Arithmetic Constant-Depth Circuit Complexity Classes

Hubie Chen

Department of Computer Science, Cornell University,
Ithaca, NY 14853, USA
hubes@cs.cornell.edu

Abstract. The boolean circuit complexity classes $AC^0 \subseteq AC^0[m] \subseteq TC^0 \subseteq NC^1$ have been studied intensely. Other than NC^1, they are defined by constant-depth circuits of polynomial size and unbounded fan-in over some set of allowed gates. One reason for interest in these classes is that they contain the boundary marking the limits of current lower bound technology: such technology exists for AC^0 and some of the classes $AC^0[m]$, while the other classes $AC^0[m]$ as well as TC^0 lack such technology.

Continuing a line of research originating from Valiant's work on the counting class $\sharp P$, the arithmetic circuit complexity classes $\sharp AC^0$ and $\sharp NC^1$ have recently been studied. In this paper, we define and investigate the classes $\sharp AC^0[m]$ and $\sharp TC^0$, new arithmetic circuit complexity classes that are defined by constant-depth circuits and are analogues of the classes $AC^0[m]$ and TC^0.

1 Introduction

The study of counting complexity was initiated by Valiant's work on $\sharp P$, the class of functions mapping a string x to the number of accepting paths of x on an NP-machine [11]. The class $\sharp L$, defined similarly but with NL-machines, has also been studied [13, 10, 6, 5]. Both $\sharp P$ and $\sharp L$ can be obtained by "arithmetizing" boolean circuit characterizations of NP and NL given in [12]. To arithmetize a boolean circuit, we propagate all NOT gates to the input level, convert OR gates to addition ($+$) gates, and convert AND gates to multiplication ($*$) gates. Viewing the inputs to the circuit as taking on the values $0, 1$ from the natural numbers, we obtain circuits which map naturally from the binary strings $\{0, 1\}^*$ to the natural numbers.

More recently, the arithmetic classes $\sharp AC^0$, $\sharp BWBP$, $\sharp NC^1$, and $\sharp SAC^1$ have been defined and studied [1, 7, 3, 8, 4, 13]. Other than $\sharp BWBP$, these classes are arithmetic versions of boolean classes typically defined by circuits, and arise from arithmetizing the corresponding boolean circuits. These classes obey the inclusion chain $\sharp AC^0 \subsetneq \sharp BWBP \subseteq \sharp NC^1 \subseteq \sharp SAC^1$, which essentially mirrors the known relationships $AC^0 \subsetneq BWBP = NC^1 \subseteq SAC^1$ of boolean classes. Lying inbetween the boolean classes AC^0 and NC^1 are a hierarchy of classes $AC^0[m]$ and the class TC^0, which have been studied extensively. (For any m, we have the inclusions $AC^0 \subseteq AC^0[m] \subseteq TC^0 \subseteq NC^1$.) Not only have these classes given insight into the structure of NC^1, but the class TC^0 captures the complexity of natural problems such as multiplication and division, while the $AC^0[m]$ hierarchy is particularly interesting since it contains the boundary marking the limits of current lower bounds technology.

In this paper, we introduce the classes $\sharp AC^0[m]$ and $\sharp TC^0$, arithmetic versions of the boolean classes $AC^0[m]$ and TC^0. Just as $AC^0[m]$ and TC^0 give a refined view of NC^1, our new arithmetic classes refine $\sharp NC^1$. Shadowing their boolean counterparts, these classes fall into the inclusion chain $\sharp AC^0 \subseteq \sharp AC^0[m] \subseteq \sharp TC^0 \subseteq \sharp BWBP \subseteq \sharp NC^1$. Both the original boolean classes and the new arithmetic classes are defined by constant-depth circuits, which in this paper are always of unbounded fan-in and polynomial size. The class $AC^0[m]$ (respectively TC^0) consists of those functions computable by constant-depth circuits with AND, OR, and MOD m (respectively MAJORITY) gates. In order to define the classes $\sharp AC^0[m]$ and $\sharp TC^0$, we introduce arithmetic extensions of the functions MOD m and MAJORITY. Then, we arithmetize $AC^0[m]$ and TC^0 as above, but in addition convert MOD m and MAJORITY gates into their arithmetic extensions.

While $\sharp TC^0$ is shown to be equal to its boolean analogue TC^0, our definition of $\sharp AC^0[m]$ begets additional complexity classes. By defining $\mathit{Diff}\,AC^0[m]$ to consist of those functions equal to the difference of two $\sharp AC^0[m]$ functions, we obtain another hierarchy of classes, which includes the already studied $\mathit{Diff}\,AC^0$ [1, 7, 3] at the bottom. Moreover, we define two generic operators on arithmetic function classes; acting on the classes $\mathit{Diff}\,AC^0[m]$ and $\sharp AC^0[m]$ with these operators gives us new language classes.

This paper focuses on studying the structure of three hierarchies: the $\sharp AC^0[m]$ hierarchy, the $\mathit{Diff}\,AC^0[m]$ hierarchy, and a hierarchy of language classes. The hierarchy of language classes includes the classes $AC^0[m]$ and the classes induced by applying the mentioned operators to the classes $\sharp AC^0[m]$ and $\mathit{Diff}\,AC^0[m]$. We prove class separations and containments where possible. Although making unconditional statements about these classes would in many cases require new lower bounds, it is often possible to show that a question in one hierarchy is equivalent to a question in another. For instance, we prove that the hierarchy of the classes $\sharp AC^0[m]$ has *exactly* the same structure as the hierarchy of the classes $AC^0[m]$: $AC^0[m] \subseteq AC^0[m']$ if and only if $\sharp AC^0[m] \subseteq \sharp AC^0[m']$. We also investigate closure properties of the classes $\sharp AC^0[m]$ and $\mathit{Diff}\,AC^0[m]$. The closure properties proved here generalize those appearing in previous work [1, 7, 3].

One reason why this plethora of new classes is interesting is that it offers *rephrasings* of open questions. Not only does the $\sharp AC^0[m]$ hierarchy have the same structure as the $AC^0[m]$ hierarchy, but the classes $\sharp AC^0[m]$ seem to give an *alternate* decomposition[1] of TC^0. As a result, any question regarding the structure of the boolean $AC^0[m]$ hierarchy can be rephrased as a question con-

[1] Note that by Theorem 7, $FAC^0[m]$ is properly contained in $\sharp AC^0[m]$, assuming that $AC^0[m] \neq TC^0$.

cerning arithmetic classes, offering a new line of attack on such questions. As mentioned, the $AC^0[m]$ hierarchy is particularly important because it contains both classes for which we have lower bounds technology, and classes for which we do not: such technology exists for the class $AC^0[m]$ when m is a prime power, but not when m is a composite with two or more distinct prime factors [9]. Note that ours are not the first results providing an interface between boolean and arithmetic circuit complexity, in the constant-depth setting: an intriguing result obtained by Agrawal et al. is that deciding whether or not two $\sharp AC^0$ circuits are equal characterizes exactly TC^0 [1].

Another reason to be interested in the classes introduced here is that they provide *refinements* of open questions, which may be more tractable than the original questions. For instance, for any odd positive integer m, new language classes sitting inbetween $AC^0[m]$ and $AC^0[2m]$ are induced by our arithmetic classes. These new classes offer us the ability to "interpolate" between existing classes. In particular, when m is an odd prime, our new classes sit inbetween a class ($AC^0[m]$) for which we have lower bound technology, and a class ($AC^0[2m]$) for which we do not. Thus, there is the natural question of whether or not one can prove lower bounds using one of these new classes. Establishing lower bounds technology for one of these new classes is necessarily no more difficult than doing so for $AC^0[2m]$, since the new classes are contained in $AC^0[2m]$. Studying these "refined" questions is not only independently interesting, but may give insight into the original questions.

The contents of this paper are as follows. In Section 2, we define the complexity classes to be studied, as well as the operators on arithmetic classes. In Section 3, we study the language classes induced by the arithmetic classes. Section 4 contains a normal form theorem, which essentially states that our arithmetic circuits need only use the arithmetic MOD and MAJORITY gates on 0-1 valued inputs. This theorem in turn allows us to show that the $\sharp AC^0[m]$ hierarchy is isomorphic to the $AC^0[m]$ hierarchy. Section 5 studies the classes $\mathit{Diff}\,AC^0[m]$; focus is given to the question of whether or not $\mathit{Diff}\,AC^0[m] = \mathit{Diff}\,AC^0[2m]$. (This equality is unconditionally true when $m = 1$.) Aided by the notion of normal form, we derive a number of closure properties in Section 6.

For more background on circuit complexity, we refer the reader to the book [14], which contains a chapter on arithmetic circuit complexity; the survey [2] is also a good source of information on arithmetic circuit complexity.

2 Preliminaries

We let $\mathbb{N}$ denote the set of natural numbers, $\{0, 1, 2, \ldots\}$; and, we let $\mathbb{N}^+$ denote the set of positive integers, $\{1, 2, 3, \ldots\}$. The complexity classes that we study in this paper are defined by constant depth circuits; what varies among the definitions of the classes are the types of gates allowed. We will instantiate the following definition with different bases of gates to define our classes.

Definition 1. *We say that a function is computable by AC^0 circuits over the basis B if the function can be computed by a family of constant depth, polynomial* *size circuits of unbounded fan-in with gates from B and inputs from $\{0, 1, x_i, \overline{x_i}\}$. (By the size of a circuit, we mean the number of gates plus the number of wires.)*

2.1 Boolean Classes

The boolean functions and classes in the next two definitions have been studied in past work.

Definition 2. *We define the following boolean functions:*

- *MOD m on inputs $x_1, \ldots, x_k$ takes on the value 1 if the number of x_i's that are nonzero is a multiple of m; and the value 0 otherwise.*
- *MAJORITY on inputs $x_1, \ldots, x_k$ takes on the value 1 if the number of x_i's that are nonzero is strictly greater than $k/2$; and the value 0 otherwise.*

Definition 3. *We define the following classes of boolean functions[2]:*

- *AC^0 (FAC^0) is the class of functions computable by AC^0 circuits over the basis of boolean functions $\{AND, OR\}$ with exactly one output gate (one or more output gates).*
- *$AC^0[m_1, \ldots, m_l]$ ($FAC^0[m_1, \ldots, m_l]$) is the class of functions computable by AC^0 circuits over the basis of boolean functions $\{AND, OR, MOD\ m_1, \ldots, MOD\ m_l\}$ with exactly one output gate (one or more output gates). (This definition is for all $\{m_1, \ldots, m_l\} \subseteq \mathbb{N}^+$.)*
- *TC^0 (FTC^0) is the class of functions computable by AC^0 circuits over the basis of boolean functions $\{AND, OR, MAJORITY\}$ with exactly one output gate (one or more output gates).*

We note a lower bound due to Smolensky.

Theorem 1. *[9] If p and q are distinct primes, then MOD $q \notin AC^0[p]$.*

This separates the classes $AC^0[q]$ and $AC^0[p]$ (for p, q distinct primes), and will allow us to derive separations of some of the classes which we introduce.

2.2 Arithmetic Classes

We give arithmetic versions of the definitions of the functions MOD m and MAJORITY.

Definition 4. *Define $\eta : \mathbb{N} \to \mathbb{N}$ so that $\eta(x)$ is 1 if $x = 0$, and equal to x otherwise.*

We define the following arithmetic functions:

[2] Although these classes are often defined so that NOT gates are allowed anywhere in the corresponding circuits, it is an easy exercise to show that such definitions are equivalent to ours. Our definitions will make proving various properties more convenient.

- *AMOD* m *on inputs* $x_1, \ldots, x_k$ *takes on the value* $\prod_{i=1}^{k} \eta(x_i)$ *if the number of* x_i*'s that are nonzero is a multiple of* m*; and the value* 0 *otherwise.*
- *AMAJORITY on inputs* $x_1, \ldots, x_k$ *takes on the value* $\prod_{i=1}^{k} \eta(x_i)$ *if the number of* x_i*'s that are nonzero is strictly greater than* $k/2$*; and the value* 0 *otherwise.*

Notice that these arithmetic functions coincide with their boolean counterparts on 0-1 valued inputs, typing issues aside.[3] With these new functions in hand, we can now define the arithmetic complexity classes to be studied in this paper. The definition is parallel to Definition 3.

Definition 5. *We define the following classes of functions from* $\{0,1\}^*$ *to* $\mathbb{N}$*; the functions* $+, *$ *denote the usual arithmetic sum and product in* $\mathbb{N}$*, and the input gates are interpreted as the values* $0, 1$ *in* $\mathbb{N}$*.*

- $\sharp AC^0$ *is the class of functions computable by* AC^0 *circuits over the basis* $\{+, *\}$*.*
- $\sharp AC^0[m_1, \ldots, m_l]$ *is the class of functions computable by* AC^0 *circuits over the basis* $\{+, *, AMOD\ m_1, \ldots, AMOD\ m_l\}$*. (This definition is for all* $\{m_1, \ldots, m_l\} \subseteq \mathbb{N}^+$*.)*
- $\sharp TC^0$ *is the class of functions computable by* AC^0 *circuits over the basis* $\{+, *, AMAJORITY\}$*.*

2.3 Operators on Arithmetic Classes

By way of some generic operators defined on arithmetic classes, we will obtain yet more complexity classes. The class *Diff* AC^0 was studied in [1, 7, 3]; we generalize it here.

Definition 6. *We define the following classes of functions from* $\{0,1\}^*$ *to* $\mathbb{Z}$*:*

- DiffAC^0 *is the class of functions expressible as the difference of two* $\sharp AC^0$ *functions. (That is,* $\mathrm{Diff}AC^0 = \{f - g : f, g \in \sharp AC^0\}$*.)*
- Diff$AC^0[m_1, \ldots, m_l]$ *is the class of functions expressible as the difference of two* $\sharp AC^0[m_1, \ldots, m_l]$ *functions.*

The following two operators allow us to obtain language classes from arithmetic classes.

Definition 7. *Suppose that* $\mathcal{C}$ *is a class of functions from* $\{0,1\}^*$ *to* $\mathbb{Z}$*.*

Define $\chi\mathcal{C}$ *to be the class of languages with characteristic function in* $\mathcal{C}$*.*

Define LowOrd$\mathcal{C}$ *to be the class of languages with characteristic function equal to the low order bit of a function in* $\mathcal{C}$ *(i.e., the value of a function in* $\mathcal{C}$ *modulo two).*

[3] In this paper, we will generally ignore such typing issues, and associate the boolean values $0, 1$ with the natural numbers $0, 1$.

2.4 Unambiguous circuits

We now observe a basic fact: that our arithmetic classes are at least as powerful as their boolean analogues.[4]

Lemma 1. *$FAC^0 \subseteq \sharp AC^0$, $FAC^0[m_1,\ldots,m_l] \subseteq \sharp AC^0[m_1,\ldots,m_l]$ (for all $m_1,\ldots,m_l \in \mathbb{N}$), and $FTC^0 = \sharp TC^0$.*

3 Language Classes

In this section, we study the language classes that result by allowing the operators χ and *LowOrd* to act on the arithmetic classes. We first show that the characteristic functions in each of the function classes $\sharp AC^0, \sharp AC^0[m_1,\ldots,m_l], \sharp TC^0$, are *exactly* the corresponding boolean language classes.[5]

Lemma 2. *$AC^0 = \chi\sharp AC^0$, $AC^0[m_1,\ldots,m_l] = \chi\sharp AC^0[m_1,\ldots,m_l]$, and $TC^0 = \chi\sharp TC^0$.*

This lemma immediately allows us to derive separations of the arithmetic classes.

Theorem 2. *For all primes p, $\sharp AC^0 \subsetneq \sharp AC^0[p]$. For all distinct primes p, q, $\sharp AC^0[p] \setminus \sharp AC^0[q]$ is nonempty.*

The other language classes we get by operating on $AC^0[m]$ and *Diff* $AC^0[m]$ fall into a chain of inclusions bounded below and above by $AC^0[m]$ and $AC^0[2,m]$, respectively.

Theorem 3. *For all odd positive integers m, we have $AC^0[m] \subseteq \chi\mathrm{Diff}AC^0[m] \subseteq \mathrm{LowOrdDiff}AC^0[m] = \mathrm{LowOrd}\sharp AC^0[m] \subseteq AC^0[2,m]$. If m is an even positive integer, all of the classes coincide.*

4 Arithmetic Classes

We now study the relationships of the arithmetic classes to each other, and to the boolean classes. We begin by proving that every function from an arithmetic class can be computed by circuits in *normal form*: circuits where the inputs (and hence outputs) of the AMOD (or AMAJORITY) gates always have 0-1 values. This notion of normal form will allow us to derive many facts concerning the structure of the classes $\sharp AC^0[m]$, and will also aid us in proving closure properties.

---

[4] Note that we view the classes FAC^0, $FAC^0[m_1,\ldots,m_l]$, and FTC^0 as classes of functions from $\{0,1\}^*$ to $\mathbb{N}$, in order to compare them with the corresponding arithmetic classes. To do this, we view a string of bits as a natural number in the usual way: the string $y_n \ldots y_0$ represents the natural number $\sum_{i=0}^{n} 2^i y_i$.

[5] By associating languages with their characteristic functions, we view the classes AC^0, $AC^0[m_1,\ldots,m_l]$, and TC^0 as language classes.

Definition 8. *Let us say that a* $\sharp AC^0[m_1, \ldots, m_l]$ *(*$\sharp TC^0$*) circuit is in* normal form *if on all inputs* x*, all AMOD (AMAJORITY) gates appearing in the circuit receive as inputs only the values* 0 *and* 1*. We say that a* $\sharp AC^0[m_1, \ldots, m_l]$ *(*$\sharp TC^0$*) circuit family is in normal form if all of its circuits are in normal form.*

Theorem 4. *For every function* f *in* $\sharp AC^0[m_1, \ldots, m_l]$ *(*$\sharp TC^0$*), there is a* $\sharp AC^0[m_1, \ldots, m_l]$ *(*$\sharp TC^0$*) circuit family in normal form computing* f*.*

In some sense, what we are showing is that only the boolean function MOD (MAJORITY) is required in arithmetic circuits, to capture the full power of $\sharp AC^0[m_1, \ldots, m_l]$ ($\sharp TC^0$).

We now show that the structure of the classes $\sharp AC^0[m]$ is isomorphic to the structure of the classes $AC^0[m]$, with respect to the subset relation $\subseteq$.

Theorem 5. *For all positive integers* m_1, m_2*,* $AC^0[m_1] \subseteq AC^0[m_2]$ *if and only if* $\sharp AC^0[m_1] \subseteq \sharp AC^0[m_2]$*.*

Corollary 1. *For all positive integers* m_1, m_2*,* $AC^0[m_1] = AC^0[m_2]$ *if and only if* $\sharp AC^0[m_1] = \sharp AC^0[m_2]$*.*

Thus far, we have sometimes restricted our attention to the classes $\sharp AC^0[m]$; the next theorem justifies this restriction, showing that any class $\sharp AC^0[m_1, \ldots, m_l]$ is equivalent to some class $\sharp AC^0[m]$.

Theorem 6. *For every subset* $\{m_1, \ldots, m_l\} \subseteq \mathbb{N}^+$*, we have* $\sharp AC^0[m_1, \ldots, m_l] = \sharp AC^0[\prod_{i=1}^{l} m_i] = \sharp AC^0[p_1, \ldots, p_k]$ *where* $p_1, \ldots, p_k$ *are the primes dividing* $\prod_{i=1}^{l} m_i$*.*

The following two corollaries can be derived by making simple modifications to the proof of Theorem 5, along with the fact that $AC^0[m] \subseteq TC^0$.

Corollary 2. *For all positive integers* m*,* $\sharp AC^0[m] \subseteq \sharp TC^0$*.*

Corollary 3. *For all positive integers* m*,* $AC^0[m] = TC^0$ *if and only if* $\sharp AC^0[m] = \sharp TC^0$*.*

The next theorem demonstrates that for a particular m, the $AC^0[m]$ versus TC^0 question is equivalent to the $FAC^0[m]$ versus $\sharp AC^0[m]$ question.

Theorem 7. *Let* m *be a positive integer. Either* $FAC^0[m] = \sharp AC^0[m] = FTC^0$*, or* $FAC^0[m] \subsetneq \sharp AC^0[m] \subsetneq FTC^0$*.*

5 Difference Classes

We now focus on the difference classes $\mathit{Diff}\,AC^0[m]$. First, we observe a separation.

Theorem 8. *For all odd primes* p*,* $\mathrm{Diff}AC^0 \subsetneq \mathrm{Diff}AC^0[p]$*.*

In the case of $\mathit{Diff}\,AC^0[2]$, we have class equality with $\mathit{Diff}\,AC^0$. Roughly, this is because the boolean function MOD 2 is contained in $\mathit{Diff}\,AC^0$.

Theorem 9. $\mathrm{Diff}AC^0 = \mathrm{Diff}AC^0[2]$.

There is the more general question of whether or not $\mathit{Diff}\,AC^0[m] = \mathit{Diff}\,AC^0[2m]$ for $m > 1$. We are not able to answer this question unconditionally, but can connect it to a question concerning language classes: equality holds if and only if $AC^0[2m] \subseteq \chi \mathit{Diff}\,AC^0[m]$.

Theorem 10. *Let m be a positive integer. The following are equivalent:*

1. $\sharp AC^0[2m] \subseteq \mathrm{Diff}AC^0[m]$
2. $\mathrm{Diff}AC^0[2m] = \mathrm{Diff}AC^0[m]$
3. $AC^0[2m] \subseteq \chi\mathrm{Diff}AC^0[m]$

6 Closure Properties

6.1 Maximum and Minimum

Theorem 11. *Let m be a positive integer. Neither $\sharp AC^0[m]$ nor $\mathrm{Diff}AC^0[m]$ is closed under MAX, unless $AC^0[2m] = TC^0$.*

The same theorem holds for MIN in place of MAX.

6.2 Division by a constant

For a positive integer c, we say that a function class $\mathcal{C}$ is closed under division by c if $f \in \mathcal{C}$ implies that $\lfloor \frac{f}{c} \rfloor \in \mathcal{C}$.

Theorem 12. *Let m be a positive integer and let p be a prime. The class $\sharp AC^0[p,(p-1),m]$ is closed under division by p.*

Corollary 4. *Let m be a positive integer and let p be a prime. The class $\mathrm{Diff}AC^0[p,(p-1),m]$ is closed under division by p.*

The idea behind the proofs of Theorem 12 and Corollary 4 is to modify $\sharp AC^0[p,(p-1),m]$ circuits inductively so that for each gate g in the original circuit, both the remainder of g (modulo p) and $\lfloor \frac{g}{p} \rfloor$ are computed in the modified circuit.[6] The MOD $p-1$ gates are used to compute the remainder of g from the remainders of its inputs g_i, when g is a multiplication gate (i.e., $g = \prod g_i$).

In the direction of proving converses of Theorem 12 and Corollary 4, we have the following theorem and corollary. Some new terminology is required for their statement. We call a function f *non-trivial* if it is not constant. We say that a function f is *symmetric* if it is non-trivial and for any $x_1, \ldots, x_n \in$

[6] In light of the equalities $\mathit{Gap}AC^0 = \mathit{Diff}\,AC^0 = \mathit{Diff}\,AC^0[2]$, Corollary 4, when instantiated with $p = 2$ and $m = 1$, gives [3, Theorem 8].

$\{0,1\}$ and $x'_1, \ldots, x'_{n'} \in \{0,1\}$, $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n'} x'_i$ implies $f(x_1, \ldots, x_n) = f(x'_1, \ldots, x'_{n'})$. We say that a symmetric function f has *period* k if $k \geq 1$, and for any $x_1, \ldots, x_n \in \{0,1\}$ and $x'_1, \ldots, x'_{n'} \in \{0,1\}$, $\sum_{i=1}^{n} x_i = k + \sum_{i=1}^{n'} x'_i$ implies $f(x_1, \ldots, x_n) = f(x'_1, \ldots, x'_{n'})$.

Theorem 13. *Let m be a positive integer. If $\sharp AC^0[m]$ is closed under division by p, then there exist symmetric functions with periods p and $p-1$ in $\mathrm{Diff}AC^0[m]$.*

Corollary 5. *Let m be a positive integer. If $\sharp AC^0[m]$ is closed under division by p, then MOD $p \in AC^0[2m]$, and there exists a divisor $q > 1$ of $p-1$ such that MOD $q \in AC^0[2m]$.*

6.3 Choose

We say that a function class $\mathcal{C}$ is closed under the choose operation if $f \in \mathcal{C}$ implies that $\binom{f}{k} \in \mathcal{C}$ for every positive integer k.

Theorem 14. *Let m be a positive integer. The classes $\sharp AC^0[m]$ and $\mathrm{Diff}AC^0[m]$ are closed under the choose operation.*

7 Future Work

We identify some open issues as possible avenues for future work.

- Are there combinatorial problems complete for the classes $\sharp AC^0[m]$? A characterization of $\sharp AC^0$ by the problem of counting paths in a certain class of graphs is given in [3].
- Can Corollary 5 be improved to show that under the hypotheses, MOD $p-1$ is in $AC^0[2m]$ (as opposed to just MOD q for a non-trivial divisor q of $p-1$)?
- Can one generalize the characterization of $\mathit{Diff}\,AC^0 = \mathit{Diff}\,AC^0[2]$ as $\mathit{Gap}AC^0$ (given in [7])?
  Access to the complex second roots of unity $\{1, -1\}$ seems to be what gives $\mathit{Gap}AC^0$ the ability to compute MOD 2. If we allow $\sharp AC^0$ the third roots of unity as constants, we obtain circuits that can compute both MOD 2 and MOD 3. What languages can be computed by $\sharp AC^0$ circuits when the pth roots of unity (for a prime p) are allowed as constants? Can give a general characterization of such circuits which includes the result of [7] as a particular case?
- Another possibility is to study the power of arithmetic circuits when the underlying algebraic structure is a group ring, such as $\mathbb{N}G$ for some finite group G. If the constants $\{1g : g \in G\}$ are allowed in circuits over $\mathbb{N}G$ that can multiply, then for all m dividing $|G|$, MOD m is computable.
- Suppose that $\mathcal{C}_1, \mathcal{C}_2$ are classes from $\{AC^0[m] : m \geq 2\} \cup \{TC^0\}$. It was demonstrated that $\mathcal{C}_1 = \mathcal{C}_2$ if and only if $\sharp\mathcal{C}_1 = \sharp\mathcal{C}_2$ (Corollaries 1 and 3). It is the case that $\sharp TC^0 = \sharp NC^1$ implies $TC^0 = NC^1$ (since $\chi\sharp TC^0 = TC^0$, and $\chi\sharp NC^1 = NC^1$). Does the converse hold?

– Of course, this list would not be complete without a request for new lower bounds. We conjecture that for odd primes p, $\chi \mathit{Diff}\,AC^0[p]$ is properly contained in $AC^0[2p]$ (and hence that $\mathit{Diff}\,AC^0[p] \neq \mathit{Diff}\,AC^0[2p]$). Can this be proved? More generally, can one prove any lower bounds using the classes $\chi \mathit{Diff}\,AC^0[p]$?

Acknowledgements. The author would like to thank Eric Allender for many interesting discussions. Riccardo Pucella deserves thanks for his useful comments on a draft of this paper.

References

1. M. Agrawal, E. Allender, and S. Datta. On TC^0, AC^0, and arithmetic circuits. In *Proceedings 12th Computational Complexity*, pages 134–148. IEEE Computer Society Press, 1997.
2. E. Allender. Making computation count: arithmetic circuits in the nineties. *SIGACT News*, 28(4):2–15, 1998.
3. E. Allender, A. Ambainis, D. A. Mix Barrington, S. Datta, and H. LêThanh. Bounded depth arithmetic circuits: Counting and closure. In *Proceedings 26th International Colloquium on Automata, Languages, and Programming (ICALP)*, Lecture Notes in Computer Science 1644, pages 149–158, 1999.
4. E. Allender, J. Jiao, M. Mahajan, and V. Vinay. Non-commutative arithmetic circuits: depth reduction and size lower bounds. *Theoretical Computer Science*, 209:47–86, 1998.
5. E. Allender and M. Ogihara. Relationships among PL, #L, and the determinant. *RAIRO – Theoretical Informatics and Applications*, 30:1–21, 1996.
6. C. Àlvarez and B. Jenner. A very hard log space counting class. *Theoretical Computer Science*, 107:3–30, 1993.
7. A. Ambainis, D. A. Mix Barrington, and H. LêThanh. On counting AC^0 circuits with negative constants. In *Proceedings 23rd Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 1450, pages 409–417, Berlin, 1998. Springer-Verlag.
8. H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.
9. R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings 19th Symposium on Theory of Computing*, pages 77–82. ACM Press, 1987.
10. S. Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Communications/Electronics/Information and Systems*, E75-D:116–124, 1992.
11. L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.
12. H. Venkateswaran. Circuit definitions of non-deterministic complexity classes. *SIAM Journal on Computing*, 21:655–670, 1992.
13. V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings 6th Structure in Complexity Theory*, pages 270–284. IEEE Computer Society Press, 1991.
14. H. Vollmer. *Introduction to Circuit Complexity*. Springer-Verlag, 1999.