g(r^2) \geq t + 1,$$

$$r^2 - g(r^2) \geq r.$$

For each such value of r we shall show how to construct a restriction of the original program on r^2 inputs to r input variables, so that the new program accepts $L(t+1, r)$. First suppose that the original program has length $r^2 \cdot k(r^2)$. Since $r < \frac{r^2}{2}$, (unless $r = 1$), we may select a set I of r input positions, each of which is consulted at most $2 \cdot k(r^2)$ times. Let us set

$g(r^2)-(t+1)$ of the positions in $\{1,\ldots,r^2\}\backslash I$ to 1, and the remaining $r^2-r-(g(r^2)-(t+1))$ to 0. The resulting restricted program has length no more than $2\cdot rk(r^2)$ and accepts its input $a_1\cdots a_r$ if and only if $\sum_{i=1}^{r} a_i \geq t+1$. By Lemma 5.1, there exists a constant c such that

$$k(r^2) \geq c \log\log r \geq \frac{c}{2}\cdot \log\log r^2$$

for sufficiently large r. Thus

$$k(n) \geq \frac{c}{2}\cdot \log\log n$$

for infinitely many values of n.
∎

Proof of Theorem 1.4. Suppose G is not solvable and that every group in N is solvable. Then $G \notin rcl(N)$. Consider now the homomorphism $\psi : G^* \rightarrow G$ given by $\psi(g) = g$ for all $g \in G$. Let $m_1, m_2 \in G$ be N-separated by ψ. We may assume, as in the proof of Theorem 1.1, that $m_1 = 1$. It now follows directly from Theorem 2.2 that any program over N recognizing L_G has length $\Omega(n \log\log n)$.∎

Proof of Theorem 1.5. The constructions of [5] and [15] associate to an ACC^0 formula a program over a solvable group whose length is equal to the size of the formula. The result now follows directly from Theorem 1.4. ∎

6. Questions for Further Study

Our principal open problem is to replace the superlinear lower bounds in Theorem 1.2 (in the case wehre G is a solvable group) and Theorem 1.4 by superpolynomial lower bounds. As mentioned before, this will show that ACC^0 is strictly contained in NC^1. The combinatorial methods used here do not appear to generalize directly to give superpolynomial bounds. However we believe that these results may be an ingredient in the proof of the stronger bounds. Indeed, a polynomial-length program over a finite monoid N can be factored into a translation of the original length n input string to a string of length n^k, followed by a program over N that makes a single scan of its input. We now know a great deal about the behavior of single scan programs, and wish to study what happens during the polynomial translation of the original input.

A second question worth investigating is how tight are our bounds on branching program length for threshold functions. The evidence of [1] and [2] suggests that a $\Omega(n \log n)$ lower bound ought to hold for the length of branching programs computing threshold functions satisfying the hypotheses of Theorem 1.3(b), however it may be possible to find shorter programs recognizing $L_{T(g)}$ if $g(n)$ increases very slowly.

7. References

[1] N. Alon and W. Maass, Meanders and their applications in lower bounds arguments, *J. Comp. Sys. Sci.* **37**, 118-129 (1988).

[2] L. Babai, P.Pudlàk, V. Rödl and E. Szemeredi, Lower bounds to the complexity of symmetric boolean functions, *Theretical Computer Science* **74,** 313-324 (1990).

[3] D. Mix Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *J. Comp. Sys. Sci.* **38**, 150-164 (1989).

[4] D. Mix Barrington, H. Straubing and D. Thérien, Non-uniform automata over groups, *Information and Computation* **89**, 109-132 (1990).

[5] D. Mix Barrington and D. Thérien, Finite monoids and the fine structure of NC^1, *J. Assoc. Comp. Mach.* **35**, 941-952 (1988).

[6] A. Borodin, D. Dolev, F. Fich and W. Paul, Bounds for width two branching programs, *SIAM J. Computing* **15**, 549-560 (1986).

[7] J. Cai and R. Lipton, Subquadratic simulations of circuits by branching programs, to appear in *SIAM J. Computing.*

[8] A. Chandra, M. Furst and R. Lipton, Multiparty protocols, *Proc. 15th ACM STOC,* 94-99 (1983).

[9] S. Eilenberg, *Automata, Languages and Machines,* vol. B., Academic Press, New York, 1976.

[10] P. Erdös and Szekeres,A combinatorial problem in geometry, *Composito Math.* **2**, 464-470 (1935).

[11] G. Lallement, *Semigroups and Combinatorial Applications,* John Wiley, 1979.

[12] J. E. Pin, *Varieties of Formal Languages,* Plenum, London, 1986.

[13] Pudlàk, P., A lower bound on the complexity of branching programs, *Proc. 11th MFCS Symposium,* Lecture Notes in Computer Science **176**, 480-485 (1984).

[14] Smolensky, R., Algebraic methods in the theory of lower bounds for Boolean circuit complexity, *Proc. 19th ACM STOC,* 77-82 (1987).

[15] H. Straubing, D. Thérien and W. Thomas, Regular languages defined with generalized quantifiers, *Proc. 15th ICALP,* Lecture Notes in Computer Science **317**, 561-575 (1988).