CONSTANT-DEPTH PERIODIC CIRCUITS

HOWARD STRAUBING*
Department of Computer Science
Boston College
Chestnut Hill, Massachusetts 02167
USA

Communicated by John Rhodes
Received 10 July 1990

AMS Mathematics Subject Classification (1985 revision): 20M, 68Q

This paper is devoted to the languages accepted by constant-depth, polynomial-size families of circuits in which every circuit element computes the sum of its input bits modulo a fixed period q. It has been conjectured that such a circuit family cannot compute the *AND* function of n inputs. Here it is shown that such circuit families are equivalent in power to polynomial-length programs over finite solvable groups; in particular, the conjecture implies that Barrington's result on the computational power of branching programs over nonsolvable groups cannot be extended to solvable groups. It is also shown that polynomial-length programs over dihedral groups cannot compute the *AND* function. Furthermore, it is shown that the conjecture is equivalent to a characterization, in terms of finite semigroups and formal logic, of the regular languages accepted by such circuit families. There is, moreover, considerable independent evidence for this characterization. This last result is established using new theorems, of independent interest, concerning the algebraic structure of finite categories.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

1. Introduction

This paper is part of a continuing study of the computing power of small-depth families of boolean circuits, and its relation to logic and the theory of automata. The principal results concern circuits all of whose gates add their input bits modulo some fixed period. These theorems are aimed, for the most part, at establishing super-polynomial lower bounds on the size of constant depth circuits of this kind that compute certain functions, notably the *AND* function of n boolean variables.

This introductory section presents background material and a brief account of the results obtained.

1.1. *Small-depth circuits and parallel complexity*

The steps of a parallel algorithm on all possible inputs of a fixed length n can be represented by a boolean circuit, with a close correspondence, on one hand, between the number of gates in the circuit and the number of processors, and, on the other, between the depth of the circuit and the running time of the algorithm. Thus one can view a language $L \subseteq \{0,1\}^+$ (where $\{0,1\}^+$ denotes the set of all strings of positive

* Research supported by a NSF Grant CCR-8902369.

International Journal of Algebra and Computation Vol. 1 No. 1 (1991) 49–87
© World Scientific Publishing Company

length over the alphabet $\{0,1\}$) as being recognizable by a fast parallel algorithm if there is a family $\{\mathscr{C}_n : n > 0\}$ of boolean circuits such that $\mathscr{C}_n$ accepts $L \cap \{0,1\}^n$, the size of $\mathscr{C}_n$ is bounded above by a polynomial in n, and the depth of $\mathscr{C}_n$ is bounded above by a slowly growing function of n (e.g., $\log n$ or a polynomial in $\log n$). Cook [10] gives a survey of problems admitting fast parallel algorithms and discusses circuit models for these classes of problems.

Here I am concerned with the class NC^1 of languages recognized by families of bounded fan-in circuits of $O(\log n)$ depth. (These conditions force the size of the circuit to be bounded above by a polynomial in the number of inputs.) No condition concerning a feasible uniform construction of the circuits in the family is imposed. Thus, strictly speaking, NC^1 and its subclasses discussed below contain non-recursive languages—our model of problems with fast parallel algorithms includes many problems for which no algorithm exists at all! Experience has shown, however, that in searching for lower bounds—which is really the concern here—such uniformity considerations are irrelevant. Indeed, the languages that have been found to separate two complexity classes in this non-uniform model also separate the analogous classes defined under very strict uniformity conditions. This phenomenon is illustrated by the lower bounds results discussed in 1.2, as well as by the results of Sec. 5.

1.2. *Unbounded fan-in constant-depth circuit models*

Furst, Saxe and Sipser [12] considered the class AC^0 of languages recognized by constant-depth polynomial-size circuit families in which the fan-in of the *AND* and *OR* gates is not bounded. Their principal achievement was the proof that the languages

$$L_m = \left\{ a_1 \cdots a_k \in \{0,1\}^+ : \sum_{i=1}^{k} a_i \equiv 0 \pmod{m} \right\},$$

where $m > 1$, are not in AC^0. It is easy to show that L_m, like all regular languages, is recognized by a highly uniform NC^1 circuit family. The results of [12] were discovered independently by Ajtai [1] and subsequently improved by Yao [31] and Hastad [13] to show that L_m cannot be recognized by a constant-depth unbounded fan-in circuit family unless the size of the circuits grows exponentially in the number of inputs.

Razborov [22] and Smolensky [23] studied the situation in which one of the L_m is adjoined as an oracle language to the unbounded fan-in circuit model. More precisely, they considered constant-depth circuit families containing, in addition to *AND* and *OR* gates, gates that determine whether or not the sum of the incoming bits is divisible by m. It was shown in [23] that if m is a prime power, then such a family cannot recognize L_p, where p is a prime not dividing m, unless the size of the circuits grows exponentially. It is conjectured that the analogous result holds without the restriction that m be a prime power.

1.3. *Programs over finite groups and finite monoids*

Barrington [2] and Barrington and Thérien [6] introduced *programs* over a finite group or finite monoid in connection with the study of the computational power of

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

branching programs of bounded width. These programs will be defined precisely in Sec. 2. The principal result of [2] is that if M is a finite monoid containing a nonsolvable group, then every language in NC^1 can be recognized by a polynomial-size family of programs over M.

In [6] it is shown that various subclasses of NC^1 defined in terms of constant-depth circuit models can also be characterized by programs over finite monoids, with a restriction imposed on the algebraic structure of the monoid. Thus AC^0 is precisely the class of languages recognized by polynomial-size families of programs over a finite aperiodic monoid (i.e., a monoid that contains no nontrivial groups). Similarly, the class $ACC(m)$ of languages recognized by constant-depth polynomial-size circuits with both unbounded fan-in boolean gates and gates that test membership in L_m, as described in the preceding subsection, is also the class of languages recognized by polynomial-size families of programs over a finite monoid in which every group is solvable of order dividing a power of m. Thus the conjecture that $ACC(m)$ is strictly contained in NC^1 is equivalent to the assertion that nonsolvable groups are essential for Barrington's completeness result [2], and not merely an artifact of the proof.

1.4. *Programs over solvable groups*

In this paper I consider programs over finite *groups*. If G is a nonsolvable group then Barrington's completeness result shows that every language in NC^1 can be recognized by a polynomial-size family of programs over G. The conjecture stated in the preceding subsection implies that no finite *solvable* group has the same property. In fact a stronger conjecture has been formulated: It is believed that no polynomial-size family of programs over a finite solvable group G can recognize the language 1^+—that is, no such family of programs can compute the AND function of its inputs. The present paper is, in essence, a contribution to the study of this conjecture.

Programs over finite solvable groups are equivalent to still another constant-depth circuit model. Here the circuits contain unbounded fan-in gates that determine whether the sum of the incoming bits is divisible by m, but contain no unbounded fan-in AND and OR gates. Thus the conjecture just cited is equivalent to: no polynomial-size constant-depth family of such periodic circuits of period m can compute the AND function. This assertion can be viewed as a kind of dual to the Theorem of Furst, Saxe and Sipser—just as one cannot simulate periodic gates with constant-depth circuits of AND and OR gates, so one cannot simulate AND and OR gates with constant-depth periodic circuits.

The precise definition of this circuit model and the proof of its equivalence with programs over solvable groups will be given in Sec. 2.

1.5. *Known lower bounds*

Programs over solvable groups were extensively investigated in Barrington, Straubing and Thérien [5]. It is shown there that no family of programs over a finite nilpotent group can recognize 1^+. Programs over solvable groups that are not nilpotent can recognize 1^+, but the proof of this uses exponential-size program families. A method based on multidimensional Fourier transforms over finite fields is used to show that

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

if G is an extension of a p-group by an abelian group, then any family of programs over G that recognizes 1^+ must be of exponential size.

In Sec. 3 below these spectral methods are extended to show that no family of programs over a finite dihedral group can recognize 1^+ with less than exponential size. (The result that is obtained is a little weaker than this, since the acceptance condition for the programs must be altered slightly.) As there are dihedral groups that are not extensions of a p-group by an abelian group (for example the dihedral group of order 30), this result enlarges the class of solvable groups for which the conjecture is known to hold.

In Sec. 7, I briefly discuss the computing power of p-groups and show that no family of programs over a finite p-group can recognize 1^+ (this is merely a special case of the theorem on nilpotent groups just cited) or add bits modulo q, where q is a prime different from p.

1.6. *Connections with formal logic*

A theorem of Büchi [8] shows that the regular languages over $\{0,1\}$ are exactly those definable by sentences in the monadic second order theory of linear order, together with a unary predicate π where $\pi(x)$ is interpreted to mean that the bit in position x is on. (Thus formulas are interpreted in strings over $\{0,1\}$, and a sentence defines the language over $\{0,1\}$ consisting of all strings that satisfy the sentence.) McNaughton [17] showed that the languages defined by first-order sentences are exactly the regular languages whose *syntactic monoids* are aperiodic. (The syntactic monoid of a regular language L will be defined precisely in Sec. 5.)

Immerman [14] proved that AC^0 consists of the languages defined by first-order sentences of a different kind: here, in place of the order relation, one is allowed to use arbitrary relations of any arity on the positive integers. The theorem of Barrington and Thérien [6] on AC^0 and aperiodic monoids can easily be derived from a comparison of McNaughton's theorem with this logical characterization of AC^0.

Straubing, Thérien and Thomas [29] studied an extension of first-order logic in which a new class of *modular quantifiers* is used: $\exists^r_d x\phi(x)$ is interpreted to mean 'the number of positions x satisfying $\phi(x)$ is congruent to r modulo d'. They showed that with these quantifiers and the predicates $<$ and π one defines exactly the regular languages whose syntactic monoids are finite solvable groups. If both modular and ordinary quantifiers are used, then the languages obtained are all the regular languages whose syntactic monoids contain no nonsolvable groups. Barrington, Compton, Straubing and Thérien [3] show that the complexity class $ACC(m)$ consists of those languages defined by sentences that use both ordinary quantifiers and modular quantifiers of modulus m, and in which arbitrary numerical predicates are permitted.

Interesting things happen when one varies the class of permitted numerical predicates between the two extremes of the single relation $<$ and that of arbitrary relations on the positive integers. Barrington, Immerman and Straubing [4] consider classes defined with a predicate $BIT(x,y)$ that says that the xth bit of the binary representation of y is on, and show that this gives rise to natural uniform versions of AC^0 and ACC. Denote by **FO**, **MOD(q)** and **MOD** respectively the classes of first-order quantifiers,

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

modular quantifiers of modulus q and all modular quantifiers; and by $\mathcal{N}$ the class of all numerical predicates. Then according to the results cited above,

$$\mathbf{FO}[\pi, <]$$

is the class of aperiodic regular languages. (In this notation, a class of languages is specified by the kinds of quantifiers and the atomic formulas allowed in sentences that define the languages.)

$$\mathbf{MOD}[\pi, <]$$

is the class of regular languages whose syntactic monoids are solvable groups, and

$$(\mathbf{FO} + \mathbf{MOD})[\pi, <]$$

is the class of regular languages whose syntactic monoids contain no nonsolvable groups.

$$\mathbf{FO}[\pi, \mathcal{N}]$$

is AC^0, and

$$(\mathbf{FO} + \mathbf{MOD(q)})[\pi, \mathcal{N}]$$

is $ACC(q)$.

Section 4 contains precise definitions of the logical framework used in this paper, and introduces the class $\mathcal{R}$ of *regular numerical predicates*—those that can be recognized by finite automata. The principal results of [3] can then be stated as

$$\mathbf{FO}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{FO}[\pi, \mathcal{R}],$$

where **Reg** denotes the class of regular languages over $\{0, 1\}$. It is conjectured that

$$(\mathbf{FO} + \mathbf{MOD(q)})[\pi, \mathcal{N}] \cap \mathbf{Reg} = (\mathbf{FO} + \mathbf{MOD(q)})[\pi, \mathcal{R}].$$

This equality, known to hold when q is a prime power, is equivalent to the conjecture that the bit sum modulo p cannot be computed in $ACC(q)$ if p is a prime that does not divide q. There appears to be a general principle at work whereby numerical predicates appearing in a formula that defines a regular language can be replaced by regular numerical predicates. Since formulas that use regular numerical predicates correspond to highly uniform circuit families, these results indicate that nonuniformity does not play a significant role in the circuit family's computational power, at least insofar as recognition of regular languages is concerned.

A precise formulation and an independent proof of the general principle on replacing arbitrary numerical predicates by regular numerical predicates would give a new proof

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

of the theorem of Furst, Saxe and Sipser and prove that $ACC(q)$ is properly contained in NC^1 for all $q > 0$.

In Sec. 5 this logical formalism is extended to apply to programs over solvable groups. It will be shown there that

$$\mathbf{MOD(q)}[\pi, \mathcal{N}]$$

is the family of languages recognized by polynomial-size families of solvable groups whose orders divide a power of q. Furthermore, it will be shown that the equation

$$\mathbf{MOD(q)}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{MOD(q)}[\pi, \mathcal{R}]$$

is equivalent to the conjecture that 1^+ cannot be recognized by a polynomial-size family of programs over a finite solvable group. As this equation is another instance of the general principle cited above, this result provides additional evidence for the conjectured limitation on the computing power of programs over solvable groups.

The proof of a somewhat weaker version of this last result is given in Sec. 5. The complete proof requires a considerable detour through semigroup theory, and depends upon a new theorem, of independent interest, concerning the algebraic structure of finite categories and semigroupoids. These details are presented in Sec. 6.

2. Constant-depth Periodic Circuits and Programs over Solvable Groups

2.1. *Programs over finite monoids*

Let M be a finite monoid; that is, M is a finite set with an associative multiplication and a multiplicative identity 1. Let $n > 0$. A *program* π over M with n inputs consists of a sequence of pairs

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

(where $1 \leq i_j \leq n$ and f_j is a function from $\{0,1\}$ into M for all j such that $1 \leq j \leq r$) together with a subset X of M. The pairs are called *instructions*, the set X is called the set of *accepting values* and the integer r is called the *size* or the *length* of the program. If $w = a_1 \cdots a_n \in \{0,1\}^n$ then the *value* of the program on w is the element

$$\prod_{j=1}^{r} f_j(a_{i_j})$$

of M. The string w is *accepted* by π if and only if this value belongs to X. The set of all strings accepted by π is denoted $L(\pi)$.

Ordinarily one considers a family of programs $\{\pi_n : n > 0\}$, indexed by the number n of inputs, over a single finite monoid M. Such a family thus accepts, or *recognizes* a language $L = \bigcup_{n>0} L(\pi_n)$. As any language can be accepted by a family of programs over a very simple monoid if the program size is allowed to grow exponentially with

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

the number of inputs, one is usually interested in families in which the length of π_n is bounded above by a polynomial in n.

It is easy to show that if a language $L \subseteq \{0,1\}^+$ is recognized by a polynomial-length family of programs over a finite monoid, then it can also be recognized by a family of bounded fan-in boolean circuits of depth $O(\log n)$; that is, every such language belongs to the complexity class NC^1. It is a remarkable fact, discovered by Barrington [2], that if M contains a nonsolvable group, then every language in NC^1 is accepted by a polynomial-size family of programs over M.

Most of the rest of this paper will be devoted to programs over finite groups. One of the principal goals of the current research is to show that Barrington's result cannot be extended to include any solvable groups: that is, to show that there are languages in NC^1 that cannot be recognized by any polynomial-size family of programs over a finite solvable group.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

2.2. *Equivalence of the circuit and program models*

For the purposes of this paper, a *circuit* with n inputs is a directed acyclic graph with $2n$ source nodes and one sink node. The *size* of the circuit is the number of nodes, and the *depth* of the circuit is the length of the longest path from a source to the sink. To each node v (also called a *gate*) is associated a function $f_v : \{0,1\}^k \to \{0,1\}$, where k is the number of edges entering the node. Given an input string $a_1 \cdots a_n \in \{0,1\}^n$, the circuit computes an output value as follows: The $2n$ source nodes are initially assigned the values $(a_1, 1 - a_1, \ldots, a_n, 1 - a_n)$. If the origins of the k edges entering a node v have been assigned the values $b_1, \ldots, b_k$, then v is assigned the value $f_v(b_1, \ldots, b_k)$. (This appears to presume an ordering on the edges entering a node, however all of the functions f_v considered in this paper are symmetric.) Because of the acyclicity, this results in a unique boolean value assigned to the sink node. The input $a_1 \cdots a_n$ is *accepted* if and only if this value is 1.

Occasionally it will be necessary, in the course of constructing a circuit, to consider circuits with $k > 1$ sink nodes. Such a circuit computes a function from $\{0,1\}^n$ to $\{0,1\}^k$.

As was the case with programs, one usually considers families $\{\mathscr{C}_n : n > 0\}$ of circuits indexed by the number of inputs. Such a family thus accepts, or recognizes, a language $L \subseteq \{0,1\}^+$.

Here I shall consider several different sorts of gates:

Boolean gates. These compute the k-ary functions AND_k, $OR_k : \{0,1\}^k \to \{0,1\}$ defined by

$$AND_k(b_1, \ldots, b_k) = b_1 \cdots b_k$$

and

$$OR_k(b_1, \ldots, b_k) = 1 - AND_k(1 - b_1, \ldots, 1 - b_k).$$

Periodic gates. These compute the functions

$$MOD_{m,k} : \{0,1\}^k \to \{0,1\}$$

defined by $MOD_{m,k}(b_1,\ldots,b_k) = 1$ if and only if $m|b_1 + \cdots + b_k$. The integer m is called the *period* of the gate.

The *fan-in* of a gate is the number of edges entering the gate. NC^1 denotes the class of languages recognized by polynomial-size, $O(\log n)$ depth families of circuits made up of boolean gates of fan-in 2. AC^0 is the class of languages recognized by polynomial-size, constant-depth families of circuits made up of boolean gates. $ACC(q)$ is the family of languages accepted by polynomial-size constant-depth families of circuits made up of boolean gates and periodic gates of period q. ACC is the union of the $ACC(q)$ over all $q > 1$. Since it is easy to simulate a boolean gate or a periodic gate with fan-in k by a boolean circuit of depth $O(\log k)$ in which every gate has fan-in 2, it follows that for all $q > 1$, $AC^0 \subseteq ACC(q) \subseteq NC^1$.

A *periodic circuit* of period q is one made up of periodic gates of period q and boolean gates with fan-in 2. $CC(q)$ denotes the class of languages recognized by constant-depth, polynomial-size families of periodic circuits of period q. CC denotes the union of the $CC(q)$ over all $q > 1$.

Clearly $CC(q) \subseteq ACC(q)$. If q is a prime power, then a result of Smolensky [23] shows that $ACC(q) \neq NC^1$, and, as will be seen in Sec. 7, $CC(q) \neq ACC(q)$. If q has two distinct prime factors then it is an open question whether either of these two inclusions is strict.

The circuit families used to construct the class $CC(q)$ should really be thought of as consisting solely of periodic gates of period q. Indeed, without changing the class of languages defined, one can alter the model so that all the boolean gates appear at the level of the inputs, or even eliminate the boolean gates altogether by putting more power in the communication between gates. A variant on this circuit model is discussed in Straubing and Thérien [28].

Periodic gates of period m as described here emit 1 if and only if the sum of the inputs is divisible by m. One can simulate gates that emit 1 if and only if the bit sum is congruent to k modulo m, for $0 \leq k < m$, by feeding $m - k$ additional 1's into the gate—this can be accomplished with a fixed amount of boolean circuitry, since the constant 1 is realized by $x \vee \bar{x}$, where x is any input bit. Thus a bank of m periodic gates of period m, together with $O(m^2)$ boolean gates of fan-in 2, serves to compute sums in $\mathbf{Z}_m$.

Furthermore, one can simulate a periodic gate of period m^k and fan-in f by a periodic circuit of period m, depth $O(k)$ and size $O(f^{k-1})$. To accomplish this, given $a_1, \ldots, a_f \in \{0,1\}$, use f gates of period m^{k-1} to determine whether $b_1 = a_1, b_2 = a_1 + a_2, \ldots, b_f = a_1 + \cdots + a_f$ are divisible by m^{k-1}, and then determine whether the number of pairs $(i, i+1)$, such that m^{k-1} does not divide b_i and m^{k-1} divides b_{i+1} is divisible by m. The resulting circuit contains a level of gates of period m^{k-1}, a level of fan-in 2 boolean gates, and a level consisting of a single gate of period m. The desired result follows by applying this construction recursively to the gates of period m^{k-1}. In particular, this proves:

2.2.1. Proposition. *For all $m, k > 1$, $CC(m) = CC(m^k)$.*

The principal result of this section is:

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

2.2.2. Theorem. *Let $L \subseteq \{0,1\}^+$, $q > 0$. $L \in CC(q)$ if and only if L is accepted by a polynomial-size family of programs over a finite solvable group whose order divides a power of q.*

Proof. I shall first show that if $L \subseteq \{0,1\}^n$ is accepted by a program of length r over a finite solvable group G whose order divides a power of q, then L is accepted by a periodic circuit of period q whose depth depends only on $|G|$ and whose size is bounded above by a polynomial in r. This immediately gives one direction of the theorem.

The proof is by induction on $|G|$. If $|G| = 1$ there is nothing to prove. Otherwise, G contains a normal subgroup N such that $|N| < |G|$ and G/N is abelian. Let $g_1, \ldots, g_k$ be a complete set of coset representatives of N. The representative of a coset of $g \in G$ will be denoted $\bar{g}$.

Given an input string in $\{0,1\}^n$, the jth instruction of the program emits an element $n_j g_{i_j}$ of G, where $n_j \in N$. The program is supposed to compute the product

$$n_1 g_{i_1} \cdots n_r g_{i_r}.$$

To accomplish this, the circuit will compute values $n_2', n_3', \ldots, n_{r+1}'$ such that

$$g_{i_1} n_2 = n_2' g_{i_1},$$

$$\overline{g_{i_1} g_{i_2}} n_3 = n_3' \overline{g_{i_1} g_{i_2}},$$

$$\vdots$$

$$\overline{g_{i_1} \cdots g_{i_{r-1}}} n_r = n_r' \overline{g_{i_1} \cdots g_{i_{r-1}}},$$

$$\overline{g_{i_1} \cdots g_{i_{r-1}} g_{i_r}} = n_{r+1}' \overline{g_{i_1} \cdots g_{i_r}}.$$

This done, the product to be computed can be rewritten as

$$n_1 n_2' \cdots n_{r+1}' \overline{g_{i_1} \cdots g_{i_r}}.$$

The circuit will be built in four levels. The first is a constant-depth layer of periodic gates of period q that computes the values $g_{i_1}, \overline{g_{i_1} g_{i_2}}, \ldots, \overline{g_{i_1} \cdots g_{i_r}}$. The second is a layer of boolean circuits that computes $n_2', \ldots, n_{r+1}'$. The third layer computes the product $n = n_1 n_2' \cdots n_{r+1}' \in N$ and the fourth determines whether the product $n\overline{g_1} \cdots \overline{g_r}$ belongs to the accepting set X of the program. These layers of circuitry have only binary inputs and outputs, so that when one says they 'compute' elements of G what is meant is that the output is a binary encoding of elements of G. Any encoding can be used, but it is simplest to represent an element of G by a $|G|$-tuple of bits that has a 1 in one position and 0 elsewhere.

In the first layer, the values $g_{i_1}, \ldots, \overline{g_{i_1} \cdots g_{i_{r-1}}}$ are computed by working in G/N. But G/N is a quotient of a direct product of t copies of $\mathbf{Z}_{q^s}$, and a periodic gate of period

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

q^s can be simulated by a periodic circuit of period q whose depth is $O(s)$ and whose size is polynomial in the fan-in. Thus $r-1$ separate periodic circuits of period q, with constant depth and size polynomial in r, can compute binary encodings of the required values.

In the second layer, given encodings of g_i, $n \in N$, a constant-size boolean circuit suffices to compute n' such that $g_i n = n' g_i$. Similarly, given encodings of g_i and g_j, a constant-size boolean circuit suffices to compute $n \in N$ such that $g_i g_j = n' \overline{g_i g_j}$. So r boolean circuits of constant size following the first layer are sufficient to compute $n_2', \ldots, n_r', n_{r+1}'$.

Third, since $|N| < |G|$, it follows from the inductive hypothesis that there is a constant-depth periodic circuit of period q whose size is polynomial in r that, given encodings of $n_1, n_2', \ldots, n_{r+1}'$ computes $n_1 n_2' \ldots n_{r+1}'$. This is because each bit of the encoding of an element of N can be viewed as an instruction that emits $1 \in N$ if the bit is off and the encoded element if the bit is on. The fourth layer is a constant-size boolean circuit that, given encodings of $n \in N$ and g_i determines whether ng_i belongs to the accepting set X. By piecing together the four levels one obtains the desired circuit.

For the converse direction, I shall show by induction on k that there is a finite solvable group G_k whose order divides a power of q such that any $L \subseteq \{0,1\}^n$ accepted by a depth k periodic circuit of period q is also accepted by a program over G_k, where the program length is a polynomial function of the circuit size.

If the depth of the circuit is 0, then $L = \{a_1 \cdots a_n : a_i = 1\}$ or its complement. This set is accepted by a program over $\mathbf{Z}_q$ with a single instruction.

Now suppose that the assertion is true for circuits of depth $k-1$. Consider a circuit $\mathscr{C}$ of depth k. There are three cases to treat: The gate at the sink node can be an *AND* or an *OR* gate of fan-in 2, or a periodic gate of period q. In the case of an *AND* gate, one can view $\mathscr{C}$ as consisting of two disjoint circuits $\mathscr{C}_1$ and $\mathscr{C}_2$ of depth less than k each feeding its output into the *AND* gate. (Converting $\mathscr{C}$ to a circuit of this form will at most double the size and leave the depth unchanged.) By the inductive hypothesis, the sets L_1 and L_2 accepted by these circuits are also accepted by programs π_1 and π_2 over G_{k-1}. Thus $L_1 \cap L_2$, which is the set accepted by $\mathscr{C}$ is accepted by a program π over the direct product $G_{k-1} \times G_{k-1}$. The program π mimics the instructions of π_1 in the first component and emits 1 in the second component, then emits 1 in the first component and mimics π_2 in the second component. The input is accepted if the resulting value has an accepting value for π_1 in the first component and an accepting value for π_2 in the second component. The lengh of π is the sum of the lengths of π_1 and π_2. The case of an *OR* gate at the output is treated identically.

For the case of a periodic gate at the output, $\mathscr{C}$ can be replaced by a circuit consisting of r disjoint subcircuits $\mathscr{C}_1, \ldots, \mathscr{C}_r$ of depth less than k each feeding its output into the periodic gate. The number r and the size of each $\mathscr{C}_i$ are bounded above by the size of the original circuit $\mathscr{C}$. By the inductive hypothesis, the set L_i accepted by $\mathscr{C}_i$ is also accepted by a program π_i over G_{k-1} with accepting set X_i. The lengths of the π_i are uniformly bounded above by a polynomial in the size of $\mathscr{C}$. The circuit $\mathscr{C}$ accepts the set L consisting of those strings that belong to exactly kq of the L_i, for some $k \geq 0$.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

I shall show that L is accepted by a program π of length no more than $(le + 1)r$ over the wreath product $\mathbf{Z}_q \circ G_{k-1}$. Here l is the maximum of the lengths of the π_i, and e is the exponent of G_{k-1}. The elements of the wreath product are pairs (f, g), where $g \in G_{k-1}$ and f is a map from G_{k-1} into $\mathbf{Z}_q$. The multiplication is given by

$$(f_1, g_1)(f_2, g_2) = (f, g_1 g_2),$$

where $f(g) = f_1(g) \cdot f_2(gg_1)$ for all $g \in G_{k-1}$. It is easy to verify that this multiplication makes $\mathbf{Z}_q \circ G_{k-1}$ a group. Since the kernel of the homomorphism $(f, g) \mapsto g$ is an abelian group of exponent q, $\mathbf{Z}_q \circ G_{k-1}$ is a solvable group whose order divides a power of q.

The sequence of instructions of π is partitioned into r blocks. The first $|\pi_j|$ instructions of the jth block simply mimic the instructions of π_j in the right co-ordinate, and, in the left co-ordinate, emit the function $I : G_{k-1} \to \mathbf{Z}_q$, where $I(g) = 0$ for all $g \in G_{k-1}$. The $(|\pi_j| + 1)$th instruction of the jth block emits $(f_j, 1)$ where $f_j(g) = 1$ if $g \in X_j$ and is 0 otherwise. (Formally one must define this instruction so that it 'consults' one of the bits of the input, although the value emitted is independent of the input.) The remaining instructions of the jth block are identical to the first π_j instructions, repeated $e - 1$ times. It is readily verified that the value of π on a given input is $(h, 1)$, with $h(0) = f_1(v_1) + \cdots + f_r(v_r)$, where v_j is the value of π_j on the input. Thus the desired program is obtained by defining the accepting set of π to be $X = \{(f, 1) : f(0) = 0\}$.

The proof of the theorem is now completed by setting $G_k = (\mathbf{Z}_q \circ G_{k-1}) \times G_{k-1}$, which has both $\mathbf{Z}_q \circ G_{k-1}$ and $G_{k-1} \times G_{k-1}$ as quotients. □

2.3. *The fundamental conjectures*

At present, work on the complexity class CC is directed toward the proof of several conjectures, each of which implies that CC is properly contained in NC^1.

2.3.1. Conjecture. For all $q > 1$, $1^+ \notin CC(q)$.

2.3.2. Conjecture. If p is a prime not dividing q, then the language

$$L_p = \left\{ a_1 \cdots a_k \in \{0,1\}^+ : \sum_{1 \le i \le k} a_i \equiv 0 \pmod{p} \right\}$$

is not in $CC(q)$.

By Theorem 2.2.2, the conjecture 2.3.1 is equivalent to asserting that no polynomial size family of programs over a finite solvable group can recognize 1^+. In the next section I shall extend the class of solvable groups for which this conjecture is known to hold. The following two sections will be devoted to a reformulation of the same conjectures in terms of logical descriptions of regular languages. The reformulated conjectures follow from a (not yet proved) general principle for which there is considerable evidence.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

3. Programs over Dihedral Groups

3.1. *Previous results obtained using spectral methods*

Let F be a finite field, and let F^* be its group of units. As is well-known, F^* is a cyclic group; let g be a generator of this group. Barrington, Straubing and Thérien [5] considered the boolean function computed by a program over a solvable group as a function from $\{1, g\}^*$ into F, where F is an appropriately chosen finite field. They showed that for a large class of solvable groups, the size of the program is polynomially related to the number of non-zero terms in the representation of this function with respect to a particular basis for the space of functions from $(F^*)^n$ into F. The proof that the *AND* function could not be computed by a polynomial-size program was completed by estimating the number of non-zero terms in its representation with respect to this basis. Since this number is difficult to estimate directly, it was necessary to consider a change of basis. This change of basis is essentially a multidimensional Fourier transform over F.

This works for the class of groups consisting of all G with a normal subgroup N such that N is a p-group and G/N is abelian. A distinction must be made between two acceptance conditions. A program over a monoid is said to *strongly recognize* a set L of strings if there are two distinct elements m_1, m_2 of M such that the program has the value m_1 on all strings in L and m_2 on all strings not in L. The usual acceptance condition will be called *weak recognition.* When the value of the program can be represented by a function with values in a field, as in [5], weak recognition implies strong recognition with only a polynomial blow-up in program size. The relationship between weak and strong recognition in general is unknown (see Sec. 7).

3.2. *Dihedral groups*

D_r, the dihedral group of degree r, is defined by the presentation

$$\langle g, h : g^2 = 1, h^r = 1, gh = h^{-1}g \rangle.$$

It follows that $|D_r| = 2r$ and that D_r consists of the elements

$$\{g^i h^j : 0 \leq i < 2, 0 \leq j < r\}.$$

I shall write h^c additively, as $c \cdot h$, with $c \in \mathbf{Z}_r$.

Observe that there are dihedral groups that are not extensions of normal p-subgroups by abelian groups. Consider, for example, D_{15}. It is easy to see, by conjugation with h, that no proper normal subgroup of D_{15} can contain an element of the form gh^i. Thus the only normal p-subgroups are the subgroups of orders 3 and 5 generated by h^5 and h^3, respectively. The elements g and h do not commute modulo either of these two subgroups, so in this case the quotient groups are not abelian.

In light of this, the next result represents an advance over previous work. The

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

acceptance condition, however, has had to be altered, and thus the theorem is somewhat weaker than one would desire.

3.2.1. Theorem. *Any program over D_r that strongly recognizes $\{1^n\}$ has size exponential in n.*

Proof. It can be supposed that r is not a power of 2, since the theorem is already known to hold for 2-groups. Let $\mathbf{Z}'_r$ denote the extension of $\mathbf{Z}_r$ obtained by adjoining, if none exists already, an element $\frac{1}{2}$ such that $\frac{1}{2} + \frac{1}{2} = 1$. If r is odd then such an element already exists, so $\mathbf{Z}'_r = \mathbf{Z}_r$. If r is even, then $\mathbf{Z}'_r = \mathbf{Z}_r \cup \{k + \frac{1}{2} : k \in \mathbf{Z}_r\}$ is a cyclic group of order $2r$ with $\frac{1}{2}$ as a generator.

For each $I \subseteq \{1, \ldots, n\}$ consider the function

$$f_I : \{0,1\}^n \to \mathbf{Z}'_r$$

defined by

$$f_I(x_1, \ldots, x_n) = (-1)^{\sum_{i \in I} x_i}.$$

Consider a program of length M over D_r. A given instruction of the program consults bit x_j of the input string $x_1 \ldots x_n$ and emits the group element

$$g^{c_1 x_j + c_2} \cdot \{(ax_j + b) \cdot h\}$$

where $c_1, c_2 \in \{0,1\}$; $a, b \in \mathbf{Z}_r$. The commutation rules can be used to rewrite this as

$$(-1)^{c_1 x_j + c_2}(ax_j + b) \cdot h \cdot g^{c_1 x_j + c_2}.$$

The commutation relations are applied repeatedly to move all the g's to the right, so the value of the program can be written as

$$\{\alpha(x_1, \ldots, x_n) \cdot h\} g^{d + \sum_{j \in J} x_j}$$

for some $d \in \{0,1\}$ and $J \subseteq \{1, \ldots, n\}$, where α is a $\mathbf{Z}_r$-linear combination of no more than M terms of the form

$$f_I(x_1, \ldots, x_n) \cdot (ax_j + b).$$

Now $x_j = \frac{1}{2} - \frac{1}{2}(-1)^{x_j}$, so α is a $\mathbf{Z}'_r$-linear combination of no more than $2M$ of the f_I. Suppose now that the program strongly recognizes the set $\{1^n\}$. Suppose further that $|J|$ is even and $d = 0$ (the argument is similar in the other cases). Let $c = \alpha(1, \ldots, 1)$. Then the program has the value

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$c \cdot h$$

at $(1,\ldots,1)$ and

$$\{e \cdot h\} \cdot g$$

on all n-tuples $(x_1,\ldots,x_n)$ such that $|J \cap \{i : x_i = 1\}|$ is odd. If there is an n-tuple $(x_1,\ldots,x_n) \neq (1,\ldots,1)$ such that $|J \cap \{i : x_i = 1\}|$ is even, then since the program strongly recognizes $\{1^n\}$, it must give the value $\{e \cdot h\} \cdot g$ on this input; however the general form of the program value shows that this is impossible. Thus J is empty, and the function α itself strongly recognizes $\{1^n\}$ in the sense that $\alpha(1^n) = c$ and $\alpha(1^n) = e \neq c$ for all other $(x_1,\ldots,x_n)$.

I shall now show that no such function can be written as a $\mathbf{Z}_r'$-linear combination of the f_I with fewer than $2^n - 1$ terms, and thus the program length must be exponential in n. If $\beta : \{0,1\}^n \to \mathbf{Z}_r'$, I define $T\beta : \{0,1\}^n \to \mathbf{Z}_r'$ by

$$T\beta(x_1,\ldots,x_n) = \sum_{(y_1,\ldots,y_n) \in \{0,1\}^n} (-1)^{x_1y_1 + \cdots + x_ny_n} \beta(y_1,\ldots,y_n).$$

If $\beta(1,\ldots,1) = a \neq 0$ and $\beta(b_1,\ldots,b_n) = 0$ for $(b_1,\ldots,b_n) \neq (1,\ldots,1)$, then $T\beta(x_1,\ldots,x_n)$ is everywhere nonzero. On the other hand,

$$Tf_I(x_1,\ldots,x_n) = \sum_{(y_1,\ldots,y_n) \in \{0,1\}^n} f_I(y_1,\ldots,y_n) f_J(y_1,\ldots,y_n)$$

where $J = \{i : x_i = 0\}$. If $I = J$, then this sum is 2^n, which is not zero, by the assumption that r is not a power of 2. If $I \neq J$ then there exists $i \in (I \backslash J) \cup (J \backslash I)$, so for each term $f_I(y_1,\ldots,y_n) f_J(y_1,\ldots,y_n)$ there is a term of opposite sign obtained by reversing the bit y_i; thus the sum is zero. Hence Tf_I is nonzero at exactly one point. Now let $\alpha(1,\ldots,1) = c$, and $\alpha(b_1,\ldots,b_n) = e \neq c$ for $(b_1,\ldots,b_n) \neq (1,\ldots,1)$. If α is a $\mathbf{Z}_r'$-linear combination of the f_I with M nonzero terms, then $\alpha - c$ can be so expressed with at most $M + 1$ terms. By the above calculation and the linearity of T, it follows that $T(\alpha - c)$ is nonzero at at most $M + 1$ points, but the previous argument showed that $T(\alpha - c)$ is everywhere nonzero, so $M \geq 2^n - 1$. □

4. Defining Formal Languages in Extensions of First-order Logic

4.1. *The logical apparatus*

Let $\{x_1,\ldots,x_k\}$ be set of variables. The logical formulas discussed here are built starting from a basis of two kinds of atomic formulas: For each $i = 1,\ldots,k$,

$$\pi(x_i)$$

is an atomic formula. The other kind of atomic formula is called a *numerical predicate*: A *variable string* is a string $S_1 \cdots S_r$ over the alphabet $2^{\{x_1,\ldots,x_k\}}$ such that $S_i \cap S_j = \varnothing$

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

whenever $i \neq j$. The set $\bigcup_{1 \leq i \leq r} S_i$ is called the set of *free variables* of the string. A *numerical predicate* is a set of variable strings all having the same underlying set of free variables.

If ϕ and ψ are formulas having the same set T of free variables, then $\neg\phi$ and $\phi \wedge \psi$ are formulas with T as the set of free variables. If ϕ is a formula whose set V of free variables contains the variable x, then $\exists x\phi$, and $\exists_d^r x\phi$ for $0 \leq r < d$, are formulas whose set of free variables is $V \backslash \{x\}$. The other boolean connectives $\vee$, $\Rightarrow$ and the universal quantifier $\forall x_i$ can be defined in the usual fashion in terms of these others.

The semantics of these logical formulas was discussed informally in Sec. 1. Variables denote positions in a bit string. Numerical predicates give information about the positions and the length of the string in which the formula is interpreted, but say nothing about what bits appear in these positions. The π predicate is used to say that the bit in a certain position is on. Here is a precise account of the semantics: A *word structure* is a string $v = (a_1, S_1)\cdots(a_m, S_m)$ over the alphabet $\{0,1\} \times 2^{\{x_1,\ldots,x_k\}}$ such that $S_i \cap S_j = \varnothing$ whenever $i \neq j$. $\hat{v}$ denotes the variable string $S_1 \cdots S_m$. I define

$$v \models \pi(x_i)$$

if and only if for some $j = 1, \ldots, m$, $x_i \in S_j$ and $a_i = 1$. If N is a numerical predicate then

$$v \models N$$

if and only if $\hat{v} \in N$. Boolean connectives are interpreted in the usual manner.

$$v \models \exists x_i \psi$$

if and only if for some $j = 1, \ldots, m$

$$(a_1, S_1)\cdots(a_j, S_j \cup \{x_i\})\cdots(a_m, S_m) \models \psi.$$

$$v \models \exists_d^r x_i \psi$$

if and only if

$$|\{j : (a_1, S_1)\cdots(a_j, S_j \cup \{x_i\})\cdots(a_m, S_m) \models \psi\}| \equiv r \pmod{d}.$$

A string $w = a_1 \ldots a_m \in \{0,1\}^*$ is identified with the word structure $(a_1, \varnothing)\ldots(a_m, \varnothing)$. If ϕ is a *sentence*, that is, a formula without free variables, then ϕ defines the language

$$L_\phi = \{w \in \{0,1\}^+ : w \models \phi\}.$$

4.1.1. Examples.

(a) $\exists_2^0 x\pi(x)$ defines the set of strings containing an even number of 1s.

(b) $\psi = (\varnothing^2)^*$ is a numerical predicate without free variables (and is thus a sentence). L_ψ is the set of strings of even length.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

(c) It is more common, of course, to write numerical predicates with ordinary relation symbols. Thus $x_i < x_j$ is a shorthand for the numerical predicate consisting of all variable strings of the form *uavbw* with $u, v\, w \in (2^{\{x_1,\ldots,x_k\}})^*$, $a, b \in 2^{\{x_1,\ldots,x_k\}}$ and $x_i \in a$, $x_j \in b$. Similarly $x_i \equiv 0 \pmod 2$ denotes the numerical predicate consisting of all variable strings of the form *uav* where $|u|$ is odd and $x_i \in a$. Observe that in the usual notation, the set of free variables in the formula is inferred from the context (i.e., there may be free variables other than those that appear in the formula) whereas in the formalism of variable strings and numerical predicates, the set of free variables is given explicitly. The set of strings over $\{0, 1\}$ of even length is thus also defined by the sentence

$$\exists x((x \equiv 0 \pmod 2) \wedge \forall y(\neg(x < y))).$$

4.1.2. Remarks.

(a) It might seem more natural to define a regular numerical predicate to be a subset of $\mathbf{N}^k$, where $\mathbf{N}$ denotes the set of natural numbers. This is the approach taken by Peladeau [19], who presents a detailed study from this standpoint of the 'regular numerical predicates' of the next subsection. The present approach has been adopted for two reasons: First, by interpreting numerical predicates as sets of strings, those numerical predicates that give rise to definitions of regular languages turn out to be themselves regular languages. Second, it is desirable to be able to include information about the length of the string in the numerical predicate; in this way the nonuniform nature of the model can be pushed down to a single atomic formula.

(b) I have confined myself to languages over a binary alphabet, but the same logical apparatus can be used to define languages over an arbitrary alphabet A. In this case the predicate π is replaced by a family of predicates $\{Q_a : a \in A\}$, where $Q_a(x)$ is interpreted to mean 'the letter in position x is a'.

4.2. *Regular numerical predicates*

Since a numerical predicate is formally simply a set of strings, one can talk about the language-theoretic properties of the predicate. A numerical predicate is thus said to be *regular* if it is a regular language. All the numerical predicates discussed in Example 4.1.1 are regular. In particular, part (c) of the example illustrates the following fundamental fact.

4.2.1. Theorem. *A numerical predicate is regular if and only if it is equivalent to a first-order formula in the atomic formulas*

$$x_i < x_j$$

and

$$x_i \equiv 0 \pmod d$$

for $d > 1$.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

Proof. (*if*) Example 4.1.1 (c) shows that the numerical predicates $x_i < x_j$ and $x_i \equiv 0 \pmod{d}$ are regular. If R, S are numerical predicates with the same underlying set of free variables, then $R \wedge S$ is $R \cap S$, and $\neg R$ is $L \backslash R$, where L is the set of all variable strings whose underlying set of free variables is the same as that of R. Since the class of regular languages is closed under complementation and intersection, and since L is a regular language, it follows that the class of regular numerical predicates is closed under boolean operations. If suffices to show that if R is a regular numerical predicate in which the variable x is free, then $\exists x R$ is regular.

Let $\mathscr{A}$ be an automaton that accepts R. Replace each edge

$$q \xrightarrow{S} q'$$

of $\mathscr{A}$ such that $x \in S$ by the edge

$$q \xrightarrow{S \setminus \{x\}} q'.$$

The result is a nondeterministic automaton $\mathscr{A}'$. The initial and final states of $\mathscr{A}'$ are the same as those of $\mathscr{A}$. Let T consist of those w accepted by $\mathscr{A}'$ such that in some successful path labelled by w, exactly one edge is one of these altered edges. It is easy to see that T is a regular language (indeed the same operation can be performed on any finite automaton with respect to any set of edges, and yields a regular language) and that $T = \exists x R$.

(*only if*) Let R be a regular numerical predicate and let $\mathscr{A}$ be an automaton that accepts R. For each variable string $w \in R$, consider the sequence θ_w of the edges of $\mathscr{A}$ traversed by w that are labelled by nonempty elements of $2^{\{x_1,\ldots,x_k\}}$:

$$\theta_w = (q_1 \xrightarrow{S_1} q_1', \ldots, q_r \xrightarrow{S_r} q_r').$$

Since there are only finitely many such sequences, R is a finite union of languages of the form

$$L(\theta_w) = L_0 S_1 L_1 \cdots S_r L_r,$$

where L_0 is the set of strings over the alphabet $\{\varnothing\}$ that lead from an initial state of $\mathscr{A}$ to q_1; L_i, for $1 \leq i < r$, is the set of strings over $\{\varnothing\}$ leading from q_i' to q_{i+1}, and L_r is the set of strings over $\{\varnothing\}$ leading from q_r' to a final state of $\mathscr{A}$. If suffices to show that each of the $L(\theta_w)$ can be represented by a formula of the required sort; R is then represented by the disjunction of these formulas.

Every regular language over the unary alphabet $\{\varnothing\}$ is a finite union of languages of the form $\varnothing^s$ and $(\varnothing^t)^*\varnothing^s$ for $s, t \geq 0$. Thus it suffices to show that each language of the form

$$(\varnothing^{t_0})^*\varnothing^{s_0} S_1 \cdots S_r (\varnothing^{t_r})^*\varnothing^{s_r}$$

is defined by an appropriate formula. Indeed, this language is defined by the formula

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$\exists x \exists y_1 \cdots \exists y_r (\alpha_1 \wedge \alpha_2 \wedge \alpha_3),$$

where α_1 is

$$\forall z(z \leq x) \wedge (y_1 > s_0) \wedge (y_2 - y_1 > s_1) \wedge \cdots \wedge (y_r - y_{r-1} > s_{r-1}) \wedge (x - y_r \geq s_r),$$

α_2 is

$$(y_1 \equiv s_0 + 1 \,(\mathrm{mod}\, t_0)) \wedge \cdots \wedge (y_r - y_{r-1} \equiv s_{r-1} + 1 \,(\mathrm{mod}\, t_{r-1}))$$

$$\wedge (x - y_r \equiv s_r \,(\mathrm{mod}\, t_r)),$$

and α_3 is

$$\bigwedge_{j=1}^{r} \bigwedge_{v \in S_j} (v = y_j).$$

It remains to show how to express the atoms of these three formulas in terms of the allowed predicates. $x = y$ can be expressed as

$$\neg(y < x) \wedge \neg(x < y).$$

$y - x > c$, where c is a positive constant, can be expressed as

$$\exists v_1 \ldots \exists v_c (x < v_1 \wedge v_1 < v_2 \wedge \cdots v_{c-1} < v_c \wedge v_c \leq y).$$

$y - x \equiv c \,(\mathrm{mod}\, t)$ can be expressed as

$$\bigvee_{m \in \mathbf{Z}_t} ((x \equiv m \,(\mathrm{mod}\, t)) \wedge (y \equiv m + c \,(\mathrm{mod}\, t))).$$

Finally, if $1 \leq m < t$, then

$$x \equiv m \qquad (\mathrm{mod}\, t)$$

can be written as

$$\exists y_0 \ldots \exists y_{m-1} ((y_0 \equiv 0 \,(\mathrm{mod}\, t))$$

$$\wedge (y_1 = y_0 + 1 \wedge \cdots \wedge y_{m-1} = y_{m-2} + 1 \wedge x = y_{m-1} + 1)),$$

where $u = v + 1$ is expressed as

$$u < v \wedge \forall w(u < w \Rightarrow \neg(v > w)).$$

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

4.3. *Connection with earlier work*

Let $\mathcal{N}$ denote the class of all numerical predicates, let $\mathcal{R}$ denote the class of all regular numerical predicates, and let **Reg** denote the class of regular languages over the alphabet $\{0, 1\}$. If **Q** is a class of quantifiers, and F a collection of quantifier-free formulas, then

$$\mathbf{Q}[F]$$

denotes the class of languages defined by sentences built using formulas from F as atoms and quantifiers from **Q**. Let **FO** denote the class consisting of the existential quantifier alone, **MOD(q)** the class of modular quantifiers with modulus q, **MOD** the class of all modular quantifiers, and **FO** + **MOD(q)**, **FO** + **MOD** the respective unions of these classes. According to a result of Immerman [14],

$$AC^0 = \mathbf{FO}[\pi, \mathcal{N}].$$

Barrington, Compton, Straubing and Thérien [3] used the lower-bounds theorem of Furst, Saxe and Sipser to show

$$AC^0 \cap \mathbf{Reg} = \mathbf{FO}[\pi, <, \{x \equiv 0 \,(\mathrm{mod}\, d) : d > 0\}],$$

and thus, by Theorem 4.2.1,

$$\mathbf{FO}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{FO}[\pi, \mathcal{R}].$$

I believe that this last equality can be demonstrated using purely qualitative considerations. That is, one ought to be able to show, independently of the circuit lower bounds, that if the language defined by a first-order sentence is accepted by a finite automaton, then the numerical predicates appearing in the sentence can be replaced by numerical predicates that are accepted by a finite automaton. What is particularly intriguing is that such a qualitative proof would give a new demonstration of the Furst-Saxe-Sipser result, since well-developed techniques from the theory of automata can be used to show that

$$\mathbf{FO}[\pi, \mathcal{R}]$$

does not contain the set of strings in which the number of 1s is divisible by q, for $q > 1$.

The same principle appears to be at work when modular quantifiers are used. Thus the results of Smolensky [23] imply

$$(\mathbf{FO} + \mathbf{MOD(q)})[\pi, \mathcal{N}] \cap \mathbf{Reg} = (\mathbf{FO} + \mathbf{MOD(q)})[\pi, \mathcal{R}]$$

when q is a prime power. The same equation for arbitrary q is equivalent to the conjecture that in $ACC(q)$ one cannot add bits modulo p, where p is a prime not

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

dividing q. In the next section analogous results will be established for the complexity class CC.

5. Characterizations of CC in Formal Logic

5.1. *The nonuniform class*

5.1.1. Theorem. *For all $q > 0$,*

$$CC(q) = \mathbf{MOD(q)}[\pi, \mathcal{N}].$$

Proof. I first prove that $CC(q) \subseteq \mathbf{MOD(q)}[\pi, \mathcal{N}]$. The argument is essentially the same as one given in [3] to prove the analgous result for $ACC(q)$. By Theorem 2.2.2, if $L \in CC(q)$ then L is accepted by a polynomial-size family $\{\pi_n : n > 0\}$ of programs over a finite solvable group G whose order divides a power of q. Let $X_n \subseteq G$ be the accepting set for the program π_n. According to a theorem of Straubing, Thérien and Thomas [29], which describes the regular languages whose syntactic monoids are solvable groups, the set of sequences

$$g_1, \ldots, g_r$$

of elements of G such that the product

$$g_1 \cdots g_r$$

belongs to a given subset X of G is defined by a sentence ϕ_X of $\mathbf{MOD(q)}[\{Q_g : g \in G\}, <]$. The program π_n transforms an input string $a_1 \cdots a_n$ into a string $\pi_n(a_1 \cdots a_n) = (g_1, \ldots, g_r)$ of elements of G, and accepts $a_1 \cdots a_n$ if and only if this output string satisfies ϕ_{X_n}. I shall show how to produce a sentence ψ_X that defines the set

$$\bigcup_{n>0} \{a_1 \cdots a_n : \pi_n(a_1 \cdots a_n) \models \phi_X\}.$$

Because of the polynomial-size condition, there is an integer k such that $|\pi_n| < n^k$ for all $n > 0$. By adding a suitable number of instructions that always emit the identity of G, one can assume that the program length is exactly n^k, and that the program makes n^{k-1} complete passes over the input. Thus a position $\mathbf{x}$ in the output string can be uniquely encoded by a k-tuple $\theta(\mathbf{x}) = (x_0, \ldots, x_{k-1})$ of positions in the input string. By the conventions concerning the form of π_n, the $\mathbf{x}$th instruction of the program consults the x_0th bit of the input. Each modular quantifier

$$\exists^s_q \mathbf{x} \alpha(\mathbf{x})$$

in ϕ_X is interpreted as: 'there exist s modulo q k-tuples $(x_0, \ldots, x_{k-1})$ satisfying α'. This can be rewritten

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$\bigvee_f \bigwedge_{j=0}^{q-1} \exists_q^{f(j)} \exists_q^j \mathbf{x}' \alpha,$$

where the inner quantification runs over all $(k-1)$-tuples $\mathbf{x}' = (x_1, \ldots, x_{k-1})$ and the outer disjunction is over all functions $f: \mathbf{Z}_q \to \mathbf{Z}_q$ such that

$$\sum_{j=0}^{q-1} j \cdot f(j) \equiv s \pmod{q}.$$

This rewriting is applied recursively to the quantifiers $\exists_q^j \mathbf{x}'$. Occurrences of $\mathbf{x} < \mathbf{y}$ in ϕ_X are rewritten as

$$\mathbf{less}(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{k-1}),$$

where **less** is a numerical predicate interpreted as

$$\theta^{-1}(x_0, \ldots, x_{k-1}) < \theta^{-1}(y_0, \ldots, y_{k-1}).$$

Occurrences of $Q_g(\mathbf{x})$ are rewritten as

$$(\pi(x_0) \wedge S_g(x_0, \ldots, x_{k-1})) \vee (\neg \pi(x_0) \wedge T_g(x_0, \ldots, x_{k-1})),$$

where S_g (respectively T_g) is a numerical predicate interpreted as: 'if the input length is n, then the $\theta^{-1}(x_0, \ldots, x_{k-1})$th instruction of π_n emits g when the x_0th bit of the input is 1 (respectively 0)'. (Recall that, according to the formalism of Sec. 4, numerical predicates can refer to the length of the string in which they are interpreted.) Finally, L itself is defined by the sentence

$$\bigvee_{X \subseteq G} (\phi_X \wedge U_X),$$

where U_X is the 0-ary numerical predicate $\{\varnothing^m : X_m = X\}$.

For the converse direction, I shall show how to build a polynomial-size, constant-depth circuit family accepting the same language L as a given sentence of $\mathbf{MOD}(q)[\pi, \mathcal{N}]$. Let

$$MOD_{q,n}^d : \{0,1\}^n \to \{0,1\}$$

be the function defined by $MOD_{q,n}^d(a_1, \ldots, a_n) = 1$ if and only if

$$\sum_{i=1}^n a_i \equiv d \pmod{q}.$$

Beginning with the outermost quantifiers, replace each occurrence of $\exists_q^d x \eta(x)$ by

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$MOD^d_{q,n}(\eta(x|1),\ldots,\eta(x|n)),$$

where $\eta(x|i)$ denotes the formula that results from η upon replacing each occurrence of x by i. When all the quantifiers have been replaced, the resulting formula is expressed entirely in terms of the functions $MOD^d_{q,n}$, for $0 \leq d < q$, and constant-size disjunctions and conjunctions of the formulas $\pi(i)$, $\neg\pi(i)$, and $R(i_1,\ldots,i_k)$, where R is a numerical predicate. The terms $\pi(i)$ and $\neg\pi(i)$ can be replaced by x_i and $\bar{x}_i$, and the terms $R(i_1,\ldots,i_k)$ can be replaced by boolean constants 0 or 1, depending on whether the k-tuple of integers $(i_1,\ldots,i_k)$ satisfies R at length n. Thus $L \cap \{0,1\}^n$ is accepted by a circuit that uses only MOD^d_q gates. In 2.2 it was shown how to simulate these gates using only MOD_q gates and constant-size boolean combinations of the inputs. The depth of nesting of these gates (that is, the depth of the circuit) is equal to the depth of nesting of the quantifiers in ϕ, and the size of the circuit is bounded by a polynomial in n, since the size of the formula increases by a factor of $O(n)$ each time a level of quantifiers is replaced.

5.2. *Syntactic invariants of regular languages*

For a thorough account of the notions of syntactic congruence, syntactic semigroup, semidirect and triple products discussed in this subsection, see Eilenberg [11], or Pin [20]. Here I present just enough of the theory to prove the results of 5.3.

Let A be a finite alphabet, and let $L \subseteq A^+$. The *syntactic congruence* of L is the equivalence relation $\cong_L$ on A^+ defined by

$$u \cong_L v$$

if and only if

$$\forall x, y \in A^*(xuy \in L \Leftrightarrow xvy \in L).$$

It is easily verified that $\cong_L$ is compatible with the multiplication in A^+, and thus the quotient $A^+/\cong_L$ is a semigroup, called the *syntactic semigroup* of L and denoted $S(L)$. The projection morphism $\eta_L : A^+ \to S(L)$ is called the *syntactic morphism* of L. $S(L)$ is finite if and only if L is a regular language. Observe that L is itself a union of equivalence classes of $\cong_L$. Indeed, $\cong_L$ has been defined to be the coarsest congruence on A^+ for which this is the case.

If $L \subseteq A^*$ then one can define a congruence on A^* in exactly the same manner. This gives rise to the *syntactic monoid* of L. In the present application (Theorem 5.3.1), the syntactic semigroup is more appropriate.

The following reduction lemma is adapted from a result in [3].

5.2.1. Lemma. *Let $L_1, L_2 \subseteq \{0,1\}^*$ be regular languages with $L_1 \in CC(q)$. Suppose there is a morphism $\alpha : T' \to S(L_2)$, where T' is a subsemigroup of $S(L_1)$, and a positive integer t, such that for every $b \in \{0,1\}$, there exists $w \in \{0,1\}^t$ such that $\eta_{L_2}(b) = \alpha\eta_{L_1}(w)$. Then $L_2 \in CC(q)$.*

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

Proof. I shall construct a constant-depth periodic circuit of period q that, given $x = b_1 \dots b_n \in \{0,1\}^n$, outputs a binary encoding of $\eta_{L_2}(x)$. Since L_2 is a union of $\cong_{L_2}$-classes, the adjunction of a fixed amount of boolean circuitry will suffice to turn this into a circuit that determines whether $x \in L_2$. By hypothesis there exist $w_1, \dots, w_n \in \{0,1\}^t$ such that $\eta_{L_2}(x) = \alpha(\eta_{L_1}(w_1, \dots, w_n))$. The first level of the circuit consists of n boolean circuits, each with one input and t outputs, that compute the strings $w_1, \dots, w_n$. It only remains to compute $\eta_{L_1}(w_1 \cdots w_n)$, since the value of α on an encoding of an element of $S(L_1)$ can be determined with a fixed amount of boolean circuitry. Since $S(L_1)$ is finite, there exist $u_1, v_1, \dots, u_r, v_r \in \{0,1\}^*$ such that for any $z \in \{0,1\}^+$, $\eta_{L_1}(z)$ can be determined by testing $u_1 z v_1, \dots, u_r z v_r$ for membership in L_1. The next level of the circuit thus consists of r subcircuits $\mathscr{C}_i$, $1 \le i \le r$, with $|u_i| + |v_i| + tn$ inputs, that determine whether $u_i w_1, \dots, w_n v_i \in L_1$. Since $L_1 \in CC(q)$ and $|u_i|, |v_i|, t$ are independent of n, this can be done with periodic circuits of period q having constant depth and size polynomial in n. The r outputs of this level can then be fed into fixed-size boolean circuits that determine $\alpha(\eta_{L_2}(w_1 \cdots w_n))$. This completes the proof. □

I conclude this section with a bit of additional terminology concerning semigroup theory. A semigroup S is said to *divide* a semigroup T if there is a homomorphism from a subsemigroup of T onto S. A semigroup S is said to *recognize* a language $L \subseteq A^+$ if there is a homomorphism $\psi : A^+ \to S$ and a subset X of S such that $L = \psi^{-1}(X)$. One sometimes says in this case that the morphism ψ recognizes L. The syntactic semigroup of L is the smallest semigroup that recognizes L; more precisely, S recognizes L if and only if $S(L)$ divides S.

Let S, T be semigroups; the product in S will be written additively. (This is a device to make the notation a little clearer, and is not intended to imply that S is commutative.) A *left action* of T on S is a map

$$(t, s) \mapsto ts$$

such that $t_1(t_2 s) = (t_1 t_2)s$ for all $s \in S$; $t_1, t_2 \in T$, and such that $t(s_1 + s_2) = ts_1 + ts_2$ for all $s_1, s_2 \in S$; $t \in T$. Given such a left action, the *semidirect product* $S * T$ is the set $S \times T$ together with the multiplication

$$(s_1, t_1)(s_2, t_2) = (s_1 + t_1 s_2, t_1 t_2).$$

If S, T_1, T_2 are semigroups, with a left action of T_1 on S and a right action of T_2 on S, there results a semigroup called the *triple product* (T_1, S, T_2). This is the set $T_1 \times S \times T_2$ with multiplication given by

$$(t_1, s, t_2)(t_1', s', t_2') = (t_1 t_1', st_2' + t_1 s', t_2 t_2').$$

U_1 denotes the semigroup $\{0,1\}$ with the standard multiplication. Let $r \ge 1$. A finite semigroup D is said to be *r-definite* if for all $d, d_1, \dots, d_r \in D$,

$$dd_1 \cdots d_r = d_1 \cdots d_r.$$

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

5.3. *Regular languages in CC*

The main theorem of this section shows that the conjecture that the *AND* function cannot be computed in CC is equivalent to certain logical and algebraic characterizations of the regular languages in CC. Moreover, the logical characterization, while not yet proved, is a consequence of general principle for which there is considerable supporting evidence. (See the discussion in 4.3.) There is a stronger version of this theorem, concerning the class $\mathbf{MOD(q)}[\pi, \mathcal{N}]$. As this extension requires considerably more technical details concerning finite semigroups, its proof will be given separately in the next section.

5.3.1. Theorem. *The following are equivalent*:

(a) *Conjecture* 2.3.1.

(b) *Let* $L \subseteq \{0,1\}^+$ *be regular. Then* $L \in CC$ *if and only if* $S(L)$ *contains no isomorphic copy of* U_1 *and no nonsolvable group.*

(c)

$$\mathbf{MOD}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{MOD}[\pi, \mathcal{R}].$$

Proof.

(a) $\Rightarrow$ (b). Suppose $L \in CC$ is regular. If $S(L)$ contains a copy $\{e_0, e_1\}$ of U_1 then there are strings $u, v \in \{0,1\}^+$ such that $\eta_L(u) = e_0$, $\eta_L(v) = e_1$. Thus $u^{|v|}$ and $v^{|u|}$ are strings of the same length mapping to e_0 and e_1, respectively. Let $L' = 1^+$. Then $S(L') = U_1$, with $\eta_{L'}(1) = 1$ and $\eta_{L'}(0) = 0$. Thus the map $\alpha : e_x \mapsto x$ and the integer $t = |u| \cdot |v|$ satisfy the conditions of Lemma 5.2.1, so $1^+ \in CC$, contradicting (a).

Suppose L contains a nonsolvable group G. By a result in [3], there is a nonabelian simple composition factor H of G with the following property: Encode each element h of H by a bit string $e(h)$ such that all the $e(h)$ have the same length. Let L_H be the language

$$\{e(h_1)\cdots e(h_r) \in \{0,1\}^+ : h_1 \cdots h_r = 1\}.$$

Then the languages $L_1 = L_H$ and $L_2 = L$ satisfy the hypotheses of Lemma 5.2.1. Thus $L_H \in CC$. It now follows from Barrington's theorem [2] that $NC^1 \subseteq CC$, which contradicts (a).

It now must be shown that if $S(L)$ has the stated property, then $L \in CC$. (This does not depend on the truth of conjecture 2.3.1.) According to a theorem of Straubing [25], $S(L)$ divides a semidirect product $G * D$, where G is a solvable group and D is an r-definite semigroup, for some $r > 0$. Thus L is recognized by a homomorphism $\phi : \{0,1\}^+ \to G * D$. Encode elements of $G * D$ by bit strings of fixed length. It suffices to show that, given encodings of

$$(g_1, d_1), \ldots, (g_n, d_n),$$

there is a periodic circuit of constant depth and size polynomial in n that computes the product

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$(g_1, d_1)\cdots(g_n, d_n)$$

in $G * D$.

This product is equal to

$$(g_1 + d_1 g_2 + \cdots + d_1 \cdots d_r g_{r+1} + \cdots + d_{n-r}\cdots d_{n-1} g_n, d_{n-r+1}\cdots d_n).$$

With $n + 1$ boolean circuits of fixed size, one can compute encodings of

$$g_1, d_1 g_2, \ldots, d_{n-r}\cdots d_{n-1} g_n, d_{n-r+1}\cdots d_n.$$

The size is fixed because each such circuit need consult only $r + 1$ of the encoded input values. It then follows from Theorem 2.2.2 that the product

$$g_1 + d_1 g_2 + \cdots + d_{n-r}\cdots d_{n-1} g_n$$

in G can be computed by a constant-depth periodic circuit with period equal to the exponent of G and size polynomial in n.

(b) $\Rightarrow$ (c). It follows immediately from the results of Straubing, Thérien and Thomas [28] that

$$\mathbf{MOD}[\pi, \mathscr{R}] \subseteq \mathbf{Reg}$$

and thus

$$\mathbf{MOD}[\pi, \mathscr{R}] \subseteq \mathbf{MOD}[\pi, \mathscr{N}] \cap \mathbf{Reg}.$$

By (b) and the results cited in the preceding part of the proof, L is recognized by a semidirect product $G * D$, where G is a solvable group and D is an r-definite semigroup for some $r > 0$. Let $\phi : \{0,1\}^+ \to G * D$ be a homomorphism that recognizes L. Then there is a subset K of $G * D$ such that $w \in L$ if and only if $\phi(w) \in K$. It suffices to show that for each $(g, d) \in G * D$, $\phi^{-1}(g, d) \in \mathbf{MOD}[\pi, \mathscr{R}]$; a formula for L is then obtained by taking the disjunction of the sentence for $\phi^{-1}(g, d)$ over all $(g, d) \in K$.

Let $\phi(0) = (g_0, d_0)$, $\phi(1) = (g_1, d_1)$. Then $\phi(a_1 \cdots a_n) = (g, d)$ if and only if

$$g_{a_1} + d_{a_1} g_{a_2} + \cdots + d_{a_{n-r}}\cdots d_{a_{n-1}} g_{a_n} = g$$

and

$$d_{a_{n-r+1}}\cdots d_{a_n} = d.$$

By a theorem of [28] (already cited in the proof of Theorem 5.1.1) the set of all strings $(h_1, \ldots, h_n)$ of elements of G such that $h_1 \cdots h_n = g$, can be defined by a formula η of $\mathbf{MOD(q)}[\{Q_h : h \in G\}, <]$. This can be transformed into a sentence η' defining the set

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$\phi^{-1}(\{g\} \times D)$ as follows: The quantifiers in η and the occurrences of $<$ are left unchanged. To translate $Q_h(x)$ consider all products

$$d_{i_1} \cdots d_{i_{s-1}} h_{i_s} = h,$$

where $s \leq r + 1$ and $i_j \in \{0, 1\}$ for $1 \leq j \leq s$, and take the disjunction over all these products of the formulas

$$\rho(x - s + 1) \wedge \cdots \wedge \rho(x),$$

where $\rho(x - s + k)$ is $\pi(x - s + k)$ if $i_k = 1$ and $\neg\pi(x - s + k)$ otherwise. It remains to show how to express $\pi(x - c)$ when c is a positive constant. Normally this would be done with

$$\exists y_1 \cdots \exists y_{c-1} \exists x(\pi(y_1) \wedge y_2 = y_1 + 1 \wedge \cdots \wedge x = y_{c-1} + 1).$$

However, in building the formulas of $\mathbf{MOD}[\pi, \mathscr{R}]$, one cannot have an occurrence of π within the scope of an ordinary existential quantifier. The problem is solved by replacing each of these quantifiers by $\exists_m^1$, where $m > 1$. Observe that $u = v + 1$ is a regular numerical predicate. To complete this part of the proof it must be shown how to express $\phi^{-1}(G \times \{d\})$ by a sentence of the appropriate sort. This is accomplished by considering all products

$$d_{i_1} \cdots d_{i_s} = d$$

with $s \leq r$ and $i_j \in \{0, 1\}$ for $1 \leq j \leq s$, and then proceeding as above.

(c) $\Rightarrow$ (a). It must be shown here that $1^+ \notin \mathbf{MOD}[\pi, \mathscr{R}]$. Using the methods of [28] one can prove that the languages in $\mathbf{MOD}[\pi, \mathscr{R}]$ are built by starting from the languages

$$LENGTH_{d,r} = \{w \in \{0, 1\}^+ : |w| \equiv r \pmod{d}\}$$

and closing under the operations

$$(L_1, L_2) \mapsto L_1 \cup L_2,$$

$$L_1 \mapsto \{0, 1\}^+ \backslash L_1,$$

and

$$(L_1, L_2) \mapsto (L_1, a, L_2, k, m),$$

where $a \in \{0, 1\}$ and $0 \leq k < m$. (L_1, a, L_2, k, m) consists of those strings w such that the number of factorizations $w = w_1 a w_2$ with $w_1 \in L_1$, $w_2 \in L_2$ is congruent to k modulo m. Since $S(1^+) = U_1$, it suffices to show that the syntactic semigroup of every language constructed in this fashion contains no copy of U_1. The syntactic semigroup

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

of $LENGTH_{d,r}$ is the cyclic group of order d, which contains no copy of U_1. The syntactic monoid of the complement of L is isomorphic to the syntactic monoid of L, and the syntactic monoid of the union of L_1 and L_2 divides the direct product of their respective syntactic monoids. Since the property 'S contains no copy of U_1' is preserved under division and direct product, it follows that if $S(L_1)$ and $S(L_2)$ contain no copy of U_1 then neither does $S(L_1 \cup L_2)$. Finally, $S(L_1, a, L_2, k, m)$ divides a triple product $(S(L_1), \mathbf{Z}_m, S(L_2))$. (See [11], [3].) It remains to show that if $S(L_1)$ and $S(L_2)$ contain no copy of U_1, then neither does this triple product. Let $\{(x_0, c_0, y_0), (x_1, c_1, y_1)\}$ be a copy of U_1 contained in such a triple product. Since neither $S(L_1)$ nor $S(L_2)$ contains a copy of U_1, $x_0 = x_1 = x$, $y_0 = y_1 = y$, and both these elements are idempotent. Thus

$$\begin{aligned}(x, c_0, y) &= (x, c_1, y)(x, c_0, y)(x, c_1, y)\\ &= (x, c_1 y + \underbrace{x c_0 y + \cdots + x c_0 y}_{k \text{ times}} + x c_1, y)\\ &= (x, c_1, y)^2\\ &= (x, c_1, y),\end{aligned}$$

a contradiction.

6. Regular Languages in $CC(q)$

In this section I give a characterization of the regular languages in $CC(q)$ equivalent to Conjectures 2.3.1 and 2.3.2. The proof requires some additional technical material concerning finite semigroups and categories, which will be presented in the first two subsections.

6.1. *Some finite semigroup varieties*

A *variety of finite semigroups* is a class of finite semigroups closed under the formation of quotients, subsemigroups, and finite direct products. A *variety of finite monoids* (respectively, of *finite groups*) is a class of finite monoids (resp. groups) closed under the formation of quotients, submonoids and finite direct products.

Let **H** be a variety of finite groups. **KH** consists of all finite semigroups S such that every subgroup of S is in **H**, and such that S contains no proper ideals (i.e., S is *simple*). **KH** is a variety of finite semigroups.

If A, B are finite sets, S a finite semigroup, and $P: B \times A \to S$ a map, then $\mathcal{M}(A, B, S; P)$, a *Rees matrix semigroup* over S, is the semigroup with underlying set $A \times S \times B$ and multiplication

$$(a, s, b) \cdot (a', s', b') = (a, s \cdot P(b, a') \cdot s', b').$$

It is easy to show that a Rees matrix semigroup over a Rees matrix semigroup over S is itself a Rees matrix semigroup over S; that is,

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$\mathcal{M}(A_2, B_2, \mathcal{M}(A_1, B_1, S; P); Q)$$

is isomorphic to

$$\mathcal{M}(A_1 \times A_2, B_1 \times B_2, S; R)$$

for an appropriate $R : B_1 \times B_2 \times A_1 \times A_2 \to S$.

A fundamental theorem of semigroup theory implies

$$\mathbf{KH} = \{\mathcal{M}(A, B, G; P) : G \in \mathbf{H}, P \in G^{A \times B}\}.$$

(See, for example, [9] or [16].)

In a Rees matrix semigroup over a group G, each $\{a\} \times G \times \{b\}$ is a group isomorphic to G. Furthermore, the map

$$(a, g, b) \mapsto (a', h, a'')(a, g, b)(b'', k, b'),$$

where $k \in G$ and

$$h = [P(b, b'') \cdot k \cdot P(b', a')]^{-1} P(a'', a)^{-1},$$

is an isomorphism from $\{a\} \times G \times \{b\}$ onto $\{a'\} \times G \times \{b'\}$.

Let S be a finite semigroup. Then the following four conditions are equivalent (see [25]).

(i) $S^n \in \mathbf{KH}$ for some n.

(ii) $eSe \in \mathbf{H}$ for all idempotents $e \in S$.

(iii) S contains no copy of the monoid U_1, and every group in S belongs to $\mathbf{H}$.

(iv) S divides a wreath product $G \circ D$, where G is a group in $\mathbf{H}$ and D is a definite semigroup (that is, $se = e$ for all $s, e \in D$ with e idempotent).

The family of finite semigroups satisfying these conditions is denoted $\mathbf{LH}$. The following lemma gives a congruential relation between $\mathbf{KH}$ and $\mathbf{LH}$. Let A be a finite alphabet, and let $B = A^n$, considered as a finite alphabet. I write $[a_1 \cdots a_n]$ to denote the letter of B corresponding to the word $a_1 \cdots a_n \in A^+$. Let μ be a congruence of finite index on B^+. An equivalence relation μ_n on A^+ is defined as follows: $w_1 \mu_n w_2$ if and only if $w_1 = w_2$, or

$$w_1 = a_1 \cdots a_{n+r}, \qquad w_2 = a'_1 \cdots a'_{n+s},$$

where $r, s \geq 0$; $a_i, a'_i \in A$, with

$$a_1 \cdots a_n = a'_1 \cdots a'_n, \qquad a_{r+1} \cdots a_{r+n} = a'_{s+1} \cdots a'_{s+n},$$

and

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$[a_1 \cdots a_n][a_2 \cdots a_{n+1}] \cdots [a_{r+1} \cdots a_{r+n}]\mu[a'_1 \cdots a'_n] \cdots [a'_{s+1} \cdots a'_{s+n}].$$

It is straightforward to verify that μ_n is a congruence on A^+ of finite index.

6.1.1. Lemma. *If* $B^+/\mu \in \mathbf{KH}$ *then* $A^+/\mu_n \in \mathbf{LH}$.

Proof. Let $\{w\}_{\mu_n}$ denote the μ_n-class of $w \in A^+$, and $\{v\}_\mu$ the μ-class of $v \in B^+$. Then

$$\{a_1 \cdots a_{n+r}\}_{\mu_n} \leftrightarrow (a_1 \cdots a_n, \{[a_1 \cdots a_n] \cdots [a_{r+1} \cdots a_{r+n}]\}_\mu, a_{r+1} \cdots a_{r+n})$$

is a well-defined one-to-one correspondence between the set of μ_n-classes of words of length at least n, and $A^n \times S \times A^n$, where S is a subset of B^+/μ. Define a map $Q: A^n \times A^n \to B^+/\mu$ by

$$Q(a_1, \ldots, a_n, a'_1, \ldots, a'_n) = \{[a_2 \cdots a_n a'_1] \cdot [a_3 \cdots a_n a'_1 a'_2] \cdots [a_n a'_1 \cdots a'_{n-1}]\}_\mu.$$

The above correspondence thus embeds $(A^+/\mu_n)^n$ into $\mathcal{M}(A^n, A^n, B^+/\mu; Q)$. As this is a Rees matrix semigroup over a semigroup in **KH**, $(A^+/\mu_n)^n \in \mathbf{KH}$, and thus $A^+/\mu_n \in \mathbf{LH}$. □

6.2. *A local-global property for finite semigroupoids*

A number of questions about finite semigroups and finite automata have been answered by studying finite *categories* as algebraic objects. This notion is implicitly present in several papers in the theory of automata ([7], [15], [17], [25]), and was given its explicit foundations by Tilson [30]. Since then, it has seen some fruitful applications ([21], [26]). In these applications, the central question is often: How can we determine whether a given finite category divides a monoid belonging to a particular variety of finite monoids?

For certain varieties **V**, the answer has turned out to be: A category C divides an element of **V** if and only if every base monoid is in **V**. The best-known example of a variety for which such a 'local-global' property holds is the variety $\mathbf{J}_1$ of idempotent and commutative monoids. The proof of this property is the central element in the characterization of locally testable languages, and the result has been applied in a number of other contexts ([7], [18]). Versions of Tilson's paper [29] that circulated in preprint form concerned *semigroupoids* (essentially categories without identity arrows). It was subsequently realized that in most of the interesting applications one could suppose the existence of identities, which made the theory somewhat cleaner. The treatment of semigroupoids in general was relegated to an appendix of the published version of [30]. As shall be seen below, questions from circuit complexity related to the considerations of the previous section lead to the following problem: Let **H** be a variety of finite groups, and let **LH** be the variety of finite semigroups all of whose submonoids are groups in **H**. If a finite semigroupoid C has all its base semigroups in **LH**, does C itself divide a member of **LH**? In this subsection I give a

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

proof of this local-global property; the application to circuit complexity will be given in Sec. 6.3.

See [30] for background on categories and semigroupoids, in particular the definition of semigroupoid division. I shall use the following notation: If C is a finite semigroupoid, then $\mathrm{Obj}(C)$ denotes the set of objects of C. If $a, b \in \mathrm{Obj}(C)$, then $\mathrm{Hom}_C(a, b)$ denotes the set of arrows from a to b. C is said to be *strongly connected* if for all $a, b \in \mathrm{Obj}(C)$, $\mathrm{Hom}_C(a, b)$ is nonempty. The *base semigroup* of C at an object a is the semigroup $\mathrm{Hom}_C(a, a)$.

6.2.1. Theorem. *Let C be a strongly connected finite semigroupoid in which each base semigroup is in* **LH**. *Then C divides a member of* **LH**.

Proof. (a) Let C^n denote the subsemigroupoid of C consisting of all arrows that are products of at least n arrows of C. In this part of the proof I shall show that for sufficiently large values of n, C^n divides a member of **KH**.

Fix an object $u \in \mathrm{Obj}(C)$. Since C is strongly connected, $\mathrm{Hom}_C(u, u)$ is nonempty. Fix a maximal subgroup G of $\mathrm{Hom}_C(u, u)$. I claim that for every $v \in \mathrm{Obj}(C)$ and every maximal subgroup K of $\mathrm{Hom}_C(v, v)$, there is an isomorphism of G onto K given by

$$h \mapsto z_K h z'_K,$$

where $z_K \in \mathrm{Hom}_C(v, u)$, $z'_K \in \mathrm{Hom}_C(u, v)$. To see this pick $x' \in \mathrm{Hom}_C(v, u)$, $y' \in \mathrm{Hom}_C(u, v)$ such that both x' and y' can be factored as the product of at least n arrows between u and v. Since

$$\mathrm{Hom}_C(u, u), \mathrm{Hom}_C(v, v) \in \mathbf{LH},$$

$x' \cdot \mathrm{Hom}_C(u, u) \cdot y'$ and $y' \cdot \mathrm{Hom}_C(v, v) \cdot x'$ are groups, provided n is chosen large enough. Using the isomorphism between maximal subgroups of elements of **KH** described in 6.1, we can left-multiply x' and right-multiply y' by appropriate arrows to obtain $x \in \mathrm{Hom}_C(v, u)$, $y \in \mathrm{Hom}_C(u, v)$ such that

$$x \cdot \mathrm{Hom}_C(u, u) \cdot y \subseteq K,$$

$$y \cdot \mathrm{Hom}_C(v, v) \cdot x \subseteq G.$$

Given $g \in G$, I define $\phi : G \to K$ by

$$\phi(g) = xg(yx)^{-1}y.$$

Thus ϕ is a homomorphism. Further, if $k \in K$, then

$$k = \phi(y(xy)^{-1}k(x(yx)^{-1}y)^{-1}x) \in \phi(G).$$

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

ϕ is thus surjective, and is consequently an isomorphism. This establishes the claim, with $z_K = x$, $z'_K = (yx)^{-1}y$.

Now I claim that if n is chosen large enough, every arrow $z \in \mathrm{Hom}_{C^n}(u_1, u_2)$ admits a factorization

$$z = yhy',$$

where $y \in \mathrm{Hom}_C(u_1, u)$, $y' \in \mathrm{Hom}_C(u, u_2)$ and $h \in G$. There exists $s \geq 0$ such that for all $v \in \mathrm{Obj}(C)$, $\mathrm{Hom}_C(v, v)^s$ is a union of groups. Let $n \geq (s+1)\cdot|\mathrm{Obj}(C)|$. Then every product of n arrows must pass by some object v at least s times. Thus if $z \in \mathrm{Hom}_{C^n}(u_1, u_2)$,

$$z = z_1 y_1 \cdots y_s z_2,$$

where $z_1 \in \mathrm{Hom}_C(u_1, v)$, $z_2 \in \mathrm{Hom}_C(v, u_2)$, and $y_1, \ldots, y_s \in \mathrm{Hom}_C(v, v)$. Because of the isomorphism between maximal subgroups cited above, we can write $y_1 \cdots y_s = z'_1 g z'_2$, for some $g \in G$. This gives the desired factorization.

Now let

$$\mathscr{A} = \bigcup_{v \in \mathrm{Obj}(C)} \mathrm{Hom}_C(v, u),$$

$$\mathscr{B} = \bigcup_{v \in \mathrm{Obj}(C)} \mathrm{Hom}_C(u, v).$$

To each arrow $z \in C^n$, associate the set of triples $(\alpha, h, \beta) \in \mathscr{A} \times G \times \mathscr{B}$ such that $z = \alpha h \beta$. These triples are said to *cover* the arrow. The result in the preceding paragraph shows that this set is nonempty. Define

$$P : \mathscr{B} \times \mathscr{A} \to G$$

by $P(\beta, \alpha) = e\beta\alpha e$, where e is the identity of G, if β and α are consecutive arrows. P is extended in an arbitrary fashion to pairs of non-consecutive arrows. If β and α' are consecutive, then

$$(\alpha g \beta)\cdot(\alpha' g' \beta') = \alpha g \cdot P(\beta, \alpha') \cdot g' \beta'.$$

Thus if (α, g, β) and (α', g', β') cover z_1 and z_2 respectively, then the product

$$(\alpha, g, \beta)\cdot(\alpha', g', \beta')$$

in $\mathscr{M}(\mathscr{A}, \mathscr{B}, G; P)$ covers $z_1 z_2$. So C^n divides $\mathscr{M}(\mathscr{A}, \mathscr{B}, G; P) \in \mathbf{KH}$, as claimed.

(b) I shall now use the result of part (a) to complete the proof of the theorem. Let X be the set of arrows of C, considered as a finite alphabet, and let $Y = X^n$, also considered as a finite alphabet. As one must distinguish between a string of elements

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

in X^+ and its product in C, if $x_1, \ldots, x_r \in X$ is a sequence of consecutive arrows, I denote by $x_1 \cdots x_r$ the string in X^+, and by $\{x_1 \cdots x_r\}$ the product in C.

Note that for every $x_1 \cdots x_n \in X^+$ that is a sequence of consecutive arrows, there is an arrow $z = z(x_1, \ldots, x_n)$ such that

$$\{x_1 \cdots x_n z x_2 \cdots x_n\} = \{x_1 \cdots x_n\}.$$

To prove this, I use the division proved in part (a): $\{x_1 \cdots x_n\}$ is covered by some $(a, g, b) \in \mathscr{M}(\mathscr{A}, \mathscr{B}, \mathscr{G}; P)$. Pick any arrow y from the end of x_n to the beginning of x_2. Then $\{yx_2 \cdots x_n\}$ is covered by $(a', g', b) \in \mathscr{M}(\mathscr{A}, \mathscr{B}, G; P)$. (We can pick y to be of the form $y'x_1$ where y' is an arrow in C^n, and thus it can be assumed that the right-hand co-ordinate b is the same in both cases.) Let m be the exponent of G. Then

$$\{x_1 \cdots x_n (yx_2 \cdots x_n)^m\}$$

is covered by

$$(a, g, b) \cdot (a', g', b)^m = (a, g \cdot (P(b, a')g')^m, b) = (a, g, b).$$

So the claim is verified by $z = (yx_2 \cdots x_n)^{m-1} y$.

Define

$$\mu : X^n \to \mathscr{M}(\mathscr{A}, \mathscr{B}, G; P)$$

by

$$\mu(x_1, \ldots, x_n) = (a, g, b),$$

where (a, g, b) covers $\{x_1 \cdots x_n \cdot z(x_1, \ldots, x_n)\}$. μ extends to a homomorphism with domain Y^+, and thus induces a congruence on Y^+ such that $Y^+/\mu \in \mathbf{KH}$. Let $x_1 \cdots x_r$ and $x_1' \cdots x_s'$ be coterminal paths such that

$$x_1 \cdots x_r \mu_n x_1' \cdots x_s'.$$

To complete the proof, I will show

$$\{x_1 \cdots x_r\} = \{x_1' \cdots x_s'\}.$$

This will imply that C divides X^+/μ_n, which by 6.1.1 is in **LH**. According to the above remarks,

$$\mu([x_1 \cdots x_n] \cdot [x_2 \cdots x_{n+1}] \cdots [x_{r-n+1} \cdots x_r]) = \mu([x_1' \cdots x_n'] \cdots [x_{s-n+1} \cdots x_s']).$$

(One can of course suppose, as I am doing here, that $r, s \geq n$, since otherwise the two strings would be equal.)

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

Thus

$$\{x_1 \cdots x_n \cdot z(x_1, \ldots, x_n) \cdot x_2 \cdots x_{n+1} \cdots x_{r-n+1} \cdots x_r \cdot z(x_{r-n+1}, \ldots, x_r)\}$$

$$= \{x'_1 \cdots x'_n \cdot z(x'_1, \ldots, x'_n) \cdots x'_{s-n+1} \cdots x'_s \cdot z(x'_{s-n+1}, \ldots, x_s)\},$$

so

$$\{x_1 \cdots x_r \cdot z(x_{r-n+1}, \ldots, x_r)\} = \{x'_1 \cdots x'_s \cdot z(x'_{s-n+1}, \ldots, x_s)\}.$$

Thus, by the equivalence of suffixes,

$$\{x_1 \cdots x_r\} = \{x_1 \cdots x_r \cdot z(x_{r-n+1}, \ldots, x_r) \cdot x_{r-n+2} \cdots x_r\}$$

$$= \{x'_1 \cdots x'_s \cdot z(x'_{s-n+1}, \ldots, x'_s) \cdot x'_{s-n+2} \cdots x'_s\}$$

$$= \{x'_1 \cdots x'_s\}. \qquad \square$$

6.3. *The characterization of CC(q)*

In this subsection the algebraic results of 6.2 are applied to obtain the following refinement of Theorem 5.3.1.

6.3.1. Theorem. *The following are equivalent:*

(a) *Conjectures* 2.3.1 *and* 2.3.2.

(b) *Let* $L \subseteq \{0,1\}^+$ *be a regular language.* $L \in \mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}]$ *if and only if* $S(L)$ *(the syntactic semigroup of* L*) contains no copy of* U_1*, and no nonsolvable groups, and furthermore, for all* $t > 0$*, every group contained in* $\eta_L(\{0,1\}^t)$ *has order dividing a power of* q*.* (η_L *is the syntactic morphism of* L.)

(c)

$$\mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}].$$

The proof will be given shortly, after the proof of this algebraic lemma:

6.3.2. Lemma. *Let* $\mathbf{H}$ *be the variety of all finite solvable groups whose order divides a power of* q*. Let* $L \subseteq \{0,1\}^+$ *be a regular language that satisfies the conditions in part* (b) *of Theorem* 3*. Then there is an* $r > 0$*, and a morphism*

$$\lambda_r : \{0,1\}^+ \to \mathbf{Z}_r$$

such that

$$\lambda_r(w) = |w| \bmod r,$$

for all $w \in \{0,1\}^+$*, and such that the derived semigroupoid of the relational morphism* $\lambda_r \eta_L^{-1}$ *divides a semigroup in* $\mathbf{LH}$.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

Proof. The derived semigroupoid D of $\lambda_r\eta_L^{-1}$ has $\mathbf{Z}_r$ as its set of objects and arrows (i, w, j), where $w \in \{0,1\}^+$, and $i + |w| \equiv j \pmod r$. Two arrows (i, w_1, j), (i, w_2, j) are identified if for all v with $|v| \equiv i \pmod r$,

$$\eta_L(vw_1) = \eta_L(vw_2).$$

Multiplication of arrows is defined by

$$(i, w, j)\cdot(j, w', k) = (i, ww', k),$$

which is consistent with the identification of arrows. This account of the derived semigroupoid is different from the one given in Tilson [30], however the two definitions are equivalent.

By 6.2.1, it is sufficient to show that each base semigroup of D is in **LH**, since D is strongly connected.

For each nontrivial group G in $S(L)$, pick $v \in \{0,1\}^+$ such that $\eta_L(v)$ is the identity of G. Let r be the least common multiple of the lengths of the chosen words v.

Observe that $\mathrm{Hom}_D(i,i)$ is a divisor of $S(L)$, since the map $w \mapsto (i, w, i)$ is a well-defined homomorphism from the subsemigroup $(\{0,1\}^r)^+$ of $\{0,1\}^+$ onto $\mathrm{Hom}_D(i,i)$ that factors through η_L. Since $S(L)$ contains no copy of U_1, and every group in $S(L)$ is solvable, $S(L) \in \mathbf{LG}_{\mathrm{sol}}$, where $\mathbf{G}_{\mathrm{sol}}$ is the variety of finite solvable groups. Thus each $\mathrm{Hom}_D(i,i) \in \mathbf{LG}_{\mathrm{sol}}$, so it remains to show that $\mathrm{Hom}_D(i,i)$ contains no group of order p, where p is a prime that does not divide q. Such a group, if it exists, is a quotient of a group in $S(L)$. Thus there is a word w such that $\eta_L(w)$ is a group element and (i, w, i) is an element of order p in $\mathrm{Hom}_D(i,i)$. Let m be the order of $\eta_L(w)$; then $p|m$. Since $r||w|$, there is a word v such that $|v| = |w|$, and such that $\eta_L(w)$ is the identity of the group generated by $\eta_L(w)$. Thus the set of words $\{v^k w^{m-k} : 1 \le k \le m\}$ maps onto a group whose order does not divide a power of q, contradicting the condition on the syntactic morphism of L.

Proof of Theorem 3. The proof follows the same outline as that of Theorem 5.3.1. Here I will simply indicate the principal modifications.

(a) $\Rightarrow$ (b). By Theorem 5.3.1, we know that (a) implies every regular language in $\mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}]$ has its syntactic semigroup in $\mathbf{LG}_{\mathrm{sol}}$. Furthermore, if the image of $\{0,1\}^t$ under η_L contains a group of order p, then one can use the techniques of [3] to show that

$$\{w \in \{0,1\}^+ : p||w|_1\}$$

is in $\mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}]$, contradicting (a). It remains to show that every regular language L satisfying the conditions on the syntactic morphism is in $\mathbf{MOD}(\mathbf{q})[\pi, \mathcal{N}]$ (this holds independently of (a).)

The result of Lemma 4 and the connection between the derived semigroupoid and the wreath product [30] implies that the syntactic morphism of L factors through a morphism

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

$$\phi : \{0,1\}^+ \to G \circ D \circ \mathbf{Z}_r,$$

where D is a definite semigroup, G is a solvable group whose order divides a power of q, and the composition of ϕ with the projection onto $\mathbf{Z}_r$ is the length modulo r.

Let L' be a language accepted by a morphism

$$\psi : \{0,1\}^+ \to D \circ \mathbf{Z}_r,$$

where D is definite, and where the composition of ψ with the projection onto $\mathbf{Z}_r$ is the length modulo r. It is not difficult to show that L' is a boolean combination of languages of the form

$$\{0,1\}^* u,$$

where $u \in \{0,1\}^+$, and

$$\{v \in \{0,1\}^+ : |v| \equiv i \,(\mathrm{mod}\, r)\}.$$

Both these languages can be described by formulas of the appropriate kind. It follows from results in [24] that L is obtained from such languages by closing under boolean operations and the operation

$$K \mapsto \langle K, a, r, q \rangle,$$

where $a \in \{0,1\}$, $0 \leq r < q$, and where $\langle K, a, r, q \rangle$ denotes the set of all strings in which the number of initial segments in Ka is congruent to r modulo q. It is furthermore easy to show that the class of languages $\mathbf{MOD(q)}[\pi, \mathcal{N}]$ is closed under this operation.

(b) $\Rightarrow$ (c). If

$$L \in \mathbf{MOD(q)}[\pi, \mathcal{N}] \cap \mathbf{Reg}$$

then, as argued in the preceding paragraph, the conditions in (b) imply that L is built from the languages

$$\{0,1\}^* u$$

and

$$\{v \in \{0,1\}^+ : v \equiv i \,(\mathrm{mod}\, m)\}$$

by boolean operations and repeated applications of the operation

$$K \mapsto \langle Ka, r, q \rangle.$$

One can then construct a formula of $\mathbf{MOD(q)}[\pi, \mathcal{R}]$ that defines L.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

(c) $\Rightarrow$ (a). The technique is the same as the corresponding part of the proof of Theorem 5.3.1. The only thing left to show is that

$$\{v \in \{0,1\}^+ : |v|_1 \equiv 0 \pmod{p}\}$$

does not belong to the class of languages built by starting from

$$\{v \in \{0,1\}^+ : |v| \equiv i \pmod{m}\}$$

and closing under boolean operations and the operation

$$(L_1, L_2) \mapsto \langle L_1, a, L_2, r, q \rangle .$$

Here $\langle L_1, a, L_2, r, q \rangle$ denotes the set of strings w for which the number of factorizations $w = uav$, with $u \in L_1$, $v \in L_2$ and $a \in \{0,1\}$ is congruent to r modulo q. This is accomplished by noting that if

$$\phi : \{0,1\}^+ \to M_1 \times M_2$$

maps a word w to a group element of prime order p, then one of the two projections

$$\phi : \{0,1\}^+ \to M_1 ,$$

$$\phi_2 : \{0,1\}^+ \to M_2$$

must map w to an element of order p, and that the projection

$$(M_1, \mathbf{Z}_q, M_2) \to M_1 \times M_2 ,$$

where $(M_1, \mathbf{Z}_q, M_2)$ is a triple product, is injective on groups of prime order not dividing q. It follows by induction on the construction of L that the syntactic morphism of L never maps two words of the same length to different elements of a group of order p.

7. Related Results and Open Problems

7.1. *The case of a prime period*

Conjectures 2.3.1 and 2.3.2 are true when the period q is a prime power. Indeed, the polynomial size bound is irrelevant here, since no constant-depth periodic circuit family of prime period p can recognize 1^+ or add bits modulo q, where q is a prime different from p. That this is so for 1^+ is a special case of a theorem in [5], where it is shown that no family of programs over a finite nilpotent group can recognize 1^+. The proof of Theorem 2.2.2 shows that a constant-depth family of periodic circuits of period p can be simulated by a family of programs over a p-group, and since all p-groups are nilpotent, the result follows. Furthermore, if one can add bits modulo q with a constant-depth circuit family of period p, where q is a prime different from p, then any

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

constant-depth periodic circuit family of period pq can be simulated by such a family of period p. Thus, again using the methods of the proof of Theorem 2.2.2, any family of programs over a solvable group of order p^iq^j can be simulated by a constant-depth periodic circuit family of period p. However, there are non-nilpotent groups of this form, and it is shown in [5] that one can recognize 1^+ with programs over any solvable, non-nilpotent group.

It thus follows from Theorem 6.3.1 that the equation

$$\mathbf{MOD(q)}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{MOD(q)}[\pi, \mathcal{R}]$$

holds when q is prime.

7.2. *Extending the spectral methods*

It is worth investigating whether the methods of Sec. 3 can be extended to treat programs over groups other than the dihedral groups. A proof of Conjecture 2.3.1 for groups that are extensions of one abelian group by another seems a promising prospect. A more difficult question is whether such methods can be extended to apply to solvable groups outside this class. For example, there has been no progress made concerning the symmetric group S_4.

Another question raised by the results of Sec. 3 is the relation between strong and weak recognition. It would be nice to see a theorem that says: if L is weakly recognized by a polynomial-size family of programs over a group G, then it is strongly recognized by a polynomial-size family of programs over a group G', with G' somehow close to G in algebraic structure. It should be noted that in light of Barrington's result [2], this is true if G is nonsolvable, with $G' = G$.

7.3. *Extending the logical methods*

As a result of the work in the present paper and the results of [3], all the known and conjectured lower bounds results concerning the classes AC^0, ACC and CC are equivalent to equalities of the form

$$\mathbf{Q}[\pi, \mathcal{N}] \cap \mathbf{Reg} = \mathbf{Q}[\pi, \mathcal{R}],$$

where $\mathbf{Q}$ is a class of quantifiers. Roughly speaking, this says that if the circuit family accepts a regular language, it can be replaced by a family that itself can be built by a finite automaton. A direct proof of this general principle would of course be very desirable. It might be worthwhile to try to prove this for some special classes of formulas. For instance, it is not difficult to give an independent proof of this equation for Σ_1 and Π_1 formulas.

Acknowledgements

While this research was being conducted I had a number of fruitful conversations with Pierre Péladeau, Pascal Weil, and especially David Barrington and Denis Thérien. Indeed, Theorem 2.2.2 is the result of collaborative work among Barrington, Thérien

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

and myself. This paper was prepared while I was visiting the Laboratoire d'Informatique Théorique et Programmation and Université de Paris VII.

References

1. M. Ajtai, Σ_1^1 *formulae on finite structures*, Ann. Pure Appl. Logic **24** (1983), 1–48.
2. D. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in* NC^1, J. Comp. Sys. Sci. **38** (1989), 150–164.
3. D. Barrington, K. Compton, H. Straubing and D. Thérien, *Regular languages in* NC^1, to appear in J. Comp. Sys. Sci.
4. D. Barrington, N. Immerman and H. Straubing, *On uniformity in* NC^1, J. Comp. Sys. Sci. **41** (1990), 274–306.
5. D. Barrington, H. Straubing and D. Thérien, *Nonuniform automata over groups*, Information and Computation **89** (1990), 109–132.
6. D. Barrington and D. Thérien, *Finite monoids and the fine structure of* NC^1, J. Assoc. Comp. Mach. **35** (1988), 941–952.
7. J. A. Brzozowski and I. Simon, *Characterizations of locally testable events*, Discrete Math. **4** (1973), 243–271.
8. J. Büchi, *Weak second-order arithmetic and finite automata*, Z. Math. Logik Grundlagen Math. **6** (1960), 66–92.
9. A. Clifford and G. Preston, *Algebraic Theory of Semigroups*, American Mathematical Society, Providence, Vol. 1, 1961, and Vol. 2, 1967.
10. S. Cook, *A taxonomy of problems with fast parallel algorithms*, Information and Control **64** (1985), 2–22.
11. S. Eilenberg, *Automata, Languages and Machines*, vol. B., Academic Press, New York, 1976.
12. M. Furst, J. Saxe and M. Sipser, *Parity, circuits and the polynomial-time hierarchy*, Math. Syst. Theory **17** (1984), 13–27.
13. J. Hastad, *Almost optimal lower bounds for small depth circuits*, Proc. 18th ACM STOC (1986), 6–20.
14. N. Immerman, *Languages that capture complexity classes*, SIAM J. Computing **16** (1987), 760–78.
15. R. Knast, *Some theorems on graph congruences*, RAIRO Inform. Théor. **17** (1983), 331–342.
16. G. Lallement, *Semigroups and Combinatorial Applications*, Wiley, New York, 1979.
17. R. McNaughton, and S. Papert, *Counter-free Automata*, MIT Press, Cambridge, 1971.
18. S. Margolis and J. E. Pin, *Graphs, inverse semigroups and languages*, University of Nebraska Computer Science Research Report, 1985.
19. P. Péladeau, *Logically defined subsets of* N^k, Research report, LITP, Paris, 1989.
20. J. E. Pin, *Varieties of Formal Languages*, Plenum, London, 1986.
21. J. E. Pin, H. Straubing and D. Thérien, *Locally trivial categories and unambiguous concatenation*, J. Pure and Applied Algebra **52** (1988), 297–311.
22. A. A. Razborov, *Lower bounds for the size of circuits of bounded depth with basis* $\{\&, \oplus\}$, Math. Notes Acad. Sci. USSR **41** (Sept. 1987), 333–338.
23. R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, Proc. 19th ACM STOC (1987), 77–82.
24. H. Straubing, *Families of recognizable sets corresponding to certain varieties of finite monoids*, J. Pure and Applied Algebra **15** (1979), 305–318.
25. H. Straubing, *Finite semigroup varieties of the form* $\mathbf{V} * \mathbf{D}$, J. Pure and Applied Algebra **36** (1985), 53–94.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.

26. H. Straubing, *Semigroups and languages of dot-depth 2*, Theor. Comp. Sci. **58** (1988), 361–378.
27. H. Straubing, and D. Thérien, *Finite automata and computational complexity*, in Formal Properties of Finite Automata and Applications, Lecture Notes in Computer Science, vol. 226, Springer, Berlin (1990), pp. 199–223.
28. H. Straubing, D. Thérien and W. Thomas, *Regular languages defined with generalized quantifiers*, Automata, Languages and Programming; Proc. 15th ICALP, Lecture Notes in Computer Science, Springer, Berlin (1988), pp. 561–575.
29. B. Tilson, *Categories as algebra: an essential ingredient in the theory of monoids*, J. Pure and Applied Algebra **48** (1987), 83–198.
30. D. Thérien and A. Weiss, *Graph congruences and wreath products*, J. Pure and Applied Algebra **36** (1985), 205–215.
31. A. C. Yao, *Separating the polynomial-time hierarchy by oracles*, Proc. 26th IEEE FOCS (1985), 1–10.

Int. J. Algebra Comput. 1991.01:49-87. Downloaded from www.worldscientific.com
by COLUMBIA UNIVERSITY on 02/02/15. For personal use only.