Electronic Colloquium on Computational Complexity, Report No. 95 (2001)

Arithmetic Versions of Constant Depth Circuit Complexity Classes

Hubie Chen[*]

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 $\sharp TC^0$. Just as the boolean classes $AC^0[m]$ and TC^0 give a refined view of NC^1, our new arithmetic classes, which fall into the inclusion chain $\sharp AC^0 \subseteq \sharp AC^0[m] \subseteq \sharp TC^0 \subseteq \sharp NC^1$, refine $\sharp NC^1$. These new classes (along with $\sharp AC^0$) are also defined by constant-depth circuits, but the allowed gates compute arithmetic functions. We also introduce the classes *Diff* $AC^0[m]$ (differences of two $AC^0[m]$ functions), which generalize the class *Diff* AC^0 studied in previous work.

We study the structure of three hierarchies: the $\sharp AC^0[m]$ hierarchy, the *Diff* $AC^0[m]$ hierarchy, and a hierarchy of language classes. We prove class separations and containments where possible, and demonstrate relationships among the various hierarchies. For instance, we prove that the hierarchy of classes $\sharp AC^0[m]$ has exactly the same structure as the hierarchy of classes $AC^0[m]$:

$$AC^0[m] \subseteq AC^0[m'] \text{ iff } \sharp AC^0[m] \subseteq \sharp AC^0[m']$$

We also investigate closure properties of our new classes, which generalize those appearing in previous work on $\sharp AC^0$ and *Diff* AC^0.

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

ISSN 1433-8092

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 the number of accepting paths of a NP-machine on x [12]. The class $\sharp L$, defined similarly but with NL-machines, has also been studied [14, 11, 6, 5]. Both $\sharp P$ and $\sharp L$ can be obtained by "arithmetizing" the boolean circuit characterizations of NP and NL given in [13]. To arithmetize a boolean circuit, we propagate all NOT gates to the input level, and convert OR gates to $+$ gates and AND gates to $*$ 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 $\{0, 1\}^*$ to the natural numbers.

More recently, the arithmetic classes $\sharp AC^0, \sharp BP, \sharp NC^1$, and $\sharp SAC^1$ have been defined and studied [1, 7, 3, 8, 4, 14]. Other than $\sharp BP$, 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 $\sharp AC^0 \subsetneq \sharp BP \subseteq \sharp NC^1 \subseteq \sharp SAC^1$, which essentially mirrors the known relationships $AC^0 \subsetneq BP = NC^1 \subseteq SAC^1$ 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 $\sharp AC^0 \subseteq \sharp AC^0[m] \subseteq \sharp TC^0 \subseteq \sharp BP \subseteq \sharp NC^1$. 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 *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 *Diff* AC^0 [1, 7, 3] at the bottom. Moreover, we define two generic operators on arithmetic function classes; acting on the classes *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 *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 *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 *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 $\sharp AC^0[m]$ give an alternate decomposition[1] of TC^0. As a result, any question regarding the structure of the

[1]Note that by Theorem 12, $FAC^0[m]$ is properly contained in $\sharp AC^0[m]$, assuming that $AC^0[m] \neq TC^0$.

boolean $AC^0[m]$ hierarchy can be rephrased as a question concerning 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 [10]. Note that ours are not the first results providing an interface between boolean and arithmetic circuit complexity: 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 *Diff* $AC^0[m]$; focus is given to the question of whether or not *Diff* $AC^0[m] =$ *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 [15], 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 *[10] 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:

- *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]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.

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

2.3 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,* 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).*

2.4 Unambiguous circuits

We now observe a basic fact: that our arithmetic classes are at least as powerful as their boolean analogues.[4] This is done by showing that a boolean circuit can be converted into an arithmetic circuit computing the same values. The conversion roughly involves replacing each boolean gate by its corresponding arithmetic gate. However, an OR gate cannot simply be replaced by a $+$ gate, but must be "disambiguated" to ensure that the output is not strictly greater than one. We also observe that $\sharp TC^0$ has no additional power over its corresponding boolean class: $\sharp TC^0$ is equal to FTC^0. The proof of the following lemma is given in the appendix.

Lemma 2 $FAC^0 \subseteq \sharp AC^0$*,* $FAC^0[m_1, \ldots, m_l] \subseteq \sharp AC^0[m_1, \ldots, m_l]$ *(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 which 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 3 $AC^0 = \chi\sharp AC^0$*,* $AC^0[m_1, \ldots, m_l] = \chi\sharp AC^0[m_1, \ldots, m_l]$*, and* $TC^0 = \chi\sharp TC^0$*.*

Proof. The $\subseteq$ direction follows from Lemma 2, in all three cases. The $\supseteq$ direction follows, again in all three cases, by converting arithmetic circuits to boolean ones: $+$ gates are changed to OR gates, $*$ gates are changed to AND gates, AMOD gates are changed to MOD gates, and AMAJORITY gates are changed to MAJORITY gates. It is easy to verify that (on any input) a gate in the new circuit has value true if and only if the corresponding gate in the old circuit had a non-zero value. □

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

[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.

Theorem 4 *For all primes p, $\sharp AC^0 \subsetneq \sharp AC^0[p]$. For all distinct primes p, q, $\sharp AC^0[p] \backslash \sharp AC^0[q]$ is nonempty.*

Proof. We prove the second statement, which implies the first, since for any prime q, $\sharp AC^0 \subseteq \sharp AC^0[q]$. By Theorem 1, MOD $p \in AC^0[p] \setminus AC^0[q]$; thus, by Lemma 3, MOD $p \in \chi\sharp AC^0[p] \setminus \chi\sharp AC^0[q]$, from which it follows that MOD $p \in \sharp AC^0[p]$ and MOD $p \notin \sharp AC^0[q]$. □

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

Proof. If m is even, then $AC^0[m] = AC^0[2,m]$ (see the proof of Theorem 9), so we prove the containments.

By Lemma 2, $AC^0[m] \subseteq \sharp AC^0[m]$, and $\sharp AC^0[m] \subseteq$ *Diff* $AC^0[m]$ by definition; thus $AC^0[m] \subseteq \chi$*Diff* $AC^0[m]$.

The containment of χ*Diff* $AC^0[m]$ in LowOrd*Diff* $AC^0[m]$ follows from the general fact that, for any class of functions $\mathcal{C}$, $\chi\mathcal{C} \subseteq$ LowOrd$\mathcal{C}$.

If $h \in$ LowOrd*Diff* $AC^0[m]$, then h is the low order bit of $f - g$ for some $\sharp AC^0[m]$ functions f and g. The function $f + g$ is in $\sharp AC^0[m]$ and has the same low order bit as $f - g$, so $h \in$ LowOrd$\sharp AC^0[m]$.

The containment LowOrd$\sharp AC^0[m] \subseteq$ LowOrd*Diff* $AC^0[m]$ is immediate from the containment $\sharp AC^0[m] \subseteq$ *Diff* $AC^0[m]$.

It remains to show that LowOrd$\sharp AC^0[m] \subseteq AC^0[2,m]$. Suppose that h is the low order bit of f for some $\sharp AC^0[m]$ function f; using a circuit family for f, we create a new $AC^0[2,m]$ circuit family computing h to show that $h \in AC^0[2,m]$. This is done by induction on the depth of each circuit for f; for each gate g, our new circuit has gates $l(g)$ and $a(g)$ such that the low order bit of g is equal to $l(g)$, and g is nonzero if and only if $a(g)$ is true.

- For input gates and constants g, we let $l(g)$ and $a(g)$ be equal to g.
- If $g = \prod_{i=1}^{k} g_i$, then $l(g) = \bigwedge_{i=1}^{k} l(g_i)$ and $a(g) = \bigwedge_{i=1}^{k} a(g_i)$.
- If $g = \sum_{i=1}^{k} g_i$, then $l(g)$ is the NOT of the MOD 2 of $l(g_1), \ldots, l(g_k)$, and $a(g) = \bigvee_{i=1}^{k} a(g_i)$.
- If g is the AMOD m of $g_1, \ldots, g_k$, then $a(g)$ is the MOD m of $a(g_1), \ldots, a(g_k)$, and $l(g) = a(g) \wedge (\bigwedge_{i=1}^{k}(l(g_i) \vee \neg a(g_i)))$.

Let the output gate of the new circuit be $l(g_o)$, where g_o is the output gate of the original circuit. It is straightforward to verify by induction that for all gates g in the original circuit, $l(g)$ and $a(g)$ behave as described. □

If m is odd, it can be verified that there is another complexity class sandwiched inbetween the first two: the class of languages computable by "stratified" AC^0 circuits, circuits with MOD 2 and MOD m gates where from any path from an input gate to the output gate, no MOD 2 gate comes before a MOD m gate. When m is an odd prime, this stratified class (and hence χ*Diff* $AC^0[m]$) properly contains both $AC^0[2]$ and $AC^0[m]$: containment is clear from the definition, and propriety follows from Theorem 1.

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.

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 6 *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 $\sharp AC^0[m_1,\ldots,m_l]$ ($\sharp TC^0$).

Proof. We prove this for $\sharp AC^0[m_1,\ldots,m_l]$; the same proof with MAJORITY and AMAJORITY gates in place of MOD and AMOD gates applies to $\sharp TC^0$. Suppose we have a $\sharp AC^0[m_1,\ldots,m_l]$ circuit family $\{C_n : n \geq 1\}$. As in the proof of LowOrd$\sharp AC^0[m] \subseteq AC^0[2,m]$ (Theorem 5), we can create a $AC^0[m_1,\ldots,m_l]$ circuit family $\{C'_n : n \geq 1\}$ such that for each gate g in C_n, there is a corresponding gate $a'(g)$ in C'_n such that $a'(g)$ is 1 if and only if g is nonzero. We can add to C'_n gates $b'(g)$ such that $b'(g)$ computes $\neg a'(g)$, without increasing the size of C'_n by more than a constant factor and without increasing the depth of C'_n.

Now, apply Lemma 2 to $\{C'_n\}$ to obtain a circuit family $\{C''_n\}$ in $\sharp AC^0[m_1,\ldots,m_l]$; notice that the resulting circuits $\{C''_n\}$ are in normal form. For each gate g in one of the original circuits C_n, there are gates $a''(g)$ and $b''(g)$ in C''_i taking on only $0-1$ values such that g is nonzero if and only if $a''(g)=1$ if and only if $b''(g)=0$. For each n, append the original circuit C_n to the bottom of the circuit C''_n, letting the output gates of the copies of C_n be the output gates of our new circuit. Finally, we "normalize" these copies of C_n by replacing every AMOD m gate (in a copy of C_n) with inputs $g_1,\ldots,g_k$ with a circuit computing

$$(\text{AMOD}\, m(a''(g_1),\ldots,a''(g_k))) * \prod_{i=1}^{k}(b''(g_i)+g_i).$$

This does not affect the output of the circuit, but does ensure that AMOD m gates receive only the values 0 and 1 as inputs. □

In the previous section, classes of the form $\sharp AC^0[m]$ were shown to be distinct by using known separations in conjunction with the fact that $AC^0[m_1] \neq AC^0[m_2]$ implies that $\sharp AC^0[m_1] \neq \sharp AC^0[m_2]$. The converse of this fact is also true: $AC^0[m_1] = AC^0[m_2]$ implies that $\sharp AC^0[m_1] = \sharp AC^0[m_2]$. We prove a slightly stronger fact, 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 7 *For all positive integers* m_1, m_2, $AC^0[m_1] \subseteq AC^0[m_2]$ *if and only if* $\sharp AC^0[m_1] \subseteq \sharp AC^0[m_2]$.

Proof. The $\Leftarrow$ direction follows from Lemma 3, so we prove the $\Rightarrow$ direction. Let f be a function in $\sharp AC^0[m_1]$, and let $\{C_n : n \geq 1\}$ be a circuit family computing f. By Theorem 6, we can assume $\{C_n\}$ to be in normal form. Since $AC^0[m_1] \subseteq AC^0[m_2]$, there exists a boolean circuit family in $AC^0[m_2]$ computing MOD m_1. By Lemma 2, there is an arithmetic circuit family $\{D_j : j \geq 1\}$ in $\sharp AC^0[m_2]$ computing AMOD m_1 on $0-1$ values. We replace each AMOD m_1 gate in $\{C_n\}$ with fan-in j with the circuit D_j, obtaining a $\sharp AC^0[m_2]$ circuit family for f. □

Corollary 8 *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 justifi es this restriction, showing that any class $\sharp AC^0[m_1,\ldots,m_l]$ is equivalent to some class $\sharp AC^0[m]$.

Theorem 9 *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* $\prod_{i=1}^{l} m_i$*.*

Proof. It is easily verifi ed that Corollary 8 is true in the case of multiple AMOD gates: $AC^0[a_1, \ldots, a_r] = AC^0[b_1, \ldots, b_s]$ if and only if $\sharp AC^0[a_1, \ldots, a_r] = \sharp AC^0[b_1, \ldots, b_s]$ (for all subsets $\{a_1, \ldots, a_r\}, \{b_1, \ldots, b_s\} \subseteq \mathbb{N}^+$). This theorem then follows immediately from the folklore theorem giving the same result in the boolean case (see for example [9, Proposition 1]), which states that $AC^0[m_1, \ldots, m_l] = AC^0[\prod_{i=1}^{l} m_i] = AC^0[p_1, \ldots, p_k]$. □

The following two corollaries can be derived by making simple modifi cations to the proof of Theorem 7, along with the fact that $AC^0[m] \subseteq TC^0$.

Corollary 10 *For all positive integers* m*,* $\sharp AC^0[m] \subseteq \sharp TC^0$*.*

Note that Corollary 10 can also be proved directly by using the same "padding" technique used to show that $AC^0[m] \subseteq TC^0$.

Corollary 11 *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 12 *Let* m *be a positive integer. Either* $FAC^0[m] = \sharp AC^0[m] = FTC^0$*, or* $FAC^0[m] \subsetneq \sharp AC^0[m] \subsetneq FTC^0$*.*

Proof. The containments $FAC^0[m] \subseteq \sharp AC^0[m] \subseteq FTC^0$ follow from Lemma 2 and Corollary 10. If $AC^0[m] = TC^0$, then $FAC^0[m] = FTC^0$, so assume $AC^0[m] \neq TC^0$.

By Corollary 11, $\sharp AC^0[m] \subsetneq FTC^0$. If $FAC^0[m] = \sharp AC^0[m]$, then $2^{\sum_{i=1}^{n} x_i} = \prod_{i=1}^{n}(1 + x_i)$ can be computed in $\sharp AC^0[m]$ and hence $FAC^0[m]$. The expression $2^{\sum_{i=1}^{n} x_i}$ is strictly greater than $2^{n/2}$ if and only if the MAJORITY of $x_1, \ldots, x_n$ is true. But, checking whether or not a string is is strictly greater than $2^{n/2}$ is easily done in $AC^0[m]$, so we have $TC^0 \subseteq AC^0[m]$, contradicting our assumption. □

5 Difference Classes

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

Theorem 13 *For all odd primes* p*,* Diff$AC^0 \subsetneq$ Diff$AC^0[p]$*.*

Proof. The function MOD p is not in $AC^0[2]$ by Theorem 1. By Theorem 5 with $m = 1$, χ*Diff* $AC^0 \subseteq AC^0[2]$ and hence MOD $p \notin \chi$*Diff* AC^0, implying that MOD $p \notin$ *Diff* AC^0. □

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

Theorem 14 Diff$AC^0 =$ Diff$AC^0[2]$.

Proof. The $\subseteq$ direction is trivial, so we prove *Diff* $AC^0[2] \subseteq$ *Diff* AC^0. It is shown in [7] that *Diff* $AC^0 =$ *GapAC*^{0}, where *GapAC*^{0} is defi ned as the class of functions computable by $\sharp AC^0$ circuits with -1 allowed as a constant. Since *GapAC*^{0} is closed under subtraction, it suffi ces to show that $\sharp AC^0[2] \subseteq$ *GapAC*^{0}.

Suppose f is in $\sharp AC^0[2]$. Let $\{C_n : n \geq 1\}$ be a $\sharp AC^0[2]$ circuit family computing f, assumed to be in normal form by Theorem 6. Replace every MOD 2 gate in $\{C_n\}$ with input gates $g_1, \ldots, g_k$ by a *GapAC*^{0}

circuit computing the expression $\sum_{i=1}^{k}(\prod_{1\leq j<i}(1-2g_j))g_i$, which is equal to the MOD 2 function on $0-1$ inputs. The new circuit family is in *GapAC*^{0} and computes f. $\square$

There is the more general question of whether or not *Diff AC*$^0[m]$ = *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 $AC^0[2m] \subseteq \chi$*Diff AC*$^0[m]$.

Theorem 15 *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. $AC^0[2m] \subseteq \chi\mathrm{Diff}AC^0[m]$

The proof of this theorem is given in the appendix.

6 Closure Properties

6.1 Maximum and Minimum

Theorem 16 *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 proof is essentially identical to the that of [3, Theorem 1][6]. We include a proof in the appendix for completeness. A symmetric argument applies to MIN.

Theorem 17 *Let m be a positive integer. Neither $\sharp AC^0[m]$ nor $\mathrm{Diff}AC^0[m]$ is closed under MIN, unless $AC^0[2m] = TC^0$.*

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 18 *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 19 *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.*

Both Theorem 18 and Corollary 19 are proved in the appendix[7]. The idea behind the proofs 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 modifi ed circuit. 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 18 and Corollary 19, we have the following theorem and corollary, also proved in the appendix. Some new terminology is required for their statement.

[6][3, Theorem 1] is our theorem in the case $m = 1$ (without the "unless" clause: by Theorem 1, $AC^0[2] \neq TC^0$).

[7]In light of the equalities *GapAC*^{0} = *Diff AC*^{0} = *Diff AC*$^0[2]$, Corollary 19, when instantiated with $p = 2$ and $m = 1$, gives [3, Theorem 8].

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 $x_1, \ldots, x_n \in \{0,1\}$ 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 20 *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 21 *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 22 *Let m be a positive integer. The classes $\sharp AC^0[m]$ and* $\mathrm{Diff}AC^0[m]$ *are closed under the choose operation.*

The proof is similar to the proof of [1, Theorem 9], which shows that $\sharp AC^0$ and *Diff* AC^0 are closed under the choose operation. Their proof demonstrates that $\binom{\sum_{i=1}^{n} f_i(x)}{k}$ and $\binom{\prod_{i=1}^{n} f_i(x)}{k}$ can be computed in $\sharp AC^0$ from the values $\{\binom{f_i(x)}{j} : 0 \leq j \leq k, i = 1, \ldots, n\}$. (The result then follows by induction on the depth of the circuit.) It is easy to see that this proof applies after noting that for circuits in normal form, $\binom{\text{AMOD } m(f_1(x), \ldots, f_n(x))}{k}$ is equal to 1 when $k = 0$, AMOD $m(f_1(x), \ldots, f_n(x))$ when $k = 1$, and 0 when $k \geq 2$. We refer the reader to their proof for details.

6.4 Weak Product

We say that a function class $\mathcal{C}$ is closed under weak product if for every $k \geq 1$, $f \in \mathcal{C}$ implies that $\prod_{i=1}^{n^k} f(x, i) = g(x) \in \mathcal{C}$. The following is proved in [7].

Theorem 23 *[7] There exists a pair of $\sharp AC^0$ circuits f, g on inputs $\{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\}$ such that for all $\mathbb{N}$-assignments to the $2n$ input variables, $\prod_{i=1}^{n}(x_i - y_i) = f - g$.*

From this, closure under weak product of the classes *Diff* $AC^0[m]$ follows immediately.

Theorem 24 *Let m be a positive integer. The class* $\mathrm{Diff}AC^0[m]$ *is closed under weak product.*

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 21 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 *Diff* $AC^0 =$ *Diff* $AC^0[2]$ as *Gap*AC^0 (given in [7])?

  If we leave the definition of AMOD m untouched, allowing the constant -1 in $\sharp AC^0[m]$ circuits, rather uninterestingly, gives us the class FTC^0. This follows from the characterization of TC^0 given in [1]; if f and g are $\sharp AC^0$ functions, we can check if they are equal (using such augmented $\sharp AC^0[m]$ circuits) by taking the AMOD m of m 1's and $f - g$, which is 0 if $f - g$ is nonzero, and 1 otherwise.

  Access to the complex second roots of unity $\{1, -1\}$ seems to be what gives *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. It is curious to note that multiplying elements from the group ring $\mathbb{N}G$ amounts to counting accepting paths of a deterministic automaton solving the word problem for G. That is, the coefficient of the identity element of G in the product $r_1 \cdots r_k$ is the number of strings accepted by the mentioned automaton, when the frequency of a string $g_1 \ldots g_k \in G^k$ is specified by the product of coefficients of the g in r_i. Perhaps there is some connection to the class $\sharp BP$, studied in [8].

- 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 8 and 11). 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, χ*Diff* $AC^0[p]$ is properly contained in $AC^0[2p]$ (and hence that *Diff* $AC^0[p] \neq$ *Diff* $AC^0[2p]$). Can this be proved? More generally, can one prove any lower bounds using the classes χ*Diff* $AC^0[p]$?

Acknowledgements. The author would like to thank Eric Allender for many interesting discussions. Riccardo Pucella deserves thanks for 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] H. Chen. Polynomial programs and the Razborov-Smolensky method. Technical Report 01-067, Electronic Colloquium on Computational Complexity, 2001.

[10] 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.

[11] 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.

[12] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[13] H. Venkateswaran. Circuit definitions of non-deterministic complexity classes. *SIAM Journal on Computing*, 21:655–670, 1992.

[14] 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.

[15] H. Vollmer. *Introduction to Circuit Complexity*. Springer-Verlag, 1999.

A Proofs

A.1 Proof of Lemma 2

Proof. The containment $\sharp TC^0 \subseteq FTC^0$ follows from the fact that iterated sum and iterated product are in TC^0 [15].

The containment $FAC^0 \subseteq \sharp AC^0$ was proved in [1]; the proofs of the containments $FAC^0[m_1, \ldots, m_l] \subseteq \sharp AC^0[m_1, \ldots, m_l]$ and $FTC^0 \subseteq \sharp TC^0$ are essentially the same, and are given here for completeness. For $i \geq 1$, let C_i be a depth two circuit computing $\prod_{j=1}^{i}(1+1)$, which takes on the value 2^i. Suppose each bit of a function f in one of these boolean circuit classes can be computed in the corresponding arithmetic class; then, f is contained in the arithmetic class by computing $\sum_{i=0}^{|f|-1}(C_i * f_i)$. Thus, it suffices to show that each of the boolean circuit classes AC^0, $AC^0[m_1, \ldots, m_l]$, and TC^0 are contained in their arithmetic counterparts.

To prove this, we show that a boolean circuit can be converted into an arithmetic circuit computing the same function, as follows. First, modify the boolean circuit so that for each gate g, there is another gate computing $\neg g$. This can be done without increasing the depth and by increasing the size by only a constant factor. Then, replace all OR gates (assumed to have inputs $g_1, \ldots, g_k$) with the circuit $\sum_{i=1}^{k}(g_i * (\prod_{j=1}^{i-1} \neg g_j))$; and, change all AND gates to $*$ gates, MOD gates to AMOD gates, and MAJORITY gates to AMAJORITY gates. It is easily verified by induction that for each gate in the original circuit, the corresponding gate in the new circuit computes the same value. $\square$

A.2 Proof of Theorem 15

Proof. $(1) \Rightarrow (2)$: obvious from closure of *Diff*$AC^0[m]$ under subtraction.

$(2) \Rightarrow (3)$: (2) immediately implies that χ*Diff*$AC^0[2m] \subseteq \chi$*Diff*$AC^0[m]$, and we have $AC^0[2m] = \chi\sharp AC^0[2m] \subseteq \chi$*Diff*$AC^0[2m]$ by Lemma 3.

$(3) \Rightarrow (1)$: Let $\{C_n : n \geq 1\}$ be a $\sharp AC^0[2m]$ circuit family in normal form. As in the proof of LowOrd$\sharp AC^0[m] \subseteq AC^0[2, m]$ (Theorem 5), we can create a $AC^0[2m]$ circuit family $\{C'_n : n \geq 1\}$ such that for each gate g in C_n, there is a corresponding gate $a'(g)$ in C'_n such that $a'(g)$ is 1 if and only if g is nonzero. For each circuit C'_n, number the gates arbitrarily and define $f(x, i)$ to be equal to the ith gate of circuit $C'_{|x|}$ on inputs x. The function f is in $AC^0[2m]$; by (3), there exist functions f_1, f_2 in $\sharp AC^0[m]$ with $f = f_1 - f_2$.

We now modify each circuit C_n, replacing each gate g with a pair of gates $X(g), Y(g)$ such that on all inputs, the value of g is equal to the value of $X(g) - Y(g)$. The modified circuit will be a $\sharp AC^0[m]$ circuit, demonstrating that the function computed by C_n is in *Diff*$AC^0[m]$. The modification is done by induction on the depth of the circuit.

- For input gates and constants g, set $X(g) = g$ and $Y(g) = 0$.
- If $g = \sum_{i=1}^{k} g_i$, then $X(g) = \sum_{i=1}^{k} X(g_i)$ and $Y(g) = \sum_{i=1}^{k} Y(g_i)$.
- If $g = \prod_{i=1}^{k} g_i$, then $X(g)$ and $Y(g)$ are computed from the values $X(g_i)$ and $Y(g_i)$ by exactly the circuits of Theorem 23.
- If $g =$ AMOD $m(g_1, \ldots, g_k)$, then let $X(g)$ and $Y(g)$ be circuits computing $f_1(x, i)$ and $f_2(x, i)$, respectively, where i was the number given to the gate $a'(g)$ (of C'_n) corresponding to g.

$\square$

A.3 Proof of Theorem 16

Proof. Let $f(x_1, \ldots, x_n) = \sum_{i=1}^n x_i$, and let $g(x_1, \ldots, x_n) = \lfloor \frac{n}{2} \rfloor$. Let x' denote the result of changing the first 1 in x to a 0 (if such a bit exists). The number of 1's in $x = x_1 \ldots x_n$ is less than or equal to $\lfloor \frac{n}{2} \rfloor$ if and only if the low-order bits of MAX$(f(x), g(x))$ and MAX$(f(x'), g(x'))$ are equal. The low-order bits of a $\sharp AC^0[m]$ or *Diff* $AC^0[m]$ function are computable in $AC^0[2m]$ (Theorem 5). Thus, if either of these classes are closed under MAX, we have MAJORITY $\in AC^0[2m]$, from which it follows that $AC^0[2m] = TC^0$. □

A.4 Proof of Theorem 18

Proof. We denote the MOD a function by MOD$_a$, and let $r : \mathbb{N} \rightarrow \{0, \ldots, p-1\}$ denote the mapping taking an integer to its remainder when divided by p.

Let f be a function in $\sharp AC^0[p, (p-1), m]$, and let $\{C_n : n \geq 1\}$ be a circuit family computing f; by Theorem 6, we assume $\{C_n\}$ to be in normal form. For each $n \geq 1$, we show by induction on the depth of C_n that for every gate g of C_n, $r(g)$ (represented in unary) and $\lfloor \frac{g}{p} \rfloor$ can be computed in constant depth.

- For input gates and constants g, $\lfloor \frac{g}{p} \rfloor = 0$ and $r(g) = g$.
- If $g = \sum_{i=1}^k g_i$, then $\lfloor \frac{g}{p} \rfloor = \lfloor \frac{\sum_{i=1}^k \sum_{j=1}^{p-1} [r(g_i) \geq j]}{p} \rfloor + \sum_{i=1}^k \lfloor \frac{g_i}{p} \rfloor$. The left summand can be expanded using the identity $\lfloor \frac{\sum_{s=1}^t x_s}{p} \rfloor = \sum_{s=p}^t \text{MOD}_p(x_1, \ldots, x_s) x_s$, which holds when the x_s are $0-1$ valued variables. $r(g)$ can be computed from $r(g_1), \ldots, r(g_k)$ using a MOD p gate.
- If $g = \prod_{i=1}^k g_i$, then $r(g)$ can be computed from $r(g_1), \ldots, r(g_k)$ using a MOD $p-1$ gate; recall that the multiplicative group of the integers modulo p is isomorphic to the cyclic group of order $p-1$ (for p prime). Note also that $r(g_i \cdots g_k)$ can be computed similarly (for any $1 \leq i \leq k$).

  We have the identity

  $$\begin{aligned} \lfloor \tfrac{g_1 \cdots g_k}{p} \rfloor &= \lfloor \tfrac{g_1}{p} \rfloor g_2 \cdots g_k + \textstyle\sum_{j=1}^{p-1} [r(g_1) = j] \lfloor \tfrac{j g_2 \cdots g_k}{p} \rfloor \\ &= \lfloor \tfrac{g_1}{p} \rfloor g_2 \cdots g_k + \textstyle\sum_{j=1}^{p-1} [r(g_1) = j] (j \lfloor \tfrac{g_2 \cdots g_k}{p} \rfloor + \lfloor \tfrac{j r(g_2 \cdots g_k)}{p} \rfloor) \\ &= \lfloor \tfrac{g_1}{p} \rfloor g_2 \cdots g_k + (\textstyle\sum_{j=1}^{p-1} [r(g_1) = j] \lfloor \tfrac{j r(g_2 \cdots g_k)}{p} \rfloor) + (\textstyle\sum_{j=1}^{p-1} j [r(g_1) = j]) \lfloor \tfrac{g_2 \cdots g_k}{p} \rfloor. \end{aligned}$$

  Applying this identity $k-1$ times gives us an expression for $\lfloor \frac{g}{p} \rfloor$ which can be evaluated in constant depth and depends only on the values g_i, $\lfloor \frac{g_i}{p} \rfloor$, $r(g_i)$, and $r(g_i \cdots g_k)$, for $i = 1, \ldots, k$. (Note that given $j \in \{1, \ldots, p-1\}$ and $r(g_2, \cdots g_k)$, $\lfloor \frac{j r(g_2 \cdots g_k)}{p} \rfloor$ can be computed using a lookup table.)
- If $g = \text{MOD}_a(g_1, \ldots, g_k)$, then g takes on only the values 0 and 1, as we assumed $\{C_n\}$ to be in normal form. Thus, $\lfloor \frac{g}{p} \rfloor = 0$ and $r(g) = g$.

□

A.5 Proof of Corollary 19

Proof. Suppose $h \in$ *Diff* $AC^0[p, (p-1), m]$. Let f and g be $\sharp AC^0[p, (p-1), m]$ functions such that $h = f - g$. Letting r denote the remainder upon division by p as in the previous proof, we observe the identity

$$\lfloor \frac{f-g}{p} \rfloor = \lfloor \frac{f}{p} \rfloor - \lfloor \frac{g}{p} \rfloor + \lfloor \frac{r(f) - r(g)}{p} \rfloor = \lfloor \frac{f}{p} \rfloor - (\lfloor \frac{g}{p} \rfloor + [r(g) > r(h)]).$$

By the proof of Theorem 18, $\lfloor\frac{f}{p}\rfloor$, $\lfloor\frac{g}{p}\rfloor$, $r(g)$, and $r(h)$ are all in $\sharp AC^0[m, p, (p-1)]$, so by the identity, $\lfloor\frac{h}{p}\rfloor$ is in *Diff* $AC^0[m, p, (p-1)]$. $\square$

A.6 Proof of Theorem 20

Proof. Let $b \in \mathbb{N}^+$ be a generator for the multiplicative group of integers mod p, that is, a positive integer such that no two of $1, b, b^2, \ldots, b^{p-2}$ are congruent mod p. Set $b' = b - 1$. Let $f = \sum_{i=1}^n x_i$, and let $g = \prod_{i=1}^n (1 + b' x_i)$. Observe that both f and g are in $\sharp AC^0$, and that $g = b^{\sum_{i=1}^n x_i}$. The functions $f - p\lfloor\frac{f}{p}\rfloor$ and $g - p\lfloor\frac{g}{p}\rfloor$ are symmetric with periods p and $p-1$, respectively. $\square$

A.7 Proof of Corollary 21

Proof. Let f_0, g_0 be the functions in *Diff* $AC^0[m]$ in given by Theorem 20. Let f_1 and g_1 be equal to the lowest order bits of f_0 and g_0, respectively. It is easily verifi ed that f_1 and g_1 are symmetric. By Theorem 5, f_1 and g_1 are in $AC^0[2m]$.

Let us say that a symmetric function f with some period has *minimum period* k if there is no $m < k$ such that f has period m. (Note that the minimum period must be strictly greater than 1, since we assume symmetric functions to be non-trivial.) In general, a function with period m has minimum period dividing m. Thus, in particular, the functions f_1 and g_1 have minimum period p and q for some divisor $q > 1$ of $p - 1$.

To check whether or not $x_1 \ldots x_n \in$ MOD q, we check whether or not $g_1(j + \sum_{i=1}^n x_i) = g_1(j)$ for all $j = 0, \ldots, q-1$; this can be done in $AC^0[2m]$. (When f is symmetric and $s \in \mathbb{N}^+$, we let $f(s)$ denote the value of f on any inputs $x_1, \ldots, x_n$ such that $\sum_{i=1}^n x_i = s$.) The same idea applies to checking membership in MOD p. $\square$