A uniform circuit lower bound for the permanent

Eric Allender*
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Vivek Gore†
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

July 23, 1992

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 where the uniformity condition is essential; it is still unknown if there is any set in Ntime $(2^{n^{O(1)}})$ that does not have nonuniform ACC circuits.

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 [Hås87, Yao85, ABFR91]. 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 [Raz87] proved that these circuits could not compute the MAJORITY function. Smolensky [Smo87] 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 various moduli m. This extension is known as the class ACC. It should be noted that this class was mentioned implicitly by Barrington in [Bar86] (see also [Bar89]). 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 [Bar89] has conjectured that $ACC \subsetneq NC^1$.

*Supported in part by NSF grant CCR-9000045.
†Supported in part by a Rutgers University Graduate Excellence Fellowship.

Yao [Yao90] proved the first nontrivial upper bounds on the power of ACC circuits; these bounds were slightly improved by Beigel and Tarui [BT91] who showed that each set in ACC is accepted by a family of deterministic depth three threshold circuits of size $2^{(\log n)^{O(1)}}$. 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., [Bor77]). Some 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 [Ruz81] 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 $(\log n)^k$ 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 [BCH84], but it remains an open question whether one can improve this result using a more restrictive uniformity condition. Similarly, [BIS90] 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. 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 functions that are bounded above by some constructible function f such that $\forall \epsilon > 0$ $f(n) = o(2^{n^\epsilon})$. Let *subsubexp* denote the class of all functions that are bounded above by some constructible function f such that for all k, $f^{(k)}(n) < 2^n$ (where $f^{(k)}$ denotes f composed with itself k times). A typical example of a function in *subexp* is $2^{n^{1/\log^* n}}$, and typical examples of functions in *subsubexp* are $n^{\log n}$ and $2^{(\log n)^{\log \log n}}$. It is not hard to prove that if s is in (*sub*)*subexp* , then so is $s^{(\log s)^k}$, for any constant k.

In this paper, we provide lower bounds for the class of languages accepted by uniform circuit families of ACC circuits of (sub)subexponential size. We show that PERM (the permanent of a matrix) is not in ACC(*subexp*) and that PP $\not\subseteq$ ACC(*subsubexp*). Our main tool in proving these results is the following theorem:

Theorem 1 There is a set Y in PP such that $\mathrm{ACC}(\mathit{subexp}) \subseteq \mathrm{Dtime}(n^2)^Y$.

Theorem 1 trivially gives us an important corollary (which also follows from a more general lower bound proved later in the paper):

Corollary 2 $\mathrm{ACC} \subsetneq \mathrm{PP}$.

Proof. Theorem 1 implies that $\mathrm{ACC} \subseteq \mathrm{Dtime}(n^2)^Y$ for some $Y \in \mathrm{PP}$. Since $\mathrm{ACC} \subseteq \mathrm{PP}$, suppose for the sake of contradiction that ACC = PP. Then ACC = P = PP. Therefore, $\mathrm{Dtime}(n^3)^Y \subseteq \mathrm{P}^Y \subseteq \mathrm{P} = \mathrm{ACC} \subseteq \mathrm{Dtime}(n^2)^Y$. But this contradicts the time hierarchy theorem of [HS65]. ∎

This seems to be one of the very few instances[1] 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. For example, the combinatorial and algebraic techniques developed in [Hås87, Raz87, Smo87] 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; 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, we will first use the results of Toda [Tod91], Yao [Yao90] and Beigel and Tarui [BT91] to convert a circuit family in ACC($(sub)subexp$) 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 [Val79] results about the class #P.

Section 2 presents some basic definitions. The following section states Theorem 3, which is a uniform version of the main result of [BT91]. Theorem 3 is then used to prove the main results of the paper. Section 4 is devoted to the proof of Theorem 3, and is the longest and most technically-involved part of the paper. Even though the basic machinery of this section was developed in [Yao90, BT91], there are many obstacles to overcome to ensure that one maintains uniformity. The final section of the paper presents conclusions and open problems.

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 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,

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

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

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

Definition 3 ([MT89, Bar89, BT88]) 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.

[1] 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).

ACC was first defined and studied in [MT89, Bar89, BT88] 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 [BT88]. Razborov [Raz87] and Smolensky [Smo87] 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 to be the least common multiple of the elements in S makes the two definitions equivalent. Yao [Yao90] 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 [BT91] improved this to show the existence of deterministic circuit families of this sort.

Definition 4 For an NP machine M, let $\#\mathrm{acc}_M$ be the function $\#\mathrm{acc}_M : \Sigma^* \to \mathbf{N}$ defined by $\#\mathrm{acc}_M(x)$ = number of accepting paths of M on input x. Then $\#\mathrm{P} = \{\#\mathrm{acc}_M : M \text{ is an NP machine}\}$.

It is well known from [Val79] that PERM is complete for #P under polynomial time many-one reductions ($\leq_m^p$).

Definition 5 Let $\{C_n\}$ be a family of circuits. Following [Ruz81], 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 6 A circuit family $\{C_n\}$ is *uniform* if its direct connection language can be recognized in polynomial time by a Turing machine. Note that the time is polynomial with respect to the length of the strings in the language[2] ($|\langle n, g_1, g_2\rangle|$) and not merely polynomial in n. The Turing machine that recognizes the direct connection language of $\{C_n\}$ will be referred to as the *uniformity machine* for $\{C_n\}$.

Definition 7 Let $\mathrm{ACC}(s(n))$ denote the class of languages accepted by uniform 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

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

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

[2]Note that this is slightly different from the notion of Dlogtime-uniformity that is generally used for small complexity classes (see [BIS90] and [Ruz81]). Our notion of uniformity can be informally referred to as Polylogtime-uniformity. The reason we use this notion is that we are dealing with circuits of (sub)subexponential size and the proofs are much simpler with this uniformity condition. It should be noted that a set has uniform $\mathrm{ACC}((sub)subexp)$ circuits with respect to our notion of uniformity if and only if it has Dlogtime-uniform $\mathrm{ACC}((sub)subexp)$ circuits. This can be established by "padding" a circuit with many dummy gates.

3 The main results

For the proof of Theorem 1, we will first show the following.

Theorem 3 Suppose L is accepted by an ACC($(sub)subexp$) circuit family $\{C_n\}$ of $s(n)$ sized circuits. Then L is accepted by a uniform, depth two circuit family[3] whose circuits have the following properties:

- Level one consists of a (sub)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 $(\log s(n))^{O(1)}$.
- 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 and most of the paper hereafter is devoted to its proof. The proof uses techniques developed by Beigel and Tarui [BT91], Yao [Yao90] and Toda [Tod91]. The reader who is willing to accept the fact that the construction of [BT91] can be carried out uniformly can simply skip Section 4 (where Theorem 3 is proved). The rest of this section assumes that Theorem 3 is true, and uses it to prove our main results.

Proof. (of Theorem 1) Let $\{C_n\}$ be a circuit family in ACC($subexp$) that accepts L. Using the result in Theorem 3, 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.

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 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 $o(n)$ fan-in, M_L can do this computation in linear time. Note that $\#\mathrm{acc}_{M_L}(x)$ 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 $\{\langle i, x, l\rangle : x \in \{0,1\}^* \text{ and } \#\mathrm{acc}_{M_i}(x) \leq l\}$. Note that Y is in PP. With oracle Y, a deterministic machine can compute $\#\mathrm{acc}_{M_L}(x)$ in time n^2. Then, since this is the number of AND gates of D_n that evaluate to 1 on input x, one can then in linear time determine if D_n accepts x, using the properties guaranteed by Theorem 3. Thus membership of x in L can be determined in time n^2 relative to oracle Y. (Note that the running time 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.) ∎

Theorem 4 The permanent function (PERM) does not have ACC($subexp$) circuits.

Proof. Let M_L be the machine from the proof of Theorem 1. Note that $\#\mathrm{acc}_{M_L}(x)$ can be computed in time n^2 with one call to PERM. Thus the proof of Theorem 1 shows that ACC($subexp$)$\subseteq$ $\mathrm{Dtime}(n^2)^{\mathrm{PERM}[1]}$ where PERM[1] refers to the case where the machines can only make at most one call to the oracle PERM. We know that $\mathrm{Dtime}(n^2)^{\mathrm{PERM}[1]} \subsetneq \mathrm{Dtime}(n^3)^{\mathrm{PERM}[1]}$ from the hierarchy

[3] This circuit family is not an ACC($(sub)subexp$) 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 22.

theorem of [HS65]. Suppose PERM has ACC($subexp$) circuits. Let $L \in \text{Dtime}(n^3)^{\text{PERM}[1]}$ 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 the answer from PERM to the query made by M on input x is z $\}$. Clearly, $L' \in$ P. Similarly, let $L'' = \{\langle x, i\rangle$: the i^{th} bit of the query by M on input x is 1$\}$. Clearly, $L'' \in$ P as well. A careful reading of Valiant's proof [Val79] reveals that the membership question for any set in PP can be reduced to PERM via uniform AC^0 circuits. Therefore, by the hypothesis, PP has ACC($subexp$) circuits and hence, 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, $L \in \text{Dtime}(n^2)^{\text{PERM}[1]}$ which contradicts the hierarchy theorem of [HS65] since we started with an arbitrary L in $\text{Dtime}(n^3)^{\text{PERM}[1]}$. ∎

Theorem 5 PP $\not\subseteq$ ACC($subsubexp$).

Proof. First we claim that if PP $\subseteq$ ACC($subsubexp$), then PPtime($subsubexp$) $\subseteq$ ACC($subsubexp$), where PPtime($t(n)$) is the class of all languages L for which there exists a nondeterministic Turing machine M running in time $t(n)$ such that $x \in L$ iff at least half of the computation paths of M on input x are accepting. To see this, note that if $L \in$ PPtime($t(n)$) for some $t \in subsubexp$, then $L' \in$ PP, where $L' = \{x10^{t(|x|)} : x \in L\}$. Since by assumption $L' \in$ ACC($subsubexp$), one can also build subsubexponential size circuits for L.

Note also that, by Theorem 4, the language $\{\langle M, j, b\rangle$: the j^{th} bit of the permanent of M is $b\}$ is not in ACC($subsubexp$). However, if PP is contained in ACC($subsubexp$), then this language is in $\text{P}^{\text{PP}} \subseteq \text{P}^{\text{ACC}(subsubexp)} \subseteq \text{P}^{\text{Dtime}(subsubexp)} \subseteq$ Dtime($subsubexp$) $\subseteq$ ACC($subsubexp$). The last step, i.e., Dtime($subsubexp$) $\subseteq$ ACC($subsubexp$) follows from the claim above. Hence, $\text{P}^{\text{PP}} \subseteq$ ACC($subsubexp$), which is a contradiction, and the theorem follows. ∎

4 Proof of Theorem 3

This section is devoted to the proof of Theorem 3. The definitions, lemmas and theorems presented in this section all lead up to the proof. Since the proof of Theorem 3 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 [BT91] 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 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. Our results concerning the "nice" circuit families are proved using a variation of the alternating Turing machine model (ATM).
The main steps in the transformation are:

- 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.
- 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.
- 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.
- 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.
- 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 [Tod91] and Yao [Yao90], 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.

4.1 Nice Circuits

Definition 8 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 9 A circuit family $\{C_n\}$ is said to have the *strong connection* property if for all n, for every connection $g \to h$ 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 10 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 11 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 12 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 13 A circuit family $\{C_n\}$ is *clean* if

- It is well-named.
- It has the strong connection property for ANDs.
- Every path from an output gate to an input gate in any circuit C_n has the same signature. (Note that only constant depth circuit families can be clean.)

Definition 14 A circuit family $\{C_n\}$ of size $s(n)$ is *nice* if it has the following properties:

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

Our main lemma in this subsection is Lemma 6, which states that any uniform ACC circuit family can be transformed into an equivalent nice family.

Lemma 6 Suppose $\{C_n\}$ is an ACC($(sub)subexp$) circuit family. Then there exists an equivalent nice family $\{D_n\}$ of (sub)subexponential size. Furthermore,

- If $\{C_n\}$ is clean, then the signature of $\{D_n\}$ is the compression of the signature of $\{C_n\}$.
- 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. The model that we use here is somewhat different from the one defined in [CKS81].

The existential and universal states in our ATMs behave as usual (for background on alternation, see [CKS81]). 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. (It is worthwhile to note that the input length can be computed deterministically in logarithmic time (see [BCGR92]) but this requires multiple accesses to the input along a given computation path.) 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 [Sip83] 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 15 For a modulus m, a MOD_m configuration (say σ) is always the root of a subtree of configurations in which all the other configurations (except the root) are existential. This tree is called the *existential 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 an existential tree associated with a MOD configuration.

Definition 16 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:

- σ_1 is of type $\exists$ and is not in the MOD-tree of any MOD configuration, and σ_2 is of type $\forall$ or MOD.
- σ_1 is of type $\exists$ and is in a MOD-tree, and σ_2 is of type $\exists$ and is not in any MOD-tree, or σ_2 is of type $\forall$ or MOD.
- 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 labelled by configuration σ_2 with parent labelled 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 17 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 18 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 *profile* 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 $t_i = \text{OR}$ (AND, MOD_m).

Definition 19 An ATM is *clean* if every path in every alternation tree of the ATM on every input has the same profile. (Note that only ATMs making $O(1)$ alternations can be clean.)

Definition 20 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 7 Let $L \subseteq \{0,1\}^*$, let s be a function in $(sub)subexp$, and let L be accepted by a uniform family $\{C_n\}$ of depth d circuits ($d = O(1)$) of type $\text{ACC}(s(n))$. 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,

- If $\{C_n\}$ is clean, then the profile of M is the compression of the signature of $\{C_n\}$.
- 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 $(\log\ s(n))^{O(1)}$.
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 (i.e., with existential nodes of fan-out 1), 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 and AND or OR gate is handled similarly. Thus if $\{C_n\}$ is clean, the profile 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 7 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 8 If L is accepted by an ATM M that makes a constant number of alternations between $\mathrm{MOD}_{m_1}, \mathrm{MOD}_{m_2}, \ldots, \mathrm{MOD}_{m_k}, \exists$ 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 $\mathrm{MOD}_{m_1}, \mathrm{MOD}_{m_2}, \ldots, \mathrm{MOD}_{m_k}, \exists$ and $\forall$ states.

Proof. Suppose M makes at most λ alternations on any input. Then N has the sequence $\mathrm{MOD}_{m_1}, \mathrm{MOD}_{m_2}, \ldots, \mathrm{MOD}_{m_k}, \exists, \forall$ (repeated λ times) hardwired into its finite control. N simply simulates M but follows the profile 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 profile, it simply proceeds with the simulation and behaves exactly as M does. In the case of a type mismatch, N behaves as follows:

- 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.)
- 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 profile) 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 9 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 profile (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 7. 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 {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 σ is an existential or MOD$_m$ non-halting configuration then
existentially guess strings w of length $2 \cdot t(n)$ and τ of
length $c \cdot t(n)$.
If w encodes a path from σ to configuration τ, where
the last step in the path involves an alternation (so τ is
a primary descendent of σ)
then enter a configuration of the same type as τ and call Eval(τ)
else call Trivial(*reject*)

If σ 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 τ be the i^{th}
primary descendent of σ. Call Eval(τ). (If there
is no such τ, then call Trivial(*accept*).)
(2) Otherwise.
Universally guess strings w of length $2 \cdot t(n)$ and τ of length $c \cdot t(n)$.
If w encodes a path from σ to configuration τ,
where the last step in the path involves an alternation (so
τ is a primary descendent of σ)
then enter a configuration of the same type as τ
and call Eval(τ)
else call Trivial(*accept*)

If σ is a halting configuration, then
Accept iff σ is accepting. (Note that this may involve accessing
the input, if σ 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 profile 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 profile calls for computation of type MOD_m, and $d = accept$ (respectively, $d = reject$), 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. ∎

Lemma 10 Let L be accepted by an ATM M satisfying the conditions of Lemma 9, running in time $t(n)$. Then there is a nice $\mathrm{ACC}(2^{O(t(n))})$ circuit family $\{C_n\}$ accepting L, such that the signature of $\{C_n\}$ is the same as the profile 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 [Ruz81]. The output gate of C_n will be labelled 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 labelled 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. ∎

Proof. (of Lemma 6) This follows immediately from Lemmas 7, 8, 9 and 10. ∎

4.2 Transformations on Circuits

Definition 21 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, $1 \leq i \leq r$, $x_i = y_i$. 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 11 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 $0 \leq i \leq s$. 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:

   - Check that $\langle n, g, g\rangle \in L_0$ and that g has type G.
   - Compute the fan-in r of g from the description of g.
   - Check that $\langle r, h, h\rangle \in L_1$.

3. Strings of the form $\langle n, g\#h, g\#h'\rangle$ with $h \neq h'$. This is done as follows:

   - Check that $\langle n, g, g\rangle \in L_0$ and that g has type G.
   - Compute the fan-in r of g from the description of g.
   - Check that $\langle r, h, h'\rangle \in L_1$.
   - 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:

   - Check that $\langle n, g', g'\rangle$ and $\langle n, g, g\rangle \in L_0$, and that g has type G.
   - Compute the fan-in r of g from the description of g.
   - Check that $\langle r, h_0, h_0\rangle \in L_1$ (h_0 is the output gate of E_r).

- Check that $\langle n, g', g\rangle \in L_0$.
- 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:
   - Check that $\langle n, g, g\rangle$ and $\langle n, g', g'\rangle$ are in L_0, where g has type G.
   - Check that $\langle n, g, g'\rangle \in L_0$.
   - Compute the fan-in r of g from its description.
   - Check that $\langle r, h, h\rangle \in L_1$.
   - Compute the number j such that g' is the j^{th} input to g (using the strong connection property).
   - Let $x_1, x_2, \ldots, x_r$ denote the inputs to E_r. Check that $\langle r, h, x_j\rangle \in L_1$.
   - 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.
   - Check that $\langle n, g, g\rangle$ and $\langle n, g', g'\rangle$ are in L_0.
   - Compute the fan-in r of g and check that h_0 is the output gate of E_r.
   - Compute the fan-in r' of g' and check that $\langle r', h, h\rangle \in L_1$.
   - 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 12 Suppose L is accepted by an ACC($(sub)subexp$) family $\{C_n\}$. Then L is accepted by a nice probabilistic ACC($(sub)subexp$) circuit family $\{D_n\}$[4] such that

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

Proof. By Lemma 6, 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:

[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.

- By a construction in the proof of Lemma 13 in [AH90], 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 [VV86]; similar constructions may be found in work by Toda [Tod91] and Kannan, Venkateswaran, Vinay and Yao [KVVY92].) 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 11 to prove that the new circuit family (now probabilistic) is clean, and thus by Lemma 6 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.

- 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 $1 \leq i \leq t-1$ and for all i, $1 \leq i \leq t$, 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, $1 \leq i \leq t$ to change G into an AND of $\mathrm{MOD}_{a_i^{e_i}}$'s. Suppose G has r inputs. For each m, the subcircuit family $\{E_r\}$ 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 11 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 6 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.

- 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. [BT91]):
x is congruent to 0 (mod 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 (mod 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:

- The top level consists of an AND gate that has fan-in e.
- 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.
- 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 11 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 (sub)subexponential size. The proof is completed by appeal to Lemma 6.

■.

4.3 Circuits with Symmetric Gates

In order to prove Theorem 3 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 22 Let $f : \mathbf{N} \to \mathbf{N}$ be a function. Then $\{C_{n,t} : n \in \mathbf{N}, 1 \leq t \leq f(n)\}$ 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

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

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

- 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.
- The symmetric gate has fan-in $f(n)$ and the output gates of $C_{n,t}$, $1 \leq t \leq f(n)$, are connected to it.
- 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 6 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 profile and of approximately the same size. (In proving the analog of Lemma 6 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 13 Let L be accepted by an ACC$((sub)subexp)$ circuit family $\{C_n\}$. Then there is a constructible (sub)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 $C_{n,t}$, $1 \leq t \leq 2^{(\log s(n))^c}$ where $\{C_{n,t} : n \in \mathbf{N}, 1 \leq t \leq 2^{(\log s(n))^c}\}$ 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 12, if L is accepted by an ACC$((sub)subexp)$ circuit family, then L is accepted by a nice ACC$((sub)subexp)$ 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 (sub)subexponential. It is immediate that the other required properties also hold. ∎

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 14 Let $\{C_{n,t}\}$ be a nice family of ACC sequences of (sub)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 (sub)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 [BT91], 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 $C_{n,t}$. Let the fan-in of G_i be n_i for all i, $1 \leq i \leq r$ and let $\{x_{ij}\}$, $1 \leq j \leq n_i$ denote the set of inputs to G_i. Finally, let $x_i = \sum_{1 \leq j \leq n_i} x_{ij}$. Consider the AND of $G_1, G_2, \ldots, G_r$. By Fermat's Little Theorem, for $1 \leq i \leq r$,

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

Therefore,

$$\bigwedge_{i=1}^{r} [x_i \equiv 0 \pmod{p}] \iff 1 - \prod_{i=1}^{r} (1 - {x_i}^{p-1}) \equiv 0 \pmod{p}$$

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

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

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. The expression $(*)$ that we have derived above is in the most general form. For our situation, we can simplify things by noticing that all the MOD_p gates $G_1, G_2, \ldots, G_r$ have the same fan-in since the sequence $\{C_{n,t}\}$ is nice. If we let this fan-in be denoted by n_0, then it is not hard to see that the number of AND gates that are input to the new MOD_p gate g is $\sum_{k=1}^{r} c_k \binom{r}{k} (n_0^{p-1})^k$.

[5]Note that this does not change the value of the expression mod p.

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 $1 \leq k \leq r$. 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, $1 \leq i \leq k$ 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:
   - check that G is the output gate of $C_{n,t}$. (This can be done because the circuits are well-named.)
   - check that $\langle n, t, G, G_i \rangle \in L_0$ for all i, $1 \leq i \leq r$, 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:
   - check that G is the output gate of $C_{n,t}$.
   - verify that $H_1, H_2, \ldots, H_k$ are all distinct.
   - check that for all i, $1 \leq i \leq k$, $\langle n, t, G, H_i \rangle \in L_0$.
   - check that m has the right value based on the parity of k.
   - 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 $\langle G \# \langle \langle H_1 \# L_1 \rangle, \langle H_2 \# L_2 \rangle, \ldots, \langle H_k \# L_k \rangle \rangle \# m, \mathrm{AND} \rangle$. We do the following to check if $\langle n, t, g_1, g_2 \rangle \in L$:

[6] The word "new" will hereafter be used to refer to gates that were created in the current step.

- check that $\langle n, t, g_1, g_1 \rangle \in L$.
- check that $\langle n, t, g_2, g_2 \rangle \in L_0$.
- 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 $g_1 = \langle G \# \langle G_1, G_2, \ldots, G_r \rangle, \mathrm{MOD}_p \rangle$ and $g_2 = \langle G \# \langle \langle H_1 \# L_1 \rangle, \langle H_2 \# L_2 \rangle, \ldots, \langle H_k \# L_k \rangle \rangle \# m, \mathrm{AND} \rangle$ 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 (sub)subexponential and r is polylogarithmic in s, the size of the new circuits is still (sub)subexponential.

Note that this step does not preserve the tree structure of the circuit so we use Lemma 6 to produce an equivalent nice circuit sequence.

■

Lemma 15 Let L be accepted by a uniform nice SYMACC circuit family $\{C_n\}$ of (sub)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 [BT91], using techniques developed in [Yao90, Tod91].

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

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 [BT91] that the polynomial P_k defined by

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

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 $A(n, \sum_{1 \le i \le f(n)} \mathrm{MOD}_p(x_{i,1}, x_{i,2}, \ldots, x_{i,n_0}))$, we see that this can also be expressed as

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

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

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

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

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

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

Note that $(\sum_{1 \le i \le f(n)} Q_k(\sum_{1 \le j \le n_0} x_{i,j}))$ is a low degree polynomial in the variables $\{x_{i,j}\}$. As in the proof of Lemma 14, 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.

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

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

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

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 $0 \leq m \leq 2k-1, 1 \leq c \leq c_m$, 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 write any such index $\langle i, m, c, j_1, \ldots, j_{(p-1)m}\rangle$ 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 (sub)subexponential number of circuits, each of which is of (sub)subexponential size, the new circuit is also of (sub)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 6, the circuit can be converted into nice form, which completes the proof. ∎

Proof. (of Theorem 3)

By Lemma 13, every language in ACC($(sub)subexp$) is accepted by a deterministic SYMACC circuit family of (sub)subexponential size, with small fan-in AND gates, no OR gates, and no MOD_m gates for composite m. Successive applications of Lemma 14 and Lemma 15 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. ∎

5 Conclusion

We have proved uniform circuit lower bounds for the permanent function and for certain sets in PP. In particular, we have shown that uniform ACC circuits of subexponential size cannot compute the permanent function. We have also proved a somewhat weaker bound for certain sets in PP by showing that these sets cannot be recognized by uniform ACC circuit families of susbsubexponential size. The proofs are based on a simulation of ACC given by Beigel and Tarui in [BT91]. We have shown how to carry out this simulation in the uniform setting. Some of the obvious open problems are:

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

- 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 [AG91], we show that P^{PP} contains sets that are immune to ACC, and that nonrelativizing proof techniques suitable for attacking the Dtime vs. Ntime questions 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 [FL92], 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 [GKT92], Green, Köbler 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 MOD_kP, 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. The improvement over the result in [BT91] is that the symmetric function at the top need not be different for different input sizes. 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 [Bar92] 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 kind

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 [BIS90] 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, the paper also shows that the simulation of Beigel and Tarui [BT91] is uniform according to this new notion of uniformity. In addition, it also shows that the simulation of Green, Köbler and Torán [GKT92] is uniform under this notion as well.

References

[ABFR91] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of Voting Polynomials. In *Proc. 23rd ACM Symposium on Theory of Computing*, pages 402–409, 1991.

[AG91] E. Allender and V. Gore. On strong separations from AC^0. In *Proc. 8th International Conference on Fundamentals of Computation Theory (FCT '91),* Lecture Notes in Computer Science 529, pages 1–15. Springer-Verlag, 1991.

[AH90] E. Allender and U. Hertrampf. On the power of uniform families of constant depth threshold circuits. In *Proc. 15th International Symposium on Mathematical Foundations of Computer Science,* Lecture Notes in Computer Science. Springer-Verlag, 1990.

[Bar86] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proc. 18th ACM Symposium on Theory of Computing*, pages 1–5, 1986.

[Bar89] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[Bar92] D.M. Barrington. Quasipolynomial size circuit classes. In *Proc. 7th Structure in Complexity Theory Conference*, 1992. To appear.

[BCGR92] S. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 1992. To appear.

[BCH84] P.W. Beame, S.A. Cook, and H.J. Hoover. Log depth circuits for division and related problems. In *Proc. 25th IEEE Symposium on Foundations of Computer Science*, pages 1–11, 1984.

[BIS90] D. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[Bor77] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6:733–743, 1977.

[BT88] D. Barrington and D. Thérien. Finite monoids and the fine structure of NC^0. *Journal of the ACM*, 35(4):941–952, 1988.

[BT91] R. Beigel and J. Tarui. On ACC. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 783–792, 1991.

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *JACM*, 28:114–133, 1981.

[FL92] U. Feige and C. Lund. On the hardness of computing the Permanent of random matrices. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 643–654, 1992.

[GKT92] F. Green, J. Köbler, and J. Torán. The power of the middle bit. In *Proc. 7th Structure in Complexity Theory Conference*, 1992. To appear.

[Hås87] J. Håstad. *Computational limitations for small depth circuits.* PhD thesis, Massachusetts Institute of Technology, 1987.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. AMS*, 117:285–306, 1965.

[KVVY92] R. Kannan, H. Venkateswaran, V. Vinay, and A. Yao. A circuit-based proof of Toda's theorem. *Information and Computation*, 1992. To appear.

[MT89] 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 372. Springer-Verlag, 1989.

[Raz87] A. 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(4):333–338, 1987.

[Ruz81] W.L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 21(2):365–383, 1981.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 61–69, 1983.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 77–82, 1987.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Val79] L. G. Valiant. The complexity of computing the Permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

[Yao90] A. Yao. On ACC and threshold circuits. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, pages 619–627, 1990.