Dual VP Classes

DUAL VP CLASSES

Eric Allender, Anna Gál, and Ian Mertz

September 24, 2016

Abstract. We consider the complexity class ACC^1 and related families of arithmetic circuits. We prove a variety of collapse results, showing several settings in which no loss of computational power results if fan-in of gates is severely restricted, as well as presenting a natural class of arithmetic circuits in which no expressive power is lost by severely restricting the algebraic degree of the circuits. We draw attention to the strong connections that exist between ACC^1 and VP, via connections to the classes $\mathsf{CC}^1[m]$ for various m. These results tend to support a conjecture regarding the computational power of the complexity class VP over finite algebras, and they also highlight the significance of a class of arithmetic circuits that is in some sense dual to VP. In particular, these dual-VP classes provide new characterizations of ACC^1 and TC^1 in terms of circuits of semiunbounded fan-in. As a corollary, we show that $\mathsf{ACC}^i = \mathsf{CC}^i$ for all $i \geq 1$.

Keywords. Circuit Complexity, Arithmetic circuits, Semiunbounded circuits, Threshold Circuits

Subject classification. F.1.3 Complexity Measures and Classes

1. Introduction

Most of the well-studied subclasses of P are defined in terms of Boolean or arithmetic circuits. The question of the relative power of $\mathsf{NC}^1, \mathsf{LogCFL}$, and AC^1, or of $\#\mathsf{NC}^1$ and $\#\mathsf{LogCFL}$ boils down to the question of how the computational power of a (log-depth, polynomial-size) circuit model depends on the *fan-in* of gates in the model.

In this paper we highlight the significance of a class of semi-unbounded fan-in arithmetic circuits that is in some sense dual to

the well-studied class VP. Our main contribution is to present several settings where fan-in can be severely restricted for log-depth, polynomial-size circuits, with *no* loss of computational power. We also present a natural class of arithmetic circuits in which no expressive power is lost by severely restricting the algebraic degree of the circuits.

1.1. Duality. Semiunbounded fan-in circuits play an important role in computational complexity theory. Over the Boolean semiring, logarithmic depth polynomial-size semiunbounded fan-in circuits (with bounded fan-in AND gates and unbounded fan-in OR gates, with NOT gates only at the input level) characterize the complexity class LogCFL, also known as SAC^1, which has been the subject of numerous investigations (Gál 1995; Gál & Wigderson 1996; Reinhardt & Allender 2000; Venkateswaran 1991). Over $\mathbb{F}_p$, logarithmic depth polynomial-size semiunbounded fan-in circuits (with bounded fan-in multiplication gates and unbounded fan-in addition gates) characterize the complexity class $\mathsf{VP}(\mathbb{F}_p)$, the study of which was initiated by (Valiant 1979). $\mathsf{VP}(\mathbb{F}_p)$ is usually defined as poly-size arithmetic circuits with degree $n^{O(1)}$; its characterization by logarithmic depth semiunbounded fan-in arithmetic circuits was shown in Allender *et al.* (1998); Valiant *et al.* (1983). These classes have received a lot of attention since then (e.g., (Bürgisser 1999, 2000; Gál & Wigderson 1996; Koiran & Perifel 2011)).

Because LogCFL is closed under complement (Borodin *et al.* 1989), it can be characterized in terms of semiunbounded fan-in circuits by restricting either the AND gates or the OR gates to have bounded fan-in. It is unknown if there is any other algebraic structure for which a similar phenomenon occurs. In particular, it is not known how the complexity of functions in $\mathsf{VP}(\mathbb{F}_p)$ compares to that of the functions in the classes defined by logarithmic depth polynomial-size semiunbounded fan-in circuits with bounded fan-in $+$ gates and unbounded fan-in $\times$ gates.

A large part of the motivation for this paper is to understand the computational power of these semiunbounded fan-in circuit classes, which are in some sense dual to Valiant's classes $\mathsf{VP}(\mathbb{F}_p)$. We use the notation $\Lambda\mathsf{P}(\mathbb{F}_p)$ to refer to the class of problems characterized by logarithmic depth polynomial-size semiunbounded fan-

in circuits with bounded fan-in addition gates and unbounded fan-in multiplication gates. Formal definitions appear in Section 2. We show that each class $\Lambda\mathsf{P}(\mathbb{F}_p)$ corresponds exactly to a particular subclass of ACC^1, and that the union over all p of $\Lambda\mathsf{P}(\mathbb{F}_p)$ is exactly equal to ACC^1 (Corollary 3.2). Our results extend to larger depths as well, yielding characterizations of ACC^i for every i in terms of semiunbounded fan-in arithmetic circuits (Corollary 3.7).

Note that here (and in several of our other results) we relate Boolean classes (e.g. ACC^1) to arithmetic circuit classes. For this purpose we work with the *Boolean part* of classes defined by arithmetic circuits. The VP classes are usually studied as classes of *polynomials*, but it is also common to study the Boolean part of VP over a given semiring R, where (following Blum *et al.* (1998)), the Boolean part of an arithmetic circuit class is the class of languages whose characteristic functions are computed by circuits in the class. Especially over finite fields, there is little to distinguish VP from its Boolean part. There is a large literature exploring the connections between Boolean and arithmetic circuit complexity; see Vollmer (1999).

1.2. Fan-in Reductions. Our results mentioned above relating ACC^1 to the semiunbounded fan-in $\Lambda\mathsf{P}$ classes can be viewed as fan-in reductions. We explore this in more detail, and obtain fan-in reductions for TC^1 as well. In addition to arithmetic circuits, we also consider fan-in reductions for both ACC^1 and TC^1 in the Boolean setting, considering their characterizations by circuits with AND and OR gates, along with MOD_m gates.

1.2.1. Fan-in Reductions in Arithmetic Circuits. First we note that both ACC^1 and TC^1 are characterized by unbounded fan-in arithmetic circuits, then we observe that unbounded fan-in is not necessary for these characterizations.

We show here that $\mathsf{ACC}^1 = \bigcup_p \#\mathsf{AC}^1(\mathbb{F}_p)$ (Corollary 3.5). On the other hand, the *semiunbounded* fan-in model, where the $+$ gates have fan-in two, also yields ACC^1 (Corollary 3.2).

The complexity class TC^1 is defined by polynomial-size threshold circuits of logarithmic depth. (Reif & Tate 1992) gave an alternative characterization of TC^1 in terms of unbounded fan-in

arithmetic circuits of logarithmic depth where the circuits for inputs of size n operate over the field $\mathbb{F}_{p_n}$, where p_n is the n-th prime. (See also the discussion of Reif and Tate's work in Buhrman *et al.* (2014).) Using standard notation (reviewed in Section 2), this characterization can be stated as $\mathsf{TC}^1 = \#\mathsf{AC}^1(\mathbb{F}_{p_n})$. We show that no computational power is lost (modulo logspace-Turing reductions) by restricting the fan-in of the + gates in this setting: We show that $\mathsf{TC}^1 = \mathsf{L}^{\Lambda\mathsf{P}(\mathbb{F}_{p_n})}$ (Theorem 4.3).

1.2.2. Fan-in Reductions in Boolean Circuits. The usual definition of ACC^1 is in terms of polynomial size logarithmic depth circuits with *unbounded* fan-in AND and OR gates, along with MOD_m gates for different m. We observe here that TC^1 has an analogous characterization: $\mathsf{TC}^1 = \mathsf{AC}^1[p_n]$ (that is, AC^1 circuits with MOD_{p_n} gates, Theorem 4.4). We show for both ACC^1 and TC^1 that unbounded fan-in is not necessary for the AND and OR gates; they can both be restricted to constant fan-in. But then, noting that MOD gates can simulate bounded fan-in AND and OR gates, we get characterizations of both ACC^1 (Theorem 4.12) and TC^1 (Theorem 4.4) by logarithmic depth polynomial size circuits using only unbounded fan-in MOD gates. These characterizations also carry over for ACC^i and TC^i for every $i \geq 1$ (Corollary 4.5 and Corollary 4.14). In particular, for all $i \geq 1, \mathsf{ACC}^i = \mathsf{CC}^i$. (For definitions of these circuit complexity classes, see Section 2 and Section 3.)

1.3. Algebraic Degree. Immerman and Landau conjectured that computing the determinant of integer matrices is complete for TC^1 (Immerman & Landau 1995). This would have several consequences, including providing a characterization of TC^1 in terms of $\mathsf{VP}(\mathbb{Q})$. (Buhrman *et al.* 2014) have argued that the Immerman-Landau conjecture is unlikely, in that this would imply that arbitrary polynomials having degree $n^{O(\log n)}$ and polynomial-size arithmetic circuits mod p_n could be simulated by arithmetic circuits of *much lower degree* over $\mathbb{Q}$. This raises the question: When can high-degree polynomials over one algebra be simulated by low-degree polynomials over another?

Our degree-reduction theorem (Corollary 4.10) gives one nat-

ural class of polynomials of degree $n^{O(\log n)}$ over one algebra ($\mathbb{F}_2$) that *can* be simulated by polynomials having much smaller degree. We show that restricting the fan-in of $\times$ gates in $\#\mathsf{AC}^1(\mathbb{F}_2)$ circuits to be logarithmic results in *no loss of expressive power*; the restricted class (whose polynomials have algebraic degree only $n^{O(\log\log n)}$) represents the same class of functions as the unrestricted class (with degree $n^{O(\log n)}$). We believe that this weakens the arguments against the Immerman-Landau conjecture that were raised in Buhrman *et al.* (2014), and we suspect that there are other such examples, where restricting the fan-in of $\times$ gates causes no loss of power. We also see no reason why degree $n^{O(\log\log n)}$ should be optimal. Lowering the degree to $n^{O(1)}$ would imply $\#\mathsf{AC}^1(\mathbb{F}_2) = \mathsf{AC}^1[2] = \mathsf{VP}(\mathbb{F}_2)$. (We omit "Boolean part" if it causes no confusion.)

1.4. A Conjecture. We conjecture that ACC^1 is precisely the class of languages logspace-Turing reducible to $\bigcup_m \mathsf{VP}(\mathbb{Z}_m)$. If the conjecture is true, then ACC^1 can be defined using either kind of semiunbounded fan-in circuits, with bounded fan-in $+$ or bounded fan-in $\times$.

ACC^1 and VP are familiar to many theoreticians. (The complexity class ACC^0 has received a great deal of attention over the years – notably including the nonuniform lower bound presented in Williams (2014) – and the corresponding class of logarithmic-depth circuits was familiar, even though comparatively little has been written about ACC^1. One example is Moore *et al.* (2000).) We believe that we are the first to conjecture that these two classes are very closely related.

2. Preliminaries

We assume that the reader is familiar with Boolean circuit complexity classes such as AC^0 and ACC^0; a good source for this background material is the excellent text by (Vollmer 1999). The following standard notation is used by Vollmer for circuit complexity classes, and we follow those conventions here:[1]

[1]We will also refer to the complexity classes $\mathsf{CC}^i[m]$, which are not discussed in Vollmer (1999). We defer the definition of those classes to Section 3,

DEFINITION 2.1. ∘ AC^i *is the class of languages accepted by logtime-uniform circuit families of polynomial size and depth* $O(\log^i n)$*, consisting of unbounded fan-in* AND*, and* OR *gates, along with* NOT *gates.*

∘ $\mathsf{AC}^i[m]$ *is defined as* AC^i*, but in addition unbounded fan-in* MOD_m *gates are allowed, which output 1 iff the number of input wires carrying a value of 1 is a multiple of* m*.*

∘ *For any finite set* $S \subset \mathbb{N}$*,* $\mathsf{AC}^i[S]$ *is defined analogously to* $\mathsf{AC}^i[m]$*, but now the circuit families are allowed to use* MOD_r *gates for any* $r \in S$*. It is known that, for any* $m \in \mathbb{N}$*,* $\mathsf{AC}^i[m] = \mathsf{AC}^i[\mathsf{Supp}(m)]$*, where – following the notation of Corrales-Rodrigáñez & Schoof (1997) –* $\mathsf{Supp}(m) = \{p : p$ *is prime and* p *divides* $m\}$ *(Smolensky 1987). Thus, in particular* $\mathsf{AC}^i[6] = \mathsf{AC}^i[2,3]$ *and* $\mathsf{AC}^i = \mathsf{AC}^i[\emptyset]$*. (When it will not cause confusion, we omit unnecessary brackets, writing for instance* $\mathsf{AC}^i[2,3]$ *instead of* $\mathsf{AC}^i[\{2,3\}]$*.)*

∘ $\mathsf{ACC}^i = \bigcup_m \mathsf{AC}^i[m]$.

∘ TC^i *is the class of languages accepted by logtime-uniform circuit families of polynomial size and depth* $O(\log^i n)$*, consisting of unbounded fan-in* MAJORITY *gates, along with* NOT *gates.*

∘ SAC^i *is the class of languages accepted by logtime-uniform circuit families of polynomial size and depth* $O(\log^i n)$*, consisting of unbounded fan-in* OR *gates and bounded fan-in* AND *gates, along with* NOT *gates at (some of) the leaves.*

Note that the restriction that NOT gates appear only at the leaves in SAC^i circuits is essential; if NOT gates were allowed to appear everywhere, then these classes would coincide with AC^i. Similarly, note that we do not bother to define a complexity class $\mathsf{SAC}^i[m]$, since a MOD_m gate with a single input wire is equivalent to a NOT gate, and thus $\mathsf{SAC}^i[m]$ would be the same as $\mathsf{AC}^i[m]$.

Definition 3.8.

A remark about uniformity: We have chosen to define the classes above in terms of logtime-uniformity, primarily because we will have occasion to mention the classes AC^0 and TC^0, and logtime-uniformity has been widely accepted as the more appropriate uniformity condition to use when discussing small classes. But it is well-known that, for circuit classes that contain SAC^1, logtime-uniformity coincides with logspace uniformity (see the discussion beginning at page 123 in Vollmer (1999)), and logspace uniformity is frequently somewhat easier to work with. Since Vollmer (1999) does not specifically discuss the equivalence of these uniformity conditions for classes such as $\mathsf{AC}^1[m]$, we include a brief discussion here. Consider a logspace-uniform circuit family $\{C_n\}$ of $\mathsf{AC}^1[m]$ circuits, and consider any unbounded fan-in gate g at depth d in C_n, receiving inputs from gates $h_1, \ldots, h_m$. (Assume without loss of generality that C_n contains only OR and MOD gates, using DeMorgan's Laws, and that C_n is leveled, with gates at level d receiving inputs from gates at level $d-1$.) C_n is equivalent to a logtime-uniform circuit D_n, where each gate g at level d in C_n is simulated by a gate g_d in D_n of the same type as g. The inputs to g_d consist of fan-in two AND gates $g_{d,h}$ for *every* gate h of C_n. The inputs to theAND gate g_h are (1) the gate h_{d-1}, and (2) a logtime-uniform SAC^1 subcircuit checking whether there is a wire from h to g in C_n. It is easy to see that each gate g at level d takes on the same value as gate g_d, and that the entire construction of D_n is logtime-uniform.

The algebraic complexity classes $\mathsf{VP}(R)$ for various algebraic structures R were originally defined (Valiant 1979) in the context of nonuniform circuit complexity, as classes of families of n-variate polynomials of degree $n^{O(1)}$ that can be represented by polynomial-size arithmetic circuits over R. (For more on VP, see, e.g. Bürgisser (1999, 2000); Gál & Wigderson (1996); Koiran & Perifel (2011); Malod & Portier (2008).) In this paper, we focus on uniform circuit families, and thus we use the notation $\mathsf{VP}(R)$ to denote the families of polynomials that result when we impose a logspace-uniformity condition on the circuit families. In the original nonuniform setting, it was shown by Valiant *et al.* (1983) that the circuits defining polynomials in $\mathsf{VP}(R)$ can be assumed to have small depth. Later

Allender *et al.* (1998) a slightly improved characterization was provided, that works also in the context of uniform circuit complexity:

THEOREM 2.2. *(Allender* et al. *1998) For any commutative semiring R, $\mathsf{VP}(R)$ coincides with the class of families of polynomials over R represented by logspace-uniform circuit families of polynomial size and logarithmic depth with unbounded fan-in $+$ gates, and fan-in two $\times$ gates.*

Note that over $\mathbb{F}_p$, many different polynomials yield the same function. For example, since $x^3 = x$ in $\mathbb{F}_3$, every function on n variables has a polynomial of degree at most $2n$. Very likely there are functions represented by polynomials in $\mathsf{VP}(\mathbb{F}_3)$ of degree, say, n^5, but not by any VP polynomial of degree $2n$. On the other hand, there is a case to be made for focusing on the *functions* in these classes, rather than focusing on the *polynomials* that represent those functions. For instance, if the Immerman-Landau conjecture is true, and TC^1 is reducible to problems in $\mathsf{VP}(\mathbb{Q})$, it would suffice for every *function* in $\mathsf{TC}^1 = \#\mathsf{AC}^1(\mathbb{F}_{p_n})$ to have a representation in $\mathsf{VP}(\mathbb{Q})$, even though the *polynomials* represented by $\#\mathsf{AC}^1(\mathbb{F}_{p_n})$ circuits have large degree, and thus cannot be in any VP class.

In the literature on VP classes, one standard way to focus on the *functions* represented by polynomials in VP is to consider what is called the *Boolean Part* of $\mathsf{VP}(R)$, which is the set of *languages* $A \subseteq \{0,1\}^*$ such that, for some sequence of polynomials (Q_n), for $x \in A$ we have $Q_{|x|}(x) = 1$, and for $x \in \{0,1\}^*$ such that $x \notin A$ we have $Q_{|x|}(x) = 0$.

When the algebra R is a finite field, considering the Boolean part of $\mathsf{VP}(R)$ captures the relevant complexity aspects, since the computation of any function represented by a polynomial in $\mathsf{VP}(R)$ (with inputs and outputs coming from R) is logspace-Turing reducible to some language in the Boolean Part of $\mathsf{VP}(R)$.

In this paper, we will be concerned exclusively with the "Boolean Part" of various arithmetic classes. For notational convenience, we will just refer to these classes using the "VP*" notation, rather*

than constantly repeating the phrase "Boolean Part".[2]

Following the standard naming conventions of Vollmer (1999), for any Boolean circuit complexity class $\mathcal{C}$ defined in terms of circuits with AND and OR gates, we define the class $\#\mathcal{C}(R)$ to be the class of functions represented by arithmetic circuits defined over the algebra R, where AND is replaced by $\times$, and OR is replaced by $+$ (and NOT gates at the leaves are applied to the $\{0,1\}$ inputs).[3] In particular, we will be concerned with the following two classes:

DEFINITION 2.3. *Let R be any suitable semiring.*[4] *Then*

- *$\#\mathsf{AC}^1(R)$ is the class of functions $f : \{0,1\}^* \to R$ represented by families of logspace-uniform circuits of unbounded fan-in $+$ and $\times$ gates having depth $O(\log n)$ and polynomial size.*
- *$\#\mathsf{SAC}^1(R)$ is the class of functions $f : \{0,1\}^* \to R$ represented by families of logspace-uniform circuits of unbounded fan-in $+$ gates and $\times$ gates of fan-in two, having depth $O(\log n)$ and polynomial size.*

Input variables may be negated. Constants from R are also allowed at the input level. Where no confusion will result, the notation

[2] An exception is when an arithmetic function is used as an oracle, as in the expressions $\mathsf{L}^{\mathsf{VP}(\mathbb{Q})}$ and $\mathsf{L}^{\mathsf{VP}(\mathbb{Z}_m)}$. Here, we want the logspace-bounded oracle Turing machine to have access to the full power of functions from $\mathsf{VP}(\mathbb{Q})$ and $\mathsf{VP}(\mathbb{Z}_m)$, respectively, and not merely the zero-one-valued functions.

[3] The classes $\#\mathsf{L}$, $\#\mathsf{P}$ and $\#\mathsf{LogCFL}$ also fit into this naming scheme, using established connections between Turing machines and circuits.

[4] Our primary focus in this paper is on *finite* semirings, as well as countable semirings such as $\mathbb{Q}$. We use the standard binary representation of constants (representing separately the numerator and denominator of a rational) when constants appear in the description of a circuit. We consider arithmetic circuits over $\mathbb{Q}$ only in the context of arithmetic circuits that have algebraic degree that is bounded by a polynomial, and thus the length of the binary representation of any number that is computed by the circuit is itself bounded by a polynomial in the input length. It is not clear to us which definition would be most useful in describing a class such as $\#\mathsf{AC}^1(\mathbb{R})$, and so for now we consider such semirings to be "unsuitable". Similarly, $\mathbb{Q}$ would be considered "suitable" for $\#\mathsf{SAC}^1$, but not for $\#\mathsf{AC}^1$, because $\#\mathsf{AC}^1$ circuits have algebraic degree that is too high.

$\#\mathcal{C}(R)$ will also be used to refer to the class of languages whose characteristic functions lie in the given class.

Hence from Theorem 2.2 we obtain:

PROPOSITION 2.4. *Let p be a prime power. Then* $\mathsf{VP}(\mathbb{F}_p) = \#\mathsf{SAC}^1(\mathbb{F}_p)$.

PROOF. The inclusion $\mathsf{VP}(\mathbb{F}_p) \subseteq \#\mathsf{SAC}^1(\mathbb{F}_p)$ is immediate from Theorem 2.2. The $\#\mathsf{SAC}^1(\mathbb{F}_p)$ circuit that is created for a $\mathsf{VP}(\mathbb{F}_p)$ circuit has no NOT gates. For the converse inclusion, given a $\#\mathsf{SAC}^1(\mathbb{F}_p)$ circuit family, each NOT gate at a leaf, connected to input x_i can be replaced by $(x_i + (p-1))^2$. □

2.1. New Definitions: Λ-classes. In this section, we introduce and define classes that are dual to the $\#\mathsf{SAC}^1(R)$ classes discussed above. Define $\#\mathsf{SAC}^{1,*}(R)$ to be the class of functions $f : \{0,1\}^* \to R$ represented by families of logspace-uniform circuits of unbounded fan-in $\times$ gates and $+$ gates of fan-in two, having depth $O(\log n)$ and polynomial size. Proposition 2.4 highlights the connection between VP and $\#\mathsf{SAC}^1$; thus we will utilize the convenient notation $\Lambda\mathsf{P}(R)$ to denote the dual notation, rather than the more cumbersome $\#\mathsf{SAC}^{1,*}(R)$.

Of course, the set of formal polynomials represented by $\Lambda\mathsf{P}$ circuits is not contained in any VP class, because $\Lambda\mathsf{P}$ contains polynomials of degree $n^{O(\log n)}$. However, as discussed in the previous section, we are considering the "Boolean Part" of these classes. More formally:

DEFINITION 2.5. *Let p be a prime power. $\Lambda\mathsf{P}(\mathbb{F}_p)$ is the class of all languages $A \subseteq \{0,1\}^*$ with the property that there is a logspace-uniform (and hence polynomial-size) family of circuits $\{C_n : n \in \mathbb{N}\}$ such that*

- *The depth of C_n is $O(\log n)$.*
- *Each C_n consists of input gates, $+$ gates, and $\times$ gates.*
- *Each $+$ gate has fan-in two, whereas there is no bound on the fan-in of the $\times$ gates.*

- *For each string x of length n, x is in A if and only if $C_n(x)$ evaluates to 1, when the $+$ and $\times$ gates are evaluated over $\mathbb{F}_p$. Furthermore, if $x \notin A$, then $C_n(x)$ evaluates to 0.*

Another way of relating arithmetic classes (such as VP and $\Lambda\mathsf{P}$) to complexity classes of languages would be to consider the languages that are logspace-Turing reducible to the polynomials in $\mathsf{VP}(R)$ or $\Lambda\mathsf{P}(R)$, via a machine M with a polynomial p as an oracle, which obtains the value of $p(x_1, \ldots, x_n)$ when M writes $x_1, \ldots x_n$ on a query tape. As a side note, throughout the rest of this work we will be using the identity $\overline{x} \equiv ((x + (m-1))^2) \mod m$ to perform complementation of Boolean values over $\mathbb{Z}_m$, and as a consequence of having this equation $\mathsf{VP} = \overline{\mathsf{VP}}$ and $\Lambda\mathsf{P} = \overline{\Lambda\mathsf{P}}$. However, another key trick we use is $(x^{p-1} \equiv 1) \mod p$, and when p is not a constant, then this manipulation is sometimes too expensive to deploy, in the context of $\mathsf{VP}(\mathbb{F}_p)$.

It is worth mentioning that (the Boolean parts of) both $\mathsf{VP}(\mathbb{F}_p)$ and $\Lambda\mathsf{P}(\mathbb{F}_p)$ are closed under logspace-Turing reductions, although this is still open for classes over $\mathbb{Z}_m$ when m is not prime.

PROPOSITION 2.6. *$\Lambda\mathsf{P}(\mathbb{F}_p) = \mathsf{L}^{\Lambda\mathsf{P}(\mathbb{F}_p)}$ and $\mathsf{VP}(\mathbb{F}_p) = \mathsf{L}^{\mathsf{VP}(\mathbb{F}_p)}$*

PROOF. We consider VP first. Note that there are only polynomially-many queries that a logspace-Turing reduction can pose, on a given input x, since the query that is posed is determined entirely by the worktape configuration of the oracle Turing machine when it begins to write the query. These queries can be denoted $y_1, \ldots, y_{n^k}$ for some k. If $A \in \mathsf{L}^{\mathsf{VP}(\mathbb{F}_p)}$, then there is a language $B \in \mathsf{L}$ such that $x \in A$ iff $(x, z) \in B$ where z is the bit string of length n^k recording the oracle answers for each query y_i. Since B is in the deterministic class L, it has "unambiguous" SAC^1 circuits, meaning that the corresponding $\#\mathsf{SAC}^1$ circuits always output 0 or 1. By connecting $\mathsf{VP}(\mathbb{F}_p)$ circuits computing the answer to each oracle query y_i to the input variables for z, one obtains $\mathsf{VP}(\mathbb{F}_p)$ circuits for A.

The argument for $\Lambda\mathsf{P}(\mathbb{F}_p)$ is similar, using the fact that $B \in \mathsf{L} \subseteq \mathsf{AC}^1$, along with the fact (which we prove later in Corollary 3.3) that $\mathsf{AC}^1 \subseteq \Lambda\mathsf{P}(\mathbb{F}_p)$ for every p. □

We mention that VP classes over different fields of the same characteristic define the same class of languages. This seems to be one way that the VP and ΛP classes differ; see Corollary 3.3.

PROPOSITION 2.7. *Let p be a prime, and let $k \geq 1$. Then*

$$\mathsf{VP}(\mathbb{F}_p) = \mathsf{VP}(\mathbb{F}_{p^k}).$$

PROOF. One inclusion follows immediately since $\mathbb{F}_p$ is a subfield of $\mathbb{F}_{p_k}$. For the other direction, observe that the finite field of size p^k is a vector space of dimension k over the field of size p, and thus can be represented by $k \times k$ matrices over $\mathbb{F}_p$, as described in von zur Gathen (1993). (See also Wardlaw (1994).) Thus each $+$ and $\times$ gate of a $\Lambda\mathsf{P}(\mathbb{F}_{p^k})$ circuit can be replaced by subcircuits implementing matrix sum and product over $\mathbb{F}_p$. (Unbounded fan-in matrix sum corresponds to unbounded fan-in sum of each component. Fan-in two multiplication is implemented by a depth-two subcircuit, with fan-in two $\times$ gates, and with addition gates of fan-in $O(1)$.) The resulting circuit is a $\mathsf{VP}(\mathbb{F}_p)$ circuit. □

It is also appropriate to use the VP and ΛP notation when referring to the classes defined by Boolean semiunbounded fan-in circuits with negation gates allowed at the inputs. With this notation, $\mathsf{VP}(B_2)$ corresponds to the Boolean class SAC^1, and $\Lambda\mathsf{P}(B_2)$ corresponds to the complement of SAC^1 (with bounded fan-in OR gates, unbounded fan-in AND gates and negation gates allowed at the inputs). It has been shown by Borodin *et al.* (1989) that SAC^1 is closed under complement. Thus we close this section with the equality that serves as a springboard for investigating the ΛP classes.

THEOREM 2.8. *(Borodin* et al. *1989)* $\mathsf{VP}(B_2) = \Lambda\mathsf{P}(B_2) (= \mathsf{SAC}^1 = \mathsf{LogCFL})$.

We do not believe that $\mathsf{VP}(\mathbb{F}_p) = \Lambda\mathsf{P}(\mathbb{F}_p)$ for any prime p; see further related discussion in Section 5.

3. Subclasses of ACC^1

In this section, we first give characterizations of the $\Lambda\mathsf{P}$ classes, and then we give characterizations of the VP classes, before comparing the classes to each other.

3.1. The $\Lambda\mathsf{P}$ classes. In this subsection, we present our characterizations of ACC^1 in terms of the $\Lambda\mathsf{P}(\mathbb{F}_{p^k})$ classes.

THEOREM 3.1. *For any prime p and any $k \geq 1$,*

$$\Lambda\mathsf{P}(\mathbb{F}_{p^k}) = \mathsf{AC}^1[\mathsf{Supp}(p^k - 1)].$$

(Recall that $\mathsf{Supp}(m)$ is defined in Definition 2.1.)

PROOF. ($\subseteq$): Consider a $\Lambda\mathsf{P}(\mathbb{F}_{p^k})$ circuit C. We will create a circuit C' that has subcircuits computing the Boolean value $[g = a]$ for each gate g in C and for each $a \in \mathbb{F}_{p^k}$. (We will use the notation "$[B]$" to refer to the truth-value of predicate B.) If g is the output gate of C, then the output gate of C' is the gate $[g = 1]$. Since the input gates of C take on only binary values (by our definition of $\Lambda\mathsf{P}(\mathbb{F}_{p^k})$), if g is an input gate of C, then the subcircuit $[g = 1]$ is just g, and the subcircuit for $[g = 0]$ is $\neg g$. If g is a constant gate, set to the value $a \in \mathbb{F}_{p^k}$, then $[g = a]$ is set to the constant 1, and $[g = a']$ is set to the constant 0, for each $a' \neq a$.

If g is a + gate of C (of fan-in 2), then any gate $[g = a]$ can be simulated with NC^0 circuitry using the $O(1)$ Boolean gates of the form $[g' = a']$, where g' feeds into g in C.

Now consider a $\times$ gate g of C, having unbounded fan-in: $g = \prod_i h_i$. The value $[g = 0]$ is obtained by simply checking if there is some i such that $h_i = 0$.

Now we show how to compute $[g = a]$ for $a \neq 0$. Let $p^k - 1 = \prod_{j=1}^{\ell} q_j^{e_j}$ where $\mathsf{Supp}(p^k - 1) = \{q_1, \ldots, q_\ell\}$. Let σ be a generator of the multiplicative group of $\mathbb{F}_{p^k}$. Then $g = \prod_i h_i = \prod_i \sigma^{\log h_i} = \sigma^{\sum_i \log h_i}$ where "$\log b$" denotes the unique number in $\{0, \ldots, p^k - 1\}$ such that $\sigma^{\log b} = b$. Hence the value $[g = a]$ is equivalent to $[\log a \equiv \sum_i \log h_i \bmod (p^k - 1)]$, which in turn is equivalent to the AND of the values $[\log a \equiv \sum_i \log h_i \bmod (q_j^{e_j})]$.

If $e_j = 1$ then the value $[\log a \equiv \sum_i \log h_i \bmod (q_j)]$ is easy to compute with a MOD_{q_j} gate, as follows. Using NC^0 circuitry, for

each i, find the unique b such that $[h_i = b]$ holds (and for simplicity, let us refer to this value as h_i). Then, for each i, compute the string $x_i = 1^{\log h_i}0^{p^k - \log h_i}$. (Note that the mapping from gates of the form $[h_i = b]$ to x_i is computable in logspace-uniform NC^0.) Let X_a be the string that results from concatenating the string $1^{p^k - \log a}$ and all of the strings x_i. Now observe that feeding X_a into a MOD_{q_j} gate computes the value $[\log a \equiv \sum_i \log h_i \bmod (q_j)]$.

Now we show how to compute $[\log a \equiv \sum_i \log h_i \bmod (q_j^{e_j})]$ when $e_j > 1$. For any expression b (such as $b = (\sum_i \log h_i \bmod (q_j^{e_j})) - \log a$), first observe that $[b \equiv 0 \bmod q_j^{e_j}]$ can be computed by checking if each of $b, \left(\begin{array}{c} b \\ q_j \end{array}\right), \left(\begin{array}{c} b \\ q_j^2 \end{array}\right), \ldots \left(\begin{array}{c} b \\ q_j^{e_j - 1} \end{array}\right)$ is equivalent to 0 mod q_j. (See, e.g. Beigel & Tarui (1994, Fact 2.2).) Observe also that $\left(\begin{array}{c} b \\ d \end{array}\right)$ can be represented as the number of different AND gates of fan-in d that evaluate to 1, taking inputs from a string with b ones. Thus all of these conditions can be checked in constant depth with MOD_{q_j} gates and bounded fan-in AND gates, by constructing the string X_a (as in the preceding paragraph), and using a layer of AND gates of fan-in at most $q_j^{e_j - 1}$.

Since C has depth $O(\log n)$, and C' consists of layers of constant-depth circuitry to replace each layer of gates in C, this completes the proof of this direction.

($\supseteq$): Given an $\mathsf{AC}^1[\mathsf{Supp}(p^k - 1)]$ circuit C, we show how to construct an arithmetic circuit C' that is equivalent to C. Each gate g of C will have an equivalent gate g in C'. The input gates of C and of C' are exactly the same.

If g is a NOT gate in C, say $g = \neg h$, then in C' we will have $g = (h + (p-1)) \times (h + (p-1))$.

If g is an AND gate (say, $g = \wedge_i h_i$), then in C' we will have $g = \prod_i h_i$. OR gates will be handled the same way, using De Morgan's Laws.

Now consider the case when g is a MOD_{q_j} gate with inputs h_i. Thus g computes the value $[\sum_i h_i \equiv 0 \bmod q_j]$. Let σ be a generator of the multiplicative cyclic subgroup of size q_j. First map each h_i to the value $h'_i = 1 + (\sigma + (p-1)) \times h_i$, and observe that $h'_i = \sigma^{h_i}$ for all $h_i \in \{0, 1\}$. Observe that $1 - \prod_i h'_i = 1 - \sigma^{\sum_i h_i}$

is equal to 0 if $\sum_i h_i$ is a multiple of q_j, and is non-zero otherwise. Thus $1 - (1 - \prod_i h'_i)^{p^k-1}$ is equal to the Boolean value $[\sum_i h_i \equiv 0 \bmod q_j]$.

It is easy to verify that C' has logarithmic depth, and uses only bounded fan-in + gates, as well as unbounded fan-in $\times$ gates. □

COROLLARY 3.2. $\mathsf{ACC}^1 = \bigcup_p \Lambda\mathsf{P}(\mathbb{F}_p)$.

PROOF. Let $A \in \mathsf{ACC}^1$. Thus $A \in \mathsf{AC}^1[m]$ for some modulus m.

By Dirichlet's Theorem, the arithmetic progression $m+1, 2m+1, \ldots$ contains some prime p. Thus $\mathsf{AC}^1[m] \subseteq \mathsf{AC}^1[\mathsf{Supp}(p-1)] = \Lambda\mathsf{P}(\mathbb{F}_p)$. □

Note also that several of the $\Lambda\mathsf{P}(\mathbb{F}_p)$ classes coincide. This is neither known nor believed to happen with the $\mathsf{VP}(\mathbb{F}_p)$ classes.

COROLLARY 3.3.
- $\Lambda\mathsf{P}(\mathbb{F}_2) = \mathsf{AC}^1$, *whereas* $\Lambda\mathsf{P}(\mathbb{F}_4) = \mathsf{AC}^1[3]$. *Note that this contrasts with the equality* $\mathsf{VP}(\mathbb{F}_2) = \mathsf{VP}(\mathbb{F}_4)$ *given by Proposition 2.7.*
- *If p is a Fermat prime (that is, $p-1$ is a power of 2, such as $p \in \{3, 5, 17, 257, 65{,}537\}$), then* $\Lambda\mathsf{P}(\mathbb{F}_p) = \mathsf{AC}^1[2]$.
- $\Lambda\mathsf{P}(\mathbb{F}_7) = \Lambda\mathsf{P}(\mathbb{F}_{13}) = \Lambda\mathsf{P}(\mathbb{F}_{19})$.
- *More generally,* $\mathsf{Supp}(p-1) = \mathsf{Supp}(q-1)$ *implies* $\Lambda\mathsf{P}(\mathbb{F}_p) = \Lambda\mathsf{P}(\mathbb{F}_q)$.

Augmenting the $\Lambda\mathsf{P}(\mathbb{F}_p)$ classes with unbounded fan-in addition gates increases their computation power only by adding MOD_p gates, as the following theorem demonstrates.

THEOREM 3.4. *For each prime p and each $k \geq 1$,* $\#\mathsf{AC}^1(\mathbb{F}_{p^k}) = \mathsf{AC}^1[\{p\} \cup \mathsf{Supp}(p^k - 1)]$.

PROOF. ($\subseteq$): Again, we use a gate-by-gate simulation, with subcircuits recording the value of $[g = a]$ for each gate g and each $a \in \mathbb{F}_{p^k}$. Multiplication gates are handled as in the proof of Theorem 3.1. Consider now the case of an addition gate $g = \sum_i h_i$.

Since $\mathbb{F}_{p^k}$ is a vector space of dimension k over $\mathbb{F}_p$, each element $b \in \mathbb{F}_{p^k}$ is represented by a vector $\vec{b} = (b_1, \ldots, b_k) \in (\mathbb{F}_p)^k$, which we will represent as a bitstring $Y_{\vec{b}} = 1^{b_1}0^{p-b_1} \ldots 1^{b_k}0^{p-b_k}$. Let us call the string $1^{b_j}0^{p-b_j}$ the j-th component of $Y_{\vec{b}}$.

Using NC^0 circuitry (as in the proof of Theorem 3.1), one can use the gates $[h_i = b]$ to compute the string $Y_{\vec{h_i}}$ (as in the proof of Theorem 3.1). Let $Z_{a,j}$ be the string that is the concatenation of the j-th component of all of the $Y_{\vec{h_i}}$ with the j-th component of $Y_{\vec{-a}}$, and feed each Z_a into a MOD_p gate. The gate $[g = a]$ is an AND gate, verifying that, for all $j \leq k$, $\mathrm{MOD}_p(Z_{a,j}) = 1$.

($\supseteq$): As in Theorem 3.1, we carry out a gate-by-gate simulation, whereby each gate g in a $\mathsf{AC}^1[\{p\} \cup \mathsf{Supp}(p^k - 1)]$ circuit C is equivalent to a gate (also called g) in a $\#\mathsf{AC}^1(\mathbb{F}_p)$ circuit C'. We only need to consider the case where g is a MOD_p gate with Boolean inputs h_i. In this case, note that $g = 1 + ((\sum_i h_i)^{p^k-1} \times (p-1))$. □

Corollary 3.5.
$\mathsf{ACC}^1 = \bigcup_p \Lambda\mathsf{P}(\mathbb{F}_p) = \bigcup_p \#\mathsf{AC}^1(\mathbb{F}_p) = \bigcup_m \#\mathsf{AC}^1(\mathbb{Z}_m)$.

Proof. All inclusions are immediate from Theorem 3.1 and Theorem 3.4, except for $\#\mathsf{AC}^1(\mathbb{Z}_m) \subseteq \mathsf{ACC}^1$. Consider a circuit C for some function in $\#\mathsf{AC}^1(\mathbb{Z}_m)$. Again, we will build an ACC^1 circuit C' with gates of the form $[g = a]$ for each gate g in C and each $a \in \mathbb{Z}_m$. Addition is handled as in the proof of Theorem 3.4. Thus consider a multiplication gate $g = \prod_i h_i = \prod_j a_j^{e_j}$, where $e_j = |\{i : [h_i = a_j]\}|$. The sequence $(a_j^0, a_j^1, a_j^2, \ldots)$ (where the product is interpreted in $\mathbb{Z}_m$) is ultimately periodic with a period less than m, and thus the value of $[a_j^{e_j} = b]$ can be computed using AC^0 circuitry and a MOD gate, using inputs of the form $[h_i = a_j]$ for various values of i. Then $[g = a]$ can be computed in NC^0 using the $O(1)$ gates of the form $[a_j^{e_j} = b]$. □

Corollary 3.6. *For any prime p there is a prime q such that $\#\mathsf{AC}^1(\mathbb{F}_p) \subseteq \Lambda\mathsf{P}(\mathbb{F}_q)$.*

Proof. By Dirichlet's Theorem, there is a prime q such that $q - 1$ is a multiple of $p(p-1)$. The claim now follows immediately from Theorem 3.4 and Theorem 3.1. □

We remark that the proofs of Theorem 3.1 and Theorem 3.4 carry over also for depths $\log^i n$ for every $i \geq 0$. (Related results for constant-depth unbounded-fan-in circuits can be found already in Agrawal *et al.* (2000); Smolensky (1987).)

COROLLARY 3.7. *For any prime p and for every $i \geq 0$, $\#\mathsf{SAC}^{i,*}(\mathbb{F}_p) = \mathsf{AC}^i[\mathsf{Supp}(p-1)]$ and $\#\mathsf{AC}^i(\mathbb{F}_p) = \mathsf{AC}^i[p \cup \mathsf{Supp}(p-1)]$. In particular, $\mathsf{ACC}^i = \bigcup_p \#\mathsf{SAC}^{i,*}(\mathbb{F}_p)$.*

3.2. The VP classes. It will be useful to bear in mind that $\mathsf{VP}(\mathbb{F}_p)$ also has a simple characterization in terms of Boolean circuits. In order to present this characterization, we present a more general definition, which will be needed later.

DEFINITION 3.8. *Let $m \in \mathbb{N}$, and let g be any function on $\mathbb{N}$. Define g-$\mathsf{AC}^i[m]$ to be the class of languages with logspace-uniform circuits of polynomial size and depth $O(\log^i n)$, consisting of MOD_m gates of unbounded-fan-in, along with* AND *gates of fan-in $O(g(n))$. Clearly g-$\mathsf{AC}^i[m] \subseteq \mathsf{AC}^i[m]$.*

The class CC^i is defined to be $\bigcup_m \mathsf{CC}^i[m]$, analogously to ACC^i.

When $g(n) = O(1)$, the class g-$\mathsf{AC}^i[m]$ coincides with the class $\mathsf{CC}^i[m]$, which was defined by (Straubing 1994, p. 141) for the special case $i = 0$, and which has been studied subsequently in e.g. Hahn *et al.* (2015); Hansen & Koucký (2010); Thérien (1994). If $m > 2$, then no AND or OR gates are needed at all (Straubing 1994, Chapter VIII, Exercise 9). Thus some authors define $\mathsf{CC}^i[m]$ in terms of circuits consisting only of MOD_m gates, but the original definition is more convenient for our purposes.

Observe that, since a MOD_m gate can simulate a NOT gate, g-$\mathsf{AC}^1[m]$ remains the same if OR gates of fan-in $O(g)$ are also allowed.

COROLLARY 3.9. *For every prime p, $\mathsf{VP}(\mathbb{F}_p) = \mathsf{CC}^1[p] \subseteq \mathsf{AC}^1[p]$.*

PROOF. Recall that $\mathsf{VP}(\mathbb{F}_p) = \#\mathsf{SAC}^1(\mathbb{F}_p)$. Thus we need only show how to simulate bounded fan-in $\times$ gates and unbounded fan-in $+$ gates. Bounded fan-in $\times$ gates can be simulated in $O(1)$ depth using AND and OR gates of fan-in two (since the values

being multiplied are of size $O(1)$). Unbounded fan-in + gates can be simulated using MOD_p gates, as in the proof of Theorem 3.4.

For the converse inclusion, consider a $\mathsf{CC}^1[p]$ circuit. Since a unary MOD_p gate is equivalent to a NOT gate, we can assume that the circuit has only fan-in two AND gates and unbounded fan-in MOD_p gates. Thus each Boolean AND gate can be simulated by a fan-in two multiplication gate, and the MOD_p gates can be simulated as in the proof of Theorem 3.4. □

We remark that the same proof shows that, for any $m \in \mathbb{N}$, $\mathsf{VP}(\mathbb{Z}_m) \subseteq \mathsf{CC}^1[m]$. However, the converse inclusion is not known, unless m is prime.

3.3. Comparing $\Lambda\mathsf{P}$ and VP. How do the $\Lambda\mathsf{P}$ and VP classes compare to each other?

As a consequence of Corollary 3.9 and Theorem 3.1, $\mathsf{VP}(\mathbb{F}_p) \subseteq \Lambda\mathsf{P}(\mathbb{F}_q)$ whenever p divides $q-1$. In particular, $\mathsf{VP}(\mathbb{F}_2) \subseteq \Lambda\mathsf{P}(\mathbb{F}_q)$ for any prime $q > 2$. No inclusion of any $\Lambda\mathsf{P}$ class in any VP class is known unconditionally, although $\Lambda\mathsf{P}(B_2)(= \mathsf{SAC}^1)$ is contained in every $\mathsf{VP}(\mathbb{F}_p)$ class in the nonuniform setting (Gál & Wigderson 1996; Reinhardt & Allender 2000), and this holds also in the uniform setting under a plausible derandomization hypothesis (Allender *et al.* 1999).

No $\Lambda\mathsf{P}(\mathbb{F}_q)$ class can be contained in $\mathsf{VP}(\mathbb{F}_p)$ unless $\mathsf{AC}^1 \subseteq \mathsf{VP}(\mathbb{F}_p)$, since $\mathsf{AC}^1 = \Lambda\mathsf{P}(\mathbb{F}_2) \subseteq \Lambda\mathsf{P}(\mathbb{F}_3) \subseteq \Lambda\mathsf{P}(\mathbb{F}_q)$ for every prime $q \geq 3$. AC^1 is not known to be contained in any VP class, although we return to this topic again in Section 4

4. Threshold circuits and small degree

In this section, we revisit the known connections between threshold circuits and arithmetic circuits over small (but non-constant) finite fields, and present some new alternative characterizations of TC^1. This leads to a discussion of the possibility of "degree reduction" – simulating classes of arithmetic circuits using circuits with smaller algebraic degree.

4.1. Circuits with growing modulus. The inspiration for the results in this section comes from the following theorem of (Reif &

Tate 1992) (as re-stated by (Buhrman *et al.* 2014)):

THEOREM 4.1. $\mathsf{TC}^1 = \#\mathsf{AC}^1(\mathbb{F}_{p_n})$.

Here, the class $\#\mathsf{AC}^1(\mathbb{F}_{p_n})$ consists of those languages whose (Boolean) characteristic functions are computed by logspace-uniform families of arithmetic circuits of logarithmic depth with unbounded fan-in $+$ and $\times$ gates, where the arithmetic operations of the circuit C_n are interpreted over $\mathbb{F}_{p_n}$, where $p_1, p_2, p_3, \ldots$ is the sequence of all primes $2, 3, 5, \ldots$ That is, circuits for inputs of length n use the n-th prime to define the algebraic structure.

This class is closed under logspace-Turing reductions – but when we consider *other* circuit complexity classes defined using $\mathbb{F}_{p_n}$, it is *not* clear that these other classes are closed under logspace-Turing reductions.

As an important example, we mention $\mathsf{VP}(\mathbb{F}_{p_n})$. As we show below, this class has an important connection to $\mathsf{VP}(\mathbb{Q})$, which is perhaps the canonical example of a VP class. (Vinay 1991) proved that $\mathsf{VP}(\mathbb{Q})$ has essentially the same computational power as $\#\mathsf{LogCFL}$ (which counts among its complete problems the problem of determining how many distinct parse trees a string x has in a certain context-free language). Here, we mention one more alternative characterization of the computational power of $\mathsf{VP}(\mathbb{Q})$.

PROPOSITION 4.2. $\mathsf{L}^{\mathsf{VP}(\mathbb{F}_{p_n})} = \mathsf{L}^{\mathsf{VP}(\mathbb{Q})} = \mathsf{L}^{\#\mathsf{LogCFL}}$.

PROOF. Consider the first equality. If one wants to compute the value of a $\mathsf{VP}(\mathbb{F}_{p_n})$ circuit on a given input of length n, in logspace one can first compute the value of p_n. Then one can use a $\mathsf{VP}(\mathbb{Q})$ oracle to evaluate the $\mathsf{VP}(\mathbb{F}_{p_n})$ circuit over the rationals instead of over $\mathbb{F}_{p_n}$, obtaining an integer result. Then one can divide the result by p_n and obtain the remainder, which is the value of the circuit in $\mathbb{F}_{p_n}$, using the fact that division is computable in logspace (Chiu *et al.* 2001; Hesse *et al.* 2002).

Conversely, if one wants to evaluate a $\mathsf{VP}(\mathbb{Q})$ circuit on a given n-tuple of rationals, one can use the standard technique of computing the numerator and denominator separately; the circuits for these functions are also in $\mathsf{VP}(\mathbb{Q})$. Thus our task boils down to

evaluating an integer-valued arithmetic circuit C_n. To do this, we use Chinese remaindering, and evaluate circuits (with some dummy variables) over the primes $p_n, p_{n+1}, \ldots, p_{n+n^c}$ for some constant c. Converting between Chinese remainder representation and binary representation can be accomplished in logspace (Chiu *et al.* 2001; Hesse *et al.* 2002), which completes the proof of the first equality.

For the second equality, we similarly use the fact that $\mathsf{VP}(\mathbb{Q})$ circuits with integer coefficients and inputs can be evaluated in $\#\mathsf{LogCFL}$, and appeal to Vinay (1991). □

When we consider arithmetic circuits of superpolynomial algebraic degree (such as the $\Lambda\mathsf{P}$ classes), evaluating the circuits over the integers can produce outputs that require a superpolynomial number of bits to express in binary. Thus, when we consider such classes, it will always be in the context of structures (such as $\mathbb{F}_{p_n}$) where the output can always be represented in a polynomial number of bits.

Our first new result in this section, is to improve Theorem 4.1. Note that this result bears some similarity to Theorem 3.1 and Corollary 3.5 (showing that arithmetic circuits can be simulated using circuits with bounded fan-in multiplication gates) and Corollary 3.6 (making explicit the change of field required for this simulation).

THEOREM 4.3. $\mathsf{TC}^1 = \#\mathsf{AC}^1(\mathbb{F}_{p_n}) = \mathsf{L}^{\Lambda\mathsf{P}(\mathbb{F}_{p_n})}$.

PROOF. The first equality is due to Reif & Tate (1992). The inclusion of $\mathsf{L}^{\Lambda\mathsf{P}(\mathbb{F}_{p_n})}$ in TC^1 follows since $\Lambda\mathsf{P}(\mathbb{F}_{p_n})$ is a subclass of $\#\mathsf{AC}^1(\mathbb{F}_{p_n})$ and TC^1 is closed under logspace-Turing reducibility.

Consider a logspace-uniform circuit family $\{C_n\}$ where C_n is a $\#\mathsf{AC}^1$ circuit over $\mathbb{F}_{p_n}$. We will show how to simulate C_n, by making calls to an appropriate function in $\Lambda\mathsf{P}(\mathbb{F}_{p_n})$. The first step is to find a prime q that is not too much larger than p_n, such that $q-1$ is a multiple of $p_n(p_n-1)$. (Xylouris 2011) has shown that the sequence $1+p_n(p_n-1), 1+2p_n(p_n-1), 1+3p_n(p_n-1)\ldots$ contains a prime of size $O(p_n^{10.4})$. Thus our logspace oracle machine will begin by enumerating the elements of this sequence, and is guaranteed to find some such prime q. Note that this means that

$q-1 = \ell p_n(p_n-1)$ for some ℓ, and note also that, for all large n, this means that $\ell < (p_n)^{10}$. We will show how to construct a logspace-uniform $\Lambda\mathsf{P}(\mathbb{F}_{q_m})$ circuit family $\{D_m\}$ defining a function that our logspace oracle Turing machine can query, in order to simulate C_n. (In order to avoid confusion, we use "m" to index the $\Lambda\mathsf{P}$ circuit family, and denote the sequence of primes as $q_1, q_2, \ldots$ for this family, although of course $p_i = q_i$ for all i.)

The logspace machine that creates D_m on input 1^m (by the logspace uniformity condition) operates as follows: Find q_m, and then find the prime factorization of $q_m - 1$. *If there is no prime p* such that $q_m - 1 = p(p-1)\ell$ for some $\ell < p^{10}$, then D_m is a circuit that computes the constant zero polynomial. Otherwise, note that there can be at most ten different primes $p_{n_1} < p_{n_2} < \ldots, p_{n_{10}}$ for which $q_m - 1 = p_{n_j}(p_{n_j}-1)\ell_{n_j}$ for $\ell_{n_j} < p_{n_j}^{10}$, since otherwise $q_m - 1 = p_{n_1}(p_{n_1}-1)p_{n_2}\cdots p_{n_{11}}\ell'$ for some ℓ', where $\ell_{n_1} = p_{n_2}\cdots p_{n_{11}}\ell' > p_{n_1}^{10}$. Assume therefore that there are $1 \le c \le 10$ such primes $p_{n_1} < p_{n_2} < \ldots, p_{n_c}$. The arithmetic circuit D_m operating over $\mathbb{F}_q$ will compute a polynomial of the form $\sum_{j=1}^{c} y_j \cdot P_{n_j}(x_1, \ldots, x_{n_j})$ on the variables $\{y_1, \ldots, y_c\} \cup \{x_1, \ldots, x_{n_c}\}$. (Note that the number of variables is at most $10 + n_c$, which is less than m for all large m.) Here, the polynomial P_{n_j} is computed by a subcircuit that is constructed to simulate C_{n_j}. (Note that if the logspace oracle machine wants to simulate C_{n_j} on input $(x_1, x_2, \ldots, x_{n_j})$, then it can query the oracle (computed by D_m) by setting variable y_{n_j} to 1 and all of the other variables y_i to zero, and providing the input $(x_1, x_2, \ldots, x_{n_j})$, (and setting all of the rest of the m variables to zero).) In what follows, we let $n = n_j$, and we show how to build the subcircuit C'_n of D_m that will allow us to simulate C_n.

For each gate g of C_n and each $a \in \mathbb{F}_{p_n}$, C'_n will have a gate computing the Boolean value $[g = a]$. If g is an input gate, say $g = x_i$, then the Boolean value $[g = a]$ is given by $((x_i - a)^{p_n - 1} + (p_n - 1))^2$.

Let us now consider the case when g is a $+$ gate, $g = \sum_i h_i$. Let γ be a generator of the cyclic subgroup of the multiplicative group of $\mathbb{F}_q$ of order p_n. Our circuit C'_n will have gates $h_{i,a}$ computing the value

$$h_{i,a} = ([h_i = a] \times (\gamma^a - 1) + 1).$$

Observe that $\prod_a h_{i,a}$ is equal to γ^{h_i}. C'_n will have a gate g' computing the value $g' = \prod_{i,a} h_{i,a}$. Note that g' is equal to $\gamma^{\sum_i h_i} = \gamma^g$ (since γ has order p_n). The value of the gate $[g = b]$ (for a given $b \in \mathbb{F}_{p_n}$) is thus $c_b^{-1} \times \prod_{\sigma \neq b}(\gamma^\sigma - g')$, where the constant $c_b = \prod_{\sigma \neq b}(\gamma^\sigma - b)$ can be computed in logspace and is thus available as a constant in C'_n.

It remains only to deal with the case when g is a $\times$ gate, $g = \prod_i h_i$. In C'_n, the gate $[g = 0]$ is $1 - \prod_i (1 - [h_i = 0])$.

Let μ be a generator of the multiplicative group of $\mathbb{F}_{p_n}$, and let α be a generator of the subgroup of the multiplicative group of $\mathbb{F}_q$ of order $p_n - 1$. If g does not evaluate to 0, then g is equal to μ^b for some b. Our circuit C'_n will have gates $h_{i,\sigma}$ computing the values

$$h_{i,\sigma} = ([h_i = \mu^\sigma] \times (\alpha^\sigma - 1) + 1).$$

Our circuit C'_n will have gates h'_i computing the value $h'_i = \prod_\sigma h_{i,\sigma}$. Observe that h'_i is equal to α^a if $h_i = \mu^a$, and h'_i is equal to 1 if $h_i = 0$.

In C'_n, there will be a gate g' that computes the following value: $g' = (1 - [g = 0]) \prod_i h'_i = ([g \neq 0]) \prod_i \alpha^{\log_\mu h_i} = ([g \neq 0])\alpha^{\sum_i \log_\mu h_i} = ([g \neq 0])\alpha^{\log_\mu g}$. Observe that, if $g \neq 0$, then $g = \mu^b$ for some b, and in this case g' evaluates to α^b. The value of the gate $[g = \mu^b]$ (for a given $b \in \mathbb{F}_{p_n}$) is thus $c_b^{-1} \times \prod_{\sigma \neq b}(\alpha^\sigma - g')$, where the constant $c_b = \prod_{\sigma \neq b}(\alpha^\sigma - \mu^b)$ can be computed in logspace and is thus available as a constant in C'_n. □

For completeness, we add two more relevant characterizations of TC^1. (Recall the definition of g-$\mathsf{AC}^1[m]$ from Definition 3.8.)

THEOREM 4.4.
$\mathsf{TC}^1 = \#\mathsf{AC}^1(\mathbb{F}_{p_n}) = \mathsf{L}^{\Lambda\mathsf{P}(\mathbb{F}_{p_n})} = \mathsf{AC}^1[p_n] = \mathsf{CC}^1[p_n]$.

PROOF. We need only consider the last two equalities.

($\supseteq$): MAJORITY gates can simulate AND, OR, and MOD_{p_n} gates in constant depth; thus this direction is easy.

($\subseteq$): Let ϵ be chosen so that $2n^\epsilon < p_n$ for every n. Any MAJORITY gate (of fan-in n^k) can be simulated by an AC^0-reduction to MAJORITY gates having fan-in n^ϵ (Allender &

Koucký 2010). Thus if $A \in \mathsf{TC}^1$, then A is accepted by a family of circuits of AND, OR, and MAJORITY gates, where the MAJORITY gates have fan-in at most n^ϵ. It suffices to show how to simulate a MAJORITY gate with inputs $h_1, \ldots, h_\ell$. Note that $\mathrm{MOD}_{p_n}(h_1, \ldots, h_\ell, 1^{p_n - b})$ computes the value $[b = \sum_i h_i]$. Thus the MAJORITY of the h_i is simply the OR, over all $b > \ell/2$ of the subcircuits computing $[b = \sum_i h_i]$.

For the final equality, first note any AND or OR gate with fan-in at least p_n can be replaced by a constant-depth tree of AND and OR gates of fan-in strictly less than p_n. Next, use DeMorgan's laws to remove all of the AND gates. Thus the circuit has only MOD gates and small fan-in OR gates. But note that if we feed the wires from an OR gate into a MOD_{p_n} gate, then the result is the NOR of the inputs (since if all of the wires are zero, the MOD gate outputs 1, and otherwise the number of wires that are one is less than p_n, and thus the MOD gate outputs zero. Negating each such NOR (again using a MOD gate) completes the proof. □

We also mention that Theorem 4.4 generalizes to other depths, in a way analogous to Corollary 3.7:

Corollary 4.5. $\mathsf{TC}^i = \#\mathsf{AC}^i(\mathbb{F}_{p_n}) = \mathsf{AC}^i[p_n] = \mathsf{CC}^i[p_n]$.

For $i \geq 1$ the equality $\mathsf{TC}^i = \mathsf{L}^{\#\mathsf{SAC}^{i,*}(\mathbb{F}_{p_n})}$ also holds, but for $i = 0$ a more careful argument is needed, using AC^0-Turing reducibility in place of logspace-Turing reducibility.

In the next section, it will be necessary to consider arithmetic circuits over certain *rings* (especially the ring of integers mod m for composite m). We present the definition here, rather than in the next section, because this new definition also provides additional characterizations of TC^1, which is the topic of this section.

Definition 4.6. *Let (m_n) be any sequence of natural numbers (where each $m_n > 1$) such that the mapping $1^n \mapsto m_n$ is computable in logspace. We use the notation $\#\mathsf{AC}^1(\mathbb{Z}_{m_n})$ to denote the class of functions f with domain $\{0,1\}^*$ such that there is a logspace-uniform family of arithmetic circuits $\{C_n\}$ of logarithmic depth with unbounded fan-in $+$ and $\times$ gates, where the arithmetic operations of the circuit C_n are interpreted over $\mathbb{Z}_{m_n}$, and*

for any input x of length n, $f(x) = C_n(x)$. We use the notation $\#\mathsf{AC}^1(\mathbb{Z}_{\mathsf{L}})$ to denote the union, over all logspace-computable sequences of moduli (m_n), of $\#\mathsf{AC}^1(\mathbb{Z}_{m_n})$.

Since the sequence of primes (p_n) is logspace-computable, $\mathsf{TC}^1 (= \#\mathsf{AC}^1(\mathbb{F}_{p_n}))$ is clearly contained in $\#\mathsf{AC}^1(\mathbb{Z}_{\mathsf{L}})$. Conversely, all of the functions in $\#\mathsf{AC}^1(\mathbb{Z}_{\mathsf{L}})$ are computable in TC^1. To see this, consider a function $f \in \#\mathsf{AC}^1(\mathbb{Z}_{\mathsf{L}})$. To evaluate $f(x)$ for an input of length n, first we compute the modulus m_n and the circuit C_n. To evaluate each gate g of C_n (in binary), first we compute the sum or product of the values that feed into g (which can be done in constant depth using threshold circuits) and then we reduce the result modulo m_n (which involves division, which can also be computed in constant depth). Thus, arithmetic circuits over the integers mod m_n for reasonable sequences of moduli m_n give yet another arithmetic characterization of TC^1.

4.2. Degree Reduction. The results of Section 3 and Section 4 gave examples of fan-in reduction for arithmetic circuits (showing that ACC^1 and TC^1 can be characterized either in terms of unbounded fan-in or semiunbounded fan-in arithmetic circuits). However, those theorems showed only how to reduce the fan-in of addition gates; thus they did not involve decreasing the algebraic degree of the circuits under consideration. Degree reduction is the topic to which we turn now.

In this subsection, we introduce a class of circuits that is intermediate between the unbounded fan-in circuit model and the semiunbounded fan-in model, for the purposes of investigating when arithmetic circuits of superpolynomial algebraic degree can be simulated by arithmetic circuits (possibly over a different algebra) with much smaller algebraic degree.

The starting point for this subsection is Theorem 4.3 in Allender *et al.* (1998), which states that every problem in AC^1 is reducible to a function computable by polynomial-size arithmetic circuits of degree $n^{O(\log \log n)}$. In this section, we refine the result of Allender *et al.* (1998), and put it in context with the theorems about TC^1 that were presented in the previous subsection. Those results show that TC^1 reduces to semiunbounded fan-in arithmetic circuits in

the $\Lambda\mathsf{P}(\mathbb{F}_{p_n})$ model, but leave open the question of whether TC^1 also reduces to semiunbounded fan-in arithmetic circuits in the $\mathsf{VP}(\mathbb{F}_{p_n})$ model (which coincides with $\mathsf{VP}(\mathbb{Q})$). We are unable to answer this question, but we do show that some interesting inclusions can be demonstrated if we relax the VP model, by imposing a less-stringent restriction on the fan-in of the $\times$ gates.

Definition 4.7. *Let (m_n) be any sequence of natural numbers (where each $m_n > 1$) such that the mapping $1^n \mapsto m_n$ is computable in logspace. $\#\mathsf{WSAC}^1(\mathbb{Z}_{m_n})$ is the class of functions represented by logspace-uniform arithmetic circuit families $\{C_n\}$, where C_n is interpreted over $\mathbb{Z}_{m_n}$, where each C_n has size polynomial in n, and depth $O(\log n)$, and where the $+$ gates have unbounded fan-in, and the $\times$ gates have fan-in $O(\log n)$. We use the notation $\#\mathsf{WSAC}^1(\mathbb{Z}_{\mathsf{L}})$ to denote the union, over all logspace-computable sequences of moduli (m_n), of $\#\mathsf{WSAC}^1(\mathbb{Z}_{m_n})$. In the special case when $m_n = p$ for all n, we obtain the class $\#\mathsf{WSAC}^1(\mathbb{F}_p)$.*

Note that with the $O(\log n)$ fan-in restriction on the $\times$ gates these circuits are not semiunbounded, but do have a "weak" form of the semiunbounded fan-in restriction. We refrain from defining a weakly semiunbounded analog of the $\Lambda\mathsf{P}$ classes, because it is easy to show that they are equivalent to the $\Lambda\mathsf{P}$ classes, since AC^0 circuits can add logarithmically-many numbers, given in binary.

We improve on Allender *et al.* (1998, Theorem 4.3) by showing AC^1 is contained in $\#\mathsf{WSAC}^1(\mathbb{F}_2)$; note that all polynomials in $\#\mathsf{WSAC}^1(\mathbb{F}_p)$ have degree $n^{O(\log\log n)}$, and note also that the class of functions considered in Allender *et al.* (1998) is not obviously even in TC^1. In addition, we improve on Allender *et al.* (1998) by reducing not merely AC^1, but also $\mathsf{AC}^1[p]$ for any prime p. This includes $\Lambda\mathsf{P}(\mathbb{F}_p)$ for any p such that $\mathsf{Supp}(p-1) \subseteq \{2\}$. Also, we obtain an exact characterization of $\mathsf{AC}^1[p]$, whereas Allender *et al.* (1998) presented merely an inclusion.

Theorem 4.8. *Let p be any prime. Then $\mathsf{AC}^1[p] = \#\mathsf{WSAC}^1(\mathbb{F}_p)$.*

Proof. The inclusion $\#\mathsf{WSAC}^1(\mathbb{F}_p) \subseteq \mathsf{AC}^1[p]$ is straightforward. The proof of Corollary 3.9 shows how to simulate semiunbounded fan-in circuits over $\mathbb{F}_p$ by $\mathsf{AC}^1[p]$ circuits. We merely need to add

to that construction, to show how to handle multiplication gates of logarithmic fan-in. Let g be a multiplication gate computing the product of the gates $h_1, \ldots, h_{c \log n}$. As in the proof of Corollary 3.9, the simulating $\mathsf{AC}^1[p]$ circuit will have gates of the form $[h_i = b]$ for all $b \in \mathbb{F}_p$. Thus the value of g depends on only $O(\log n)$ binary bits of the simulating circuit, and the value of $[g = a]$ can be computed by a logspace-uniform DNF expression. This yields the desired $\mathsf{AC}^1[p]$ circuit.

For the proof of the converse inclusion, the main technical ingredient involved is the following lemma from Allender *et al.* (1998). (In Allender *et al.* (1998) the lemma is stated only for MOD_2, but the proof carries over to any MOD_m gate with only trivial changes. (See also the very similar result of Hansen & Koucký (2010, Proposition 3.4).) For completeness, a detailed proof may be found in Appendix A.)

LEMMA 4.9. *(Allender* et al. *1998) Let m be any natural number, $m > 1$. For each $\ell \in \mathbb{N}$, there is a family of constant-depth, polynomial-size, probabilistic circuits consisting of unbounded-fan-in* MOD_m *gates,* AND *gates of fan-in $O(\log n)$, and $O(\log n)$ probabilistic bits, computing the* OR *of n bits, with error probability $< 1/n^\ell$.*

Now we follow closely the proof of Allender *et al.* (1998, Theorem 4.3).

Take an $\mathsf{AC}^1[p]$ circuit, replace all AND gates by OR and MOD_p gates (using DeMorgan's laws), and then replace each OR gate in the resulting circuit with the sub-circuit guaranteed by Lemma 4.9 (for ℓ chosen so that n^ℓ is much larger than the size of the original circuit), with the *same* $O(\log n)$ probabilistic bits re-used in each replacement circuit. The result is a probabilistic, polynomial-size circuit that, with high probability, provides the same output as the original circuit. (This assertion may not be obvious to the reader. We provide a careful proof in Appendix B.)

Note that replacing AND gates by $\times$ and replacing each MOD_p gate g having wires from h_i with a subcircuit of the form $1 + (p - 1)(\sum_i h_i)^{p-1}$, one obtains an arithmetic circuit over the integers, whose value mod p is equal to the output of the original $\mathsf{AC}^1[p]$

circuit with high probability. (This is one place where we use the fact that p is prime.) The circuit has depth $O(\log n)$, and has unbounded fan-in + gates, and all $\times$ gates have fan-in $O(\log n)$, and thus it is a weakly semiunbounded fan-in circuit.

Create $n^{O(1)}$ copies of this probabilistic circuit, one copy for each sequence of probabilistic bits; call these circuits $D_1, \ldots, D_{n^c}$. Note that each D_i computes a value in $\{0, 1\}$. Note also that $1 - D_i$ is also computable in $\#\mathsf{WSAC}^1(\mathbb{F}_p)$. Thus we can feed these values into an arithmetic NC^1 circuit computing MAJORITY (using the fact that all functions in NC^1 are in $\#\mathsf{NC}^1$ (Caussinus *et al.* 1998)). The resulting circuit is equivalent to our original $\mathsf{AC}^1[p]$ circuit. □

We especially call attention to the following corollary, which shows that, over $\mathbb{F}_2$, polynomial size logarithmic depth arithmetic circuits of degree $n^{O(\log n)}$ and of degree $n^{O(\log \log n)}$ represent precisely the same functions!

COROLLARY 4.10. $\#\mathsf{WSAC}^1(\mathbb{F}_2) = \#\mathsf{AC}^1(\mathbb{F}_2) = \mathsf{AC}^1[2] = \Lambda\mathsf{P}(\mathbb{F}_3)$.

PROOF. The containment $\#\mathsf{WSAC}^1(\mathbb{F}_2) \subseteq \#\mathsf{AC}^1(\mathbb{F}_2)$ is immediate from the definition (since $\#\mathsf{WSAC}^1(\mathbb{F}_2)$ circuits are a restricted form of $\#\mathsf{AC}^1(\mathbb{F}_2)$ circuits). The second equality is from Theorem 3.4. The equality $\mathsf{AC}^1[2] = \Lambda\mathsf{P}(\mathbb{F}_3)$ is from Theorem 3.1. The inclusion $\mathsf{AC}^1[2] \subseteq \#\mathsf{WSAC}^1(\mathbb{F}_2)$ is from Theorem 4.8. □

If we focus on the Boolean classes, rather than on the arithmetic classes, then we obtain a remarkable collapse.

THEOREM 4.11. *Let* $1 < m \in \mathbb{N}$. *Then* $\mathsf{AC}^1[m] = \text{log-}\mathsf{AC}^1[m]$.

PROOF. The proof of Theorem 4.8 begins with the statement of Lemma 4.9, which holds for any modulus m. The proof then uses Lemma 4.9 to replace a general $\mathsf{AC}^1[m]$ circuit by an equivalent probabilistic circuit with unbounded fan-in MOD_m gates and AND gates with logarithmic fan-in, using only $O(\log n)$ probabilistic bits.

The proof of Theorem 4.8 proceeds to modify this to obtain an arithmetic circuit. Instead, we simply make polynomially-many copies of this Boolean circuit (one copy for each probabilistic sequence), and take the majority vote of these copies. □

Using Theorem Theorem 3.4 it follows that arithmetic AC^1 circuits over any finite field $\mathbb{F}_p$ can be simulated by Boolean circuits with MOD gates and small fan-in AND gates. It remains open whether this in turn leads to small-degree arithmetic circuits over $\mathbb{F}_p$ when $p > 2$, and also whether the fan-in of the AND gates can be sublogarithmic, without loss of power.

When m is composite, Theorem 4.11 can be improved to obtain an even more striking collapse, by invoking the work of (Hansen & Koucký 2010).

THEOREM 4.12. *Let $m > 1$ not be a prime power. Then $\mathsf{AC}^1[m] = \mathsf{CC}^1[m]$.*

PROOF. Let $p \neq q$ where $\{p, q\} \subseteq \mathsf{Supp}(m)$. It suffices to show how to construct a family of $\mathsf{CC}^1[m]$ circuits to simulate a given $\mathsf{AC}^1[m]$ circuit family.

Hansen and Koucký showed (Hansen & Koucký 2010, Lemma 3.5) that, for every $c > 1$ there is a constant-depth probabilistic circuit composed of MOD_{pq} gates that computes the OR of n variables, using only $O(\log n)$ probabilistic bits, and having error probability less than $1/n^c$. Thus we can replace each unbounded fan-in AND and OR gate in the $\mathsf{AC}^1[m]$ circuit with the corresponding circuit (possibly with negation gates) guaranteed by Hansen & Koucký (2010). The MOD_{pq} gates can be replaced with MOD_m gates via standard techniques, as in the proof of Theorem 3.1. By choosing a suitably large value for c, the resulting probabilistic circuit simulates the original circuit with small error probability.

Now, as in the proof of Theorem 4.11 we can make polynomially-many copies of the probabilistic circuit, hardwiring in different values for the probabilistic bits, and take the majority vote.

□

COROLLARY 4.13.
$\mathsf{ACC}^1 = \bigcup_p \Lambda\mathsf{P}(\mathbb{F}_p) = \bigcup_p \#\mathsf{AC}^1(\mathbb{F}_p) = \bigcup_m \#\mathsf{AC}^1(\mathbb{Z}_m) = \mathsf{CC}^1$.

COROLLARY 4.14. $\mathsf{ACC}^i = \mathsf{CC}^i$ *for all* $i \geq 1$.

This equality is still open for the case $i = 0$, although Hansen and Koucký show that the probabilistic versions of ACC^0 and CC^0 coincide (Hansen & Koucký 2010).

Note that

$$\bigcup_{p \text{ prime}} \mathsf{CC}^1[p] = \bigcup_{p \geq 2} \mathsf{VP}(\mathbb{F}_p) \subseteq \bigcup_{m} \mathsf{L}^{\mathsf{VP}(\mathbb{Z}_m)} \subseteq \mathsf{ACC}^1 = \mathsf{CC}^1.$$

The right-most class corresponds to uniform families of MOD_m gates (for *composite* m), and to arithmetic circuits of degree $n^{O(\log n)}$. The left-most class consists of uniform families of MOD_p gates for *prime* p, and to arithmetic circuits of degree $n^{O(1)}$. The intermediate class corresponds to arithmetic circuits of polynomial degree, but having access to composite moduli. It is natural to wonder how much the composite moduli can help, in simulating higher-degree arithmetic circuits using small degree.

It might be useful to have additional examples of algebras, where some degree reduction can be accomplished. Thus we also offer the following theorem:

THEOREM 4.15. *Let p be any prime. Then*

$$\mathsf{AC}^1[p] \subseteq \mathsf{L}^{\#\mathsf{WSAC}^1(\mathbb{Z}_\mathsf{L})}.$$

PROOF. As in Theorem 4.8, here we need to simulate $\mathsf{AC}^1[p]$ circuits. The proof proceeds precisely as in the proof of Theorem 4.8, up to the construction of the sequence of circuits $D_1, D_2, \ldots, D_{n^c}$ (in the final paragraph of the proof of Theorem 4.8). (These are the copies of the probabilistic circuit simulating the original $\mathsf{AC}^1[p]$ circuit, with different copies of the probabilistic bits hardwired in.)

We now make use of the "Toda polynomials" introduced in Toda (1991). For example, there is an explicit construction in Beigel & Tarui (1994) of a polynomial P_k of degree $2k-1$ such that $(y \bmod p) \in \{0, 1\}$ implies $P_k(y) \bmod p^k = y \bmod p$. It is observed in Allender & Gore (1994) that, for $k = O(\log n)$, the polynomial P_k can be implemented via logspace-uniform constant-depth circuits over the integers. Thus, by replacing each multiplication gate with a tree of fan-in two, the polynomial can be implemented by a semiunbounded fan-in circuit of logarithmic depth.

Applying this polynomial to the output of each circuit D_i, we obtain a $\#\mathsf{WSAC}^1(\mathbb{Z})$ circuit whose value mod p is the same as the output of the original $\mathsf{AC}^1[p]$ circuit with high probability, and with the additional property that the output of the circuit, when represented in p-ary notation, has all of the $c \log n$ low-order symbols of the result equal to zero (except possibly the lowest-order symbol). We will choose c to be the constant such that there are $c \log n$ probabilistic bits). Call the resulting circuit E_i.

Now create a circuit whose output gate computes $\sum_i E_i$. The output gate of the resulting $\#\mathsf{WSAC}^1(\mathbb{Z})$ circuit records a number whose low-order $c \log n$ positions (in p-ary notation) records the number of the n^c copies that output 1. If this number is greater than $n^c/2$, then the original circuit accepted its input; otherwise it rejected its input.

In order to compute this number using $\#\mathsf{WSAC}^1(\mathbb{Z}_{\mathsf{L}})$ instead of $\#\mathsf{WSAC}^1(\mathbb{Z})$, we use this logspace-computable sequence of moduli: $m_n = p^n$. Evaluating the arithmetic over $\mathbb{Z}_{p^n}$ gives the number represented by the low-order n positions of the result, in p-ary notation. A logspace oracle machine, upon being given this number (say, in binary notation) can compute the value of this number modulo $p^{1+c \log n}$ and determine if that number is greater than $n^c/2$, and can thereby determine if the original circuit accepted its input. □

It is natural to wonder whether this theorem can be extended, to allow composite moduli. A direct application of the techniques of Allender & Gore (1994); Beigel & Tarui (1994); Yao (1990) requires multiple applications of the Toda polynomials, and this in turn results in circuits of superlogarithmic depth.

Using Theorem 3.1 and Theorem 4.15 we obtain the following.

Corollary 4.16. *If p is a Fermat prime, then*

$$\Lambda\mathsf{P}(\mathbb{F}_p) \subseteq \mathsf{L}^{\#\mathsf{WSAC}^1(\mathbb{Z}_{\mathsf{L}})}.$$

5. Conclusions, Discussion, and Open Problems

We have introduced the complexity classes $\Lambda\mathsf{P}(R)$ for various algebraic structures R, and have shown that they provide alternative characterizations of well-known complexity classes. Furthermore, we have shown that arithmetic circuit complexity classes corresponding to polynomials of degree $n^{O(\log\log n)}$ also yield new characterizations of complexity classes, such as the equality

$$\mathsf{AC}^1[p] = \log\text{-}\mathsf{AC}^1[p] = \#\mathsf{WSAC}^1(\mathbb{F}_p).$$

Furthermore, in the case when $p = 2$, we obtain the additional collapse

$$\#\mathsf{AC}^1(\mathbb{F}_2) = \mathsf{AC}^1[2] = \log\text{-}\mathsf{AC}^1[2] = \#\mathsf{WSAC}^1(\mathbb{F}_2),$$

showing that algebraic degree $n^{O(\log n)}$ and $n^{O(\log\log n)}$ have equivalent expressive power, in this setting.

We have obtained new characterizations of ACC^1 in terms of restricted fan-in:

$$\mathsf{ACC}^1 = \bigcup_p \#\mathsf{AC}^1(\mathbb{F}_p) = \bigcup_p \Lambda\mathsf{P}(\mathbb{F}_p) = \mathsf{CC}^1.$$

That is, although ACC^1 corresponds to unbounded fan-in arithmetic circuits of logarithmic depth, and to unbounded fan-in Boolean circuits with modular counting gates, no power is lost if the addition gates have bounded fan-in (in the arithmetic case) or if only the modular counting gates have unbounded fan-in (in the Boolean case). It remains unknown if every problem in ACC^1 is reducible to a problem in $\bigcup_m \mathsf{VP}(\mathbb{Z}_m)$, although we believe that our theorems suggest that this is likely. It would be highly interesting to see such a connection between ACC^1 and VP.

We believe that it is fairly likely that several of our theorems can be improved. For instance:

- Perhaps Theorem 4.11 and Theorem 4.12 can be improved, to show that for all m, $\mathsf{AC}^1[m] = \mathsf{CC}^1[m]$. Note that this is already known to hold if m is not a prime power. By Corollary 3.9 this would show that $\mathsf{VP}(\mathbb{F}_p) = \mathsf{AC}^1[p]$ for all

primes p. It would also show that $\#\mathsf{AC}^1(\mathbb{F}_2) = \mathsf{VP}(\mathbb{F}_2) = \Lambda\mathsf{P}(\mathbb{F}_p)$ for every Fermat prime p. (We should point out that this would imply that $\mathsf{AC}^1 \subseteq \mathsf{VP}(\mathbb{F}_p)$ for every prime p, whereas even the weaker inclusion $\mathsf{SAC}^1 \subseteq \mathsf{VP}(\mathbb{F}_p)$ is only known to hold non-uniformly (Gál & Wigderson 1996).)

- Can Corollary 4.16 be improved to hold for all primes p, or even for $\Lambda\mathsf{P}(\mathbb{F}_{p_n})$? The latter improvement would show that $\mathsf{TC}^1 \subseteq \mathsf{L}^{\#\mathsf{WSAC}^1(\mathbb{Z}_\mathsf{L})}$.

- Perhaps one can improve Theorem 4.15, to achieve a simulation of degree $n^{O(1)}$. Why should $n^{O(\log\log n)}$ be optimal? Perhaps this could also be improved to hold for composite moduli?

- If some combinations of the preceding improvements are possible, TC^1 would reduce to $\mathsf{VP}(\mathbb{Q})$, which would be a significant step toward the Immerman-Landau conjecture.

We began this investigation, wondering if the equality $\mathsf{VP}(B_2) = \Lambda\mathsf{P}(B_2)$ could carry over to any other algebraic structure. We think that it appears as if $\mathsf{VP}(\mathbb{F}_p)$ and $\Lambda\mathsf{P}(\mathbb{F}_p)$ are incomparable for every non-Fermat prime $p > 2$, since $\mathsf{VP}(\mathbb{F}_p) = \mathsf{CC}^1[p]$ and $\Lambda\mathsf{P}(\mathbb{F}_p) = \mathsf{CC}^1[\mathsf{Supp}(p-1)]$. That is, these classes correspond to circuits consisting of modular counting gates for completely different sets of primes. For Fermat primes we have $\Lambda\mathsf{P}(\mathbb{F}_p) = \text{log-}\mathsf{AC}^1[2]$ and again the VP and $\Lambda\mathsf{P}$ classes seem incomparable.

For the special case of $p = 2$, we have $\mathsf{VP}(\mathbb{F}_2) = \mathsf{CC}^1[2]$ and $\Lambda\mathsf{P}(\mathbb{F}_2) = \mathsf{AC}^1$. We hold out some hope that $\mathsf{VP}(\mathbb{F}_2) = \mathsf{AC}^1[2]$, in which case it would appear that the VP class could be *more* powerful than the $\Lambda\mathsf{P}$ class – but based on current knowledge it also appears possible that the VP and $\Lambda\mathsf{P}$ classes are incomparable even for $p = 2$.

Some of our theorems overcome various hurdles that would appear to stand in the way of a proof of our conjecture that $\mathsf{ACC}^1 = \bigcup_m \mathsf{L}^{\mathsf{VP}(\mathbb{Z}_m)}$.[5] First, recall that $\mathsf{VP}(\mathbb{Z}_m) \subseteq \mathsf{CC}^1[m]$ (Corol-

[5]Here, "$\mathsf{VP}(\mathbb{Z}_m)$" refers to the class of *functions* defined on $\mathbb{Z}_m$ that are represented by VP circuits, rather than to a class of *languages*. The distinction is significant, as is discussed in Allender & Goodwillie (2015).

lary 3.9). Thus, if the conjecture is correct, then *unbounded* fan-in AND and OR gates would have to be simulated efficiently with *bounded* fan-in AND and OR gates (which in turn can be replaced by MOD gates). But this is true in this context: $\mathsf{AC}^1[m] = \mathsf{CC}^1[m]$, if m is not a prime power (Theorem 4.12). If m is a prime power, then the fan-in can be reduced to $\log n$ (Theorem 4.11). If the fan-in can be reduced to $O(1)$ also in the case of prime power moduli, then $\mathsf{AC}^1[p] = \mathsf{CC}^1[p] = \mathsf{VP}(\mathbb{F}_p)$. If CC^1 circuits can be simulated using an oracle for functions in $\mathsf{VP}(\mathbb{Z}_{m'})$ for some m', then the conjecture holds. (The latter simulation is possible if the MOD gates in the CC^1 circuits are for a prime modulus; see Corollary 3.9.)

A second objection that might be raised against the conjecture deals with algebraic degree. ACC^1 corresponds precisely to polynomial-size logarithmic depth unbounded fan-in arithmetic circuits over finite fields (Corollary 3.5). Such circuits represent polynomials of degree $n^{O(\log n)}$, whereas VP circuits represent polynomials of degree only $n^{O(1)}$. One might assume that there are languages represented by polynomial-size log-depth arithmetic circuits of degree $n^{O(\log n)}$ that actually *require* such large degree in order to be represented by arithmetic circuits of small size and depth.

Our degree-reduction theorem (Corollary 4.10) shows that this assumption is incorrect. Every Boolean function that can be represented by an arithmetic AC^1 circuit over $\mathbb{F}_2$ (with algebraic degree $n^{O(\log n)}$) can be represented by an arithmetic AC^1 circuit over $\mathbb{F}_2$ where the multiplication gates have fan-in $O(\log n)$ (and thus the arithmetic circuit has algebraic degree $n^{O(\log \log n)}$).

Acknowledgements

A preliminary version of this work appeared as Allender *et al.* (2015). The first and third authors acknowledge the support of NSF grants CCF-0832787, CCF-1064785, and CCF-1555409. The second author was supported in part by NSF grant CCF-1018060. Part of this work was done while the second author was visiting the Simons Institute for the Theory of Computing, at the University of California, Berkeley. We also acknowledge stimulating conversations with Meena Mahajan, which occurred at the 2014 Dagstuhl Workshop on the Complexity of Discrete Problems (Dagstuhl Sem-

inar 14121), and illuminating conversations with Stephen Fenner and Michal Koucký, which occurred at the 2014 Dagstuhl Workshop on Algebra in Computational Complexity (Dagstuhl Seminar 14391). We also thank Igor Shparlinski and our Rutgers colleagues Richard Bumby, John Miller and Steve Miller, for helpful pointers to the literature, as well as helpful feedback from Pascal Koiran, Daniel Kane, Michaël Cadilhac, Charles Paperman, and Russell Impagliazzo. Finally, we thank the anonymous referees for their careful reading and insightful comments, which improved both the readability and correctness of our arguments.

References

M. Agrawal, E. Allender & S. Datta (2000). On TC^0, AC^0, and Arithmetic Circuits. *Journal of Computer and System Sciences* **60**, 395–421.

E. Allender & V. Gore (1994). A uniform circuit lower bound for the permanent. *SIAM Journal on Computing* **23**, 1026–49.

E. Allender, J. Jiao, M. Mahajan & V. Vinay (1998). Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theoretical Computer Science* **209**, 47–86.

E. Allender, K. Reinhardt & S. Zhou (1999). Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences* **59**(2), 164–181.

Eric Allender, Anna Gál & Ian Mertz (2015). Dual VP Classes. In *Proc. 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, number 9235 in Lecture Notes in Computer Science, 14–25. Springer Berlin Heidelberg.

Eric Allender & Asa Goodwillie (2015). Arithmetic circuit classes over $\mathbb{Z}_m$. Technical Report 15-145, Electronic Colloquium on Computational Complexity (ECCC).

Eric Allender & Michal Koucký (2010). Amplifying Lower Bounds by Means of Self-Reducibility. *Journal of the ACM* **57**, 14:1 – 14:36.

R. Beigel & J. Tarui (1994). On ACC. *Computational Complexity* **4**, 350–366. Special issue on circuit complexity.

L. Blum, F. Cucker, M. Shub & S. Smale (1998). *Complexity and Real Computation*. Springer.

A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo & M. Tompa (1989). Two applications of inductive counting for complementation problems. *SIAM Journal on Computing* **18**, 559–578. See Erratum in SIAM J. Comput. 18, 1283.

Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff & Florian Speelman (2014). Computing with a full memory: catalytic space. In *ACM Symposium on Theory of Computing (STOC)*, 857–866.

Peter Bürgisser (1999). On the Structure of Valiant's Complexity Classes. *Discrete Mathematics & Theoretical Computer Science* **3**(3), 73–94.

Peter Bürgisser (2000). Cook's versus Valiant's hypothesis. *Theoretical Computer Science* **235**(1), 71–88.

Hervé Caussinus, Pierre McKenzie, Denis Thérien & Heribert Vollmer (1998). Nondeterministic NC^1 Computation. *Journal of Computer and System Sciences* **57**(2), 200–212.

Suresh Chari, Pankaj Rohatgi & Aravind Srinivasan (1995). Randomness-Optimal Unique Element Isolation with Applications to Perfect Matching and Related Problems. *SIAM Journal on Computing* **24**(5), 1036–1050.

A. Chiu, G.I. Davida & B. Litow (2001). Division in logspace-uniform NC^1. *RAIRO Theoretical Informatics and Applications* **35**, 259–276.

Capi Corrales-Rodrigáñez & René Schoof (1997). The support problem and its elliptic analogue. *Journal of Number Theory* **64**(2), 276–290.

Ofer Gabber & Zvi Galil (1981). Explicit Constructions of Linear-Sized Superconcentrators. *Journal of Computer and System Sciences* **22**(3), 407–420.

Anna Gál (1995). Semi-Unbounded Fan-In Circuits: Boolean vs. Arithmetic. *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995* 82–87.

Anna Gál & Avi Wigderson (1996). Boolean complexity classes vs. their arithmetic analogs. *Random Struct. Algorithms* **9**(1-2), 99–111.

Michael Hahn, Andreas Krebs, Klaus-Jörn Lange & Michael Ludwig (2015). Visibly Counter Languages and the Structure of NC^1. In *Symposium on Mathematical Foundations of Computer Science (MFCS)*, number 9235 in Lecture Notes in Computer Science, 384–394. Springer.

Kristoffer Arnsfelt Hansen & Michal Koucký (2010). A New Characterization of ACC^0 and Probabilistic CC^0. *Computational Complexity* **19**(2), 211–234.

William Hesse, Eric Allender & David A. Mix Barrington (2002). Uniform Constant-Depth Threshold Circuits for Division and Iterated Multiplication. *Journal of Computer and System Sciences* **65**, 695–716.

N. Immerman & S. Landau (1995). The Complexity of Iterated Multiplication. *Information and Computation* **116**, 103–116.

Russell Impagliazzo & David Zuckerman (1989). How to Recycle Random Bits. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 248–253.

Pascal Koiran & Sylvain Perifel (2011). Interpolation in Valiant's Theory. *Computational Complexity* **20**(1), 1–20.

Guillaume Malod & Natacha Portier (2008). Characterizing Valiant's algebraic complexity classes. *J. Complexity* **24**(1), 16–38.

Cristopher Moore, Denis Thérien, François Lemieux, Joshua Berman & Arthur Drisko (2000). Circuits and Expressions with Nonassociative Gates. *Journal of Computer and System Sciences* **60**(2), 368–394.

J. Reif & S. Tate (1992). On threshold circuits and polynomial computation. *SIAM Journal on Computing* **21**, 896–908.

K. Reinhardt & E. Allender (2000). Making Nondeterminism Unambiguous. *SIAM Journal on Computing* **29**, 1118–1131.

R. Smolensky (1987). Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *ACM Symposium on Theory of Computing (STOC)*, 77–82.

H. Straubing (1994). *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston.

Denis Thérien (1994). Circuits constructed with MOD_q gates cannot compute "AND" in sublinear size. *Computational Complexity* **4**(4), 383–388.

S. Toda (1991). PP Is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing* **20**, 865–877.

L.G. Valiant (1979). Completeness classes in algebra. In *Proc. 11th ACM STOC*, 249–261.

L.G. Valiant, S. Skyum, S. Berkowitz & C. Rackoff (1983). Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing* **12**(4), 641–644.

H. Venkateswaran (1991). Properties That Characterize LOGCFL. *Journal of Computer and System Sciences* **43**, 380–404.

V Vinay (1991). Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of 6th Structure in Complexity Theory Conference*, 270–284.

H. Vollmer (1999). *Introduction to Circuit Complexity: A Uniform Approach.* Springer-Verlag New York Inc.

J. von zur Gathen (1993). Parallel linear algebra. In *Synthesis of Parallel Algorithms*, J. Reif, editor, 574–615. Morgan Kaufmann.

William P Wardlaw (1994). Matrix representation of finite fields. *Mathematics Magazine* 289–293.

Ryan Williams (2014). Nonuniform ACC circuit lower bounds. *Journal of the ACM* **61**(1), 2.

T. Xylouris (2011). On the least prime in an arithmetic progression and estimates for the zeros of Dirichlet L-functions. *Acta Arithmetica* **150**, 65–91.

Andrew Chi-Chih Yao (1990). On ACC and Threshold Circuits. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 619–627.

A. Appendix: Proof of Lemma 4.9

In this section, we present a detailed proof of Lemma 4.9, showing the adjustments that need to be made, in order to deal with arbitrary MOD_m gates.

Here is a reminder of the statement of Lemma 4.9: *Let m be any natural number, $m > 1$. For each $\ell \in \mathbb{N}$, there is a family of constant-depth, polynomial-size, probabilistic circuits consisting of unbounded-fan-in* MOD_m *gates,* AND *gates of fan-in $O(\log n)$, and $O(\log n)$ probabilistic bits, computing the* OR *of n bits, with error probability $< 1/n^\ell$.*

Proof. Our presentation here is a slight adjustment of the proof in Allender *et al.* (1998). There are no significant changes in the proof, which relies crucially on the fact that one can replace an OR gate with a MOD gate, when there is a guarantee that at most one of the inputs to the OR gate evaluates to 1.

The construction in Chari *et al.* (1995) gives a depth 5 probabilistic circuit that computes the NOR correctly with probability at least $\frac{1}{2}$ and uses $O(\log n)$ random bits. More precisely, using the terminology of Chari *et al.* (1995), let $m' = \lceil \log n \rceil$, let $S = \{1, \ldots, m'\}$, and let $\mathcal{F}$ be the collection of subsets of S, such that $A \in \mathcal{F}$ iff the bit string k of length $m' = \lceil \log n \rceil$ representing the characteristic sequence of A corresponds to a binary number $k \leq n$ such that the k-th bit of the input sequence $x_1, \ldots, x_n$ has value 1. That is, the OR of $x_1, \ldots, x_n$ evaluates to 1 iff $\mathcal{F}$ is not empty. The strategy of Chari *et al.* (1995) is to use probabilistic bits to define a way of assigning a "weight" to each set $A_k \in \mathcal{F}$ so that if $\mathcal{F}$ is not empty, then with high probability there is a

unique element of $\mathcal{F}$ having minimum weight. The next paragraph explains how this is done.

Let $c = \lceil \log m' \rceil$ and let $t = \lceil m'/c \rceil$. For any $1 \leq i \leq m'$ and $0 \leq j \leq t-1$, define $b_{i,j}$ as follows:

$$b_{i,j} = \begin{cases} 2^{i-jc} & \text{if } jc < i \leq (j+1)c \\ 0 & \text{otherwise} \end{cases}$$

(It may help the reader's intuition to consider an m'-bit sequence $k = k_1, \ldots, k_{m'}$. Divide this sequence into blocks; Block(j) has positions $k_{jc+1}, k_{jc+2}, \ldots, k_{(j+1)c}$. Clearly, $k_{m'}$ is in Block(k_{t-1}). Now, if $k_i \notin$ Block(j), then $b_{i,j} = 0$, else $b_{i,j} = 2^{i-jc}$. Note that $i - jc$ is the position of k_i within Block(j).)

Choose t numbers $r_0, \ldots, r_{t-1}$ in the range $1 \leq r_j \leq 50 \log^5 n$ uniformly and independently at random (and note that this involves choosing $O(\log n)$ random bits). Finally, define w_i to be equal to $\sum_{j=0}^{t-1} b_{i,j} r_j$. The weight of a set A is then $\sum_{i \in A} w_i$. The analysis in Proposition 2 of Chari *et al.* (1995) shows that if $\mathcal{F}$ is not empty, then with probability at least .99, there is a unique minimal weight set in $\mathcal{F}$.

This paragraph explains how to implement this system as a uniform constant-depth circuit. Note first that for any $k \leq n$ and for any fixed $p \leq \log^7 n$, there is a depth 2 circuit of MOD_m gates and small-fan-in AND gates that evaluates to 1 iff the weight of A_k is equal to p. Here A_k is that subset of S whose characteristic sequence is the binary representation of k. (To see this, note that the only inputs to this circuit are the $O(\log n)$ probabilistic bits. Thus the DNF for this function can be computed in logspace, and the OR gate at the root can be replaced by a MOD_m gate with $m-1$ additional 1 inputs. Here we are making use of the fact that there can only be one of the AND gates that feed into to the MOD_m gate that returns 1, namely the one where the weight of $A_k = p$.)

Taking the AND of this circuit with the input bit x_k results in a depth three circuit that evaluates to 1 iff $A_k \in \mathcal{F}$ and the weight of A_k is equal to p. Thus there is a polynomial-size depth-4 circuit with a MOD_m gate at the root (with $m-1$ additional 1 inputs) that evaluates to 1 iff the number of sets in $\mathcal{F}$ that have weight p

is equivalent to 1 mod m. Hence there is a uniform depth-5 circuit with an OR at the root that evaluates to 1 iff there is some weight $p \leq \log^7 n$ such that the number of sets in $\mathcal{F}$ having weight p is equivalent to 1 mod m. By the remarks in the preceding paragraph, if the OR of $x_1, \ldots, x_n$ evaluates to 1, then with probability at least .99, our depth-5 circuit will also. (Clearly, if the OR is zero, then the depth-5 circuit also evaluates to zero.) If we replace the OR gate at the root with AND and negate each of the MOD_m gates that feed into that OR gate (recalling that a unary MOD_m gate is a NOT gate) we obtain our desired circuit for the NOR function – except that the fan-in of the gate at the root is $\log^7 n$, and our goal is for the AND gates to have fan-in at most $\log n$. But replacing this AND gate with a depth-7 tree of AND gates of fan-in $\log n$ yields an equivalent circuit of the desired form. Let us denote this circuit by $C(x, r)$.

It remains only to reduce the error probability from $\frac{1}{100}$ to $\frac{1}{n^l}$, without using too many additional probabilistic bits. We accomplish this using a standard construction, as in Allender *et al.* (1998): Consider a graph with vertices for each of our $O(\log n)$-bit probabilistic sequences, the edge relation is given by the construction of an expander graph presented in Gabber & Galil (1981), where each vertex has degree five. Inspection of Gabber & Galil (1981) shows that, in logspace, one can take as input one of our original probabilistic sequences r as well as a new probabilistic sequence $s \in \{1, 2, 3, 4, 5\}^{c\ell \log n}$ (for some constants c and ℓ) and output the vertex r' reached by starting in vertex r and following the sequence of edges indicated by s. Since this function depends on only $O(\log n)$ bits, the DNF for this function can be computed in logspace, and (as above) can be implemented using a MOD_m gate and AND gates of small fan-in. Let this circuit be denoted by $R(r, s)$.

Thus we can construct a constant-depth circuit that computes the AND for all $i \leq cl \log n$ of $C(x, R(r, s[1..i]))$ (where $s[1..i]$ denotes the prefix of s of length i, where r and s are probabilistically chosen. By Section 2 of Impagliazzo & Zuckerman (1989), this circuit computes the NOR correctly with probability $1 - \frac{1}{n^l}$. Adding

a MOD_m gate at the root allows us to compute the OR, as desired. This completes the proof of the lemma. □

B. Appendix: An assertion from Theorem 4.8

In this appendix, we provide a careful proof of the following assertion, which was made in the proof of Theorem 4.8:

Let C be a polynomial-size circuit of MOD_p gates and OR gates. Replace each OR gate in the resulting circuit with the sub-circuit guaranteed by Lemma 4.9 (for ℓ chosen so that n^ℓ is much larger than the size of the original circuit C), with the *same* $O(\log n)$ probabilistic bits re-used in each replacement circuit. Call the resulting circuit C'. Then C' is a probabilistic, polynomial-size circuit that, with high probability, provides the same output as the original circuit.

This follows from the following, slightly stronger claim.

Let C_g be the probabilistic subcircuit of C' that replaces an OR gate g of C. Let g' be the output gate of C_g. We claim that, for every input x, for most settings of the probabilistic bits, the values of each gate g of C agrees with the value of the corresponding gate g' of C' on input x.

To establish this claim, consider some topological sort of the OR gates; i.e., a linear order so that if g comes before h, then there is no path from h to g. Choose any input x. Let E_g be the event that g is the first gate in this order such that g' and g take on different values on input x. The probability that there is any gate g such that g and g' take on different values is equal to $\Pr(\bigcup_g E_g) \leq \sum_g \Pr(E_g)$. Let z_g be the sequence of bits that is input to gate g in C on input x. Then

$$\begin{aligned} \Pr(E_g) &= \Pr(g' \neq \mathrm{OR}(z_g) \text{ and } \neg E_h \text{ for all } h < g) \\ &\leq \Pr(g' \neq \mathrm{OR}(z_g)). \end{aligned}$$

By Lemma 4.9, $\Pr(g' \neq \mathrm{OR}(z_g)) \leq 1/n^\ell$. Thus $\Pr(\bigcup_g E_g) \leq \sum_g 1/n^\ell$, which can be made as small as n^{-c} by appropriate choice of the constant ℓ.)

C. Appendix: diagram of new macro and micro inclusions

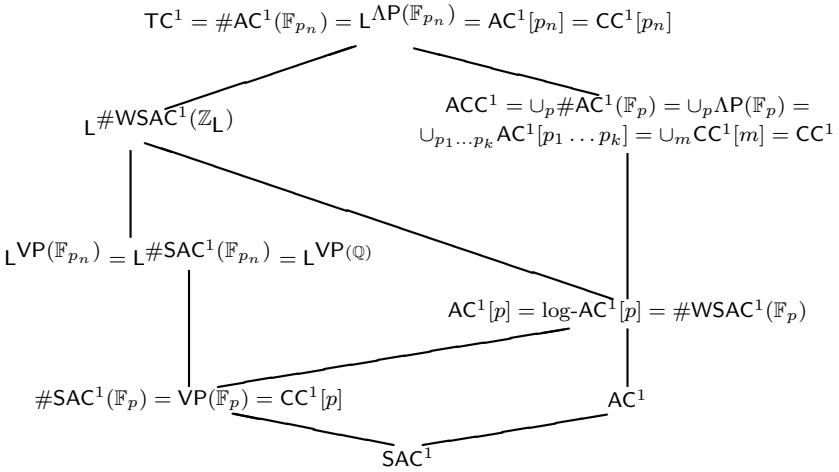

Figure C.1: Macro inclusions within TC^1

Manuscript received 25 September, 2015

Eric Allender
Department of Computer Science,
Rutgers University
Piscataway, NJ, USA
allender@cs.rutgers.edu

Anna Gál
Department of Computer Science,
University of Texas
Austin, TX, USA
panni@cs.utexas.edu

Ian Mertz
Department of Computer Science,
Rutgers University
Piscataway, NJ, USA
iwmertz@gmail.com

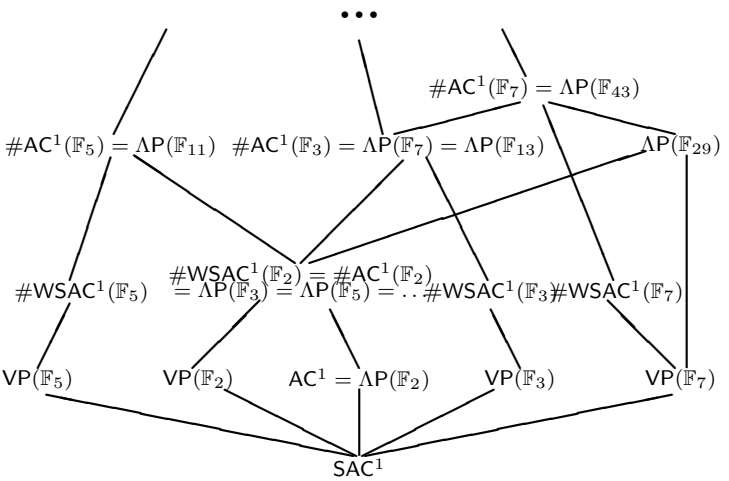

Figure C.2: Micro inclusions within ACC^1