A UNIFORM CIRCUIT LOWER BOUND FOR THE PERMANENT

ERIC ALLENDER* AND VIVEK GORE†

Abstract. We show that uniform families of ACC circuits of subexponential size cannot compute the permanent function. This also implies similar lower bounds for certain sets in PP. This is one of the very few examples of a lower bound in circuit complexity whose proof hinges on the uniformity condition; it is still unknown if there is any set in Ntime $(2^{n^{O(1)}})$ that does not have nonuniform ACC circuits.

Key words. circuit complexity, uniformity, permanent, lower bounds, complexity classes

AMS subject classifications. 68Q05, 68Q15, 03D15

1. Introduction. Circuit complexity classes consisting of circuits of constant depth and polynomial size have been intensely studied in the last decade. The first such class to be studied was AC^0, the class of languages accepted by polynomial size, constant depth circuits consisting of NOT gates and unbounded fan-in AND and OR gates. Machinery for proving lower bounds for AC^0 has been developed in a series of papers, culminating in the powerful and elegant techniques of [18, 29, 3]. These papers provide exponential size lower bounds for constant depth circuits computing the PARITY function. These lower bounds prompted people to look at constant depth, polynomial size circuits with PARITY gates along with AND, OR and NOT gates but Razborov [22] proved that these circuits could not compute the MAJORITY function. Smolensky [25] extended Razborov's method to show that an AC^0 circuit with MOD_p gates cannot compute the MOD_q function if p and q are distinct primes. This implies that no AC^0 circuit containing MOD gates for a single prime can compute the MAJORITY function. Therefore, the next natural extension of the above class was to allow MOD_m gates for composite moduli m. This extension is known as the class ACC, and it was introduced (implicitly) by Barrington in [4]. Though there has been a fair amount of research on ACC, we still do not know much about this class except the trivial fact that $AC^0 \subsetneq ACC \subseteq NC^1$ where NC^1 is the class of languages accepted by polynomial size, $O(\log n)$ depth circuits with NOT gates and bounded fan-in AND and OR gates. Barrington [4] has conjectured that ACC $\subsetneq NC^1$.

Yao [30] proved the first nontrivial upper bounds on the power of ACC circuits, showing that each set in ACC is accepted by a family of depth three threshold circuits of size $2^{(\log n)^{O(1)}}$; these bounds were slightly improved by Beigel and Tarui [10]. These results have been proved for nonuniform ACC. We are, however, interested in the uniform version of ACC.

A circuit family consists of a sequence of circuits $C_1, C_2, \ldots$, where circuit C_n takes n Boolean inputs. The circuit family is *uniform* if a description of C_n can be computed efficiently from n; otherwise the circuit family is said to be *nonuniform*. The original motivation for studying uniform circuit families came from a desire to relate time and space complexity classes to circuit complexity (see, e.g., [11]). Some

* Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903 (allender@cs.rutgers.edu). Supported in part by NSF grants CCR-9000045 and CCR-9204874. Some of this work was done while the author was on sabbatical leave at Princeton University.

† Program in Computing, Department of Mathematics, University of California, Los Angeles, California 90024-1555 (gore@math.ucla.edu). This work was done while the author was a student at Rutgers University, and was supported in part by a Rutgers University Graduate Excellence Fellowship, and by NSF grant CCR-9204874.

sort of uniformity condition is essential for this endeavor to succeed, since it is an easy observation that there are sets with trivial circuit complexity that are not even recursive. The question of exactly which uniformity condition one should use has proved to be somewhat controversial, and largely it has been a matter of taste. When providing upper bounds, or when defining complexity classes, as a practical matter it usually makes no difference which uniformity condition one uses. For example, Ruzzo [23] considers a number of related uniformity conditions, and shows that, for all $k \geq 2$, NC^k consists of languages defined by uniform circuits of polynomial size and $O((\log n)^k)$ depth, no matter which of those uniformity conditions is considered. For very small complexity classes, however, the uniformity condition is sometimes crucial. For example, P-uniform NC^1 circuits are known for division [8], but it remains an open question whether one can improve this result using a more restrictive uniformity condition. Similarly, [6] presents a number of beautiful characterizations of subclasses of NC^1 using Dlogtime uniformity, but these characterizations are not believed to hold if less restrictive uniformity conditions are used. In this paper, we consider uniform circuits out of necessity. The lower bounds that we present are not known to hold in the nonuniform setting.

Before we can state our results, we need a few technical definitions. We are interested in two classes of subexponential functions that we call *subexp* and *subsubexp*. Let us call a function f *constructible* if $f(n) = 2^{g(n)}$, where $g(n)$ can be computed from n (in binary) in time polynomial in $g(n)$. Let *subexp* denote the class of all monotonic functions that are bounded above by some constructible function f such that $\forall \epsilon > 0, f(n) = o(2^{n^{\epsilon}})$. Let *subsubexp* denote any class of monotonic functions closed under composition with polynomials, such that for any two functions f and g in this class, the composition of f and g is in *subexp*.

A typical example of a function in *subexp* is $2^{n^{1/\log^* n}}$, and typical choices for *subsubexp* are $n^{\log^{O(1)} n}$ or $2^{(\log n)^{O(\log\log n)}}$. It is not hard to prove that if s is in *subexp*, then so is $s^{(\log s)^k}$, for any constant k.

In this paper, we provide lower bounds for the classes of languages accepted by uniform circuit families of ACC circuits of subsubexponential and subexponential size. Let those classes be denoted by ACC(*subexp*) and ACC(*subsubexp*). Formal definitions can be found in Section 2 (Definition 2.10). For the rest of this section, we assume that ACC, ACC(*subexp*) and ACC(*subsubexp*) denote the uniform versions of these classes for the notion of uniformity defined in Section 2 (Definition 2.9). Any other notions of uniformity that we use will be mentioned explicitly. We show that PERM (the permanent of a matrix) is not in ACC(*subexp*) and that PP $\not\subseteq$ ACC(*subsubexp*). We are also able to show that ACC $\subsetneq$ $\mathrm{C}_=\mathrm{P}$ and that $\mathrm{C}_=\mathrm{P} \not\subseteq$ ACC(*subsubexp*). Our main tool in proving these results is the following theorem:

THEOREM 1.1. *There is a set Y in PP such that ACC(subexp), $\subseteq$ Dtime$(n^2)^Y$.*

Theorem 1.1 trivially gives us an important corollary (which also follows from a more general lower bound proved in Theorem 3.5 later in the paper):

COROLLARY 1.2. *ACC $\subsetneq$ PP.*

Proof. Theorem 1.1 implies that ACC $\subseteq$ Dtime$(n^2)^Y$ for some $Y \in$ PP. Since ACC $\subseteq$ PP, suppose for the sake of contradiction that ACC = PP. Then ACC = P = PP. Therefore, Dtime$(n^3)^Y \subseteq \mathrm{P}^Y \subseteq \mathrm{P} =$ ACC $\subseteq$ Dtime$(n^2)^Y$. But this contradicts the time hierarchy theorem of [17]. □

This seems to be one of the very few instances where lower bounds are known for the uniform circuit complexity of certain languages or functions, but where nothing is known about the nonuniform circuit complexity. In fact, the only other instance that we are aware of is that it is not known if EXPTIME contains sets that are not in P/poly (the class of languages accepted by nonuniform circuit families of polynomial size), whereas it does contain sets that are not in P (which is the class of languages accepted by uniform circuit families of polynomial size). In contrast with our results, the combinatorial and algebraic techniques developed in [18, 22, 25] make no use of uniformity, and thus they provide lower bounds on nonuniform circuit size. The uniformity condition is critical in the proof of Theorem 1.1; it is still unknown if PP = Dlogspace-uniform ACC. Although Dlogspace-uniform ACC is trivially seen to be properly contained in PSPACE, it is not known if P-uniform ACC = PSPACE. In fact, it is even unknown if there is any set in Ntime $(2^{n^{O(1)}})$ that is not accepted by a nonuniform ACC circuit family.

To prove Theorem 1.1, we will first use the results of Toda [26], Yao [30] and Beigel and Tarui [10] to convert a circuit family in ACC(*subexp*) into an equivalent circuit family of depth two circuits with a symmetric gate at level two, AND gates of small fan-in at level one and the input gates at level zero. However, since we need the resulting circuit family to be uniform as well, we need to show that the above conversion process can be done uniformly. We then show that the language recognized by the new circuit family can be quickly recognized by a deterministic Turing machine that has access to a particular oracle set in PP. Results about PERM then follow from Valiant's [27] results about the class #P.

Section 2 presents some basic definitions. The following section states Theorem 3.1, which is a uniform version of the main result of [10]. Theorem 3.1 is then used to prove the main results of the paper. The final section of the paper presents conclusions and open problems.

The proof of Theorem 3.1, which is the longest and most technically-involved part of the paper is presented in the Appendix. Even though the basic machinery of the proof was developed in [30, 10], there are many obstacles to overcome to ensure that one maintains uniformity.

2. Preliminaries.

We will assume that the reader is familiar with circuits and standard complexity classes such as NP, PP, PH, etc., and the various notions of reducibility.

DEFINITION 2.1. *Let m be a positive integer. A* MOD_m gate *outputs 1 if the sum of its (binary) inputs is 0 modulo m; 0, otherwise. That is,*

$$MOD_m(x_1,\ldots,x_n) = \begin{cases} 1 & \textit{if } \sum_i x_i \equiv 0 \pmod{m} \\ 0 & \textit{otherwise.} \end{cases}$$

DEFINITION 2.2. *A* MAJORITY gate *with n inputs outputs 1 if $\frac{n}{2}$ or more of its inputs are 1; 0, otherwise. That is,*

$$\text{MAJORITY}(x_1,\ldots,x_n) = \begin{cases} 1 & \textit{if } \sum_i x_i \geq \frac{n}{2} \\ 0 & \textit{otherwise.} \end{cases}$$

DEFINITION 2.3. ([21, 4, 7]) *A language L is in ACC if there exists a positive integer m such that L is recognized by a family of constant depth polynomial size circuits containing NOT gates and unbounded fan-in AND, OR and MOD_m gates.*

ACC was first defined and studied in [21, 4, 7] under the name ACC^0. Barrington and Thérien showed that ACC is equal to the class of languages recognized by polynomial length programs over solvable monoids [7]. Razborov [22] and Smolensky

[25] also studied bounded depth circuits containing AND, OR and MOD gates. Yao's definition of ACC is slightly different from the one given by Barrington et al; it allows a fixed finite set S of moduli instead of a single modulus m. It is easy to see that a MOD_m gate can simulate a MOD_k gate for any k that divides m. Letting m be the least common multiple of the elements in S makes the two definitions equivalent. Yao [30] showed that every language in ACC is recognized by a family of depth two probabilistic circuits with a symmetric gate at level two and $2^{(\log n)^{O(1)}}$ AND gates having fan-in $(\log n)^{O(1)}$ at level one. Beigel and Tarui [10] improved this to show the existence of deterministic circuit families of this sort.

Definition 2.4. *For an NP machine M, let $\#M$ be the function $\#M : \Sigma^* \to \mathrm{N}$ defined by $\#M(x) =$ number of accepting paths of M on input* x. *Then $\#P = \{\#M : M$ is an NP machine$\}$.*

It is well known from [27] that PERM is complete for #P under polynomial time many-one reductions ($\leq_m^p$). (See also [31, 9].)

Definition 2.5. *A language L is said to be in $PrTime(t(n))$ if there exists a nondeterministic machine M that runs in time $t(n)$ such that for all $x \in \Sigma^*$,*
$x \in L \iff$ more than half of the computation paths of M on input x are accepting.

Definition 2.6. *A language L is said to be in $C_=Time(t(n))$ if there exists a nondeterministic machine M that runs in time $t(n)$ such that for all $x \in \Sigma^*$,*
$x \in L \iff$ exactly half of the computation paths of M on input x are accepting.

In particular, for polynomial running times we get the well known classes PP $=$ $\mathrm{PrTime}(n^{O(1)})$ and $\mathrm{C_=P} = \mathrm{C_=Time}(n^{O(1)})$.

Definition 2.7. *Let $\{C_n\}$ be a family of circuits. Following* [23], we define the *direct connection language L* of $\{C_n\}$ as:

$$L = \{\langle n, g_1, g_2\rangle : g_1 = g_2 \text{ and } g_1 \text{ is a gate in } C_n \text{ or } g_1 \neq g_2 \text{ and } g_2 \text{ is an input to } g_1 \text{ in } C_n\}.$$

Here g_1 and g_2 are names of gates and n is in binary notation.

Definition 2.8. *A circuit family $\{C_n\}$ is* dlogtime-uniform *if its direct connection language can be recognized in linear time by a deterministic Turing machine. The Turing machine that recognizes the direct connection language of $\{C_n\}$ will be referred to as the* uniformity machine *for $\{C_n\}$.*

The above notion of uniformity is the one that is generally used for small complexity classes (see [6, 12, 23]). However, we are going to use a slightly less restrictive notion of uniformity for our results. Our notion of uniformity can be informally referred to as Polylogtime-uniformity. The reason that we use this notion is that we are dealing with circuits of possibly superpolynomial size and the proofs are much simpler with this uniformity condition. It should be noted that a set has uniform ACC(*subexp*) circuits with respect to our notion of uniformity if and only if it has Dlogtime-uniform ACC(*subexp*) circuits. This can be established by "padding" a circuit with many dummy gates.

Definition 2.9. *A circuit family $\{C_n\}$ is* uniform *if its direct connection language can be recognized in polynomial time by a deterministic Turing machine. Note that the time is polynomial with respect to the length of the strings in the language ($|\langle n, g_1, g_2\rangle|$) and not merely polynomial in n.*

Definition 2.10. *Let $ACC(s(n))$ denote the class of languages accepted by circuit families of constant depth circuits with NOT gates and unbounded fan-in AND, OR and MOD_m gates (for some integer $m \geq 2$) of size $s(n)$. Then*

$$ACC(subexp) = \bigcup_{s \in subexp} ACC(s(n))$$

$$ACC(subsubexp) = \bigcup_{s \in subsubexp} ACC(s(n))$$

Throughout the rest of the paper, classes ACC, ACC(*subexp*) and ACC(*subsubexp*) denote <u>uniform</u> circuit classes according to the notion of uniformity in Definition 2.9 unless the uniformity condition is mentioned explicitly.

3. The main results.

For the proof of Theorem 1.1, we will first show the following.

THEOREM 3.1. *Suppose L is accepted by an ACC(subexp) circuit family. Then L is accepted by a uniform, depth two circuit family*[1] *of $s(n)$ sized circuits that have the following properties:*

1. *Level one consists of a subexponential number of AND gates having fan-in $(\log s(n))^{O(1)}$. Furthermore, given the name of one of these AND gates, the exact fan-in of this AND gate can be computed deterministically in time $(\log s(n))^{O(1)}$.*
2. *There is a symmetric gate at level two. Furthermore, given the number m of AND gates that evaluate to one, it can be determined deterministically in time $(\log s(n))^{O(1)}$ if the symmetric gate will evaluate to one.*

The above theorem is the most important part of the argument. It is equivalent to saying that the main theorem of [10] holds also in the setting of uniform circuit complexity. Unfortunately, transformations that are obvious in the nonuniform setting require considerable care when undertaken in the uniform setting; we present a complete proof of Theorem 3.1 in the Appendix. The rest of this section assumes that Theorem 3.1 is true and uses it to prove our main results.

Proof. (of Theorem 1.1) Let $\{C_n\}$ be a circuit family in ACC(*subexp*) that accepts L. Using the result in Theorem 3.1, we can get a uniform family of circuits $\{D_n\}$ such that for every n, D_n is a deterministic depth two circuit having the properties mentioned in the statement of Theorem 3.1.

Let M_L be a nondeterministic Turing machine that, on input x, guesses the name of one of the AND gates of D_n ($n = |x|$) and the names of all the inputs of D_n that are connected to this gate. It verifies that the guesses are correct (using the uniformity machine for $\{D_n\}$). It then accepts if and only if the AND gate evaluates to 1 when x is the input to D_n. Since $\{D_n\}$ is uniform and the AND gates have fan-in $o(n^\epsilon)$ (for every ϵ), M_L can do this computation in linear time. Note that $\#M_L(x)$ is the number of AND gates of D_n that evaluate to 1 on input x.

Let $M_1, M_2, \ldots$ be an enumeration of nondeterministic machines running in linear time. Define the set Y to be $\{\langle i, x, l\rangle : x \in \{0,1\}^* \text{and} \#M_i(x) > l\}$. Note that Y is in PP[2]. With oracle Y, a deterministic machine (say M) can compute $\#M_L(x)$ in time n^2 using the binary search technique. Then, since this is the number of AND gates of D_n that evaluate to 1 on input x, it can then in linear time determine if D_n accepts x, using the properties guaranteed by Theorem 3.1. Thus membership of x in L can be determined in time n^2 relative to oracle Y. (Note that the running time

[1] This circuit family is not an ACC(*subexp*) circuit family because the circuits have arbitrary symmetric gates at their roots. When we say that it is uniform, we are using a slightly different notion of uniformity which is explained in Definition A.22.

[2] Let M be a nondeterministic machine that is given input $\langle i, x, l\rangle$. Suppose $t(|x|)$ is the total number of paths of M_i on input x. The computation of M will have $2t(|x|)$ paths; the first $t(|x|)$ of those consist of $t(|x|) - l$ trivially accepting and l trivially rejecting paths, and the other $t(|x|)$ paths will simulate the computation of M_i on x. It is easy to see that $\langle i, x, l\rangle \in Y$ iff $\#M_i(x) > l$ iff more than half of the paths of M are accepting.

can actually be brought down to $o(n)$ by modifying the oracle Turing machine model, but we choose not to do so for the sake of clarity.) □

COROLLARY 3.2. *The following statements are true:*

1. *ACC(subexp) $\subseteq$ Dtime$(n^9)^{PERM[1]}$ where PERM[1] refers to the case when only one call is made to PERM.*

2. *There is a set Z in $C_=P$ such that ACC(subexp) $\subseteq$ Ntime$(n^2)^Z$.*

Proof.

1. Let M_L and M be the machines from the proof of Theorem 1.1. Note that if M has access to PERM, it can compute $\#M_L(x)$ in time n^9 with just one call to PERM because PERM gives the exact number of accepting paths. The bound n^9 comes from a naïve analysis of Valiant's reduction [27] applied to nondeterministic Turing machines running in linear time.

2. As before, let $M_1, M_2, \ldots$ be an enumeration of nondeterministic Turing machines running in linear time. Let Z be the set $\{\langle i, x, l\rangle : x \in \{0,1\}^* \text{ and } \#M_i(x) = l\}$. It is not hard to see that Z is in $\mathrm{C_=P}$ (much like $Y \in \mathrm{PP}$ in Theorem 1.1). Let M_L be as above. A nondeterministic machine can compute $\#M_L(x)$ in time n^2 using Z as an oracle. It guesses a value l for $\#M_L(x)$ and asks the appropriate query $\langle i, x, l\rangle$ to Z.

□

THEOREM 3.3. *ACC $\subsetneq$ $C_=P$.*

Proof. Corollary 3.2 implies that $\mathrm{ACC} \subseteq \mathrm{Ntime}(n^2)^Z$ for a set $Z \in \mathrm{C_=P}$. Since $\mathrm{ACC} \subseteq \mathrm{C_=P}$, for the sake of contradiction assume that $\mathrm{ACC} = \mathrm{C_=P}$. Since co-NP $\subseteq$ $\mathrm{C_=P}$ and ACC is closed under complement, it follows that $\mathrm{ACC} = \mathrm{P} = \mathrm{NP} = \mathrm{C_=P}$. Therefore, $\mathrm{Ntime}(n^3)^Z \subseteq \mathrm{NP}^Z \subseteq \mathrm{NP}^{\mathrm{ACC}} = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP} = \mathrm{ACC} \subseteq \mathrm{Ntime}(n^2)^Z$, which contradicts the hierarchy theorem of [15] for nondeterministic time classes. □

THEOREM 3.4. *The permanent function (PERM) does not have ACC(subexp) circuits.*

Proof. Corollary 3.2 states that ACC(*subexp*)$\subseteq$ $\mathrm{Dtime}(n^9)^{\mathrm{PERM}[1]}$. By the hierarchy theorem of [17], we know that $\mathrm{Dtime}(n^9)^{\mathrm{PERM}[1]} \subsetneq \mathrm{Dtime}(n^{10})^{\mathrm{PERM}[1]}$. Suppose PERM has ACC(*subexp*) circuits. Let $L \in \mathrm{Dtime}(n^{10})^{\mathrm{PERM}[1]}$ and let M be the oracle machine that accepts L making at most one call to PERM. Let $L' = \{\langle x, z\rangle : M$ accepts x if z is used as the answer to the query made by M to PERM on input x $\}$. Clearly, $L' \in \mathrm{P}$. Similarly, let $L'' = \{\langle x, i\rangle :$ the i^{th} bit of the query by M on input x is $1\}$. Clearly, $L'' \in \mathrm{P}$ as well. A careful reading of Valiant's proof [27] reveals that the membership question for any set in P can be reduced to PERM via uniform AC^0 circuits. (In brief, Valiant's reduction takes an input y to a #P function f, builds a CNF formula ϕ such that $f(y)$ is equal to the number of satisfying assignments to ϕ, and then builds a weighted graph whose permanent is equal to $f(y)$. It has been noted before (e.g., in [19]) that ϕ can be built in uniform AC^0. An inspection of Valiant's graph construction shows that the presence or absence of each edge depends only on the presence of a literal in a given clause, and thus can be computed in uniform AC^0.) Therefore, by the hypothesis, P has ACC(*subexp*) circuits. Now we can describe an ACC(*subexp*) circuit family for L. On any input, the query made to PERM is constructed using the circuits for L'', the circuits for PERM are then used to get the answer to the query and finally we use the circuits for L' to determine whether $x \in L$. Since L', L'' and PERM all have ACC(*subexp*) circuit families, the resulting family for L is also in ACC(*subexp*). Therefore, using the result in Theorem 1.1, $L \in \mathrm{Dtime}(n^9)^{\mathrm{PERM}[1]}$ which contradicts the hierarchy theorem of [17] since we started with an arbitrary L in $\mathrm{Dtime}(n^{10})^{\mathrm{PERM}[1]}$. □

THEOREM 3.5. *PP $\not\subseteq$ ACC(subsubexp).*

Proof. We claim that if PP $\subseteq$ ACC(*subsubexp*), then PrTime(*subsubexp*) $\subseteq$ ACC(*subexp*). To see this, note that if $L \in \mathrm{PrTime}(t(n))$ for some $t \in$ *subsubexp*, then $L' \in \mathrm{PP}$, where $L' = \{x10^{t(|x|)} : x \in L\}$. Since by assumption $L' \in$ ACC(*subsubexp*), one can build subexponential size circuits for L because the composition of two functions in *subsubexp* is in *subexp*. This implies that PrTime(*subsubexp*) $\subseteq$ ACC(*subexp*).

Note that using the result in Theorem 1.1 and the hierarchy theorem of [17], we know that there are sets in P^{PP} that are not in ACC(*subexp*). However, if PP is contained in ACC(*subsubexp*), then

$$\begin{aligned}
\mathrm{P}^{\mathrm{PP}} &\subseteq \mathrm{P}^{\mathrm{ACC}(subsubexp)}\\
&\subseteq \mathrm{P}^{\mathrm{Dtime}(subsubexp)}\\
&\subseteq \mathrm{Dtime}(subsubexp)\\
&\subseteq \mathrm{PrTime}(subsubexp)\\
&\subseteq \mathrm{ACC}(subexp)
\end{aligned}$$

The last step follows from the claim above. Hence, $\mathrm{P}^{\mathrm{PP}} \subseteq$ ACC(*subexp*), which is a contradiction, and the theorem follows. □

Theorem 3.6 below is stronger than Theorem 3.5; we include both results to demonstrate the proof technique.

THEOREM 3.6. *$C_=P \not\subseteq$ ACC(subsubexp).*

Proof. We note, as above, that if $\mathrm{C_=P} \subseteq$ ACC(*subsubexp*), then ACC(*subexp*) contains $\mathrm{C_=Time}$(*subsubexp*). We also have that co-$\mathrm{C_=Time}$(*subsubexp*) $\subseteq$ ACC(*subexp*) since ACC(*subexp*) is closed under complement.

Using the result in Corollary 3.2 and the hierarchy theorem of [15] for nondeterministic time, we know that there are sets in $\mathrm{NP}^{\mathrm{C_=P}}$ that are not in ACC(*subexp*). If $\mathrm{C_=P} \subseteq$ ACC(*subsubexp*), then

$$\begin{aligned}
\mathrm{NP}^{\mathrm{C_=P}} &\subseteq \mathrm{NP}^{\mathrm{ACC}(subsubexp)}\\
&\subseteq \mathrm{NP}^{\mathrm{Dtime}(subsubexp)}\\
&\subseteq \mathrm{Ntime}(subsubexp)\\
&\subseteq \text{co-}\mathrm{C_=Time}(subsubexp)\\
&\subseteq \mathrm{ACC}(subexp)
\end{aligned}$$

which is a contradiction. □

4. Conclusion.

We have shown that uniform ACC circuits of subexponential size cannot compute the permanent function. We have also proved a somewhat weaker bound for some sets in PP. The proofs are based on a simulation of ACC given by Beigel and Tarui in [10]. We have shown how to carry out this simulation in the uniform setting. Some of the obvious open problems are:

1. Is uniformity really necessary? Our lower bound proofs work only in the uniform setting. Can we prove a lower bound for the permanent with respect to nonuniform ACC circuits?

2. How powerful are nonuniform ACC circuits? It is still unknown if Ntime ($2^{n^{O(1)}}$) contains sets that are not accepted by nonuniform ACC circuit families.

3. The lower bound that we have for PP is not as strong as the one for permanent. Can it be improved? Even though the permanent function seems to provide more information about the number of accepting paths of NP machines (the permanent gives us all the bits whereas PP only gives us the most significant bit) we still think that a subexponential lower bound can be proved for PP as well.

The work presented here originally started off as the study of sets that are immune to small complexity classes such as AC^0 and ACC. An infinite set L is immune to a complexity class $\mathcal{C}$ if no infinite subset of L is in $\mathcal{C}$. In [1], we show that P^{PP} contains sets that are immune to ACC, and that nonrelativizing proof techniques suitable for attacking the Dtime vs. Ntime question about exponential time would result from a proof of existence as well as a proof of nonexistence of sets in NP that are immune to AC^0.

It should be emphasized that our results about the complexity of PERM do not rely on any unproven complexity-theoretic assumptions. This is in contrast to other results such as [14], which proves stronger intractability results about PERM under the hypothesis that the polynomial hierarchy is infinite.

We conclude with a few remarks about some related work that has been done recently. In [16], Green, Köbler, Regan, Schwentick and Torán have studied the class of languages that can be recognized in polynomial time with the information about just one bit from the value of a #P function. They define the class MidBitP and show that the classes $\mathrm{MOD}_k\mathrm{P}$, for every k, and the class PH are all low for MidBitP. They have also improved the existing upper bounds for ACC by introducing the idea of MidBit gates. A MidBit gate over w inputs $x_1, x_2, \ldots, x_w$ is a gate that outputs the value of the middle bit in the binary representation of the number $\sum_{i=1}^{w} x_i$. They show that every language in ACC can be accepted by a family of depth two deterministic circuits of size $2^{(\log n)^c}$ with a MidBit gate at the root and AND gates of fan-in $(\log n)^c$ at the second level. It would be interesting to see if our techniques can be used in this setting to obtain stronger lower bounds.

Barrington has written a very nice article [5] about the power of circuits of constant depth and $2^{(\log n)^{O(1)}}$ (quasipolynomial) size. The article surveys many results that deal with these kinds of circuits and provides an overview of the new complexity classes that have been introduced. The paper also shows that the notion of uniformity introduced for constant depth circuit families of polynomial size in [6] can be extended to quasipolynomial size as well. It should be noted that this extended notion of uniformity coincides with the one that we have used. Independently of our work, Barrington's paper outlines a proof that shows that the simulation of Beigel and Tarui [10] is uniform according to this new notion of uniformity; thus [5] may be consulted for an alternative approach to proving Theorem 3.1. (The proof in [5] leaves many details to the reader.) In addition, it also shows that the simulation of Green, Köbler, Regan, Schwentick and Torán [16] is uniform under this notion as well.

Acknowledgments. We thank the anonymous referees for suggestions that improved the presentation of these results. We also thank Richard Beigel for suggesting that our results concerning PP could be extended to $C_=P$.

REFERENCES

[1] E. Allender and V. Gore, *On strong separations from AC^0*, in Advances in Computation Theory, Jin-Yi Cai, ed., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 13, American Mathematical Society, Providence, RI, pp. 21–37, 1993.

[2] E. Allender and U. Hertrampf, *Depth reduction for circuits of unbounded fan-in*, Inform. Comput., to appear.

[3] J. Aspnes, R. Beigel, M. Furst, and S. Rudich, *The expressive power of voting polynomials*, in Proc. 23rd Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1991, pp. 402–409.

[4] D. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*, J. Comput. System Sci., 38 (1989), pp. 150–164.

[5] ———, *Quasipolynomial size circuit classes*, in Proc. 7th Annual IEEE Structure in Complexity Theory Conference, IEEE Computer Society Press, Washington, DC, 1992, pp. 86–93.

[6] D. Barrington, N. Immerman, and H. Straubing, *On uniformity within NC^1*, J. Comput. System Sci., 41 (1990), pp. 274–306.

[7] D. Barrington and D. Thérien, *Finite monoids and the fine structure of NC^1*, J. Assoc. Comput. Mach., 35 (1988), pp. 941–952.

[8] P. Beame, S. Cook, and H. Hoover, *Log depth circuits for division and related problems*, SIAM J. Comput., 15 (1986) pp. 994–1003.

[9] A. Ben-Dor and S. Halevi, *Zero-one permanent is #P-complete, a simpler proof*, in Proc. 2nd Israel Symposium on Theory of Computing and Systems, IEEE Computer Society Press, Washington, DC, 1993.

[10] R. Beigel and J. Tarui, *On ACC*, in Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Washington, DC, 1991, pp. 783–792.

[11] A. Borodin, *On relating time and space to size and depth*, SIAM J. Comput., 6 (1977), pp. 733–743.

[12] S. Buss, S. Cook, A. Gupta, and V. Ramachandran, *An optimal parallel algorithm for formula evaluation*, SIAM J. Comput., 21 (1992), pp. 755–780.

[13] A. Chandra, D. Kozen, and L. Stockmeyer, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114–133.

[14] U. Feige and C. Lund, *On the hardness of computing the permanent of random matrices*, in Proc. 24th Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1992, pp. 643–654.

[15] M. Fischer, A. Meyer, and J. Seiferas, *Separating nondeterministic time complexity classes*, J. Assoc. Comput. Mach., 25 (1978), pp. 146–167.

[16] F. Green, J. Köbler, K. Regan, T. Schwentick and J. Torán, *The power of the middle bit of a #P function*, submitted. Preliminary versions appeared in Proc. 7th Annual IEEE Structure in Complexity Theory Conference, IEEE Computer Society Press, Washington, DC, 1992, pp. 111–117, and in Proc. 4th Italian Conference on Theoretical Computer Science, World Scientific Press, Singapore, 1992, pp. 317–329.

[17] J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Trans. AMS, 117 (1965), pp. 285–306.

[18] J. Håstad, Computational Limitations for Small Depth Circuits, MIT Press, Cambridge, MA, 1987.

[19] N. Immerman, *Languages that capture complexity classes*, SIAM J. Comput., 16 (1987), pp. 760–778.

[20] R. Kannan, H. Venkateswaran, V. Vinay, and A. Yao, *A circuit-based proof of Toda's theorem*, Inform. Comput., 104 (1993), pp. 271–276.

[21] P. McKenzie and D. Thérien, *Automata theory meets circuit complexity*, in Proc. 16th Annual International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 372, Springer-Verlag, Berlin, New York, 1989, pp. 589–602.

[22] A. Razborov, *Lower bounds for the size of circuits of bounded depth with basis* $\{\wedge, \oplus\}$, Math. notes of the Academy of Sciences of the USSR, 41 (1987), pp. 333–338.

[23] W. Ruzzo, *On uniform circuit complexity*, J. Comput. System Sci., 21 (1981), pp. 365–383.

[24] M. Sipser, *Borel sets and circuit complexity*, in Proc. 15th Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1983, pp. 61–69.

[25] R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, in Proc. 19th Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1987, pp. 77–82.

[26] S. Toda, *PP is as hard as the polynomial-time hierarchy*, SIAM J. Comput., 20 (1991), pp. 865–877.

[27] L. Valiant, *The complexity of computing the Permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.

[28] L. Valiant and V. Vazirani, *NP is as easy as detecting unique solutions*, Theoret. Comput. Sci., 47 (1986), pp. 85–93.

[29] A. Yao, *Separating the polynomial-time hierarchy by oracles*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Washington, DC, 1985, pp. 1–10.

[30] ——— *On ACC and threshold circuits*, in Proc. 31st Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Washington, DC, 1990, pp. 619–627.

[31] V. Zankó, *#P-completeness via many-one reductions*, International Journal of Foundations of Computer Science, 2 (1991), pp. 77–82.

A. Appendix.

The appendix is devoted to the proof of Theorem 3.1, which can be regarded as a uniform version of the main theorem of [10]. The definitions, lemmas and theorems presented in this section all lead up to the proof. Since the proof of Theorem 3.1 is fairly involved, we first start with a very high level outline.

Outline: Since our goal in this section is to prove that the construction of [10] can be done uniformly, it is necessary to prove some preliminary results about uniform constant depth circuits. To that end, we define the notions of "clean" and "nice" circuits, which are circuits that have certain properties that we find essential in presenting our uniformity results. The proof of Theorem 3.1 consists of a number of transformations of a circuit. Without loss of generality, we start out with a "nice" circuit family. After each transformation, we will have a circuit that may not obviously satisfy the "niceness" condition, but at least satisfies the weaker notion of being "clean". We show that this clean circuit can then be transformed into a nice circuit of the same depth, and the process repeats.
The main steps in the transformation are:

1. All the AND and OR gates in the circuits are replaced by constant depth probabilistic subcircuits. This step removes all the OR gates from the circuits and the only remaining AND gates have small fan-in. The circuits are probabilistic but the number of probabilistic bits used in each case is small and is in fact a simple function of the size of C_n.

2. All the MOD gates in the circuit with composite moduli are replaced with equivalent subcircuits so that the resultant circuits consist only of MOD gates with prime moduli.

3. The circuits are now made deterministic by taking separate copies of those for each setting of the probabilistic bits and connecting all outputs to a MAJORITY gate.

4. A general technique is used, showing how nice circuits with small fan-in AND gates can be replaced by equivalent circuits with the same depth, whose outputs are MOD gates.

5. An induction is begun, where each step reduces the depth of the circuit. At the beginning of the inductive step, the circuit consists of a symmetric gate on the output level, where the inputs to the symmetric gate are "nice" ACC circuits with MOD_p gates feeding into the symmetric gate. Then, using techniques developed by Toda [26], Yao [30] and Beigel and Tarui [10], we create an equivalent circuit with a new symmetric gate that "absorbs" the level of MOD_p gates; thus the new circuit has smaller depth.

A.1. Nice Circuits.

In this section, we present a series of "niceness" conditions, and prove that it is no loss of generality to deal only with "nice" circuits.

Definition A.1. *A circuit family $\{C_n\}$ is* well-named *if for every n, the name of the output gate of C_n can be computed from n (in binary) in polynomial time (i.e., in $(\log n)^{O(1)}$ time).*

Definition A.2. *A circuit family $\{C_n\}$ is said to have the* strong connection property *if for all n, for every connection $g \rightarrow h$ in C_n, where i is the number such that g is the i^{th} input to h (assuming lexicographic ordering), it is the case that h can be computed in polynomial time from $\langle n, g, i\rangle$, and additionally, given $\langle n, h, g\rangle$, the number i can be computed in polynomial time. Under the weaker assumption that this condition holds whenever h is an AND gate then $\{C_n\}$ is said to have the* strong connection property for ANDs.

DEFINITION A.3. *A circuit family* $\{C_n\}$ *is said to have* small fan-in AND gates *if for every* n*, the fan-in of each AND gate in* C_n *is polylogarithmic in the size of* C_n*.*

DEFINITION A.4. *Let* C *be a circuit and let* P *be a path in* C *from the output gate to an input gate (say* t*). Let* $G_1, G_2, \ldots, G_k$ *be the sequence of the types of gates occurring on* P *so that* G_1 *is the type of the output gate of* C *and* G_k *is the type of the gate that* t *is connected to. Then the sequence* $(G_1, G_2, \ldots, G_k)$ *is defined as the* signature *of the path* P*.*

DEFINITION A.5. *The* compression *of a signature* s *is the sequence* s' *that results from applying the following operation as many times as possible to* s*: replace "AND, AND" by "AND" and replace "OR, OR" by "OR". That is, the compression of* s *contains no two adjacent ANDs or ORs.*

DEFINITION A.6. *A circuit family* $\{C_n\}$ *is* clean *if*

1. *It is well-named.*
2. *It has the strong connection property for ANDs.*
3. *Every path from an output gate to an input gate in* C_n *(for every* n*) has the same signature. (Note that only constant depth circuit families can be clean, since the signature does not depend on* n*.)*

DEFINITION A.7. *A circuit family* $\{C_n\}$ *of size* $s(n)$ *is* nice *if it has the following properties:*

1. *It is clean.*
2. *For every* n*, the fan-in of every gate* g *in* C_n *can be computed from* g *in time* $(\log s(n))^{O(1)}$*.*
3. *For every* n*, the depth of a gate* g *in* C_n *can be computed from* g *in time* $(\log s(n))^{O(1)}$*.*
4. *Each circuit* C_n *is in tree form (excluding the inputs and negated inputs, which may fan out to many gates at level 1).*
5. *It has the strong connection property.*
6. *For all input lengths* n*, all the MOD gates in* C_n *have the same fan-in.*

Our main lemma in this section is Lemma A.8, which states that any uniform ACC(*subexp*) circuit family can be transformed into an equivalent nice family.

LEMMA A.8. *Suppose* $\{C_n\}$ *is an ACC(subexp) circuit family. Then there exists an equivalent nice family* $\{D_n\}$ *of subexponential size. Furthermore,*

1. *If* $\{C_n\}$ *is clean, then the signature of* $\{D_n\}$ *is the compression of the signature of* $\{C_n\}$*.*
2. *If* $\{C_n\}$ *is clean and has small fan-in AND gates, then* $\{D_n\}$ *has small fan-in AND gates.*

The proof of the above lemma follows from a sequence of lemmas that are presented below. The proofs of these lemmas make use of a version of the Alternating Turing Machine (ATM) model of computation. For background on alternation, see [13]. It should be noted that the model that we use here is somewhat different from the one defined in [13]. Some of the lemmas that follow are similar in flavor to the results in [23], in which correspondences between ATMs and uniform circuits were first established; the reader may wish to consult [23].

The following "road map" is intended to explain how the following lemmas combine to prove Lemma A.8:

(i) Lemma A.15: circuit $\mapsto$ ATM.
(ii) Lemma A.16: ATM $\mapsto$ clean ATM.
(iii) Lemma A.17: clean ATM $\mapsto$ nice ATM.

(iv) Lemma A.18: nice ATM $\mapsto$ nice circuits.

The transformations in Lemmas A.15, A.17, and A.18 preserve various properties, such as the property of having small fan-in AND gates.

The existential and universal states in our ATMs behave as usual. Each configuration of an ATM has either zero, one, or two successor configurations (i.e., the fan-out of any node in the computation tree is at most two). We follow the convention that the ATM is always provided the length of the input (in binary) on the work tape as part of its initial configuration on a particular input. This convention has been introduced to simplify the proof.[3] We consider ATMs that access their input only at the leaves. (That is, the only configurations that depend on the input are halting configurations. These are of two types: those that accept if and only if bit i of the input is 1, and those that accept if and only if the complement of bit i is 1 (for some i that is recorded on the address tape). The results in [24] show that this convention can be introduced without loss of generality.)

The MOD states and other aspects of our ATM model are described in the following definitions.

DEFINITION A.9. *For a modulus m, a MOD$_m$ configuration (say σ) is the root of a subtree of associated configurations. This tree is called the* subtree associated with *σ and is represented as T_σ. We say that σ is* accepting *if and only if the number of leaves of T_σ that are accepting is congruent to 0 modulo m. We also use the term* *MOD-tree* *at times to refer to a subtree associated with a MOD configuration.*

DEFINITION A.10. *There is said to be an* alternation *between two configurations σ_1 and σ_2 of an ATM if and only if σ_2 follows from σ_1 via one step of the ATM and one of the following conditions hold:*

1. *σ_1 is the leaf of a MOD-tree, and σ_2 is of type $\exists$, $\forall$ or MOD.*
2. *σ_1 is of type $\exists$ and σ_2 is of type $\forall$ or MOD.*
3. *σ_1 is of type $\forall$ and σ_2 is of type $\exists$ or MOD.*

Let T denote the computation tree of an ATM M on a particular input. The root of the tree is said to have alternation depth *1, and a node in the tree labeled by configuration σ_2 with parent labeled by configuration σ_1 is defined to have alternation depth one greater than the alternation depth of σ_1 if there is an alternation between σ_1 and σ_2, and the alternation depth of σ_2 is equal to that of σ_1 otherwise. The alternation depth of a tree is the maximum alternation depth of all nodes in the tree. The alternation depth of an ATM is the maximum alternation depth of all its alternation trees.*

It is necessary for us to define a notion of "clean" ATMs corresponding to our notion of "clean" circuit families. This is accomplished using the following definitions:

DEFINITION A.11. *Let σ and τ be two different configurations of an ATM. If τ is reached from σ via a path that contains an alternation only in the step at which τ is reached, then τ is called a* primary descendent *of σ.*

DEFINITION A.12. *For a computation path of an ATM on an input, let $C_1, C_2, \ldots, C_k$ be the sequence of configurations such that C_1 is the initial configuration, and C_{i+1} is a primary descendent of C_i. The* signature *of the path is the sequence $t_1, t_2, \ldots t_k$ such that if configuration C_i is existential (universal, MOD$_m$), then t_i = OR (AND, MOD$_m$).*

DEFINITION A.13. *An ATM is* clean *if every path in every alternation tree of the ATM on every input has the same signature. (Note that only ATMs making $O(1)$ alternations can be clean.)*

[3] It is worthwhile to note that the input length can be computed deterministically in logarithmic time (see [12]) but this requires multiple accesses to the input along a given computation path.

DEFINITION A.14. *An ATM running in time $t(n)$ has* well-behaved universal configurations *if each universal configuration has $t(n)^{O(1)}$ primary descendents, and given a universal configuration σ and a number i, the i^{th} primary descendent of σ can be computed in time $t(n)^{O(1)}$.*

LEMMA A.15. *Let $L \subseteq \{0,1\}^*$, let s be a function in subexp, and let L be accepted by a uniform family $\{C_n\}$ of depth d circuits ($d = O(1)$) of type $ACC(s(n))$. Then L is accepted by an ATM M that has existential ($\exists$), universal ($\forall$) and MOD states (for the same set of moduli), that runs in time $(\log s(n))^{O(1)}$ and has alternation depth $a = O(1)$. Moreover,*

1. *If $\{C_n\}$ is clean, then the signature of M is the compression of the signature of $\{C_n\}$.*
2. *If $\{C_n\}$ is clean and has small fan-in AND gates, then M has well-behaved universal configurations.*

Proof. Suppose L is accepted by a uniform circuit family $\{C_n\}$. Let U be the uniformity machine for $\{C_n\}$. M behaves as follows:

```
On input $x$, (with $n = |x|$ on the work tape)
    ($\exists$) guess the name of the output gate (say $g$) of $C_n$ of length $(\log s(n))^{O(1)}$.
    Use $U$ to verify that $g$ is indeed a gate in $C_n$. (I.e., check that $U$
    accepts $\langle n, g, g\rangle$.)
    ($\forall$) gates $h$ of length $(\log s(n))^{O(1)}$ check that $U$ rejects $\langle n, h, g\rangle$
    (so that $g$ is indeed the output gate).
    Call Eval($g$).

Eval($g$)
    If $g$ is an OR gate then
        ($\exists$) guess $h$ (an input to $g$) of length $(\log s(n))^{O(1)}$.
        If $U$ rejects $\langle n, g, h\rangle$ then reject
        else call Eval($h$).
    If $g$ is an AND gate then
        ($\forall$) guess $h$ (an input to $g$) of length $(\log s(n))^{O(1)}$.
        If $U$ rejects $\langle n, g, h\rangle$ then accept
        else call Eval($h$).
    If $g$ is a MOD$_m$ gate then
        Switch to a MOD$_m$ configuration.
        ($\exists$) guess $h$ (an input to $g$) of length $(\log s(n))^{O(1)}$.
            (This is the subtree associated with the MOD$_m$ configuration.)
        If $U$ rejects $\langle n, g, h\rangle$ then reject
        else call Eval($h$).
    If $g$ is a constant gate then
        Accept iff $g$ is the constant 1 gate.
    If $g$ is an input gate then
        Accept iff the corresponding input is 1.
end (Eval).
```

It is fairly obvious that M accepts x iff $C_{|x|}$ evaluates to 1 on input x. Note that M consults its input only at the leaves. It is clear that M makes a constant number of alternations and runs in time $(\log s(n))^{O(1)}$. Indeed, the most time-consuming part of the simulation involves running the uniformity machine U. The constructibility conditions on s are also essential here.

If $\{C_n\}$ is clean, then it is well-named, and thus the name of the output gate g can be computed deterministically. Also, since all circuits in $\{C_n\}$ have the same signature, each output gate is of the same type. If the type of the output gate is MOD$_m$, for instance, we can avoid the extra two levels of alternation caused by the processing outside the routine Eval, by starting out in a MOD$_m$ configuration, deterministically computing g, existentially guessing h, rejecting if U rejects $\langle n, g, h\rangle$, and otherwise proceeding to Eval(h). The case when the output gate is an AND or OR gate is handled similarly. Thus if $\{C_n\}$ is clean, the signature of M can easily be seen to be the compression of the signature of $\{C_n\}$.

If $\{C_n\}$ has the strong connection property for ANDs and all AND gates have fan-in $(\log s(n))^c$, then instead of universally guessing an input h to an AND gate g, universally guess a number $i \leq (\log s(n))^c$ and deterministically compute the name of the gate h. If M is simulating r consecutive levels of AND gates of C_n, it is not hard to see that each universal configuration of M will have at most $(\log s(n))^{rc}$ primary descendents, and M thus has well-behaved universal configurations.

The other claims of the Lemma are easily seen to hold. □

Lemma A.15 does not guarantee the existence of a clean ATM accepting a language when the given circuit family is not already clean. This is remedied by the following lemma.

LEMMA A.16. *If L is accepted by an ATM M that makes a constant number of alternations between $MOD_{m_1}, MOD_{m_2}, \ldots, MOD_{m_k}, \exists$ and $\forall$ states and runs in time $t(n)$ then L is accepted by a clean ATM N running in $O(t(n))$ time with a constant number of alternations between $MOD_{m_1}, MOD_{m_2}, \ldots, MOD_{m_k}, \exists$ and $\forall$ states.*

Proof. Suppose M makes at most λ alternations on any input. Then N has the sequence MOD$_{m_1}$, MOD$_{m_2}$, ..., MOD$_{m_k}$, $\exists, \forall$ (repeated λ times) hardwired into its finite control. N simply simulates M but follows the signature in its finite control. If N is trying to simulate a move that does not involve an alternation or that involves moving into a state that has the same type as the next type in its signature, it simply proceeds with the simulation and behaves exactly as M does. In the case of a type mismatch, N behaves as follows:

1. If the next state in the sequence is universal (existential), then it executes a one-ary universal (existential) branch and continues the simulation. (Note that amounts to adding a "dummy" node in the alternating tree.)
2. If the next state in the sequence is a MOD$_m$ state for some m, then it executes a m-way MOD$_m$ branch. It trivially accepts along $m-1$ of these branches (following the signature) and continues the simulation on the remaining one.

It is fairly obvious that N is clean and for every x, N accepts x iff M accepts x. □

Our main reason for introducing the ATM model is the following lemma, which enables us to construct "nice" circuits.

LEMMA A.17. *Let $2^{t(n)}$ be constructible, and suppose L is accepted by a clean ATM M running in time $t(n)$. Then L is accepted by a clean ATM N with the same signature (and hence with the same alternation depth) that runs in time $t(n)^{O(1)}$ and also has the following properties:*

1. *Given a configuration σ on an input of length n, the number of primary descendents of σ is computable from σ in time $t(n)^{O(1)}$.*
2. *Given a configuration σ on an input of length n, the alternation depth of σ is computable from σ in time $t(n)^{O(1)}$.*
3. *Given a configuration σ and number $i \leq$ the number of primary descendents*

of σ, the i^{th} primary descendent of σ (under the usual lexicographic ordering) can be computed in time time $t(n)^{O(1)}$ from the encoding of σ.

4. *All the MOD configurations in the computation tree have the same number of primary descendents.*

5. *If M has well-behaved universal configurations, then N also has this property.*

Proof. The proof is very similar to the proof of Lemma A.15. We will need to settle on some convention of encoding paths in an alternation tree, with the property that for every path of length $i \leq t(n)$ in an alternation tree, there is exactly one string of length $2 \cdot t(n)$ that denotes that path. This can easily be accomplished by encoding sequences in $\{\text{left, right, stop}\}^*$ in the obvious way; note that there will be many strings that do not correspond to any path in the tree. Similarly, pick some encoding of configurations of M so that any configuration σ of M on inputs of length n has a unique encoding using $c \cdot t(n)$ bits (for some constant c). Again, many strings of length $c \cdot t(n)$ will not correspond to any configuration of M.

N will begin its computation on x by first computing (deterministically) $t(n)$. (Note that this can be done regardless of whether the initial configuration of N is existential, universal, or MOD_m.) If M has well-behaved universal configurations, then let $I(n) = b \log t(n)$ for some constant b; otherwise let $I(n) = t(n)$. (Note that the decision of which value to use for $I(n)$ can be encoded in the finite control of N.) Then N will set σ to be equal to the initial configuration of M, and run the routine Eval(σ).

```
Eval(σ)
    If $\sigma$ is an existential or $\text{MOD}_m$ non-halting configuration then
        existentially guess strings $w$ of length $2\cdot t(n)$ and $\tau$ of length $c\cdot t(n)$.
        If $w$ encodes a path from $\sigma$ to configuration $\tau$, where the last step
        in the path involves an alternation (so $\tau$ is a primary descendent of $\sigma$)
            then enter a configuration of the same type as $\tau$ and call Eval($\tau$)
            else call Trivial(reject)
    If $\sigma$ is a universal non-halting configuration then there are two cases:
        (1) We are simulating a machine $M$ with well-behaved universal
        configurations.
            Universally guess $i \leq b \log t(n)$. Let $\tau$ be the $i^{\text{th}}$ primary descendent
            of $\sigma$. Call Eval($\tau$).
            (If there is no such $\tau$, then call Trivial(accept).)
        (2) Otherwise.
            Universally guess strings $w$ of length $2\cdot t(n)$ and $\tau$ of length $c\cdot t(n)$.
            If $w$ encodes a path from $\sigma$ to configuration $\tau$, where the last step in
            the path involves an alternation (so $\tau$ is a primary descendent of $\sigma$)
                then enter a configuration of the same type as $\tau$ and call Eval($\tau$)
                else call Trivial(accept)
    If $\sigma$ is a halting configuration, then
        Accept iff $\sigma$ is accepting. (Note that this may involve accessing the input,
        if $\sigma$ depends on input bit $i$ for some $i$.)
end (Eval).
```

The routine Trivial(d) (for $d \in \{accept, \ reject\}$) used in the routine Eval is a simple routine that depends on the number of alternations executed thus far by N in its simulation of M. If the next step in the signature calls for computation of type $\exists$ ($\forall$), then N executes a $2^{(c+2)t(n)}$-way existential (universal) branch, all of which in turn call Trivial(d). If the next step in the signature calls for computation of type MOD_m, and $d = \mathit{accept}$ (respectively, $d = \mathit{reject}$), then N enters a MOD_m state, executes a $2^{(c+2)t(n)}$-way existential branch all of which call Trivial(*reject*) (respectively, the first of which calls Trivial(*accept*) and the rest of which call Trivial(*reject*)).

Machine N uses its worktape to record the path in the alternation tree leading to the current configuration. Thus no configuration of N will label two distinct nodes in the alternation tree.

Let us now verify the various properties claimed in the statement of the lemma.

Given σ a configuration of N, one can trace through the path in the alternation tree leading to σ (since this information is recorded in σ). This allows one to compute the alternation depth of σ, as well as to find the configuration τ reached after the last alternation on this path, and compute the number j of moves with fan-out 2 that have occurred along this path between τ and σ. If σ is an $\exists$ or MOD configuration, the number of primary descendents of σ is $2^{(c+2)t(n)-j}$. If σ is a $\forall$ configuration, then this number is $2^{(c+2)I(n)-j}$. In the particular case that σ is a MOD configuration, note that $j = 0$; thus all the MOD configurations have the same number of primary descendents. Furthermore, if σ' is the i^{th} primary descendent of σ, then the number i is encoded in $(c+2)t(n) - j$ consecutive positions in the bit string encoding the path leading to σ', thus enabling us to compute σ' given $\langle n, \sigma, i\rangle$. The other claims of the lemma are easy to verify. $\Box$

LEMMA A.18. *Let L be accepted by an ATM M satisfying the conditions of Lemma A.17, running in time $t(n)$. Then there is a nice $ACC(2^{O(t(n))})$ circuit family $\{C_n\}$ accepting L, such that the signature of $\{C_n\}$ is the same as the signature of M. Furthermore, if M has well-behaved universal configurations, then $\{C_n\}$ has small fan-in AND gates.*

Proof. The proof of this lemma is by a standard simulation of the sort introduced by [23]. The output gate of C_n will be labeled by the initial configuration of N on an input of length n (i.e., with n recorded on the worktape, as per the conventions of our ATM model). The inputs to any gate labeled with configuration σ will be all of the primary descendents of σ. Universal configurations are represented by AND gates, existential configurations by OR gates, and MOD_m configurations by MOD_m gates. Halting configurations are either constant 1 or 0 gates (if they do not depend on the input) or are input gates connected to (negated) input i (if they access input bit i).

It is easily verified that $\{C_n\}$ satisfies the requirements of the lemma. $\Box$

Proof. (of Lemma A.8) This follows immediately from Lemmas A.15, A.16, A.17 and A.18. $\Box$

A.2. Transformations on Circuits.

In this section we prove a general lemma, enabling us to replace gates by equivalent subcircuits. (This, of course, is completely obvious in the nonuniform setting. However, in the uniform setting, where we need the additional property that the fan-in of a circuit be easy to compute, we need all of the "niceness" conditions guaranteed by the preceding section.) Then we apply this lemma to remove OR gates, large fan-in AND gates, and composite MOD gates from ACC circuits.

DEFINITION A.19. *Suppose G is a particular type of gate. Let $\{G_r\}$ denote a family of gates such that the gate G_r is of type G and takes r inputs. Let $\{E_r\}$ be a family of subcircuits so that for every r, E_r takes r inputs and has a single output. We will assume an ordering on the inputs of G_r and E_r and let $x_1, x_2, \ldots, x_r$ denote the inputs to G_r and $y_1, y_2, \ldots, y_r$ denote the inputs to E_r. We say that E_r* replaces

G_r in a circuit C if we remove G_r from C and put E_r in its place in such a way that the output gate of E_r is connected to exactly the gates that G_r is connected to in C, and the inputs to G_r now become inputs to E_r so that for all i, $1 \le i \le r$, $x_i = y_i$. In general, when we talk about replacing a gate type G in a circuit, we will mean that all occurrences of G in the circuit are replaced simultaneously.

LEMMA A.20. *Suppose $\{C_n\}$ and $\{E_r\}$ are nice circuit families. Let G denote a particular type of gate used in the circuits of $\{C_n\}$. For every n, let $\{D_n\}$ denote the circuit family obtained by replacing* all *occurrences of G (of the form G_r for various r) by a subcircuit E_r. Then the circuit family $\{D_n\}$ is clean.*

Proof. It is clear that $\{D_n\}$ is well-named and that every path from output to input has the same signature. Thus we need only show that $\{D_n\}$ is uniform and has the strong connection property.

Consider the transformation from C_n to D_n for a particular value of n. Let g (with fan-in r) be an instance of G in C_n and let E_r be the subcircuit that replaces g. Suppose E_r consists of the gates $h_0, h_1, \ldots, h_s$ where h_0 is the output gate of E_r. The names of these gates in the new circuit D_n will be $g\#h_i$, for $0 \le i \le s$. Let L_0 be the direct connection language for $\{C_n\}$, L_1 for $\{E_r\}$ and L for $\{D_n\}$. Similarly, let f_0, f_1, and f be the functions that, given $\langle n, g, h\rangle$, compute the number i such that h is the i^{th} input to g in C_n, E_n and D_n, respectively, and let f_0', f_1' and f' be the related functions that compute h given $\langle n, g, i\rangle$. To accept L, and to compute f, one has to consider the following cases:

1. Strings of the form $\langle n, g, h\rangle$ where neither g nor h are of type G. In this case $\langle n, g, h\rangle \in L \iff \langle n, g, h\rangle \in L_0$. Also, $f(n, g, h) = f_0(n, g, h)$.
2. Strings of the form $\langle n, g\#h, g\#h\rangle$. This is done as follows:
   a. Check that $\langle n, g, g\rangle \in L_0$ and that g has type G.
   b. Compute the fan-in r of g from the description of g.
   c. Check that $\langle r, h, h\rangle \in L_1$.
3. Strings of the form $\langle n, g\#h, g\#h'\rangle$ with $h \ne h'$. This is done as follows:
   a. Check that $\langle n, g, g\rangle \in L_0$ and that g has type G.
   b. Compute the fan-in r of g from the description of g.
   c. Check that $\langle r, h, h'\rangle \in L_1$.
   d. Note that $f(n, g\#h, g\#h') = f_1(n, g\#h, g\#h')$.
4. Strings of the form $\langle n, g', g\#h_0\rangle$ where G is not the type of g'. This is done as follows:
   a. Check that $\langle n, g', g'\rangle$ and $\langle n, g, g\rangle \in L_0$, and that g has type G.
   b. Compute the fan-in r of g from the description of g.
   c. Check that $\langle r, h_0, h_0\rangle \in L_1$ (h_0 is the output gate of E_r).
   d. Check that $\langle n, g', g\rangle \in L_0$.
   e. Note that $f(n, g', g\#h_0) = f_0(n, g', g)$.
5. Strings of the form $\langle n, g\#h, g'\rangle$, where G is not the type of g'. This is done as follows:
   a. Check that $\langle n, g, g\rangle$ and $\langle n, g', g'\rangle$ are in L_0, where g has type G.
   b. Check that $\langle n, g, g'\rangle \in L_0$.
   c. Compute the fan-in r of g from its description.
   d. Check that $\langle r, h, h\rangle \in L_1$.
   e. Compute the number j such that g' is the j^{th} input to g (using the strong connection property).
   f. Let $x_1, x_2, \ldots, x_r$ denote the inputs to E_r. Check that $\langle r, h, x_j\rangle \in L_1$.
   g. Note that $f(n, g\#h, g') = j$.

6. Strings of the form $\langle n, g'\#h, g\#h_0\rangle$ where both g and g' are of type G.
   a. Check that $\langle n, g, g\rangle$ and $\langle n, g', g'\rangle$ are in L_0.
   b. Compute the fan-in r of g and check that h_0 is the output gate of E_r.
   c. Compute the fan-in r' of g' and check that $\langle r', h, h\rangle \in L_1$.
   d. As in the previous case, check that g is the j^{th} input to g' and that h is connected to input j of $E_{r'}$.

It is not hard to see that all the above cases can be checked within the required time bounds and hence the new circuit family $\{D_n\}$ is uniform as well.

A similar analysis shows that f' can also be computed in time polynomial in the length of its input, and thus $\{D_n\}$ has the strong connection property. □

LEMMA A.21. *Suppose L is accepted by an ACC(subexp) family $\{C_n\}$. Then L is accepted by a nice probabilistic ACC(subexp) circuit family $\{D_n\}$*[4] *such that*

1. *$\{D_n\}$ has no MOD$_m$ gates for composite modulus m.*
2. *$\{D_n\}$ has small fan-in AND gates.*
3. *For every n, the number of probabilistic inputs in D_n is polylogarithmic in the size of D_n.*

Proof. By Lemma A.8, we may assume that $\{C_n\}$ is nice.

Let n be fixed. The transformation $C_n \to D_n$ is carried out by performing the following sequence of steps:

1. By a construction in the proof of Lemma 13 in [2], one can replace the AND and OR gates in the circuit by nice depth 6 probabilistic circuits with MOD$_2$ gates and small fan-in AND gates. (This construction is based on an idea of Valiant and Vazirani in [28]; similar constructions may be found in work by Toda [26] and Kannan, Venkateswaran, Vinay and Yao [20].) The size of the new circuit is only polynomially more than that of the old one. If the AND or OR gate being replaced has r inputs, then the probabilistic circuit that replaces it uses $O((\log r)^3)$ random bits. The probabilistic circuits have the property that the probability of error for the whole circuit is less than $\frac{1}{4}$ after all the AND and OR gates have been replaced by these probabilistic circuits, even when the same $O((\log s(n))^3)$ probabilistic bits are fed into the probabilistic inputs of each of these subcircuits. (Even though Allender and Hertrampf discuss space uniformity, it is clear from their proof that the probabilistic circuits are uniform even in our sense of uniformity.) We can now apply Lemma A.20 to prove that the new circuit family (now probabilistic) is clean, and thus by Lemma A.8 there is an equivalent nice circuit family $\{C_n^1\}$. Note that $\{C_n^1\}$ has small fan-in AND gates and has no OR gates.
2. Suppose the circuit C_n^1 contains a MOD$_m$ gate (call it G) where m is composite. Let
$$m = \prod_{i=1}^{t} a_i^{e_i}$$
where $a_i < a_{i+1}$ for all i such that $1 \le i \le t-1$ and for all i, $1 \le i \le t$, a_i is prime and $e_i > 0$. We use the elementary fact that $x \equiv 0 \pmod m$ $\iff x \equiv 0 \pmod{a_i^{e_i}}$ for all i, $1 \le i \le t$ to change G into an AND of MOD$_{a_i^{e_i}}$'s. Suppose G has r inputs. For each m, the subcircuit family $\{E_r\}$

[4] Note that the circuits in $\{D_n\}$ are probabilistic and hence also have probabilistic inputs, but when we say D_n we mean the circuit that has n nonprobabilistic inputs. We follow this convention because the proof shows how to convert C_n into D_n for every n.

that replaces the MOD_m gates is easily seen to be nice. The subcircuit E_r has depth two, with an AND gate at the top level and $\mathrm{MOD}_{a_i^{e_i}}$ gates at the bottom level for all i, $1 \leq i \leq t$. The top level AND gate has fan-in t and is connected to each of the MOD gates at the second level. All the MOD gates at the second level have fan-in r and are all connected to each of the inputs of the gate G. We can now use the result of Lemma A.20 to conclude that the new circuit family is clean. Moreover, the family contains MOD gates with only prime power moduli. The subcircuit E_r, other than its input gates, contains only a constant number of gates that depends on m. Since the original circuit family $\{C_n\}$ only has MOD gates for a fixed set of moduli, the size of the circuit after this step goes up by at most a constant factor. We again use Lemma A.8 to get a nice family of probabilistic circuits $\{C_n^2\}$ that has no composite MOD gates, no OR gates, and small fan-in AND gates.

3. This step eliminates all the MOD gates that have moduli of the form p^e where p is prime and $e > 1$ from C_n^2 and replaces them with subcircuits consisting of AND and MOD_p gates. Suppose C_n^2 contains a MOD_{p^e} gate G for some prime p and $e > 1$. This step uses the following result (for references, see e.g., [10]):

   x is congruent to $0 \pmod{p^e}$ if and only if each of $x, \binom{x}{p}, \binom{x}{p^2}, \ldots, \binom{x}{p^{e-1}}$ are congruent to $0 \pmod{p}$. If $x = \sum_{i=1}^{r} x_i$, then for $1 \leq j \leq e-1$:

$$\binom{x}{p^j} = \binom{x_1+x_2+\cdots+x_r}{p^j} = \sum_{S \subseteq \{1,2,\ldots,r\}, |S|=p^j} \bigwedge_{k \in S} x_k .$$

   The subcircuit that replaces G is a three level subcircuit that is described as follows:
   (a) The top level consists of an AND gate that has fan-in e.
   (b) The middle level consists of e MOD_p gates and each of those is connected to the top level AND gate. For all j, $0 \leq j \leq e-1$, the j^{th} MOD_p gate outputs 1 if and only if $\binom{x}{p^j} \equiv 0 \pmod{p}$. If G has fan-in r, then the j^{th} MOD_p gate at this level has fan-in $\binom{r}{p^j}$, one corresponding to each subset of the inputs of size p^j.
   (c) The bottom level consists of $\sum_{j=1}^{e-1} \binom{r}{p^j}$ AND gates divided into $e-1$ groups. For all j, $1 \leq j \leq e-1$, the j^{th} group consists of $\binom{r}{p^j}$ AND gates, one corresponding to each subset of the inputs of size p^j. The inputs to a particular gate in the j^{th} group are the p^j inputs in the subset to which it corresponds and it fans out to the j^{th} MOD_p gate at the middle level. Note that all the AND gates introduced here have constant fan-in.

   It is not hard to see that the subcircuit family described above is nice for every prime power p^e. (The only point that is not completely obvious is checking that the strong connection property holds, but this is straightforward to verify.) Using Lemma A.20 we can now replace every MOD gate with a prime power modulus with a subcircuit that consists only of MOD gates with prime moduli and we now get a clean circuit family that only has AND gates and MOD gates with prime moduli. The size of the subcircuit that replaces a MOD_{p^e} gate is $O(\sum_{j=1}^{e-1} \binom{r}{p^j})$ which is a polynomial in the size of the circuit C_n^2, and thus the new circuit family also has subexponential size. The proof is completed by appeal to Lemma A.8.

□.

A.3. Circuits with Symmetric Gates. In order to prove Theorem 3.1 we need to show how to convert an ACC(*subexp*) circuit family into a uniform deterministic depth two circuit family that has a symmetric gate at the root and AND gates of small fan-in at the bottom level. So far we have only dealt with ACC type circuits. To proceed, we need to deal with circuits that have arbitrary symmetric gates (but only at the root). However, since most of the results proved so far only deal with uniform ACC type circuits, we need to expand the notion of uniformity a little so that the results can also be used with circuits that have arbitrary symmetric gates at the root. The new notion of uniformity is explained in the following definition:

DEFINITION A.22. *Let $f : \mathbf{N} \to \mathbf{N}$ be a function. Then $\{C_{n,t} : n \in \mathbf{N},\ 1 \le t \le f(n)\}$ is a* uniform family of ACC sequences *if there is a constant d and a finite set S such that for all n and for all t, $C_{n,t}$ is a circuit of depth d taking inputs from the set $\{x_1, x_2, \ldots, x_n\}$ and having AND, OR and MOD$_m$ gates (for $m \in S$) and the direct connection language defined as*

$$\{\langle n, t, g_1, g_2\rangle : g_1 = g_2 \text{ and } g_1 \text{ is a gate in } C_{n,t} \text{ or } g_1 \ne g_2 \text{ and } g_2 \text{ is an input to } g_1 \text{ in } C_{n,t}\}$$

can be recognized in polynomial time. A uniform family of ACC sequences $\{C_{n,t} : n \in \mathbf{N},\ 1 \le t \le f(n)\}$ together with a function SYM : $\mathbf{N}\times\mathbf{N} \to \{0,1\}$, defines a uniform SYMACC circuit family $\{D_n\}$ such that for every n,

1. *D_n is a circuit with a symmetric gate at the output level that computes SYM(n, i) where i is the number of its inputs that evaluate to 1.*
2. *The symmetric gate has fan-in $f(n)$ and the output gates of $C_{n,t}$, $1 \le t \le f(n)$, are connected to it.*
3. *Given n and i, $f(n)$ and SYM(n, i) can be computed in time polylogarithmic in the size of D_n.*

Note that the results proved so far also hold with this new notion of uniformity. In particular, letting $f(n) = 1$ for all n and letting SYM be the identity function reduces this to the old notion of uniformity. Also, we will use the fact that Lemma A.8 also holds in this new setting. That is, given a uniform *clean* family of ACC sequences, there is an equivalent *nice* family of ACC sequences with the same signature and of approximately the same size. (In proving the analog of Lemma A.8 in this new setting, the index t of circuit $C_{n,t}$ would be provided to the ATM as an additional parameter on the worktape, along with n.)

LEMMA A.23. *Let L be accepted by an ACC(subexp) circuit family $\{C_n\}$. Then there is a constructible subexponential function s and there is a constant c such that L is accepted by a deterministic circuit family $\{D_n\}$ where for every n, D_n has a MAJORITY gate at the root, connected to the output gates of $C_{n,t}$, $1 \le t \le 2^{(\log s(n))^c}$ where $\{C_{n,t} : n \in \mathbf{N},\ 1 \le t \le 2^{(\log s(n))^c}\}$ is a uniform family of ACC sequences with small fan-in AND gates, no OR gates, and no MOD$_m$ gates for composite m.*

Proof. By Lemma A.21, if L is accepted by an ACC(*subexp*) circuit family, then L is accepted by a nice ACC(*subexp*) family of *probabilistic* circuits with small fan-in AND gates, no OR gates, and no MOD$_m$ gates for composite m, using at most $(\log s(n))^c$ probabilistic bits (for some constant c), where $s(n)$ bounds the size of C_n.

Now construct the sequence of circuits $\{C_{n,t}\}$ where t is a bit string of length $(\log s(n))^c$. The gates in $\{C_{n,t}\}$ will have names of the form $\langle t, g\rangle$ where g is a gate in C_n, and the connections among all gates are the same, except that if gate g in C_n is connected to probabilistic bit number j, then gate $\langle t, g\rangle$ will be connected to the

j^{th} bit of t. (i.e., the new circuit sequence consists of identical copies of C_n, with particular choices of probabilistic bits hardwired in.)

Let D_n consist of a MAJORITY gate with inputs from the various $C_{n,t}$. It is clear that the new circuit accepts the same language as $\{C_n\}$. The size of D_n is $O(s(n)2^{(\log s(n))^c})$ which is subexponential. It is immediate that the other required properties also hold. $\square$

The following lemma shows how one can in effect "push" an AND gate below a level of MOD gates (much as multiplication distributes over addition).

LEMMA A.24. *Let $\{C_{n,t}\}$ be a nice family of ACC sequences of subexponential size, having small fan-in AND gates, no OR gates, and no MOD$_m$ gates where m is composite, where the output gate of each circuit is an AND gate, and the inputs to that AND gate are MOD$_p$ gates. Then there is an equivalent nice sequence $\{D_{n,t}\}$ with the same depth, also of subexponential size with small fan-in AND gates, no OR gates, and no MOD$_m$ gates where m is composite, where the output gate of each circuit is a MOD$_p$ gate, and the inputs to that MOD$_p$ gate are AND gates.*

Proof. Our proof again follows the outline given in [10] (see also [2, 20]), where we must be careful to see that the transformation can be done uniformly.

Suppose G is an AND gate (the output gate of some $C_{n,t}$ that has r MOD$_p$ gates $G_1, G_2, \ldots, G_r$ as inputs). Note that r is polylogarithmic in $s(n)$, where $s(n)$ bounds the size of $C_{n,t}$. Since the sequence $C_{n,t}$ is nice, all the MOD$_p$ gates $G_1, G_2, \ldots, G_r$ have the same fan-in. Let this fan-in be denoted by n_0 and let $\{x_{ij}\}$, $1 \le j \le n_0$ denote the set of inputs to G_i. Finally, let $\boldsymbol{x}_i = \sum_{1 \le j \le n_0} x_{ij}$. Consider the AND of $G_1, G_2, \ldots, G_r$. By Fermat's Little Theorem, for $1 \le i \le r$,

$$1 - \boldsymbol{x}_i{}^{p-1} \equiv \begin{cases} 0 \pmod p & \text{if } \boldsymbol{x}_i \not\equiv 0 \pmod p \\ 1 \pmod p & \text{otherwise} \end{cases}$$

Therefore,

$$\bigwedge_{i=1}^{r} [\boldsymbol{x}_i \equiv 0 \pmod p] \Longleftrightarrow 1 - \prod_{i=1}^{r} (1 - \boldsymbol{x}_i{}^{p-1}) \equiv 0 \pmod p$$

Note that $1 - \prod_{i=1}^{r} (1 - \boldsymbol{x}_i{}^{p-1})$ is a polynomial of degree $r(p-1)$ in the variables x_{ij}, $1 \le i \le r$, $1 \le j \le n_0$. Let $[r]$ denote the set $\{1, 2, \ldots, r\}$.

$$\begin{aligned}
1 - \prod_{i=1}^{r} (1 - \boldsymbol{x}_i{}^{p-1}) &= 1 - (\prod_{i=1}^{r} (1 - (\sum_{j=1}^{n_0} x_{ij})^{p-1})) \\
&= \sum_{k=1}^{r} \sum_{I \subseteq [r], |I|=k} (-1)^{k-1} \prod_{i \in I} (\sum_{j=1}^{n_0} x_{ij})^{p-1} \\
&= \sum_{k=1}^{r} (-1)^{k-1} \sum_{I \subseteq [r], |I|=k} \prod_{i \in I} (\sum_{j=1}^{n_0} x_{ij})^{p-1} \\
&= \sum_{k=1}^{r} (-1)^{k-1} \sum_{I \subseteq [r], |I|=k} \prod_{i \in I} \sum_{J_i = \langle j_{i,1}, j_{i,2}, \ldots, j_{i,p-1} \rangle \in [n_0]^{p-1}} \prod_{l=1}^{p-1} x_{ij_{i,l}} \\
&= \sum_{k=1}^{r} (-1)^{k-1} \sum_{I \subseteq [r], I = \{i_1, i_2, \ldots, i_k\}} \sum_{J_{i_1}, J_{i_2}, \ldots, J_{i_k}} \prod_{s=1}^{k} \prod_{l=1}^{p-1} x_{i_s j_{i_s,l}} \quad (*)
\end{aligned}$$

This expression can be realized by a MOD_p gate (call it g) with AND gates of fan-in at most $r(p-1)$ as inputs. Since r is $(\log s(n))^{O(1)}$, the fan-in of these AND gates is also polylogarithmic in $s(n)$. The only thing we need to take care of are the negative coefficients in the above expression. That is done by multiplying[5] the negative coefficients by $(1-p)$. The expression is changed slightly and the term $(-1)^{k-1}$ is replaced by c_k where $c_k = 1$ if k is even and $c_k = p-1$ if k is odd. Now we interpret scalar multiplication as repeated addition and the multiplication of variables is realized by AND gates. From the expression $(*)$, it is not hard to see that the number of AND gates that are input to the new MOD_p gate g is $\sum_{k=1}^{r} c_k \binom{r}{k} (n_0^{p-1})^k$. Note that since r is polylogarithmic in $s(n)$, this expression (i.e., the fan-in of the new MOD gate) can be computed in time $(\log s(n))^{O(1)}$ from $\langle G, G_1, \ldots, G_r \rangle$.

To show that this step can be done uniformly, we must show how the new gates created in this step should be named so that the direct connection language of the new circuit family can be recognized within the required time bound. The name of the new MOD_p gate is $g = \langle G\#\langle G_1, G_2, \ldots, G_r\rangle, \mathrm{MOD}_p\rangle$. Looking at the expression $(*)$, it is clear that a typical AND gate has $k(p-1)$ inputs where $1 \le k \le r$. The $k(p-1)$ inputs can be divided up into k groups of size $(p-1)$ each. Every group represents a distinct gate in the set $\{G_1, G_2, \ldots, G_r\}$. The $(p-1)$ inputs in a particular group (representing say G_i) are simply some of the input gates to G_i (with repetitions allowed) in $C_{n,t}$. Depending on the value of c_k, such an AND gate either appears once or $(p-1)$ times. The name of such an AND gate is $\langle G\#\langle\langle H_1\#L_1\rangle, \langle H_2\#L_2\rangle, \ldots, \langle H_k\#L_k\rangle\rangle\#m, \mathrm{AND}\rangle$ where $H_1, H_2, \ldots, H_k$ are distinct gates from the set $\{G_1, G_2, \ldots, G_r\}$ and each L_i, $1 \le i \le k$ is a list of $(p-1)$ of the gates that are input to H_i in the original circuit. Note that L_i is allowed to have repetitions. The number m is either 0 (indicating only one copy of the gate; this will be the case if k is even) or between 1 and $(p-1)$ and is used for indexing the $(p-1)$ different copies.

We now show how to recognize the direct connection language for the new circuit family that we get after applying this transformation. Let L_0 be the direct connection language before the step and L the one after the step. Note that the strings in L conform to the naming scheme discussed above. The following cases must be considered:

1. Let g be a new[6] gate. To check if $\langle n, t, g, g\rangle \in L$, we have the following two subcases:
   (a) g is a new MOD_p gate of the form $\langle G\#\langle G_1, G_2, \ldots, G_r\rangle, \mathrm{MOD}_p\rangle$. We do the following:
      i. check that G is the output gate of $C_{n,t}$. (This can be done because the circuits are well-named.)
      ii. check that $\langle n, t, G, G_i\rangle \in L_0$ for all i, $1 \le i \le r$, where r is the fan-in of G. (Recall that r can be computed from G, by one of the niceness properties.)
   (b) g is a new AND gate of the form

$$\langle G\#\langle\langle H_1\#L_1\rangle, \langle H_2\#L_2\rangle, \ldots, \langle H_k\#L_k\rangle\rangle\#m, \mathrm{AND}\rangle.$$

   We do the following:
      i. check that G is the output gate of $C_{n,t}$.
      ii. verify that $H_1, H_2, \ldots, H_k$ are all distinct.
      iii. check that for all i, $1 \le i \le k$, $\langle n, t, G, H_i\rangle \in L_0$.

[5] Note that this does not change the value of the expression mod p.
[6] The word "new" will hereafter be used to refer to gates that were created in the current step.

iv. check that m has the right value based on the parity of k.
v. For all i, $1 \leq i \leq k$, verify that L_i is indeed a list of $(p-1)$ gates that are all input to H_i.

2. Let g_1 be an old gate and g_2 a new gate. Then $\langle n, t, g_1, g_2\rangle \notin L$.
3. Let g_1 be a new gate and g_2 an old gate. The only way for $\langle n, t, g_1, g_2\rangle \in L$ to hold is that g_1 is a new AND gate created in this step.
Hence g_1 has the form $\langle G\#\langle\langle H_1\#L_1\rangle, \langle H_2\#L_2\rangle, \ldots, \langle H_k\#L_k\rangle\rangle\#m, \text{AND}\rangle$.
We do the following to check if $\langle n, t, g_1, g_2\rangle \in L$:
(a) check that $\langle n, t, g_1, g_1\rangle \in L$.
(b) check that $\langle n, t, g_2, g_2\rangle \in L_0$.
(c) verify that $\exists\, i$, $1 \leq i \leq k$, such that g_2 belongs to the list of gates L_i.
4. Let g_1 and g_2 both be new gates. The only way for g_2 to be an input to g_1 is if
$$g_1 = \langle G\#\langle G_1, G_2, \ldots, G_r\rangle, \text{MOD}_p\rangle \text{ and}$$
$$g_2 = \langle G\#\langle\langle H_1\#L_1\rangle, \langle H_2\#L_2\rangle, \ldots, \langle H_k\#L_k\rangle\rangle\#m, \text{AND}\rangle$$
where $\{H_1, \ldots, H_k\} \subseteq \{G_1, \ldots, G_r\}$. This is obviously easy to check.

The only remaining property that needs to be checked is the strong connection property for ANDs. However this is immediate using the naming system that we use, since the name of each new AND gate explicitly lists the names of each of its inputs.

Let us now consider the size of the new circuit after a single level of AND gates has been pushed below a level of MOD gates. The increase in size comes mainly because of all the new AND gates that get created. For a circuit of size s, the number of new AND gates created to change an AND of r MOD_p gates is $\leq \sum_{k=1}^{r} c_k \binom{r}{k} s^{(p-1)k} \leq (p-1)2^r s^{(p-1)r}$. Therefore, the overall size of the new circuit is at most $O(2^r s^{(p-1)r+1})$. Since s is subexponential and r is polylogarithmic in s, the size of the new circuits is still subexponential.

Note that this step does not preserve the tree structure of the circuit so we use Lemma A.8 to produce an equivalent nice circuit sequence.

□

Lemma A.25. *Let L be accepted by a uniform nice SYMACC circuit family $\{C_n\}$ of subexponential size, with small fan-in AND gates, no OR gates, and no MOD_m gates for composite m, such that each path from the output gate to an input passes through $k \geq 1$ MOD gates. Then there is an equivalent SYMACC circuit family $\{D_n\}$ satisfying the same conditions, such that each path from the output gate to an input gate passes through $k-1$ MOD gates.*

Proof. Our proof follows the outline in [10], using techniques developed in [30, 26].

Let L and $\{C_n\}$ be as in the statement of the lemma, where the output gate of C_n computes the function $A(n, \eta)$, where η is the number of elements of $\{C_{n,i} : 1 \leq i \leq f(n)\}$ that evaluate to 1. By Lemma A.24, we may assume without loss of generality that the output of each circuit $C_{n,i}$ is a MOD_p gate. Since $\{C_n\}$ is nice, for each n there is some n_0 so that each MOD_p gate in $C_{n,t}$ has fan-in n_0 (where n_0 can be computed in $(\log s(n))^{O(1)}$ time from n). For each $i \leq f(n)$, let the inputs to the i^{th} of these MOD_p gates be denoted by $x_{i,j}$, $1 \leq j \leq n_0$. Then the value of $\{C_n\}$ can be expressed as $A(n, \sum_{1 \leq i \leq f(n)} \text{MOD}_p(x_{i,1}, x_{i,2}, \ldots, x_{i,n_0}))$.

Let $k(n) = 1 + \lfloor \log_p f(n) \rfloor$ so that $p^{k(n)} > f(n)$. Note that $k(n)$ is computable in time $(\log s(n))^{O(1)}$. For the rest of this discussion, fix n, and let k denote $k(n)$.

It is shown in [10] that the polynomial P_k defined by

$$P_k(y) = (-1)^{k+1}(y-1)^k(\sum_{j=0}^{k-1} \tbinom{k+j-1}{j} y^j) + 1$$

satisfies the property that for every $m \geq 1$ and $y \geq 0$,

$$y \equiv 0 \pmod{m} \Longrightarrow P_k(y) \equiv 0 \pmod{m^k}$$

and

$$y \equiv 1 \pmod{m} \Longrightarrow P_k(y) \equiv 1 \pmod{m^k}$$

Let $Q_k(y) = 1 - P_k(y^{p-1})$. Then

$$Q_k(y) \equiv \begin{cases} 1 \pmod{p^k} & \text{if } y \equiv 0 \pmod{p} \\ 0 \pmod{p^k} & \text{otherwise.} \end{cases}$$

If $y = \sum_{i=1}^{r} y_i$ then

$$Q_k(\sum_{i=1}^{r} y_i) \equiv \mathrm{MOD}_p(y_1, y_2, \ldots, y_r) \pmod{p^k}$$

Thus, recalling that the value of the circuit C_n is

$$A(n, \sum_{1 \leq i \leq f(n)} \mathrm{MOD}_p(x_{i,1}, x_{i,2}, \ldots, x_{i,n_0})),$$

we see that this can also be expressed as

$$A(n, \sum_{1 \leq i \leq f(n)} (Q_k(\sum_{1 \leq j \leq n_0} x_{i,j}) \pmod{p^k})).$$

Since $f(n) < p^k$ and Q_k is always 0 or 1 $\pmod{p^k}$, we can bring the outer sum inside the modulus to obtain the equivalent expression

$$A(n, (\sum_{1 \leq i \leq f(n)} Q_k(\sum_{1 \leq j \leq n_0} x_{i,j})) \pmod{p^k}).$$

Let $B(n, i)$ be defined to be $A(n, (i \bmod p^k))$. Thus the value of $\{C_n\}$ is equal to

$$B(n, \sum_{1 \leq i \leq f(n)} Q_k(\sum_{1 \leq j \leq n_0} x_{i,j})).$$

Note that B is computable in time polylogarithmic in $s(n)$.

Note that $(\sum_{1 \leq i \leq f(n)} Q_k(\sum_{1 \leq j \leq n_0} x_{i,j}))$ is a low degree polynomial in the variables $\{x_{i,j}\}$. As in the proof of Lemma A.24, our strategy will be to implement scalar multiplication with AND gates, and multiply the negative coefficients by $(1 - p^k)$ to make them positive[7], to obtain a realization of this polynomial in terms of circuits.

[7] This does not change the value of the expression mod p^k.

Our first task is to compute the coefficients in the polynomial

$$(\sum_{1\le i\le f(n)} Q_k(\sum_{1\le j\le n_0} x_{i,j})).$$

Since this is just a sum of $f(n)$ similar polynomials, we can consider each of them separately.

Recall that $Q_k(y) = 1 - P_k(y^{p-1})$. Let $z = y^{p-1}$. After a little simplification we get $1 - P_k(z) = (1-z)^k(\sum_{j=0}^{k-1}\binom{k+j-1}{j}z^j)$. This is a polynomial of degree $2k-1$. For $i \ge 0$, let $b_i = 1$ if i is even, and $b_i = p^k - 1$ if i is odd. The coefficients of z^m, say c_m, are given by

$$c_m = \begin{cases} 1 & \text{if } m = 0. \\ 0 & \text{if } 1 \le m \le k-1. \\ \displaystyle\sum_{0\le i\le k, 0\le j\le k-1, i+j=m} b_i\binom{k}{i}\binom{k+j-1}{j} & \text{if } k \le m \le 2k-1. \end{cases}$$

These coefficients c_m can be computed in $(\log s(n))^{O(1)}$ time, because we only need to compute $O(k)$ binomial coefficients each involving numbers that are $O(k \log k)$ bits long. It can be verified that $O(k^4 \log k)$ time suffices which is polylogarithmic in the size of the circuit since k is logarithmic in the size.

Now observe that the value of circuit $\{C_n\}$ is given by

$$\begin{aligned} B(n, \sum_{i=1}^{f(n)} Q_k(\sum_{j=1}^{n_0} x_{i,j})) &= B(n, \sum_{i=1}^{f(n)} 1 - P_k((\sum_{j=1}^{n_0} x_{i,j})^{p-1})) \\ &= B(n, \sum_{i=1}^{f(n)} \sum_{m=0}^{2k-1} c_m(\sum_{j=1}^{n_0} x_{i,j})^{(p-1)m}) \\ &= B(n, \sum_{i=1}^{f(n)} \sum_{m=0}^{2k-1} \sum_{c=1}^{c_m} \sum_{\langle j_1, j_2, \ldots, j_{p-1}\rangle \in [n_0]^{(p-1)m}} \bigwedge_{l=1}^{(p-1)m} x_{i,j_l}) \end{aligned}$$

In place of each circuit $C_{n,t}$ in the original sequence of circuits, there will be several new circuits, each of the form $D_{n,\langle i,m,c,j_1,\ldots,j_{(p-1)m}\rangle}$ where $0 \le m \le 2k-1, 1 \le c \le c_m$, and each j_l is in $[n_0]$. (Of course, by our conventions, there will also be circuits $D_{n,t}$ where t is not of this form; each such circuit $D_{n,t}$ will be a trivial rejecting circuit that will therefore have no effect on the output of the symmetric gate.)

The output gate of each circuit $D_{n,\langle i,m,c,j_1,\ldots,j_{(p-1)m}\rangle}$ will be an AND gate with the name

$$g = \langle n, i, m, c, j_1, \ldots, j_{(p-1)m}, \text{AND}\rangle.$$

The inputs to g will be the $(p-1)m$ gates that are the j_l^{th} inputs to the MOD$_p$ gate G_i in the original circuit $C_{n,i}$. Note that since $C_{n,i}$ has the strong connection property, one can show that $D_{n,\langle i,m,c,j_1,\ldots,j_{(p-1)m}\rangle}$ does, too.

Note that the number of bits needed to express $\langle i, m, c, j_1, \ldots, j_{(p-1)m}\rangle$ is bounded by $(\log s(n))^b$ for some constant b, and thus if we define $f'(n)$ to be equal to $2^{(\log s(n))^b}$, it follows that the symmetric gate computing B in circuit D_n has fan-in $f'(n)$, where $f'(n)$ is computable in time polylogarithmic in $s(n)$.

Since the new circuit consists of a subexponential number of circuits, each of which is of subexponential size, the new circuit is also of subexponential size.

The depth of the new circuit family is the same as $\{C_n\}$ but the top layer of MOD_p gates has been "absorbed" into the symmetric gate computing B and been replaced by a layer of AND gates of small fan-in. Now, by an appeal to Lemma A.8, the circuit can be converted into nice form, which completes the proof. □

Proof. (of Theorem 3.1)

By Lemma A.23, every language in ACC(*subexp*) is accepted by a deterministic SYMACC circuit family of subexponential size, with small fan-in AND gates, no OR gates, and no MOD_m gates for composite m. Successive applications of Lemma A.24 and Lemma A.25 remove all MOD gates from the circuit, while maintaining the property that all AND gates have small fan-in. This suffices to prove the theorem. □