Electronic Colloquium on Computational Complexity, Report No. 67 (2001)

Polynomial Programs and the Razborov-Smolensky Method

Hubie Chen[*]

Abstract

Representations of boolean functions as polynomials (over rings) have been used to establish lower bounds in complexity theory. Such representations were used to great effect by Smolensky, who established that MOD $q \notin AC^0[\text{MOD}\, p]$ (for distinct primes p, q) by representing $AC^0[\text{MOD}\, p]$ functions as low-degree multilinear polynomials over fields of characteristic p. Another tool which has yielded insight into small-depth circuit complexity classes is the program-over-monoids model of computation, which has provided characterizations of circuit complexity classes such as AC^0 and NC^1.

We introduce a new model of computation, the polynomial program, which naturally unifies both the polynomial (over ring) model of computation and the program-over-monoids model of computation. Our motivation is to extend Smolensky's result to prove $AC^0[\text{MOD}\, m]$ lower bounds, where m is a composite with at least two distinct prime factors.

In this paper, we first study the basic properties of this new model of computation and its relationship to previous work. Then, we show that Smolensky's proof of the "hardness" of certain functions for the polynomial model of computation has an analog in the polynomial program model. We also prove a dichotomy theorem on finite groups, essentially showing that a finite group G either is so "complicated" that every boolean function has a degree one polynomial program over G, or is too "simple" in that the function MOD p can be computed by a low-degree polynomial program over G for at most one prime p. The property of nilpotence gives the dividing line. This dichotomy theorem rigorously demonstrates limitations of the Razborov-Smolensky method for polynomial programs over finite groups.

[*]Department of Computer Science, Cornell University, Ithaca, NY 14853. E-mail: hubes@cs.cornell.edu.

ISSN 1433-8092

1 Introduction

One research program of central importance in complexity theory aims to develop techniques for proving resource lower bounds on explicit functions. A sequence of papers from the 1980s [7, 1, 16, 8] contributed to this program by establishing that the parity function cannot be computed by AC^0 circuits (circuits of constant depth, polynomial size, and unbounded fan-in). Building on the work of Razborov [11], Smolensky gave a generalization of this result in 1987 [14], demonstrating that for distinct primes p and q, the language MOD q is not computable by $AC^0[\text{MOD}\, p]$ circuits (AC^0 circuits with oracle gates for the MOD p function). In contrast to the combinatorial methods which were used to obtain the parity function lower bounds, Smolensky's proof technique, which we call the *Razborov-Smolensky Method*, was algebraic in flavor. This proof technique involved showing that $AC^0[\text{MOD}\, p]$ circuits can be closely approximated by low degree multilinear polynomials over fields of characteristic p, and correspondingly that any such close approximation of MOD q necessarily is of high degree.

Although there have been many new results concerning circuits with depth restrictions since Smolensky's result,[1] the Razborov-Smolensky method remains a state-of-the-art separation technique in that there are no analogous results characterizing the power of $AC^0[\text{MOD}\, m]$ circuits when m is a composite number with at least two distinct prime factors. Indeed, it is an open question as to whether or not MOD k can be computed by $AC^0[\text{MOD}\, m]$ circuits when k is a prime not dividing m.

In fact, the class $AC^0[\text{MOD}\, m]$ is not known to be distinct from NP. The $AC^0[\text{MOD}\, m]$ versus NP question, then, is representative of the wide gap between our ability to formally demonstrate lower bounds and our intuition concerning the power of various complexity classes.

In this paper, we introduce a new model of computation, which we refer to as the polynomial program. A polynomial program is defined over a semigroup (a set with an associative binary operation), and encompasses the polynomial model of computation used in the Razborov-Smolensky method and the program-over-monoids model studied for example in [4]. In particular, when R is a ring, a multilinear polynomial from the ring of polynomials $R[X_1, \ldots, X_n]$ is a polynomial program over the additive group of R. Also, a program over the monoid M is a polynomial program of degree one over M.

We study this model of computation in hopes of extending the Razborov-Smolensky method to address the power of $AC^0[\text{MOD}\, m]$ circuits. In addition, we seek a better understanding of this method's limitations, and an explanation for why results on the complexity classes $AC^0[\text{MOD}\, m]$ are not forthcoming. Moreover, this new model of computation is a natural unification of two models of computation in which useful and interesting results have been derived. The use of polynomials over rings to represent boolean functions, which dates back to the work of Minsky and Papert on perceptrons [10], has been a successful technique in proving complexity lower bounds for a variety of models of computation;[2] and, polynomial length programs over various classes of monoids have provided characterizations of circuit complexity classes such as AC^0 and NC^1. The notion of polynomial program offers a framework in which such results might be extended.

The contents of this paper are as follows. We define the *polynomial program* model of computation, highlight some of its basic properties, and describe how previous work relates to this new model (Sections 3 and 4). One key ingredient of the Razborov-Smolensky method is a proof that certain explicit functions are "complete" for specific rings in that they require high degree to represent as polynomials over such rings; in Section 5 we generalize this proof to the polynomial program framework. In Section 6 we prove a dichotomy theorem for finite groups, essentially showing that a finite group G either is so "complicated" that every boolean function has a degree one polynomial program over G, or is too "simple" in that the function MOD p can be computed by a low degree polynomial program over G for at most one prime p. The property of nilpotence will give the dividing line. This dichotomy theorem rigorously demonstrates limitations of the

[1]For example, the characterizations of circuit complexity classes given in [2] and [4].

[2]For more information on polynomial representations of boolean functions, we refer the reader to the surveys [5] and [12].

Razborov-Smolensky method for polynomial programs over finite groups.

2 Preliminaries

2.1 Languages and Classes

The languages MOD m and MAJORITY are defined as follows:

MOD $m = \{x \in \{0,1\}^* : \sum_{i=1}^{|x|} x_i \equiv 0 \text{ (mod } m)\}$

MAJORITY $= \{x \in \{0,1\}^* : \sum_{i=1}^{|x|} x_i \geq |x|/2\}$

Following a common abuse of notation, we will interchange languages and their characteristic functions; whether we are referring to one or the other will be clear from the context.

An AC^0 (NC^0) circuit family is a sequence of circuits of constant depth, unbounded (respectively bounded) fan-in, and polynomial size over the standard basis $\{\neg, \wedge, \vee\}$. The class AC^0 (NC^0) consists of the languages decidable by AC^0 (NC^0) circuit families. In this paper, we do not discuss issues of uniformity, and our complexity classes are non-uniform.

If $S = \{g_1, \ldots, g_k\}$ is a set of boolean functions, then $AC^0[S]$ or $AC^0[g_1, \ldots, g_k]$ is used to denote the class of languages decidable by AC^0 circuit families with oracle gates for the functions $g_1, \ldots, g_k$, where such oracle gates may have unbounded fan-in. Define TC^0 to be AC^0[MAJORITY].

2.2 Known Relationships

Here we present some basic relationships in the region of interest, each of which has been observed or is implicit in previous work. We provide brief justifications.

Proposition 1 *The following relationships among complexity classes hold:*

1. *If $a, b \geq 1$ and a divides b, AC^0[MOD a] $\subseteq AC^0$[MOD b].*
2. *If $a, b \geq 1$ are relatively prime, then AC^0[MOD a, MOD b] $= AC^0$[MOD ab].*
3. *For all $k \geq 1$, and primes p, AC^0[MOD p^k] $= AC^0$[MOD p].*
4. *For all $m \geq 1$, AC^0[MOD m] $= AC^0[\cup_{p|m}\{$MOD $p\}]$, where the union is over primes p dividing m.*

Proof. (1) follows from the fact that, if a divides b, to test whether or not a string $x_1 \ldots x_n$ is in MOD a, we can test whether or not $x_1^{b/a} \ldots x_n^{b/a}$ is in MOD b.

The $\subseteq$ direction of (2) follows from the previous justification; for the other direction, to test whether or not a string x is in MOD ab, we can test whether or not x is in each of MOD a and MOD b.

(3) follows from the fact that for every natural number a, $a \equiv 0$ (mod p^k) if and only if for all $i = 0, \ldots, k-1$, $\binom{a}{p^i} \equiv 0$ (mod p). To test whether or not a string $x_1 \ldots x_n$ is in MOD p^k, for each $i = 0, \ldots, k-1$ we test whether or not $y_1 \ldots y_{\binom{n}{p^i}}$ is in MOD p, where each y_j is the AND of a different subset of $\{x_1, \ldots, x_n\}$ of size p^i. By the fact, the string $x_1 \ldots x_n$ is in MOD p^k if and only if each of the k tests is true (i.e., the AND of the tests is true). Notice that we can assume that every MOD m gate in a circuit has constant fan-in from any other gate (since we can reduce the number of wires coming into the MOD m gate from another gate modulo m), and hence we can assume that every MOD gate in an AC^0 circuit has polynomial fan-in.

The $\supseteq$ direction of (4) follows from (1) and (2); the $\subseteq$ direction follows from (2) and (3). $\square$

Proposition 2 *The following results concerning languages and complexity classes hold:*

1. *[14] If p and q are distinct primes, then MOD $q \notin AC^0[\text{MOD } p]$.*
2. *If p is prime and m is divisible by a prime other than p, then MOD $m \notin AC^0[\text{MOD } p]$.*
3. *All symmetric functions (and hence the functions MOD m for all $m \geq 1$) are contained in TC^0.*

Proof. (1) is the principal result of [14]. (2) follows from (1) and Proposition 1(1). Verifying that the "exact" functions $\{x \in \{0,1\}^* : \sum_{i=1}^{|x|} x_i = k\}$ for $k = 0, \ldots, n$ are in TC^0 gives (3). □

Our next proposition implies that there are infinitely many incomparable AC^0-degrees in between AC^0 and TC^0, as for any two distinct primes p, q, we have MOD $p \in AC^0[\text{MOD } p] \setminus AC^0[\text{MOD } q]$.

Proposition 3 *For every prime p, $AC^0 \subsetneq AC^0[\text{MOD } p] \subsetneq TC^0$.*

Proof. MOD $p \in AC^0[\text{MOD } p] \setminus AC^0$ as MOD $p \notin AC^0$ by Proposition 2 (1); and for any prime q distinct from p, MOD $q \in TC^0 \setminus AC^0[\text{MOD } p]$ by Proposition 2(1) and Proposition 2(3). □

2.3 Notation and Conventions

We use e_G to denote the identity element of a group G. All rings are assumed to have multiplicative identities, and when R is a ring we denote by 0_R the additive identity of R, and 1_R the multiplicative identity of R. We use R^+ to denote the additive group of R. If $p \in R[X_1, \ldots, X_n]$ is a multivariate polynomial over a ring R with indeterminates from the set $\{X_1, \ldots, X_n\}$, by the value of p at an assignment $a \in \{0,1\}^n$ we mean the value of p when, for all $i \in \{1, \ldots, n\}$, $X_i = 0_R$ or 1_R depending on whether $a_i = 0$ or 1. We generally associate 0_R and 1_R with the boolean values 0 and 1, and say that a polynomial $g \in R[X_1, \ldots, X_n]$ *strongly represents* a boolean function $f : \{0,1\}^n \to \{0,1\}$ if at all points $a \in \{0,1\}^n$, $f(a) = 0$ implies $g(a) = 0_R$, and $f(a) = 1$ implies $g(a) = 1_R$. We use $\mathbb{Z}_m$ to denote the ring of integers modulo m.

3 Polynomial Programs

A *polynomial program* P over the group G (on n inputs) is a sequence of instructions $\langle M_1, g_1 \rangle, \ldots, \langle M_l, g_l \rangle$, where M_i is a subset of the set of input variables $\{X_1, \ldots, X_n\}$ and $g_i \in G$ (for all $i = 1, \ldots, l$.) We call the M_i's the *monomials* of P. Define the *size* of P to be the number of instructions in P; and the *degree* of P, denoted by $\deg(P)$, to be the maximum cardinality of a monomial of P.

For a $0-1$ assignment $a \in \{0,1\}^n$ to the input variables $\{X_1, \ldots, X_n\}$, define the value of the ith instruction $\langle M_i, g_i \rangle$ at a as

$$\text{val}_i(P, a) = \begin{cases} g_i & \text{if } a_j = 1 \text{ for all } X_j \in M_i \\ e_G & \text{otherwise} \end{cases}$$

We define the value of P, denoted by val(P, a), to be the product (in G) of the values of the instructions of P, $\text{val}_1(P, a) \cdots \text{val}_l(P, a)$. When A is a subset of $\{0,1\}^n$, we say that a polynomial program P over G (on n inputs) *g-accepts* or *g-computes* A if $g \in G$ is not the identity element of G; for all $a \in A$, $\text{val}(P, a) = g$; and for all $a \notin A$, $\text{val}(P, a) = e_G$.

We say that $\{P_n\}_{n \geq 1}$ is a *polynomial program family* over the group G if for each $n \geq 1$, P_n is a polynomial program over the group G on n inputs. We say that a polynomial program family $\{P_n\}_{n \geq 1}$ *g-accepts* or *g-computes* a language $L \subseteq \{0,1\}^*$ if P_n g-accepts $L \cap \{0,1\}^n$ for all $n \geq 1$.

We say that a polynomial program (family) over G is an *accepting polynomial program (family)* if, for some element $g \in G$, the polynomial program (family) g-accepts a set. In other words, a polynomial program (family) is accepting if it takes on only the values $\{g, e_G\}$, for some non-identity $g \in G$.

3.1 Polynomial Programs over Semigroups

More generally, one can define polynomial programs over a semigroup S by defining a program to be a sequence of instructions where each instruction is a triple $\langle M, a, b\rangle$ such that M s a subset of the set of input variables and their negations, $\{X_1, \ldots, X_n\} \cup \{\overline{X_1}, \ldots, \overline{X_n}\}$; $a, b \in S$; and the value of an instruction is a if $Y = 1$ for all $Y \in M$, and b otherwise. In fact, one can define a polynomial program over a set with a non-associative binary operation (for instance, the program-over-groupoid model of [6] can be naturally generalized to a notion of "polynomial program over groupoid"). When S is a group, however, it can be verified the two definitions are equivalent (up to a size difference). Our focus in this paper is on polynomial programs over groups, and we use the first definition in this paper where instructions are pairs $\langle M, g\rangle$ for notational convenience. Moreover, "polynomial program" will be assumed to refer to a polynomial program over a group, unless otherwise specified.

3.2 New polynomial programs from old

Suppose $P = \langle M_1, g_1\rangle, \ldots, \langle M_l, g_l\rangle$ and $P' = \langle M'_1, g'_1\rangle, \ldots, \langle M'_{l'}, g'_{l'}\rangle$ are polynomial programs over the same group G.

Denote by $P \circ P'$ the polynomial program $\langle M_1, g_1\rangle, \ldots, \langle M_l, g_l\rangle, \langle M'_1, g'_1\rangle, \ldots, \langle M'_{l'}, g'_{l'}\rangle$; we call this the concatenation of P and P'. Notice that val$(P \circ P', a) =$ val(P, a)val(P', a), for all assignments a.

Denote by X_jP the program $\langle M_1 \cup \{X_j\}, g_1\rangle, \ldots, \langle M_l \cup \{X_j\}, g_l\rangle$. Notice that val$(X_jP, a) = e_G$ on assignments a with $a_j = 0$ and val$(X_jP, a) =$ val(P, a) on assignments a with $a_j = 1$.

Denote by P^{-1} the program $\langle M_l, g_l^{-1}\rangle, \ldots, \langle M_1, g_1^{-1}\rangle$, which has value val$(P^{-1}, a) =$ val$(P, a)^{-1}$.

If $h : G \to G'$ is a group homomorphism, denote by $h(P)$ the program $\langle M_1, h(g_1)\rangle, \ldots, \langle M_l, h(g_l)\rangle$. Notice that val$(h(P), a) = h(val(P, a))$, for all assignments a.

Suppose that $P = \langle M_1, g_1\rangle, \ldots, \langle M_l, g_l\rangle$ and $P' = \langle M'_1, g'_1\rangle, \ldots, \langle M'_l, g'_l\rangle$ are accepting polynomial programs on the same number of inputs and of the same length (but possibly over different groups). We say that the programs P and P' are *equivalent* if they accept the same set and, for all $i \in \{1, \ldots, l\}$, $M_i = M'_i$. In particular, equivalent polynomial programs are of the same degree. The following fundamental fact, which we will use a number of times throughout this paper, allows us to take a polynomial program over a group, and obtain an equivalent polynomial program over a second group, under certain circumstances.

Lemma 4 *(Homomorphism property) If P is a g-accepting polynomial program over the group G and $h : G \to G'$ is a group homomorphism such that g is not in the kernel of h, then there is a polynomial program over G' equivalent to P.*

Proof. The polynomial program $h(P)$ takes on the value $h(g)$ or $h(e_G) = e_{G'}$ depending on whether P takes on the value g or e_G (respectively). Since g is not in the kernel of h, $h(g)$ is not the identity of G', and the polynomial program $h(P)$ $h(g)$-accepts the set which the polynomial program P g-accepts. □

4 Previous work

We now describe how our new model of computation, the polynomial program, relates to previously studied models of computation, proving some basic facts along the way; and we describe the results from previous work that we will use here.

4.1 Polynomials over rings

As mentioned, there is a large body of work in which boolean functions are represented using polynomials over rings. In such work, it is typical that a polynomial f in $R[X_1, \ldots, X_n]$ (where R is a ring) is considered a representation of a boolean function $g : \{0,1\}^n \to \{0,1\}$ if there are disjoint "rejecting" and "accepting" subsets $E, C \subseteq R$ such that at all assignments $a \in \{0,1\}^n$, the value of f at a is an element of E or C depending on whether the value of g at a is 0 or 1, respectively. The Razborov-Smolensky method focuses on the so-called *strong representation*, where E and C are $\{0_R\}$ and $\{1_R\}$, respectively.

Our first observation is that for any polynomial $f \in R[X_1, \ldots, X_n]$, there is a polynomial program with value matching f at all points in $\{0,1\}^n$, and with degree not greater than that of f.

Proposition 5 *If R is a ring with identity, then for every polynomial $f \in R[X_1, \ldots, X_n]$, there is a polynomial program P over R^+ with value equal to f at all assignments in $\{0,1\}^n$, and with degree less than or equal to the degree of f.*

Proof. If an indeterminate X_i appears more than once in a monomial, we can reduce its power to one without changing its value at the points in $A = \{0,1\}^n$. Thus, we can assume that f is of the form $\sum_{a\in A} c_a \prod_{a_i=1} X_i$ (where $c_a \in R$ for all $a \in A$). The polynomial program $P_a = \langle \{X_i : a_i = 1\}, c_a \rangle$ has value equal to the monomial $c_a \prod_{a_i=1} X_i$, and the concatenation of all of the polynomial programs $\{P_a : a \in A\}$ computes $\sum_{a\in A} c_a \prod_{a_i=1} X_i$. $\Box$

It is easy to confirm that the measure of degree applied to a polynomial f by the Razborov-Smolensky method matches the degree of the polynomial program which results by applying Proposition 5 to f.

In analogy to Smolensky's demonstration [14] that every function $g : \{0,1\}^n \to R$ can be represented by a polynomial $p \in R[X_1, \ldots, X_n]$ (in the sense that the values of p and g are equal at all points in $\{0,1\}^n$), it is natural to wonder if every function $f : \{0,1\}^n \to G$ is the value of some polynomial program over G. This is indeed the case.

Proposition 6 *Assume G is a finite group. For every function $f : \{0,1\}^n \to G$, there is a polynomial program over G with value equal to f.*

Proof. It suffices to show that for any $a \in \{0,1\}^n$, there is a polynomial program over G with value equal to $f(a)$ at a, and value e_G everywhere else. (The desired polynomial program is simply the concatenation of such programs over all assignments a.)

Let m be the order of $f(a)$ in G and define δ_a to be the polynomial $\prod_{a_i=1} X_i \prod_{a_i=0}(1 - X_i) \in \mathbb{Z}_m[X_1, \ldots, X_n]$. Notice that $\delta_a(a) = 1$ and $\delta_a(x) = 0$ for all $x \neq a$. By Proposition 5, there is a polynomial program γ_a over $\mathbb{Z}_m$ 1-computing δ_a. Let $h_a : \mathbb{Z}_m \to G$ be the injective homomorphism defined by $h_a(1) = f(a)$; then, the polynomial program $h_a(\gamma_a)$ has value $f(a)$ at a and value e_G everywhere else. $\Box$

Let us say that a polynomial g over a ring R e-approximates a boolean function f on n inputs if g strongly represents f at all but e points. That is, associating 0_R and 1_R with 0 and 1, f and g are different at e or fewer points. The following two statements are among the key results of the Razborov-Smolensky method, and together with the fact that there are finite fields of characteristic p with qth roots of unity, imply that MOD $q \notin AC^0$[MOD p].

Theorem 7 *[14] Let p and q be distinct primes.*

For every boolean function $f : \{0,1\}^n \to \{0,1\}$ computable in AC^0[MOD p], there is a family of degree $o(\sqrt{n})$ polynomials over $\mathbb{Z}_p$ (and hence over any field of characteristic p) $o(2^n)$-approximating f.

If f is a family of degree $o(\sqrt{n})$ polynomials over a finite field with qth roots of unity, then f differs from the MOD q function in at least $c2^n - o(2^n)$ points (for some constant $c > 0$ depending only on q).

As a consequence of Theorem 7, fields of characteristic p are seen to be algebraic settings which differentiate between the prime p and all other primes q: MOD $p \in AC^0$[MOD p] has a low degree (indeed, constant degree) representation, whereas MOD q does not. In the search for a proof that MOD $k \notin AC^0[\text{MOD}\, pq]$ for some triple of distinct primes p, q, k, then, it is natural to seek algebraic settings where two primes p and q both have low degree representations, but another prime k does not; the results in Section 6 on polynomial programs over finite groups are along these lines.

The polynomial $1 - (\sum_{i=1}^{n} X_i)^{p-1} \in \mathbb{Z}_p[X_1, \ldots, X_n]$ strongly represents MOD p, and indeed is the polynomial which results by applying Theorem 7 to the function MOD p. More generally, given an element g of prime power order p^k in a group, there is a constant degree polynomial program family over G which g-computes MOD p^j, for any $j \geq 1$.

Proposition 8 *If $g \in G$ is of order p^k for a prime p and $k \geq 1$, then for any $j \geq 1$, there is a polynomial program family over G g-computing MOD p^j of constant degree.*

Proof. The polynomial $[1 - (\sum_{i=1}^{n} X_i)^{\phi(p^k)}]^{\phi(p^k)} \in \mathbb{Z}_{p^k}[X_1, \ldots, X_n]$ has value equal to (the characteristic function of) MOD p: if $\sum_{i=1}^{n} X_i$ is divisible by p, then so is $(\sum_{i=1}^{n} X_i)^{\phi(p^k)}$, and thus $[1 - (\sum_{i=1}^{n} X_i)^{\phi(p^k)}]$ is coprime to p^k, so the polynomial has value $1 \in \mathbb{Z}_{p^k}$. If $\sum_{i=1}^{n} X_i$ is not divisible by p, then $[1 - (\sum_{i=1}^{n} X_i)^{\phi(p^k)}]$ is equal to $0 \in \mathbb{Z}_{p^k}$.

Using this as a primitive, we can strongly represent MOD p^j with a constant degree polynomial in $\mathbb{Z}_{p^k}[X_1, \ldots, X_n]$ using the construction given in the proof of Proposition 1(3) by representing AND, OR, and NOT gates as in the proof of Lemma 22.

By Proposition 5 there is a polynomial program P over $\mathbb{Z}_{p^k}$ of constant degree which 1-accepts MOD p^j. Let $h : \mathbb{Z}_{p^k} \to G$ be the injective homomorphism defined by $h(1) = g$; then, by Lemma 4, $h(P)$ g-accepts MOD p^j. $\square$

4.2 The "programs over monoids" model

The programs over monoids model of computation is a generalization of the branching program, which was introduced in [9]. We refer the reader to [3] for a description of this model of computation. The polynomial program model encompasses this model in that a program over a monoid M is a polynomial program of degree one over M, and vice-versa[3]. A number of nice characterizations of circuit complexity classes in terms of programs over monoids have been given, among them the following.

Theorem 9 *[4] A language is in AC^0 iff it is recognizable by a polynomial length program family over some aperiodic monoid.*

Theorem 10 *[4] A language is in $ACC^0 \overset{\text{def}}{=} \cup_{m \geq 2} AC^0[\text{MOD}\, m]$ iff it is recognizable by a polynomial length program family over some solvable monoid.*

Theorem 11 *[2] A language is in NC^1 iff it is recognizable by a polynomial length program family over some finite monoid.*

Theorem 12 *[2] For any non-solvable group G, a language is in NC^1 iff it is recognizable by a polynomial length program family over G.*

Let us say that an element g of a group G is G-universal if for any boolean function f, there is a degree one polynomial program over G (equivalently, a program over G) which g-accepts f. Let us say that a group G is universal if there exists an element $g \in G$ such that g is G-universal. We have the following classification theorem.

Theorem 13 *[15] A finite group is universal iff it is not nilpotent.*

[3]This is true so long as the acceptance criterion for polynomial programs is defined appropriately. A program over a monoid M is typically said to accept a string if the product of the values of the instructions is in some "accepting subset" of M.

4.3 Discussion

Just as space and time are resources in the classical Turing machine model of computation which, when restricted, yield interesting and useful complexity classes; degree, size, and semigroup are resources in the polynomial program model of computation which, when restricted in different ways, yield complexity classes, which in some cases coincide with more traditional complexity classes.

The characterizations of depth-bounded circuit complexity classes given by Theorems 9, 10, 11, and 12 demonstrate that polynomial programs of degree one and polynomial size yield traditional classes for different classes of monoids. The polynomial size restriction is crucial when the monoid is sufficiently "complex", since without a size restriction, polynomial programs of degree one over non-nilpotent groups can compute any function (Theorem 13).

On the other hand, Smolensky's insight that functions in $AC^0[\text{MOD } p]$ can be closely approximated by degree $o(\sqrt{n})$ polynomials over $\mathbb{Z}_p$ (Theorem 7) gives a characterization of a traditional complexity class in terms of polynomial programs of restricted degree (degree $o(\sqrt{n})$) over an extremely simple group ($\mathbb{Z}_p$), with no size restriction. Here, fixing one of the resources (namely, the group) but allowing degree to be unbounded allows one to compute any function (Proposition 6).

Thus, it is appropriate to say that the line of work represented by the four characterization theorems in Section 4.2 focuses on obtaining different classes of boolean functions by looking at extremely low degree (namely, degree one), polynomial size polynomial programs over monoids of varying complexity, whereas the Razborov-Smolensky method obtains different classes of boolean functions by looking at polynomial programs over extremely simple groups (namely, cyclic groups of prime order) of varying degree, with no regard for size. In one theory, the critical resource is the "complexity" of the underlying semigroup; in the other, the critical resource is degree.

5 Complete Functions

Smolensky's argument that every low degree (degree $o(\sqrt{n})$) polynomial in a finite field with qth roots of unity must differ from the MOD q function at many points (Theorem 7) is along the following lines. It is first demonstrated that MOD-FAMILY-q, a family of functions essentially equivalent in complexity to MOD q, is "complete" in such a field. This means that if f is a polynomial (over the field) computing the functions in MOD-FAMILY-q up to an error set $I \subseteq \{0,1\}^n$, then *every* boolean function can be computed up to the same error set I in degree $\leq \deg f + (n/2)$. However, if f is of low degree and purports to compute MOD-FAMILY-q up to an error set I, a counting argument shows that there are only so many polynomials of degree $\leq \deg f + (n/2)$, but many more boolean functions. Therefore, there is a polynomial representing many boolean functions, and I must be large.

Here, we generalize the notion of completeness defined by Smolensky, and prove that q-completability, defined below, is a sufficient condition for MOD-FAMILY-q to be complete for a group, meaning that the same degree upper bound ($\deg(\text{MOD-FAMILY-}q) + (n/2)$) holds for all functions in the polynomial program model of computation. As we will see below, this proof generalizes Smolensky's proof which states (in our terminology) that MOD-FAMILY-q is complete for polynomial programs over the additive group of a field with qth roots of unity. The notion of completeness is crucial to the Razborov-Smolensky method, and we hope that this result will lead to a full extension of the method to the polynomial program model.

Note that there is no finiteness restriction on the groups in this section.

For a subset $I \subseteq \{0,1\}^n$ and group G, let us say that a polynomial program P *I-approximates* a function $f : \{0,1\}^n \to G$ if for all $a \notin I$, the value of P at a is equal to $f(a)$, that is, the value of the polynomial program P matches f at the points not in the error set I. Let us say that a polynomial program P over G on n inputs *I-approximates* a boolean function $f : \{0,1\}^n \to \{0,1\}$ via $g \in G$ if g is a non-identity element of G such that for all $a \notin I$: $f(a) = 0 \Leftrightarrow P$ has value g on a, and $f(a) = 1 \Leftrightarrow P$ has value e on a. In other words, the polynomial program P g-computes f at the points not in the error set I.

For a function $f : \{0,1\}^n \to G$, define $\deg_G^I(f)$ to be the minimum degree of a polynomial program I-approximating f. When $f : \{0,1\}^n \to \{0,1\}$ is a boolean function on n inputs, define $\deg_{g,G}^I(f)$ to be the minimum degree of a polynomial program I-approximating f via g. For a set S of boolean functions on n inputs, define $\deg_G^I(S) = \max_{g \in G} \max_{f \in S} \deg_{g,G}^I(f)$.

A set S of boolean functions is *complete for* G if for all n, for all $I \subseteq \{0,1\}^n$, for all $f : \{0,1\}^n \to G$, $\deg_G^I(f) \leq \deg_G^I(S) + (n/2)$.

Define MOD-FAMILY-q to be the set of boolean functions $\{\{x \in \{0,1\}^* : \sum_{i=1}^{|x|} x_i \equiv a \text{ (mod } q)\} : 0 \leq a < q\}$. A group G is *q-completable* if q is a prime and there exists a set of generators $\{g_i\}_{i\in\omega}$ for G and automorphisms $\{\sigma_i\}_{i\in\omega}$, $\{\tau_i\}_{i\in\omega}$ of G such that for all i, j, $\sigma_i(g_i) \neq g_i$, $\sigma_i^q(g_i) = g_i$, and $\tau_i(\sigma_i^{j+1}(g_i)\sigma_i^{-j}(g_i)) = \sigma^j(g_i)$. As the terminology might suggest, q-completability of G is a sufficient condition for MOD-FAMILY-q to be complete for G, and this is the main result for this section.

Theorem 14 *If G is q-completable, then MOD-FAMILY-q is complete for G.*

We prove two lemmas in order to establish this theorem. For each of these lemmas and in the proof of the theorem, we assume that G is q-completable via a set of generators $\{g_i\}_{i\in\omega}$ and automorphisms $\{\sigma_i\}_{i\in\omega}$, $\{\tau_i\}_{i\in\omega}$. For disjoint $M_x, M_y \subseteq \{X_1, \ldots, X_n\}$, define $Z(M_x, M_y, g_i) : \{0,1\} \to G$ to be e_G if there exists $X_i \in M_x$ such that $X_i = 0$, and $\sigma^{\sum_{X_i \in M_y} X_i}(g_i)$ otherwise.

Lemma 15 *For every $I \subseteq \{0,1\}^n$, $M_y \subseteq \{X_1, \ldots, X_n\}$, and $i \in \omega$, there is a polynomial program I-approximating $Z(\emptyset, M_y, g_i)$ with degree $\leq \deg_G^I$(MOD-FAMILY-q) $+ (n/2)$.*

Proof. Fix i. We first show by induction that for all M_y, there is a polynomial program over the variable set M_y with value $Z(\emptyset, M_y, g_i)$ of degree $\leq |M_y|$. This establishes the lemma for $|M_y| \leq n/2$.

This is obviously true for $M_y = \emptyset$: the polynomial program with the single instruction $\langle \emptyset, g_i \rangle$ computes $Z(\emptyset, \emptyset, g_i)$ and is of degree 0. For $|M_y| > 0$, fix $X_j \in M_y$ and let P be a polynomial program of degree $\leq |M_y| - 1$ computing $Z(\emptyset, M_y \setminus \{X_j\}, g_i)$. Then, the polynomial program $\sigma(X_j P) \circ X_j P^{-1} \circ P$ has degree $\leq |M_y|$ and has value equal to $\sigma(P)$ if $X_j = 1$, and value equal to P otherwise.

It remains to prove the lemma for $|M_y| > n/2$. Fix both i and I. We prove by induction on $|\overline{M_y}|$ that there exists a program I-approximating $Z(\emptyset, M_y, g_i)$ of degree $\leq \deg_G^I$(MOD-FAMILY-q) $+ |\overline{M_y}|$.

If $|\overline{M_y}| = 0$, let P_a be the minimum-degree program I-approximating $\{x \in \{0,1\}^* : \sum_{i=1}^{|x|} x_i \equiv a(\text{mod q})\}$ via $\sigma^a(g_i)$. Each P_a has degree $\leq \deg_G^I$(MOD-FAMILY-q), and $P_0 \circ \cdots \circ P_{q-1}$ is an I-approximation of $Z(\emptyset, M_y, g_i)$ with degree $\leq \deg_G^I$(MOD-FAMILY-q).

If $|\overline{M_y}| > 0$, pick $X_j \in \overline{M_y}$. By induction, there is a program P I-approximating $Z(\emptyset, M_y \cup \{X_j\}, g_i)$ of degree $\leq \deg_G^I$(MOD-FAMILY-q) $+ |\overline{M_y}| - 1$. I claim that $P' = \sigma^{-1}(X_j P) \circ (X_j P)^{-1} \circ P$ I-approximates $Z(\emptyset, M_y, g_i)$. For assignments $a \notin I$ with $a_j = 0$, val$(P', a) =$ val$(P, a) = Z(\emptyset, M_y \cup \{X_j\}, g_i) = Z(\emptyset, M_y, g_i)$. For assignments $a \notin I$ with $a_j = 1$, val$(P', a) =$ val$(\sigma^{-1}(P) \circ P^{-1} \circ P, a) =$ val$(\sigma^{-1}(P), a) = \sigma^{-1}(val(P, a)) = \sigma^{-1}(Z(\emptyset, M_y \cup \{X_j\}, g_i)) = Z(\emptyset, M_y, g_i)$. □

Lemma 16 *For every $I \subseteq \{0,1\}^n$, disjoint subsets $M_y, M_x \subseteq \{X_1, \ldots, X_n\}$, and $i \in \omega$, there is a polynomial program I-approximating $Z(\emptyset, M_y, g_i)$ with degree $\leq \deg_G^I$(MOD-FAMILY-q) $+ (n/2)$.*

Proof. Fix I, M_y, and i. We prove the lemma by induction on $|M_x|$. For $|M_x| = 0$, this lemma is precisely Lemma 15.

For $|M_x| > 0$, pick $X_k \in M_x$. Let P_1 denote the program with value $Z(M_x \setminus \{X_k\}, M_y \cup \{X_k\}, g_i)$, and let P_2 denote the program with value $Z(M_x \setminus \{X_k\}, M_y, g_i)$. Both P_1 and P_2 exist by induction.

Let P denote the program $\tau_i(P_1 \circ P_2^{-1})$. For assignments $a \notin I$ with $a_k = 0$, val$(P_1, a) =$ val(P_2, a) and so val$(P, a) = \tau_i(e) = e = Z(M_x, M_y, g_i)$. Now suppose $a \notin I$ is an assignment with $a_k = 1$. Define j so that val$(P_2, a) = \sigma^{j+1}(g_i)$ at a. Then, at a we have val$(P_1, a) = \sigma^{j+1}(g_i)$, implying that val$(P, a) = \tau_i(\sigma^{j+1}(g_i)\sigma^j(g_i^{-1})) = \sigma^j(g_i) =$ val$(P_2, a) = Z(M_x \setminus \{X_k\}, M_y, g_i) = Z(M_x, M_y, g_i)$. □

Proof. (Theorem 14) Suppose $I \subseteq \{0,1\}^n$ and let $f : \{0,1\}^n \to G$ be any function. It suffices to show that there is a polynomial program P I-approximating f with $\deg(P) \leq \deg_G^I$(MOD-FAMILY-q) $+ (n/2)$.

By Proposition 6, there is a polynomial program P' with value equal to f. Assume without loss of generality that the instructions of P' contain only group elements from the set of generators $\{g_i\}_{i\in\omega}$. Every instruction of P' is then of the form $\langle M_x, g_i\rangle$ for some $M_x \subseteq \{X_1, \ldots, X_n\}$. Such an instruction has the same value as $Z(M_x, \emptyset, g_i)$, so to obtain P from P', we expand every instruction $\langle M_x, g_i\rangle$ of P' into a polynomial program I-approximating $Z(M_x, \emptyset, g_i)$ of degree $\leq \deg_G^I$(MOD-FAMILY-q) $+ (n/2)$, which exists by Lemma 16. □

It is easy to see that the class of groups which are q-completable is closed under taking direct products.

Theorem 17 *If G_1 and G_2 are both q-completable, then $G_1 \times G_2$ is q-completable.*

The additive group of any field with qth roots of unity is a concrete example of a q-completable group. Let F be such a field and $h \in F$ be a qth root of unity. Every non-zero element $a \in F$ induces an automorphism $\rho_a(x) = ax$ of F^+. We can take our set of generators to be F itself, since for any element $x \in F$, $\rho_h(x) \neq x$, $\rho_h^q(x) = x$, and $\rho_{(h-1)^{-1}}(\rho_h^{j+1}(x) - \rho_h^j(x)) = (h-1)^{-1}(h^{j+1}x - h^j x) = h^j x = \rho_h^j(x)$.

6 Polynomial Programs over Finite Groups

The Razborov-Smolensky method uses degree as the crucial measure of the complexity of a function: taking F to be a field of characteristic p, functions in AC^0[MOD p] can be closely approximated by low degree 1_F-accepting polynomial program families over F^+, whereas MOD q has high degree if closely approximated by such programs, for distinct primes p, q (Theorem 7).

In the search for a proof that MOD $k \notin AC^0$[MOD pq] (for distinct primes k, p, q) it is natural to ask the question: is there a group G and a group element $g \in G$ such that all functions in AC^0[MOD pq] can be closely approximated by low degree g-accepting polynomial program families, but where MOD k cannot? A priori, it may seem that since G may now be chosen to be any group (as opposed to just one of the groups appearing as the additive group of some field), such pairs of groups and group elements may exist.

The results in this section show that no pairing of a *finite* group with one of its elements has the above property. We prove a dichotomy theorem which shows that for any finite group G and element $g \in G$, either g is G-universal, implying that all of the languages MOD m for $m > 1$ can be computed by degree one g-accepting polynomial program families; or, any g-accepting polynomial program family can be translated into a polynomial program family over $\mathbb{Z}_p$ (for some prime p) with at most a constant factor degree increase. In the latter case, we have that the limitations to computing MOD q for any prime q other than p given in Theorem 7 apply: no low degree ($o(\sqrt{n})$) g-accepting polynomial program family over G can closely approximate MOD q, otherwise we would have a low degree polynomial program family over $\mathbb{Z}_p$ closely approximating MOD q.

Theorem 18 *For every finite group G and non-identity element $g \in G$, either g is G-universal; or, for any g-accepting polynomial program family $\{P_n\}_{n\geq 1}$ over G of degree $d(n)$, there is a polynomial program family over $\mathbb{Z}_p$ (for some prime p) of degree $\leq c \cdot d(n)$ accepting the same set as $\{P_n\}_{n\geq 1}$, for a constant c depending only on the group G.*

Proof. This is immediate from Theorem 24 and Corollary 26, proved below. □

We now investigate the cases of abelian groups, nilpotent groups, and non-nilpotent groups. While our results on nilpotent groups apply to abelian groups, in the case of abelian groups we are able to obtain a sharper result with a simpler proof, so we treat them separately.

6.1 Abelian Groups

In this section, we show that a polynomial program over any finite abelian group can be converted into a polynomial program over a cyclic group of prime order, without changing the degree. This is stronger than our result for nilpotent groups, which involves potentially increasing the degree by a constant factor.

We begin by establishing the result for cyclic p-groups, which are the building blocks for finite abelian groups.

Lemma 19 *If C is a cyclic p-group for some prime p, and P is an accepting polynomial program over C, then there is a polynomial program over $\mathbb{Z}_p$ equivalent to P.*

Proof. Proof by induction on $|C| = p^k$. If $k = 1$, then P itself is a polynomial program with the desired properties. If $k > 1$, then suppose P is g-accepting, and let N denote the unique subgroup (of C) of order p^{k-1}. We break into two cases.

Case 1: If $g \notin N$, then by Lemma 4 applied to P and the canonical homomorphism from C to C/N, we obtain a polynomial program P' over the group C/N equivalent to P. (Since C is abelian, N is a normal subgroup.) The group C/N is of order p and so is isomorphic to $\mathbb{Z}_p$; thus, P' is a polynomial program with the desired properties.

Case 2: If $g \in N$, then we prove that if P is a program only taking on values in N, then the group elements in the instructions of P can be modified so that they are all in N. This is sufficient, as P is then equivalent to a program over a group of size p^{k-1}, and so there is a program over $\mathbb{Z}_p$ equivalent to P by induction.

Our proof is by induction on the number of inputs.

If P is a program over 1 input, then the only possible monomials in P are $\emptyset$ and $\{X_1\}$; if we take the instructions $\langle M_{i_1}, g_{i_1}\rangle, \ldots, \langle M_{i_k}, g_{i_k}\rangle$ where $M_{i_j} = \emptyset$, we can rewrite them as $\langle M_{i_1}, e\rangle, \ldots, \langle M_{i_{k-1}}, e\rangle, \langle M_{i_k}, g_{i_1} \ldots g_{i_k}\rangle$ without changing the value of the program (as C is abelian). If we perform the same rewriting on the instructions with monomial $\{X_1\}$, every instruction has a group element in N.

If P is a program over $n > 1$ inputs, let $P' = \langle M_{i_1}, g_{i_1}\rangle, \ldots, \langle M_{i_k}, g_{i_k}\rangle$ denote the instructions of P where $X_n \notin M_{i_j}$ (for all $j = 1, \ldots, k$). The sequence of instructions P' must itself be a g-accepting polynomial program on $n-1$ inputs, since the value of P' at a $0-1$ assignment to $\{X_1, \ldots, X_{n-1}\}$ is precisely the value of P at the same $0-1$ assignment extended so that $X_n = 0$. By induction, then, we can modify the g_{i_j}'s so that they are all in N, without changing the value of P anywhere. Similarly, if we take the instructions $\langle M'_{i_1}, g'_{i_1}\rangle, \ldots, \langle M'_{i_m}, g'_{i_m}\rangle$ of P where $X_n \in M'_{i_j}$ (for all $j = 1, \ldots, m$), the sequence of instructions $P'' = \langle M'_{i_1} \setminus \{X_n\}, g'_{i_1}\rangle, \ldots, \langle M'_{i_m} \setminus \{X_n\}, g'_{i_m}\rangle$ viewed as a program over the set of inputs $\{X_1, \ldots, X_{n-1}\}$ takes on only values in N, as the value of P'' at a $0-1$ assignment to $\{X_1, \ldots, X_{n-1}\}$ multiplied by the value of P' at the same $0-1$ assignment is precisely the value of P at the assignment extended so that $X_n = 1$. Since P' and P only take on values in N, P'' only takes on values in N, and we can modify the g'_{i_j}'s so that they are all in N, without changing the value of P anywhere. The modified version of P has the same value as the original polynomial program on all $0-1$ assignments to the variables $\{X_1, \ldots, X_n\}$. □

Theorem 20 *If G is a finite abelian group, and P is an accepting polynomial program over G, then there is a polynomial program over $\mathbb{Z}_p$, for some prime p, equivalent to P.*

Proof. Suppose that P is g-accepting, where $g \in G$. By the classification theorem for finite abelian groups (see, for example, [13, pp. 102-103]), G is isomorphic to the direct product of cyclic p-groups: $G = C_1 \times \ldots \times C_k$, where each C_i is a cyclic p_i-group. For $i = 1, \ldots, k$, let $h_i : G \to C_i$ denote the canonical "projection" homomorphism. Since $g \neq e_G$, there exists j such that $h_j(g)$ is not the identity element of C_j. By Lemma 4 applied to the program P and the homomorphism h_j, we have a $h_j(g)$-accepting polynomial program P' over a cyclic p_j-group equivalent to P. Applying Lemma 19 to the program P', we obtain a program over $\mathbb{Z}_{p_j}$ equivalent to P' (and hence P). □

Because the additive group of any ring is abelian, Theorem 20 implies that no finite ring is an appropriate algebraic setting in which to attempt to represent the functions in $AC^0[\text{MOD}\, pq]$ as "low degree" polynomials. At first blush, it may seem that taking the direct product of a finite field with characteristic p and a finite field with characteristic q might give an appropriate ring, since in some sense it contains the structure of both fields; this theorem demonstrates that such a ring does not give "simultaneous" access to the included field structures. On the other hand, MOD k (assuming p, q, k distinct primes) is "hard" for such rings: taking R to be the direct product of two fields with kth roots of unity results in MOD-FAMILY-k being complete for R^+ (Theorems 14 and 17).

6.2 Nilpotent Groups

We now show that for any accepting polynomial program P over a nilpotent group, there is an polynomial program of degree at most a constant times the degree of P accepting the same set over $\mathbb{Z}_p$, for some prime p. This has been observed [3, 15] in the special case of degree one polynomial programs; we give an alternative proof.

Let $NC^0[\text{MOD}\, p]$ denote the class of functions decidable by NC^0 circuits with oracle gates for the MOD p function, where the MOD p gates may have unbounded fan-in (but where all other gates must have bounded fan-in). We begin by showing that the word problem over any p-group can be computed by $NC^0[\text{MOD}\, p]$ circuits.[4]

Lemma 21 *Suppose G is a p-group for some prime p. The word problem over G can be computed by an $NC^0[\text{MOD}\, p]$ circuit family.*

Our proof of this lemma is along the lines of Barrington's proof [2] that the word problem for any fixed solvable group G is in $AC^0[\text{MOD}\, |G|]$. (When elements of G are inputs to circuits, we assume that they are represented by bit strings of length $\lceil \log_2 |G| \rceil$.)
Proof. Proof by induction on $|G| = p^k$. Suppose $k > 0$. Every non-trivial finite p-group has a normal subgroup of index p; let N be such a normal subgroup of G. The group G/N is cyclic of order p; fix an element $a \notin N$, so that aN generates G/N.

Given a product $g_1 \cdots g_k$ of elements of G, we can write each element g_i in the form $a^{\epsilon_i} n_i$, where $n_i \in N$. (Notice that converting between any two bit representations of elements of a fixed group G can be done in NC^0.) Let b_i be the product $a^{\epsilon_1} \cdots a^{\epsilon_i}$. Each b_i can be computed from the a^{ϵ_i} in NC^0, since, for any fixed $k \geq 1$, a MOD p^k gate with unbounded fan-in can be simulated in $NC^0[\text{MOD}\, p]$ (see the proof of Proposition 1(3)).

We have $a^{\epsilon_1} n_1 \cdots a^{\epsilon_k} n_k = (b_1 n_1 b_1^{-1}) \cdots (b_k n_k b_k^{-1}) b_k$. Each product $b_i n_i b_i^{-1}$ can be computed from the n_i's and b_i's in NC^0. Since N is normal, each product $b_i n_i b_i^{-1}$ is in N and thus by induction there is a $NC^0[\text{MOD}\, p]$ circuit computing the product $(b_1 n_1 b_1^{-1}) \cdots (b_k n_k b_k^{-1})$. From this product, $g_1 \cdots g_k = (b_1 n_1 b_1^{-1}) \cdots (b_k n_k b_k^{-1}) b_k$ can be computed in NC^0. □

[4]The word problem is the question of determining whether or not the product of a sequence of elements from a group is equal to the identity element.

The next lemma shows that the circuit family computing the word problem resulting from Lemma 21 (indeed, every $NC^0[\text{MOD}\, p]$ circuit family) can be converted to a polynomial over $\mathbb{Z}_p$.

Lemma 22 *For every language L computed by a $NC^0[\text{MOD}\, p]$ circuit family, there is a polynomial program family over $\mathbb{Z}_p$ of constant degree accepting L.*

Proof. Let $\{X_1, \ldots, X_m\}$ denote the inputs to the mth circuit in the circuit family, which decides $L \cap \{0,1\}^m$. By Proposition 5, it suffices to show that there is a polynomial in $\mathbb{Z}_p[X_1, \ldots, X_m]$ which at an assignment $a \in \{0,1\}^m$ takes on the value 0 or 1 depending on whether or not $a \in L$.

We represent this circuit using a polynomial, by induction on the height of the circuit. Each input gate X_i has polynomial representation X_i. We can represent the NOT of a gate with representation f as $1 - f$; we can represent the AND of two gates with representations f_1, f_2 as $f_1 f_2$; we can represent the OR of two gates with representations f_1, f_2 as $1 - (1 - f_1)(1 - f_2)$; and we can represent the MOD p of k gates with representations $f_1, \ldots, f_k$ as $(\sum_{i=1}^{k} f_i)^{p-1}$. In each case, we represent a gate at depth d using a polynomial which has degree at most a constant (namely, $\max(2, p-1)$) times the maximum degree of the gates at depth $d - 1$. $\Box$

In fact, it can be proved that $NC^0[\text{MOD}\, p]$ is precisely the class of languages computed by accepting polynomial program families over p-groups of constant degree, or, equivalently, the class of languages computed by accepting polynomial program families over $\mathbb{Z}_p$ of constant degree.

Theorem 23 *Suppose G is a p-group for some prime p. There is a constant c (depending only on G) such that for every accepting polynomial program P over G, there is a polynomial program over $\mathbb{Z}_p$ of degree $\leq c \cdot \deg(P)$ accepting the same set as P.*

Proof. It suffices to show that there is a polynomial in $\mathbb{Z}_p[X_1, \ldots, X_n]$ satisfying the degree bound which, at every $0 - 1$ assignment to the X_i's, takes on the value 1 or 0 depending on whether or not P accepts or rejects.

Set $b = \lceil \log_2 |G| \rceil$. Let $\{C_l\}_{l \geq 1}$ denote the $NC^0[\text{MOD}\, p]$ circuit family of Lemma 21, where C_l decides the word problem for products of length l. Let $\{Y_{i,j}\}_{1 \leq i \leq l, 1 \leq j \leq b}$ denote the inputs to the lth circuit C_l in this circuit family. For each $i \in \{1, \ldots, l\}$, $Y_{i,1}, \ldots, Y_{i,b}$ is intended to be the bit string representation of the ith group element in the product.

By Lemma 22, there is a constant c such that for all l, there is a polynomial f_l over $\mathbb{Z}_p$ with indeterminates $\{Y_{i,j}\}$ of degree $\leq c$ accepting the same set as C_l. That is, the polynomial f has value 0 or 1, depending on whether or not the product of the group elements is the identity or not.

Suppose $P = \langle M_1, g_1 \rangle, \ldots, \langle M_l, g_l \rangle$ is a polynomial program on n inputs. Starting from f, create a new polynomial as follows. For each $i = 1, \ldots, l$ and $j = 1, \ldots, b$, replace $Y_{i,j}$ with $r_j(g_i) \prod_{X_k \in M_i} X_k \in \mathbb{Z}_p[X_1, \ldots, X_n]$, where $r_j(g_i)$ is $0 \in \mathbb{Z}_p$ or $1 \in \mathbb{Z}_p$ depending on the value of the jth bit of the bit string representing g_i. Assuming without loss of generality that e_G is represented by the bit string 0^b (i.e., the all-zero string of length b), the resulting polynomial is in $\mathbb{Z}_p[X_1, \ldots, X_n]$, has degree $\leq c \cdot \deg(P)$, and is 0 or 1 depending on whether or not P rejects or accepts. $\Box$

Because a nilpotent group is isomorphic to the direct product of p-groups (see for example [13]), we obtain the desired result using the same idea as in the proof of Theorem 20.

Theorem 24 *Suppose G is a finite nilpotent group. There is a constant c (depending only on G) such that for every accepting polynomial program P over G, there is a polynomial program over $\mathbb{Z}_p$ of degree $\leq c \cdot \deg(P)$ accepting the same set as P (for some prime p).*

6.3 Non-Nilpotent Groups

Theorem 25 *Suppose G is a finite non-nilpotent group. Then for every nonidentity element $g \in G$, either g is G-universal or, for every g-accepting polynomial program P, there is a polynomial program equivalent to P over a nilpotent group.*

When G is a group, define $L_1(G) = G$ and $L_n(G) = [L_{n-1}(G), G]$ for $n > 1$. A group is nilpotent if and only if there is a N such that $L_N(G) = \{e_G\}$ [13].
Proof. Since G is non-nilpotent, there is a N such that $L_N(G) = L_{N+1}(G)$, and $L_N(G)$ is a non-trivial subgroup. That the elements of $L_N(G)$ are G-universal is implicit in both [3] and [15].

If $g \notin L_N(G)$, then let $h : G \to G/L_N(G)$ be the canonical homomorphism. The subgroup $L_N(G)$ is normal (in fact, characteristic) in G, and it is straightforward to show that $G/L_N(G)$ is nilpotent. Thus, for every g-accepting polynomial program P over G, the accepting polynomial program $h(P)$ is equivalent to P (Lemma 4) and over a nilpotent group. □

From Theorems 24 and 25, we derive the following corollary.

Corollary 26 *Suppose G is a finite non-nilpotent group. There is a constant c (depending only on G) such that if $g \in G$ is a non-identity element that is not G-universal, then for every g-accepting polynomial program over G, there is a polynomial program over $\mathbb{Z}_p$ of degree $\leq c \cdot \deg P$ accepting the same set as P (for some prime p).*

7 Future Work

The dichotomy theorem of Section 6 suggests that the framework of accepting polynomial program families over a single finite group G will not allow one to use degree to differentiate between languages in the class AC^0[MOD m], and the languages MOD k (for prime k not dividing m, and for composite m with at least two distinct prime factors). Although the Razborov-Smolensky method does sucessfully use degree in this setting to so differentiate when m is prime, our results indicate that a more refined framework is necessary to handle the composite case. One possibility might be to use polynomial program families parameterized with *sequences* of finite group and element pairs, one pair for each input length. Because of the universality of non-nilpotent groups and the fact that a polynomial program over a nilpotent group induces at least one polynomial program over a p-group (see Theorem 24), the natural candidate for a group sequence would be a sequence of finite p-groups. In this direction, it might be worthwhile to understand what the optimal constants of Theorem 23 are, and whether certain sequences of p-groups admit a low degree "bias" towards particular languages MOD q, for primes q not equal to p. It may also be interesting to investigate the possibility of using a polynomial program family over an infinite group as the base algebraic setting.

Perhaps, in place of degree, a more refined notion of complexity is required to develop a setting where the functions of AC^0[MOD m] have "low complexity" and certain functions not in AC^0[MOD m] can be shown to have "high complexity." For instance, might it be possible to assign costs to individual elements of the group (or other structure) over which the polynomial program family is defined, in a meaningful way?

The proof of Theorem 14, which shows that q-completability of a group G implies that MOD-FAMILY-q is complete for G, seems to rely heavily on the existence of "inverse" polynomial programs P^{-1}. This suggests that quasigroups may be the appropriate algebraic structures with which to parameterize polynomial programs to obtain a characterization of AC^0[MOD m] *à la* the Razborov-Smolensky method. Can the counting argument of Smolensky (described at the beginning of Section 5), showing that complete elements cannot be closely approximated by low degree polynomials, be extended to quasigroups?

Given two polynomial programs over the additive group of a ring, multiplication of the two corresponding polynomials in the ring is a natural means of obtaining a third polynomial program, and this operation has a number of nice properties (distributivity, etc.). As regards the Razborov-Smolensky method, a key property of multiplication of polynomials over rings is that the resulting polynomial has degree linear in each of the original polynomials. In order to "combine" two polynomial programs which are not defined over the additive group of a ring, it may be important to develop an analogous "multiplication" operation over whatever algebraic structure (quasigroup, etc.) one wishes to use, that is distinct from the existing single operation on the algebraic structure. Attempts to develop such "multiplication" operations may be of independent interest, as one could potentially obtain algebraic structures with two operations more general than rings.

As mentioned, the polynomial program model of computation is a natural unification of the polynomial over a ring model and the program over monoids model, and it may be possible to extend the existing results from either model in this new framework. In general, it should be interesting to further investigate how the resources of degree, size, and semigroup interact in the polynomial program model.

Lastly, one could study extensions of the polynomial program model which include non-determinism or randomness; or, counting questions concerning this model.

Acknowledgements

We would like to thank Dexter Kozen for many useful discussions. We would also like to thank Eric Allender, David A. Mix Barrington, Steve Chase, Keith Dennis, Riccardo Pucella, and Pascal Tesson for their help.

References

[1] M. Ajtai. Σ_1^1 formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

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

[3] D. A. Mix Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89:109–132, 1990.

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

[5] R. Beigel. The polynomial method in circuit complexity. In *Proceedings 8th Structure in Complexity Theory*, pages 82–95. IEEE Computer Society Press, 1993. Revised version, 1995.

[6] H. Caussinus and F. Lemieux. The complexity of computing over quasigroups. *Foundations of Software Technology and Theoretical Computer Science*, 1994.

[7] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[8] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Procedings 18th Symposium on Theory of Computing*, pages 6–20. IEEE Computer Society Press, 1986.

[9] C. Y. Lee. Representation of switching functions by binary decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.

[10] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1988. Expanded edition. Originally published in 1969.

[11] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Matematicheskie Zametki*, 41:598–607, 1987. In Russian. English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 41:333–338, 1987.

[12] K. Regan. Polynomials and combinatorial definitions of languages. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 261–293. Springer-Verlag, New York, 1997.

[13] D. Robinson. *A Course in the Theory of Groups*. Springer-Verlag, second edition, 1996.

[14] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings 19th Symposium on Theory of Computing*, pages 77–82. ACM Press, 1987.

[15] P. Tesson and D. Thérien. The computing power of programs over finite monoids. Technical Report 01-005, Electronic Colloquium on Computational Complexity, 2001.

[16] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings 26th Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, 1985.