NON-UNIFORM AUTOMATA OVER GROUPS

David A. Mix Barrington,
Howard Straubing, Denis Therien

Computer and Information Science Department
University of Massachusetts

COINS Technical Report 89-56

Non-Uniform Automata Over Groups

David A. Mix Barrington[1, 2]
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003, U.S.A.

Howard Straubing[3]
Computer Science Department
Boston College
Chestnut Hill, MA 02167, U.S.A.

Denis Thérien[4]
School of Computer Science
McGill University
Montréal, P.Q. H3A 2K6, Canada

June 20, 1989

[1] Former name David A. Barrington.

[2] Supported by NSF grants CCR-8714714 and MCS-8304769 and by US Air Force grant AFOSR-82-0326.

[3] Supported by NSF grant CCR-8700700.

[4] Supported by NSERC grant A4546 and FCAR grant 86-EQ-2933.

Abstract

A new model, non-uniform deterministic finite automata (NUDFA's) over general finite monoids, has recently been developed as a strong link between the theory of finite automata and low-level parallel complexity. Achievements of this model include the proof that width 5 branching programs recognize exactly the languages in non-uniform NC^1 (Barrington, 1989), NUDFA characterizations of several important subclasses of NC^1 (Barrington and Thérien, 1988), and a new proof (Barrington and Thérien, 1988) of the old result (Brzozowski and Knast, 1978) that the dot-depth hierarchy is infinite, using Sipser's work (1983) on constant depth circuits.

Here we extend this theory to NUDFA's over solvable groups (NUDFA's over non-solvable groups have the maximum possible computing power (Barrington, 1989)). We characterize the power of NUDFA's over nilpotent groups and prove some optimal lower bounds for NUDFA's over certain groups which are solvable but not nilpotent. Most of these results appeared in preliminary form in (Barrington and Thérien, 1987).

1. Introduction

A large body of recent work in combinatorial complexity has focused on classes of languages recognizable by circuit families with tight restrictions on depth. For example, the class (non-uniform) NC^1, which consists of the languages where the inputs of length n can be recognized by boolean circuits of fan-in two and depth $O(\log n)$, has proved to be quite robust. It is equal to the class of languages definable by families of boolean formulas of polynomial length (Spira, 1971) and to those recognizable by branching program families of constant width and polynomial size (Barrington, 1989). It is also important as the base class in the parallel complexity theory outlined by Cook (1985).

Still, we are unable to prove any natural problems to be outside of NC^1. (For example, the hypothesis $NP = NC^1$, in either a uniform or non-uniform setting, is perfectly consistent with known results.) This suggests that we consider even smaller complexity classes, in which membership might be easier to determine. Furst, Saxe, and Sipser (1984) showed that the parity language is not in the class AC^0 (circuits of constant depth, polynomial size, and unbounded fan-in) and thus showed that this natural subclass of NC^1 is in fact a proper subclass. Further work has shed more light on the internal structure of NC^1 (Razborov, 1987; Smolensky, 1987; Barrington and Thérien, 1988).

One of the most familiar complexity classes of all, the class of regular languages, lies entirely within NC^1. We will take the algebraic view of finite automata — an automaton consists of a transformation of the state set for each letter, generating a homomorphism from the monoid of words under concatenation to the finite monoid of transformations on the state set. The behavior of the automaton on an input of n letters is given by an iterated multiplication of n elements of the monoid, which can easily be performed by an NC^1 circuit.

The complexity theory for automata is comparatively well-developed. We can prove languages not regular, and tell in some detail what kinds of automata can recognize what kinds of languages. By identifying an automaton with an algebraic object, its syntactic monoid, we can describe automata as combinations of various primitive components (Krohn, Rhodes and Tilson, 1968). These components perform the basic operations of AND, OR, modular counting, and multiplication in a simple group, and are described in sections 3 and 4.

Schützenberger (1965) showed that automata built up in this way using only the AND and OR operations can recognize only the star-free regular languages, i.e., those languages which can be defined using only boolean operations and concatenation. In particular these "aperiodic automata" cannot count modulo 2. This is quite reminiscent of the Furst-Saxe-Sipser result, suggesting a general analogy between circuit classes and classes of automata. Barrington (1989) showed that the ability to perform multiplication in a non-abelian simple group is surprisingly powerful in the circuit setting (gates of this type, used along with AND and OR, can simulate all NC^1 circuits in constant depth and polynomial size). Finally Barrington and Thérien (1988) made this analogy explicit in the work outlined in section 4, giving characterizations of AC^0 and other classes in terms of a new model, non-uniform finite automata. The gate types of unbounded fan-in circuits appear to correspond exactly to the basic components of finite automata.

The main open question in (Barrington and Thérien, 1988) was to prove limits on non-uniform automata made up only from AND, OR, and modular counting components. This would separate NC^1 from its subclass ACC of languages recognized by circuit families of constant depth and polynomial size made up of AND, OR, and modular counting gates. Here we attack this question by considering the power of modular counting components by themselves. This corresponds to considering automata whose syntactic monoids are solvable groups.

In effect we are looking for a dual result to the Furst-Saxe-Sipser theorem. We know that AND and OR gates cannot be used in a polynomial-size constant-depth circuit to simulate modular counting, but can gates for modular counting in such a circuit simulate AND or OR? We conjecture that they cannot and here offer some partial results in this direction. With careful definitions (see, e.g. (Barrington, Straubing, and Thérien, 1988)) circuits of modular counting gates correspond exactly to non-uniform automata over solvable groups. While we cannot yet prove lower bounds for general solvable groups, we can do so for a large class of groups.

Our main results are as follows (exact definitions will be given below). We prove that no non-uniform finite automaton of any size over a nilpotent group can calculate the AND of n variables, for sufficiently large n. We prove that if G is an extension of a p-group by an abelian group, then no non-uniform automaton over G with size subexponential in n can calculate the AND of n variables. This is an improvement over a similar result in the preliminary version of this paper (Barrington and Thérien, 1987). Our principal conjecture is that this latter result extends to any solvable group G.

2. The Model and Related Definitions

The *non-uniform deterministic finite automaton* (NUDFA) was developed originally as an equivalent form of the bounded width branching program (Barrington, 1985). An NUDFA over a monoid M on n inputs from an alphabet A is defined by a *program* of *length* ℓ. This is a sequence of ℓ *instructions*, each of which consist of a variable number i (from 1 to n) and a function from A to M. On a given input setting $a_1, \ldots, a_n$ the instruction *yields* the monoid element corresponding to the value of the input a_i, and the entire NUDFA yields the ordered product of the yields of its instructions.

As an example, consider the case of an ordinary deterministic finite automaton with input alphabet A, state set S, and transition function $\delta : (A \times S) \rightarrow S$. The transition function can just as easily be viewed as assigning a transformation of S (a function from S to itself) to each element of A. If we let M be the monoid of transformations of S (under the operation of functional composition) then δ induces a function $\hat{\delta}$ from A to M. Now consider the following program over M:

$$\langle (1, \hat{\delta}), \ldots, (n, \hat{\delta}) \rangle.$$

The yield of this program on an input word $a_1 \ldots a_n$ is the transformation on S corresponding to the action of the original deterministic finite automaton on that input word.

In this way the NUDFA is an extends the algebraic view of the theory of finite automata (Eilenberg, 1976; Pin, 1986). In this setting the automaton is simply viewed as a map from A to M as above, which induces a homomorphism from A^* (the monoid of strings on A, under concatenation) into M. A language $L \subseteq A^*$ is said to be recognized by M if L is the inverse image, under this homomorphism, of some subset of M. A variant of Kleene's theorem asserts that a language is regular iff it is recognized by some finite monoid. Furthermore, subclasses of the regular languages can be put in correspondence with families of finite monoids in a systematic way, using the theory of pseudo-varieties developed by Eilenberg (1976). One example of such a correspondence is the result of Schützenberger (1965) that a regular language is star-free iff it is recognizable by a finite aperiodic monoid.

Ou basic object of study will be a *program family* over a particular finite monoid. Formally, a program family $\langle P_1, P_2, \ldots \rangle$ is an infinite sequence of programs $P_n = \{\langle i_{n,k}, f_{n,k} \rangle : 1 \leq k \leq \ell(n)\}$, with each $i_{n,k} \in \{1, \ldots, n\}$ and each $f_{n,k}$ a function from A to M. P_n defines a mapping ϕ_n from A^n to M, given by $\phi_n(a_1 \ldots a_n) = m_{n,1} \cdots m_{n,\ell(n)}$, where $m_{n,k} = f_{n,k}(a_{i_{n,k}})$. A program family $\langle P_1, \ldots \rangle$ thus defines a mapping ϕ from A^* to M, given by $\phi(w) = \phi_{|w|}(w)$. Just as with the homomorphic mapping for an ordinary DFA, we say that a language L is recognized by a program family if it is the inverse image, under this map ϕ, of a subset of M.

Non-uniform models of computation have a long history (see, for example, Savage (1976) for earlier work on circuits and formulas). Many discrete models of computation, such as boolean circuits, boolean formulas, or branching programs, take a fixed number of bits as input rather than a string of unknown length. To compare these models to models which recognize languages, we must speak of a family of computing elements, one for each input length. It is often mathematically convenient to put no constraint on the manner in which the individual elements depend on the input length, but only look at the resources needed for each element as a function of the length.

For example, the class of languages recognizable by boolean circuit families where the size of the circuits grows as a polynomial in the input size forms a non-uniform analogue of P, the class of languages recognizable by Turing machines in polynomial time. The analogy can be made exact by speaking of circuits with a uniformity condition (e.g., each circuit must be constructible by a polynomial time Turing machine which is given the input size in unary). Alternatively, we can speak of non-uniform Turing machines, which are given an advice string along with their input, of length polynomial in the input size.

If we restrict the depth of polynomial size boolean circuit families, we can produce the NC hierarchy of parallel complexity classes, originally developed by Pippenger (1979) and extensively described in the survey article of Cook (1985). These classes, in their uniform versions, are important because they correspond to the problems which can be solved quickly in various models of parallel computation. In their non-uniform versions they are still of considerable theoretical importance. In this paper we will work with two of these classes. Non-uniform NC^1 is the class of languages recognizable by circuit families of fan-in two and depth $O(\log n)$ (and hence polynomial size). This is equal to the class of languages recognizable by boolean formulas of polynomial length (or circuits of polynomial size which are trees) (Spira, 1971) and to the class recognizable by branching programs of constant width and polynomial size (Barrington, 1989). Non-uniform AC^0 is the class recognizable by circuit families of constant depth, polynomial size, and unbounded fan-in. Furst, Saxe, and Sipser (1984) and Ajtai (1983) showed that the parity language is not in AC^0, and thus that AC^0 is a proper subset of NC^1. Recently the internal structure of NC^1 and AC^0 has been the subject of extensive research (Sipser, 1983; Chandra, Fortune, and Lipton, 1983; Chandra, Stockmeyer, and Vishkin, 1984; Fagin, Klawe, Pippenger, and Stockmeyer, 1985; Barrington, 1989; Barrington and Thérien, 1988; Razborov, 1987; Smolensky, 1987), which will be described below.

A *branching program* (see Barrington (1989) for background) is a directed acyclic graph of bounded out-degree, where nodes are labelled by input variables and edges by the possible values of the variable corresponding to the node they leave. A setting of the input variables defines a path from a special start node to one of the sinks of the graph, and the sinks are labelled as accepting or rejecting the input. The *width* of a branching program has various definitions, but as defined by Barrington (1985, 1989) a branching program of width w is equivalent to an NUDFA over the transformation monoid T_w of all transformations on a base set of size w.

We will use the following standard terminology from algebraic automata theory and group theory throughout. The reader is referred to standard texts such as those of Eilenberg (1976) and Zassenhaus (1958) respectively for more background.

A *group* is a monoid which contains an inverse for every element. (All groups in this paper will be finite.) A *permutation group* is a group whose elements are bijections of some finite set and whose operation is functional composition (i.e., a group G together with a one-to-one homomorphism from G into the group S_X of permutations of a finite set X). If $f \in S_X$ and $x \in X$, we will write xf for the image of x under f. If the underlying set of a permutation group is not given, it is to be assumed that the group is acting on itself by right multiplication.

A *subgroup* is a subset of a group which is closed under the group multiplication. A subgroup H of G is *normal* if for any $g \in G$ and $h \in H$ we have $g^{-1}hg \in H$. In this case we define the *factor group* G/H in the usual way and say that G is an *extension* of H by G/H.

The *commutator* of two group elements g and h is $ghg^{-1}h^{-1}$. If H_1 and H_2 are each subgroups of a group G, their *commutator* $[H_1, H_2]$ is the subgroup generated by all elements $h_1h_2h_1^{-1}h_2^{-1}$ for $h_1 \in H_1$ and $h_2 \in H_2$. The *derived series* of a group G is defined by $G^0 = G, G^{i+1} = [G^i, G^i]$. A group is *solvable* if its derived series terminates with G^m the trivial group. The *lower central series* of a group G is defined by $G_0 = G, G_{i+1} = [G_i, G]$. A group is *nilpotent* of *class* m if its lower central series terminates with G^m the trivial group. An important special case of nilpotent groups are the *p-groups*, those groups whose order is a power of some prime p. In fact any nilpotent group is a direct product of p-groups. The *exponent* of a group is the least common multiple of the orders of its elements, i.e., the least $q \geq 1$ such that $a^q = e$ for all $a \in G$.

One group G *divides* another group H if there is a homomorphism from a subgroup of H onto G. The relation "G divides H" is a partial order on the set of finite groups — it is the transitive closure of the union of the relations "G is a subgroup of H" and "G is a homomorphic image of H". A *variety* of groups is a class of groups closed under division and under direct product. (This definition is at variance with standard usage in universal algebra (where our "varieties" are often called "pseudo-varieties"), but is appropriate because we are dealing only with finite groups.) The p-groups for a given p, the nilpotent groups, and the solvable groups each form a variety. These notions can be extended to permutation groups (see, e.g., Eilenberg (1976)).

Let G and H be permutation groups with underlying sets X and Y respectively. The *wreath product* $G \circ H$ is a group of permutations of $X \times Y$, given (as a set) by

$\{\langle f,h\rangle : f \in G^Y, h \in H\}$. The permutation $\langle f,h\rangle$ acts on an element (x,y) of $X \times Y$ to give $(xf(y), yh)$. The wreath product is associative on permutation groups, i.e., if I is also a permutation group with underlying set Z, then $(G \circ H) \circ I$ and $G \circ (H \circ I)$ consist of exactly the same permutations of $X \times Y \times Z$. We will use the following facts about the wreath product (see, e.g., Eilenberg (1976)):

- If G divides H and I divides J, then $G \circ I$ divides $H \circ J$.
- Any extension of G by H (including $G \times H$) divides $G \circ H$.
- If V and W are varieties of groups, then a group divides the wreath product of a group in V and a group in W iff it is an extension of a group in V by a group in W. The set of all such groups form a variety, which we will denote $V \circ W$.
- A permutation group is a p-group iff it divides a wreath product $Z_p \circ \ldots \circ Z_p$ of permutation groups Z_p (the group of integers mod p acting on itself by addition).
- A permutation group is solvable iff it divides a wreath product $Z_{q_1} \circ \ldots \circ Z_{q_r}$ of cyclic permutation groups.

A monoid is *aperiodic* if all of its subsets which form groups under the monoid operation are trivial groups. A monoid is *solvable* if all such groups are solvable groups. A language is recognizable by an aperiodic monoid iff it is star-free, i.e., can be defined from the one-letter languages using concatenation and boolean operations but not the Kleene star operation (Schützenberger, 1965). Aperiodic monoids are parametrized by their ***dot-depth***, which is the minimum number of times concatenation must be used to define all the languages they can recognize (see Straubing (1986) for background on dot-depth).

3. Previous Work

The NUDFA model grew out of the study of bounded width branching programs initiated by Borodin, Dolev, Fich, and Paul (1983) and by Chandra, Furst, and Lipton (1983). The former group conjectured that no program of constant width and polynomial size (in our language, no program family of polynomial size over any finite monoid) could calculate the majority function. Barrington (1985) found that constant width branching programs could be forced into a normal form equivalent to the NUDFA's defined here, and worked with NUDFA's over the permutation group S_3 in an attempt to learn more about branching programs of width 3. He proved that there exist NUDFA's of exponential length over S_3 computing any boolean function, and that computing the AND function ($f(x_1, \ldots, x_n) = 1$ iff for all i, $x_i = 1$) in a restricted form requires exponential length.

Barrington (1989) then refuted the conjecture of Borodin et al. (1983) about the majority function by proving the following theorem, which we restate in terms of NUDFA's.

Theorem 1: If G is a nonsolvable group, the class of languages recognized by NUDFA's over G of polynomial length is exactly (non-uniform) NC^1.

Proof: (outline) Determining the output of an NUDFA over G requires an iterated multiplication of elements in G. If the length of this multiplication is polynomial,˙it can be performed by a binary tree of binary multiplications of height $O(\log n)$ and thus by an NC^1 circuit.

For the converse, let $H \neq 1$ be a subgroup of G which is its own commutator subgroup, i.e., $[H, H] = H$ (such an H exists iff G is not solvable). For every element h of H other than the identity and every circuit C of depth d, we construct an NUDFA of length $2^{O(d)}$ which yields h on input settings accepted by C and the identity on other settings. The key step in this construction occurs when C is the AND of two circuits C_1 and C_2. By induction NUDFA's $B_{1,h}$ and $B_{2,h}$ exist for any h yielding h or the identity depending on C_1 and C_2 respectively. The NUDFA obtained by concatenating $B_{1,g}$,$B_{2,h}$,$B_{1,g^{-1}}$, and $B_{2,h^{-1}}$ yields $ghg^{-1}h^{-1}$ if both C_1 and C_2 accept and the identity otherwise. As H is generated by commutators from H, the desired NUDFA's for C may all be constructed from such concatenations.□

The power of polynomial length NUDFA's over a solvable group appears to be more limited. They may be simulated by circuit families of constant depth, polynomial size, and unbounded fan-in where the individual gates compute AND, OR, or MOD q for some integer q fixed for the circuit family (Barrington, 1989). This will follow from the results we outline below.

We define ACC to be the class of languages recognizable by such circuit families (the AC^0 closure of counters) — it is conjectured (Barrington, 1989) that majority is not in ACC and thus that $ACC \neq NC^1$. Recently Razborov (1987) has proved that such circuits with AND, OR, and MOD 2 gates cannot do majority (confirming an earlier conjecture of Furst et al. (1984)), and Smolensky (1987) has extended this to MOD p gates (for p a single prime fixed for the family) by showing that circuits of AND, OR, and MOD p gates cannot compute the MOD q function if q is prime to p.

Barrington and Thérien (1988) showed that various subclasses of NC^1 are exactly the classes of languages recognizable by polynomial length NUDFA's over various classes of monoids. We summarize these results in three theorems, which are proved using the classification by Thérien (1981) of solvable finite monoids.

Theorem 2: A language is recognizable by polynomial length NUDFA's over a solvable monoid iff it is in ACC.□

Theorem 3: A language is recognizable by polynomial length NUDFA's over an aperiodic monoid iff it is in AC^0.□

Theorem 4: A language is recognizable by polynomial length NUDFA's over a monoid of dot-depth k iff it is in depth k AC^0.□

Theorem 4 allows Sipser's proof (1983) that the depth k circuit hierarchy is strict to be converted into a new proof that the dot-depth hierarchy is infinite (Brzozowski and Knast, 1978). In particular, the circuit and input language constructed for depth k circuits has dot-depth exactly k. If it had dot-depth $k-1$, any depth k circuit family could be converted (via NUDFA's over this language) to an equivalent depth $k-1$ family of only polynomially greater size. Repeated application of this process could reduce any AC^0 circuit family to an equivalent depth k family, contradicting Sipser's theorem. The conversion does not work in the other direction to provide a new proof of Sipser's theorem, because the dot-depth result *a priori* does not rule out the possibility that a dot-depth k regular language has circuits of depth $k-1$.

4. NUDFA's over Solvable Groups

The above results show that allowing NUDFA's to operate over more complicated monoids increases their power. But at the highest level, that of non-solvable monoids, the power comes entirely from the embedded groups, i.e., non-solvable groups have as much power as non-solvable monoids. It is natural, then, to examine this relationship between greater complication and greater power in the group setting. What is the power of NUDFA's operating over solvable groups of various less complicated forms? We present some results to this end, which we believe form the beginnings of a program which could add significantly to our knowledge of the fine structure of NC^1.

We focus upon the power of an NUDFA over a group to simulate an unbounded fan-in threshold counter, as, for example, by computing the AND of the input variables. This is possible over non-solvable groups in polynomial length (Theorem 1) but as solvable groups are built up purely from modular counters, one might think that NUDFA's over them could not simulate the AND function at all. The actual situation is more complicated. We will see in the next section that NUDFA's over nilpotent groups cannot do AND at all. But over groups which are solvable but not nilpotent, we can carry out an analogue of the construction of Theorem 1.

Theorem 5: If G is a group which is not nilpotent, there is a family of exponential-size NUDFA programs over G that calculates the AND function.

Proof: Let H be a normal subgroup of G with $[G, H] = H$ — such a subgroup must exist in any non-nilpotent group G. By induction we construct NUDFA programs $B(h,i)$ for all $h \in H$ and $i \leq n$, where the value of $B(h,i)$ on an input setting is h if x_1 through x_i are all on and e otherwise. Each $B(h,1)$ is a single instruction. Each $h \in H$ is a product of commutators $g_k h_k g_k^{-1} h_k^{-1}$ with $g_k \in G, h_k \in H$. We define $B(h, i+1)$ to be the concatenation for all k of $B(h_k, i)C(g_k, i+1)B(h_k^{-1}, i)C(g_k^{-1}, i+1)$, where $C(g,i)$ is the single instruction whose value is g if x_i is on and e otherwise. $B(h,n)$ calculates the AND of all n variables and has size at most $(4|H|)^n$, where $|H|$ is the order of H.□

Conjecture: If G is solvable, any family of NUDFA programs over G calculating the AND function has exponential size. Thus if G is solvable but not nilpotent, Theorem 5 is optimal.

In Section 8 we will prove this conjecture in the special case of some relatively uncomplicated groups, but it remains unknown in general. Proving it might allow us to separate out the group and aperiodic behavior of an NUDFA over an arbitrary solvable monoid, which might give a better characterization of the languages within ACC and help to prove $ACC \neq NC^1$.

5. NUDFA's over Nilpotent Groups

In this section we will characterize the power of all NUDFA's over nilpotent groups, and thus show that they cannot calculate threshold functions. One might begin by considering the easier case of abelian groups. There the behavior of NUDFA's is the sum (in the group) of the behaviors with respect to each input variable, as the order of the instructions does not matter. In fact the NUDFA calculates a linear function from the variables to the group. If the input size n is sufficiently large with respect to the group, some large number of variables must have the same coefficient in this linear map. Flipping a number of these variables equal to the exponent of the group from all zeroes to all ones or vice versa will not affect the output, so the NUDFA cannot, for example, calculate the AND function. Note also that any such NUDFA may be simulated by an NUDFA with only n instructions, one per variable. We will now see that all of these properties of NUDFA's over abelian groups are special cases of similar properties for nilpotent groups.

To establish this generalization, we will need to develop the notion of representing functions from $\{0,1\}^n$ to a ring by polynomials, used in the recent work of Razborov

(1987) and Smolensky (1987). If R is a commutative ring with identities 1_R and 0_R, any function from $\{0,1\}^n$ to R is uniquely represented by a polynomial over R in the n boolean variables $x_1, \ldots, x_n$. Formally, this is the ring $R[x_1, \ldots, x_n]$ with the identity $x_i^2 = x_i$ for each i. Given a setting of the n variables, a polynomial is evaluated by plugging in 1_R for 1 and 0_R for 0.

We will say that a language is *strongly represented* by a family of polynomials $p_1, p_2, \ldots$ if each p_n represents the characteristic function of $L \cap \{0,1\}^n$. We will say that it is *weakly represented* if there are subsets B_n of R for each n such that $p_n(\mathrm{x}) \in B_n$ iff $\mathrm{x} \in L \cap \{0,1\}^n$. We get non-uniform complexity classes by bounding the size (number of monomials with nonzero coefficient) and the degree (maximum number of variables in a monomial) of the polynomials in the family by functions of n.

If R is a field, strong and weak representation are closely related. If the polynomial family $\langle p_n \rangle$ weakly represents a language L (so that for each n $p_n(\mathrm{x}) \in B_n$ iff $\mathrm{x} \in L$), then L is strongly represented by

$$\sum_{i \in B_n} 1 - (p_n - i)^{|R|-1}$$

. This polynomial has only polynomially greater size and degree greater only by a constant factor. However, with other rings the two concepts can differ remarkably. For example, over Z_6 the set $\{\mathrm{x} : |\mathrm{x}| = 1 \bmod 2\}$ is weakly represented by the polynomial $\sum 3x_i$ but strongly represented only by a polynomial whose coefficient on a term $\prod_{i \in S} x_i$ is $2^{|S|-1}$mod 6 (or 0 if $S = \emptyset$). This polynomial has exponentially greater size.

Theorem 6: A language is recognized by a family of NUDFA's over a nilpotent group iff it is weakly represented by a family of polynomials of constant degree over a direct product of cyclic rings. More precisely, it is so recognized by a family of programs over a nilpotent group of class m and exponent q iff it can be weakly represented by polynomial of degree m over Z_q^k for some k. (Z_q^k is the k-fold direct product of the ring of integers mod q.)

Proof: We use the results of Thérien (1983) on nilpotent groups and subword counting. Let A be any finite alphabet. For words $x = x_1 \ldots x_n \in A^*$ and $u = u_1 \ldots u_k$, define $\binom{x}{u}$ to be the number of occurrences of u as a subword of x, i.e. the number of sequences $1 \leq i_1 < \ldots < i_k \leq n$ such that $u = x_{i_1} \ldots x_{i_k}$. It is proved in (Thérien, 1983) that if N is a nilpotent group of class m and exponent q, and x and y satisfy $\binom{x}{u} = \binom{y}{u}$ mod q for all u of length $\leq m$, then any monoid homomorphism from A^* into the group N maps x and y to the same value (in other words, no automaton whose syntactic monoid divides N can distinguish x and y).

Furthermore, there is a homomorphism ϕ from A^* into a particular nilpotent group N of class m and exponent q which simultaneously counts occurrences of all subwords u of length at most m. That is, it maps each x to a value $\phi(x) \in N$ such that for each u, the value $\binom{x}{u}$ mod q can be determined from $\phi(x)$. Thus we have an exact combinatorial characterization of the languages recognized (in the finite automata sense) by such groups.

Let N be a nilpotent group of class m and let P be an NUDFA program over N which converts an input string $x \in \{0,1\}^n$ into a string $P(x)$ in $N^{p(n)}$. For each word $u \in N^*$ of length $\leq m$, the number mod q of occurrences of u as a subword of $P(x)$ is given by a polynomial over Z_q in the boolean input variables, of degree m. This is because each possible set of positions in $P(x)$ where u might occur gives rise to a term of degree $\leq m$ (the product of the boolean variables corresponding to these positions). This implies the first half of the theorem, with k being the total number of subwords counted.

For the second half, for any given q and k, we must show how to use an NUDFA program to compute the value of any polynomial $f(x)$ of degree at most m. The main idea is to have the program yield a word over some alphabet A, which will encode the value of $f(x)$ in the number mod q of its occurrences of various subwords of length at most m. We know (Thérien, 1983) that for the appropriate nilpotent group N of class m and exponent q, there is a homomorphism ϕ from A^* into N such that the value of $\phi(x)$ determines all the subword counts for a word x.

We will define an NUDFA program $\begin{bmatrix} f(x) \\ a_1 \ldots a_k \end{bmatrix}$ over this group N, where $f(x)$ is any polynomial of degree at most m over Z_q and the a_i are distinct letters in A. Each instruction of this program will yield either the identity or an element $\phi(a)$ of N, so that we can think of the yield on input x as a word $w(x)$ in A^* which will be mapped by ϕ into N. The number mod q of occurrences of $a_1 \ldots a_k$ as a subword of $w(x)$ will be exactly $f(x)$. Thus the yield of the program will determine the value of $f(x)$. By using an independent set of letters for each of k copies of Z_q, we can extend this construction to give an NUDFA program, over a nilpotent group of class m and exponent q, calculating any polynomial of degree at most m over Z_q^k.

We begin by defining $\begin{bmatrix} x_i \\ a \end{bmatrix}$, for a single input x_i and single letter a, as the appropriate single instruction. That is, the program $\begin{bmatrix} x_i \\ a \end{bmatrix}$ yields $\phi(a)$ if x_i is on and the identity of N if x_i is off. Next, $\begin{bmatrix} -x_i \\ a \end{bmatrix}$ is $q-1$ copies of $\begin{bmatrix} x_i \\ a \end{bmatrix}$. In general, $\begin{bmatrix} f(x)+g(x) \\ a \end{bmatrix}$ is the concatenation of $\begin{bmatrix} f(x) \\ a \end{bmatrix}$ and $\begin{bmatrix} g(x) \\ a \end{bmatrix}$. For strings u and v of input letters, where no letter occurs more than once, $\begin{bmatrix} f(x)g(x) \\ uv \end{bmatrix}$ can be defined as $\begin{bmatrix} f(x) \\ u \end{bmatrix} \begin{bmatrix} g(x) \\ v \end{bmatrix} \begin{bmatrix} -f(x) \\ u \end{bmatrix} \begin{bmatrix} -g(x) \\ v \end{bmatrix}$. This program produces zero subwords mod q in combination with anything before or after it, but does produce subwords uv if $f(x)g(x)$ has a nonzero value. In this way programs for arbitrary polynomials and subwords can be built up, as long as the degree of the polynomial is at most the length of the subword and the subword has no repeated letters.□

Corollary: No NUDFA of any size over a nilpotent group can calculate the AND function (i.e., weakly recognize the language $1^* \subseteq \{0,1\}^*$).

Proof: The polynomial for the AND function is easily seen to have degree n over any ring. However, we must show that no constant-degree polynomial can weakly recognize the AND language by having a value on input 1^n which differs from the value in any other setting. To do this we will use Ramsey's Theorem to establish the following periodicity property of the functions calculated by constant degree polynomials. The AND function clearly does not have this periodicity, and so the Corollary is immediately implied by the following:

Lemma: For n sufficiently large, any polynomial of degree t in n variables over Z_q^k has the following property: In any setting there exist $q \cdot (t!)$ variables set alike (all zeroes or all ones) which can all be flipped without changing the value of the polynomial.

Proof: Assume without loss of generality that the setting contains a majority of zeroes — otherwise work with ones and dualize the following (note that the dualization preserves the degree of polynomials). Simplify the polynomial by plugging in ones for the variables set to one, getting a polynomial of degree at most t in the variables originally set to zero. By Ramsey's Theorem, for sufficiently large n, there must be a set A of $q \cdot (t!)$ variables satisfying the following conditions: Let T_i, $1 \leq i \leq t$, be the set of all monomials of length i in the variables of A. Note that the cardinality of T_i is $\binom{|A|}{i}$. Then every monomial in T_i has a coefficient in the new polynomial that only depends on i. That is, all linear terms from A have some coefficient c_1, all quadratics have the same coefficient c_2, and so on up to c_t. Of course, the number of variables we have available must be very large compared to t, q, and k.

Now consider the new setting obtained by flipping all the variables in A. The value of the polynomial changes by

$$\sum_{i=1}^{t} c_i \binom{q \cdot (t!)}{i} = 0$$

because q divides each of these binomial coefficients.□

This periodicity property might seem to be a logical consequence of working only with modular counters of constant modulus. However, we have seen that NUDFA's over solvable groups can compute functions which are not at all periodic, such as AND. Does such a periodicity property hold for NUDFA's of polynomial length over solvable groups?

Corollary: Any NUDFA over a nilpotent group of class m has an equivalent NUDFA of length $O(n^m)$ over the same group.

Proof: Simply keep track of the length in the construction of Theorem 6. One can also prove this directly by converting any program into an equivalent one of polynomial length, using a variant of the "formal commutator" construction of (Thérien, 1983) to rearrange the instructions until similar instructions can be collapsed.□

This work has recently been extended by Péladeau and Thérien (1988), who investigate the subsets of $\{0,1\}^n$ which can be recognized by a given nilpotent group. We have just shown that such a subset cannot be a singleton, but they prove that any such subset has exponential cardinality (at least $2^n/c$ elements, where c is a constant depending on the group).

6. Representing Languages by Linear Forms

We are left with the case of groups which are solvable but not nilpotent. NUDFA's over one such group, S_3, have been studied by Barrington (1985). He showed, in effect, that exponential program size is required for these NUDFA's to compute the AND function. The general method was to show that calculating a function with an S_3 program corresponds to expressing it as a linear combination of certain basis functions over a finite field, and then showing that all such linear combinations for the AND function contain exponentially many elements.

Here we extend the ideas there to show a similar bound in the case of certain other groups which are solvable but not nilpotent. In order to do this, we must develop some machinery for the representation of functions by linear forms over a finite field. In particular, we will define a multidimensional version of the discrete Fourier transform, and derive certain properties which will prove to have computational significance.

Let F be a finite field of order at least 3, and let F^* be the set of nonzero elements of F. As is well known, F^* is a cyclic group under the field multiplication. Let k denote the order of F^*, so that $k \geq 2$. Let us fix a generator g of this group. Now if $h \in F^*$, there is a unique m such that $0 \leq m < k$ and $g^m = h$. We thus define $\log h = m$, with the understanding that the definition of the logarithm depends on the choice of the generator g.

Let $n > 0$. We will be concerned with the F-vector space $\mathcal{A}^n$ of functions from $(F^*)^n$ to F. Our first task will be to define a particular basis for this vector space. For each $\mathrm{w} = (w_1, \ldots, w_n) \in (F^*)^n$ we define a function $P_{\mathrm{w}} : (F^*)^n \to F$ by

$$P_{\mathrm{w}}(\mathrm{x}) = P_{\mathrm{w}}(x_1, \ldots, x_n) = w_1^{\log x_1} \ldots w_n^{\log x_n}.$$

Observe that $P_{\mathbf{w}}(\mathbf{x}) = P_{\mathbf{x}}(\mathbf{w})$.

Now let $\mathbf{v}, \mathbf{w} \in (F^*)^n$. We denote $(w_1^{-1}, \ldots, w_n^{-1})$ by $\mathbf{w}^{-1}$. If $\mathbf{v} = \mathbf{w}^{-1}$ then

$$\sum_{\mathbf{x} \in (F^*)^n} P_{\mathbf{w}}(\mathbf{x}) P_{\mathbf{v}}(\mathbf{x}) = k^n = (-1)^n.$$

If $\mathbf{v} \neq \mathbf{w}^{-1}$ then for some $i \in \{1, \ldots, n\}, u = w_i v_i \neq 1$. Then for some $c \in F$,

$$\sum_{\mathbf{x} \in (F^*)^n} P_{\mathbf{w}}(\mathbf{x}) P_{\mathbf{v}}(\mathbf{x}) = c \cdot \sum_{\mathbf{x} \in F^*} u^{\log x} = c \cdot (u^k - 1)/(u - 1) = 0.$$

So $\{P_{\mathbf{w}} | \mathbf{w} \in (F^*)^n\}$ is a basis for $\mathcal{A}^n$, and this basis is orthogonal with respect to the inner product

$$\langle f_1, f_2 \rangle = \sum_{\mathbf{x} \in (F^*)^n} f_1(\mathbf{x}) \cdot f_2(\mathbf{x}^{-1}).$$

Given $f \in \mathcal{A}^n$, we denote by *supp*(f) the cardinality of the set $\{\mathbf{w} \in (F^*)^n | f(\mathbf{w}) \neq 0)\}$ and by *weight*(f) the number of nonzero coefficients that occur when f is written as an F-linear combination of the $P_{\mathbf{w}}$. The *Fourier transform* of f is the function $\mathbf{T}f \in \mathcal{A}^n$ defined by

$$\mathbf{T}f(\mathbf{w}) = \sum_{\mathbf{x} \in (F^*)^n} f(\mathbf{x}) \cdot P_{\mathbf{w}^{-1}}(\mathbf{x}).$$

If *supp*$(f) = 1$, with f taking its only nonzero value at $\mathbf{x}_0$, then $\mathbf{T}f = f(\mathbf{x}_0) \cdot P_{\mathbf{x}_0^{-1}}$, which has weight 1. Since $\mathbf{T}$ is linear, it follows that for any $f \in \mathcal{A}^n$, *supp*(f) = *weight*$(\mathbf{T}f)$. Moreover, the orthogonality relations imply that $\mathbf{T}P_{\mathbf{w}}$ is nonzero only at $\mathbf{w}^{-1}$. Thus, by linearity, for any f, *weight*(f) = *supp*$(\mathbf{T}f)$.

Given $f_1, f_2 \in \mathcal{A}^n$, we define the *convolution* $f_1 * f_2 \in \mathcal{A}^n$ by

$$(f_1 * f_2)(\mathbf{x}) = \sum_{\mathbf{w} \in (F^*)^n} f_1(\mathbf{w}) \cdot f_2(\mathbf{w}^{-1}\mathbf{x}).$$

It is then easy to show that $\mathbf{T}(f_1 \cdot f_2) = \mathbf{T}f_1 * \mathbf{T}f_2$ and that $\mathbf{T}(f_1 * f_2) = \mathbf{T}f_1 \cdot \mathbf{T}f_2$, where the dot denotes pointwise multiplication.

For each $\mathbf{w} = (w_1, \ldots, w_n) \in (F^*)^n$, we define a function $Q_{\mathbf{w}}$ from $\{0, 1\}^n$ to F by

$$Q_{\mathbf{w}}(u_1, \ldots, u_n) = w_1^{u_1} \ldots w_n^{u_n}.$$

Observe that $Q_{\mathbf{v}} \cdot Q_{\mathbf{w}} = Q_{\mathbf{vw}}$, where $\mathbf{vw}$ denotes the componentwise product of $\mathbf{v}$ and $\mathbf{w}$. Note also that $P_{\mathbf{w}}(x_1, \ldots, x_n) = Q_{\mathbf{w}}(\log x_1, \ldots, \log x_n)$, provided that $(x_1, \ldots, x_n) \in \{1, g\}^n$. In particular, the functions $Q_{\mathbf{w}}$ span the vector space of functions from $\{0,1\}^n$ to F, but are not linearly independent. Nonetheless we are able to prove some lower bounds on the number of $Q_{\mathbf{w}}$ required to express certain boolean functions. We will be able to translate these bounds into lower bounds on program length for programs over certain solvable groups.

Recall that the AND function from $\{0,1\}^n$ into $\{0,1\}$ is that function taking the value 1 if all components are 1 and taking the value 0 otherwise. Since $\{0,1\} \subseteq F$, we can view AND as taking its values in F.

Theorem 7: The AND function cannot be written as an F-linear combination of fewer than $(\frac{k}{k-1})^n$ of the functions $Q_{\mathbf{w}}$.

Proof: Let $h_1 \in \mathcal{A}^n$ be the characteristic function of the n-tuple $(g,\ldots,g)$, and let $h_2 \in \mathcal{A}^n$ be the characteristic function of the set $\{1,g\}^n$. Since $weight(\mathbf{T}h_1) = supp(h_1) = 1$, $\mathbf{T}h_1$ is equal to $P_{\mathbf{w}}$ for some $\mathbf{w}$ and is nowhere zero, so that $weight(h_1) = supp(\mathbf{T}h_1) = k^n$.

A simple calculation shows that

$$\mathbf{T}h_2(w_1,\ldots,w_n) = \sum_{S\subseteq\{0,1\}^n}\prod_{j\in S} w_j = \prod_{1\leq i\leq n}(w_i+1).$$

Thus $weight(h_2) = supp(\mathbf{T}h_2) = (k-1)^n$.

Now suppose AND $= \sum c_{\mathbf{w}}Q_{\mathbf{w}}$ and let $f = \sum c_{\mathbf{w}}P_{\mathbf{w}}$. Then $f \cdot h_2 = h_1$. Since $weight(h_2 \cdot f) \leq weight(h_2) \cdot weight(f)$ we obtain $weight(f) \geq (\frac{k}{k-1})^n$, and thus that at least the required number of $c_{\mathbf{w}}$ are nonzero.□

The preceding argument suggests that, in general, functions with small weights have large supports, and vice versa. This is made precise in the following proposition.

Proposition: For any nonzero $f \in \mathcal{A}^n$, $supp(f) \cdot weight(f) \geq k^n$.

Proof: Consider the square matrix, with rows and columns indexed by $(F^{\cdot})^n$, whose $(\mathbf{w},\mathbf{x})$ entry is $P_{\mathbf{w}}(\mathbf{x})$. Since the $P_{\mathbf{w}}$ are linearly independent, this matrix is nonsingular and hence its columns are linearly independent. Now consider the matrix M whose $(\mathbf{w},\mathbf{x})$ entry is $f(\mathbf{x}) \cdot P_{\mathbf{w}}(\mathbf{x})$. All but $supp(f)$ columns of M are zero, and the remaining columns are each obtained by multiplying the corresponding column of the original matrix by a nonzero constant. Thus M has exactly $supp(f)$ linearly independent columns, and its rank is $supp(f)$. We can therefore extract $supp(f)$ linearly independent rows, which span the subspace of $\mathcal{A}^n$ consisting of functions which are zero on the zero-set of f. This subspace has dimension $supp(f)$. In particular, there is some linear combination at most $supp(f)$ of the functions $f \cdot P_{\mathbf{w}}$ that has support 1. Taking transforms, we obtain a linear combination of at most $supp(f)$ of the functions $\mathbf{T}(f \cdot P_{\mathbf{w}})$ that has support k^n. Now $supp(\mathbf{T}(f \cdot P_{\mathbf{w}})) = supp(\mathbf{T}f * \mathbf{T}P_{\mathbf{w}}) = supp(\mathbf{T}f)$, by the definition of the convolution and the fact that $\mathbf{T}P_{\mathrm{w}}$ has support 1. Since the support function is subadditive, we obtain $k^n \leq supp(f) \cdot supp(\mathbf{T}f)$, as claimed.□

7. Lower Bounds for Certain Solvable Groups

We are now ready to apply the results of the preceding section to put lower bounds on the program length needed for NUDFA's over certain solvable groups to calculate the AND function. We conjecture, of course, that a similar bound holds for any solvable group. In the section following this one we will indicate how our methods might be extended in this direction. The treatment in this section is an extension beyond that in the preliminary version of this paper (Barrington and Thérien, 1987), and the results obtained are somewhat stronger.

We begin by considering an interesting special case of groups which are closely related to particular finite fields. For a field F as above, we define the group G_F to be a semidirect product of F and F^* as follows. Elements of G_F are pairs (i, j) with $i \in F$ and $j \in F^*$, and the product is given by $(i,j)(k,\ell) = (i + jk, j\ell)$.

Proposition: Any NUDFA over G_F calculating the AND function has length $2^{\Omega(n)}$.

Proof: An instruction of an NUDFA over G_F has value $(r_0 + r_1 x_i, s_0 s_1^{x_i})$ for constants r_0, r_1, s_0, and s_1 and some input variable x_i. By induction, it is easy to see that the yield of a sequence of ℓ instructions is given by $(f(\mathrm{x}), g(\mathrm{x}))$, where $f \in \mathcal{A}^n$ has weight at most 2ℓ and $g \in \mathcal{A}^n$ has weight 1. This is because the functions $s_0 s_1^{x_i}$ have weight 1 and weight is submultiplicative.

Suppose that some NUDFA of length ℓ calculates the AND function. That is, the yield $N(\mathrm{x}) = (f(\mathrm{x}), g(\mathrm{x}))$ is in some set $S \subseteq G_F$ if $\mathrm{x} = 1^n$ and is not in S otherwise. Define f_i for $i \in F$ to be the product for $i' \neq i$ of $f - i'$. Note that f_i has nonzero value when $f(x) = i$ and is zero otherwise. Define g_j similarly. Now let h be the sum, for all (i, j) in S, of $f_i g_j$. So $h(\mathrm{x}) \neq 0$ exactly when $N(\mathrm{x}) \in S$. Now consider h^k (under pointwise multiplication of functions in $\mathcal{A}^n$). This function has weight polynomial in ℓ and value $h^k(\mathrm{x}) = 1$ iff $N(\mathrm{x}) \in S$, and $h^k(\mathrm{x}) = 0$ otherwise. By hypothesis this is exactly the AND function shown to require weight $(k/(k-1))^n$ in Theorem 7. Since this weight is polynomial in ℓ, $\ell = 2^{\Omega(n)}$.□

Examples of groups G_F include S_3 (for the three-element field) and A_4 (for the four-element field). Thus the above Proposition implies that width 3 permutation branching programs and width 4 even permutation branching programs (as defined by Barrington (1989)) require exponential length to compute the AND of their inputs.

In the width 3 case this improves the result of Barrington (1985) by extending the lower bound from strong recognition to weak recognition, as defined above. The techniques here can easily show a lower bound of $\Omega(2^{n/2})$ for the length of width 3 permutation branching programs which weakly recognize the AND function, as conjectured by Barrington (1985).

In the preliminary version of this paper (Barrington and Thérien, 1987) it was shown that this lower bound result could be extended to the variety of groups generated by G_F for each particular field F. Here we are able to improve this somewhat. For each prime p, let V_p be the variety of groups which divide the wreath product of a p-group and an abelian group. (V_p can also be defined as the set of all extensions of p-groups by abelian groups.) We extend the lower bound to the union of the V_p (which is not itself a variety). In fact every group from (Barrington and Thérien, 1987) is contained in some V_p.

To prove our most general result we will need a theorem about the languages recognized by wreath products of cyclic groups, due to Straubing (1979). Let A be a finite alphabet and $L \subset A^*$ a language, $a \in A$, and $r \in Z_m$. The language $\langle La, r, m\rangle$ is defined as those $w \in A^*$ such that the number of initial segments of w in the language Lc is congruent to r mod m.

Theorem 8:(Straubing, 1979) Any language in A^* recognized by a wreath product $Z_m \circ X$, where X is any monoid, is a boolean combination of languages of the form L_1 and of the form $\langle L_2a, r, m\rangle$, where L_1 and L_2 are languages recognized (in the original finite-automaton sense) by X. □

Theorem 9: Let G_p be a p-group and B an abelian group, and let $L \in \{0,1\}^n$ be recognized by a program of length $\ell \geq n$ over $G_p \circ B$. Then there is a finite field F such that the characteristic function of L has weight at most ℓ^K, where K is a constant depending only on G_p and B.

Corollary: Any NUDFA program family over $G_p \circ B$ computing the AND function has length $2^{\Omega(n)}$.□

Proof of Theorem 9: Without loss of generality, we will take G_p to be an r-fold wreath product of groups Z_p (which we will denote $Z_p^{[r]}$), and take B to be Z_m^k for some m prime to p. This is because the original $G_p \circ B$ divides some $Z_p^{[r]} \circ Z_m^k$, as we will now show. First, we may assume that p does not divide the order of B. This is because as an abelian group, B may be written as a direct product $P_1 \times B'$, where P_1 is a p-group and p does not divide the order of B'. Then B divides $P_1 \circ B'$, and we may replace $G_p \circ B$ by $(G_p \circ P_1) \circ B'$, using the associativity of the wreath product. $G_p \circ P_1$ is a p-group and divides some $Z_p^{[r]}$ by one of our basic facts (see Eilenberg (1976)). Finally, B' is abelian and must divide a direct product Z_m^k, where m is its exponent.

Since m divides $p^j - 1$ for some j, we can find a field F of characteristic p containing an element g of order m. We will prove the theorem by induction on r.

Lemma: Let $L_1, \ldots, L_k$ be any family of languages whose characteristic functions each have weight at most f. Then the characteristic function of any boolean combination of the L_i has weight $f^{O(1)}$ (with k considered as a constant).

Proof of Lemma: The boolean combination, expressed in conjunctive normal form, is an AND of at most 2^k terms, each an OR of at most k of the L_i. The characteristic functions of the ORs have weight at most f^k, and thus the characteristic function of the AND has weight at most $f^{k2^k} = f^{O(1)}$.□

Proof of Theorem 9: (continued) In the case $r = 0$ we have a program over Z_m^k, which can be thought of as k independent programs over Z_m. A language recognized by such a program is thus a constant-sized boolean combination of languages recognized by programs over Z_m. But any program over Z_m computes a linear map, which is a function of weight $O(n) = O(\ell)$. Applying the Lemma, the language recognized by programs over Z_m^k has a characteristic function of weight $\ell^{O(1)}$.

Now, for the inductive step, let L be recognized by programs of length ℓ over the group $Z_p^{[r+1]} \circ Z_m^k$, which is $Z_p \circ X$, where $X = Z_p^{[r]} \circ Z_m^k$. $\mathbf{x}$ is in L iff the yield $N(\mathbf{x})$ is in some regular language T recognized by $Z_p \circ X$. By Theorem 8 T is a boolean combination of languages of the form T_1 and $\langle T_2 a, q, p \rangle$ for various languages T_1 and T_2 recognized by X. Hence L is a boolean combination of languages $\{\mathbf{x} : N(\mathbf{x}) \in T_1\}$ and $\{\mathbf{x} : N(\mathbf{x}) \in \langle T_2 a, q, p \rangle\}$. By the Lemma, it suffices to show that languages of this kind have characteristic functions of weight $\ell^{O(1)}$. This is immediate by the inductive hypothesis in the first case, as such languages are themselves recognized by programs of length ℓ over X.

Let H be the characteristic function of $\{\mathbf{x} : N(\mathbf{x}) \in \langle T_2 a, q, p \rangle\}$. To compute H, we need to know, for each i with $1 \leq i \leq \ell$, whether the yield of the first i instructions of the program is a word in $T_2 a$. Let P_i be the function which is 1 if this is the case and zero otherwise. Then $H = 1 - ((\sum_{i=1}^{\ell} P_i) - q)^{|F^*|}$, and H has weight polynomial in that of the P_i. But P_i is the characteristic function of the AND of two languages: $\{\mathbf{x} :$ the i'th character of $N(\mathbf{x})$ is an $a\}$, and $\{\mathbf{x} :$ the first $i-1$ characters of $N(\mathbf{x})$ are in $T_2\}$. The characteristic function of the first of these languages is $x_j, 1 - x_j, 0$, or 1 for some j, and the second language is clearly recognized by programs of length $\leq \ell$ over X. So the weight of P_i is polynomial in ℓ by the inductive hypothesis.□

8. The Constant-Degree Hypothesis

In this section we consider a generalization of the weight of a function, and formulate a conjecture which would extend our lower bounds to some additional groups — those which divide wreath products of a p-group and a nilpotent group. For a positive integer r, let $u_1, \ldots, u_N$ be all monomials of degree at most r in the variables $x_1, \ldots, x_n$. For each vector $\mathbf{y} = (y_1, \ldots, y_N) \in (F^*)^N$, let the function $P_{\mathbf{y}}^{(r)}$ from $\{0,1\}^n$ to F be defined by $P_{\mathbf{y}}^{(r)}(x_1, \ldots, x_n) = y_1^{u_1} \cdots y_N^{u_N}$. The basis functions $P_{\mathbf{w}}$, used in our definition of weight above, are exactly the functions $P_{\mathbf{w}}^{(1)}$. In a similar way, we define the (r)-weight of a function f from $\{0,1\}^n$ to F to be the minimal number of $P_{\mathbf{y}}^{(r)}$ functions in a linear combination summing to f.

Conjecture: (The "constant-degree hypothesis") For fixed $r > 0$, the AND function has (r)-weight $2^{\Omega(n)}$.

Note first that this hypothesis does not follow directly from the $r = 1$ version proved above, as some functions $P_{\mathbf{y}}^{(2)}$ have exponential (1)-weight. However, we do not believe that these new basis functions bring one substantially closer to the AND function. We have shown some evidence of a duality between the functions of (1)-weight 1 and the functions of support 1 (such as AND), and the functions of low (r)-weight appear to be far more similar to the former.

One consequence of the constant-degree hypothesis would be a new and simpler proof of our Corollary to Theorem 6, that no program family of any size over a nilpotent group can calculate the AND function. We will show this in the course of proving the following, the main result of this section.

Theorem 10: (Assuming the constant-degree hypothesis.) Any program family computing the AND function over $G_p \circ N$, where G_p is a p-group and N is a nilpotent group, has length $2^{\Omega(n)}$.

Proof: N is a direct product of p_i-groups for distinct primes $p_1, \ldots, p_t$. We may assume that no p_i is equal to p, as any p-groups in H may be merged into G_p (if N_1 is a p-group, with $N = N_1 \times N'$, then $G_p \circ (N_1 \times N')$ divides $(G_p \circ N_1) \circ N'$, and $G_p \circ N_1$ is a p-group). As in the proof of Theorem 9, we will induct on the number of groups Z_p wreathed together to form G_p. We will work over a field F which contains elements $g_1, \ldots, g_t$ of order $p_1, \ldots, p_t$ respectively.

Consider first the case where G_p is trivial (the special case mentioned above). From Theorem 6, it is easy to see that the language is recognized by a t-tuple of polynomials of some constant degree r, where the j'th polynomial f_j is over Z_{p_j}. Over F, then, the characteristic function of $\{\mathbf{x} : f_j(\mathbf{x}) = b\}$ is $1 - (g_j^{f_j(\mathbf{x})} - g_j^b)^{|F^*|}$.

Since $g_j^{f_j(x)}$ is just a single $P_y^{(r)}$ term, it has (r)-weight 1. The characteristic function of our language then has constant (r)-weight, independent of both n and the program length. With the constant-degree hypothesis, this immediately implies that the AND function cannot be computed at all over a nilpotent group.

The inductive step proceeds exactly as in the proof of Theorem 9, except that now it is the (r)-weight rather than the (1)-weight which is proved to be polynomial in ℓ at each step. □

9. Open Problems

The connection between automata theory and the internal structure of NC^1 offers a great opportunity for progress in both, either by new results or new proofs of and insights about known results. Proving lower bounds in the NUDFA setting, for example, could give new proofs of many structure results, but so far we can only obtain such bounds for very uncomplicated monoids (except by using the existing structure results, as in the case of dot-depth). The next goal with respect to groups would be to prove our conjecture for groups which are solvable but not nilpotent. At the same time, we must look for analogous results among aperiodic monoids and eventually general solvable monoids. A full understanding of the latter should provide a proof that ACC is strictly contained in NC^1, a result which has so far evaded the methods of Razborov and Smolensky. Even proofs of known structure results without the use of the random restriction technology (Furst, Saxe, and Sipser, 1984) would be of considerable interest.

As a more immediate goal, one could look at the simplest solvable groups for which our conjecture is not yet proven. One avenue toward this would be to prove the constant-depth hypothesis of section 9. As far as specific groups rather than varieties, the next target for analysis would appear to be S_4 (showing width 5 to be necessary for polynomial size permutation branching programs to recognize all languages in NC^1). However, as this group has three steps in its upper central series, it appears that new techniques will be needed. Eventually we hope to generalize these methods to arbitrary wreath products and/or semidirect products of cyclic groups (as any solvable group divides a wreath product of cyclic groups).

One could examine other models similar to NUDFAs, for example, non-uniform stack machines. This might extend these techniques to complexity classes larger than NC^1. Straubing (1986a) has developed a framework generalizing both circuits and branching programs, which provides new proofs of some of the results of Barrington and Thérien (1988).

Finally, we would like to develop an algebraic theory to explain all this, analogous to Eilenberg's treatment of finite automata (1976). In particular, we suggest an analogue of the Krohn-Rhodes theorem which would imply many of the known structure results and $ACC \neq NC^1$ — that if the word problem for a simple group G can be solved (or perhaps "approximately solved", as in (Razborov, 1987)) by polynomial length NUDFA's over the wreath product of two monoids, it can be so solved by such NUDFA's over one monoid or the other.

10. Acknowledgements

We would like to thank Pierre McKenzie, who first noticed the connections between our efforts and organized the meeting in Montréal where we obtained the first of these results. We would also like to thank Martin Cohn, Mike Sipser, and Roman Smolensky for helpful discussions, and to thank the referees for several helpful comments.

11. References

Ajtai, M. (1983), Σ_1^1 formulae on finite structures. *Annals of Pure and Applied Logic* **24**, 1-48.

Barrington, D. A. (1985), "Width 3 Permutation Branching Programs", Technical Memorandum TM-291, M.I.T. Laboratory for Computer Science.

Barrington, D. A. (1989), Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *J. Comp. Syst. Sci.* **38:1**, 150-164.

Barrington, D. A. (1986) "Bounded-width branching programs," Ph.D. thesis, Dept. of Mathematics, M.I.T., also Technical Report TR-361, M.I.T. Laboratory for Computer Science.

Barrington, D. A. M., Straubing, H., and Thérien, D. (1988), "Finite monoids and circuit complexity", in preparation.

Barrington, D. A., and Thérien, D. (1987), Non-uniform automata over groups, *in* "Automata, Languages, and Programming: 14th International Colloquium", Springer-Verlag, Berlin, 163-173.

Barrington, D. A. M., and Thérien, D. (1988), Finite monoids and the fine structure of NC^1, *J. ACM* **35:4**, 941-952.

Borodin, A., Dolev, D., Fich, F. E., and Paul, W. (1983), Bounds for width two branching programs, *in* "Proc. 15th Ann. ACM Symposium on the Theory of Computing", Association for Computing Machinery, New York, 87-93.

Brzozowski, J. A., and Knast, R. (1978), The dot-depth hierarchy of star-free languages is infinite, *J. Comp. Sys. Sci.* **16**, 37-55.

Chandra, A. K., Fortune, S. and Lipton, R. (1983), Unbounded fan-in circuits and associative functions, *in* "Proc. 15th Ann. ACM Symposium on the Theory of Computing", Association for Computing Machinery, New York, 52-60.

Chandra, A. K., Furst, M. L., and Lipton, R. J. (1983), Multi-party Protocols, *in* "Proc. 15th Ann. ACM Symposium on the Theory of Computing", Association for Computing Machinery, New York, 94-99.

Chandra, A. K., Stockmeyer, L. and Vishkin, U. (1984), Constant-depth reducibility, *SIAM J. Computing* **13**, 423-439.

Cohen, R. S., and Brzozowski, J. A. (1971), Dot-depth of star-free events, *J. Comp. Sys. Sci.* **5**, 1-16.

Cook, S. A. (1985), The taxonomy of problems with fast parallel algorithms, *Information and Control* **64**, 2-22.

Eilenberg, S. (1976), "Automata, Languages, and Machines, Vol. B," Academic Press, New York.

Fagin, R., Klawe, M. M., Pippenger, N. J., and Stockmeyer, L. (1985), Bounded depth, polynomial-size circuits for symmetric functions, *Theoretical Computer Science* **36**, 239-250.

Furst, M, Saxe, J. B., and Sipser, M. (1984), Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory* **18**, 13-27.

Johnson, D. S. (1986), The NP-completeness column: An ongoing guide, *Journal of Algorithms* **7:2**, 289-305.

Lallement, G. (1979) "Semigroups and Combinatorial Applications", John Wiley & Sons, New York.

Péladeau, P., and Thérien, D. (1988), Sur les langages reconnus par des groupes nilpotents, *Comptes Rendus de L'Academie des Sciences (Serie I — Mathématique)* **306:2**, 93-95.

Pippenger, N. (1979), On simultaneous resource bounds (preliminary version), *in* "Proc. 20th Ann. IEEE Symposium on Foundations of Computer Science," IEEE Computer Society, Los Angeles, 307-311.

Pin, J. E., 1986, "Varieties of Formal Languages," Plenum Press, New York.

Razborov, A. A. (1987), Lower bounds for the size of circuits of bounded depth with basis $\{\&, \oplus\}$, *Mathematicheskie Zametki* 41:4, 598-607 (in Russian). English translation *Mathematical Notes of the Academy of Sciences of the USSR* **41**:4, 333-338.

Savage, J. E. (1976), "The Complexity of Computing," J. Wiley & Sons, New York.

Schützenberger, M. P. (1965), On finite monoids having only trivial subgroups. *Information and Control* **8**, 190-194.

Sipser, M. (1983), Borel sets and circuit complexity, *in* "Proc. 15th ACM Symposium on the Theory of Computing," Association for Computing Machinery, New York, 61-69.

Smolensky, R. (1987), Algebraic methods in the theory of lower bounds for boolean circuit complexity, *in* "Proc. 19th ACM Symposium on the Theory of Computing," Association for Computing Machinery, New York, 77-82.

Spira, P. M. (1971), On time-hardware complexity tradeoffs for boolean functions, *in* "Proc. 4th Hawaii Symposium on System Sciences," Western Periodicals Co., North Hollywood, Calif., 525-527.

Straubing, H. (1979), Families of recognizable sets corresponding to certain varieties of finite products, *J. Pure and Applied Algebra* **15**, 305-318.

Straubing, H (1985), Finite semigroup varieties of the form $V * D$, *J. of Pure and Applied Algebra* **36**, 53-94.

Straubing, H. (1986), Semigroups and languages of dot-depth 2, *in* "Automata, Languages, and Programming: 13th International Colloquium," Lecture Notes in Computer Science **226**, Springer Verlag, Berlin, 416-423.

Straubing, H. (1986a), "Finite monoids and boolean circuit complexity", manuscript.

Thérien, D. (1981), Classification of finite monoids: the language approach, *Theoretical Computer Science* **14**, 195-208.

Thérien, D. (1983), Subword counting and nilpotent groups, *in* "Combinatorics on Words: Progress and Perspectives" (L. J. Cummings, Ed.), Academic Press, New York, 297-305.

Zassenhaus, H. J. (1958) "The Theory of Groups", 2nd ed., Chelsea Publ. Co., New York.