DIMACS Technical Report 97-51
September 1997

The Permanent Requires Large Uniform Threshold Circuits[1]

by

Eric Allender[2,3]
Department of Computer Science
Rutgers University
P.O. Box 1179
Piscataway, NJ 08855-1179
allender@cs.rutgers.edu.

[1] A preliminary version of this work appeared in Proc. 2nd International Computing and Combinatorics Conference (COCOON '96).
[2] Permanent Member
[3] Supported in part by NSF grant CCR-9509603.

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

A recent paper by Caussinus, McKenzie, Thérien, and Vollmer [CMTV96] shows that there are problems in the counting hierarchy that require superpolynomial-size uniform TC^0 circuits. The proof in [CMTV96] uses "leaf languages" as a tool in obtaining their separations, and their proof does not immediately yield larger lower bounds for the complexity of these problems, and it also does not yield a lower bound for any particular problem at any fixed level of the counting hierarchy. (It only shows that hard problems must exist at *some* level.) In this paper, we give a simple direct proof, showing that any problem that is hard for the complexity class $C_=P$ requires more than size $T(n)$, if for all k, $T^{(k)}(n) = o(2^n)$. Thus, in particular, the permanent, and any problem hard for PP or #P require circuits of this size. Related and somewhat weaker lower bounds are also presented, extending the theorem of [CMTV96] showing that ACC^0 is properly contained in ModPH.

1 Introduction

1.1 Motivation and Background

The central problem in complexity theory is the task of proving lower bounds on the complexity of specific problems. Circuit complexity, in particular the study of constant-depth circuits, is one of the (few) areas where complexity theory has succeeded in actually providing lower bounds, and even in the study of constant-depth circuits one quickly arrives at the limits of current lower-bound technology. It is known that constant-depth circuits of AND, OR, and NOT gates (so-called AC^0 circuits) require exponential size even to compute the parity of n input bits [Hås87, Yao85], and similar lower bounds are known for constant-depth circuits of AND, OR, NOT, and MODp gates where p is prime [Raz87, Smo87]. When MODm gates are allowed for composite m, however, almost nothing is known. It remains an open question if there is any problem in NTIME($2^{n^{O(1)}}$) that cannot be done with polynomial size and constant depth with AND and MOD6 gates.

There is considerable reason to be interested in circuits with AND, OR, and MODm gates; circuits of this sort are called ACC^0 circuits (for "Alternating Circuits with Counters"; the superscript 0 refers to the fact that we are considering circuits of depth $O(\log^0 n)$.). The lovely result of [Bar89] characterizing NC^1 (log-depth fan-in two circuits) in terms of constant-width branching programs relies heavily on algebraic techniques, and shows that NC^1 corresponds to computation over *non-solvable* algebras. Barrington also defined the corresponding notion of computation over *solvable* algebras, and it is shown in [BT88] that this notion corresponds exactly to ACC^0 circuits. To restate these two points:

1. The results of [Bar89] establish intimate connections between circuit complexity and algebraic structure.

2. In this algebraic setting, ACC^0 is the most important subclass of NC^1.

Although, as mentioned above, it is unknown if small ACC^0 circuits suffice to compute all problems in NEXPTIME, lower bounds for *uniform* ACC^0 circuits were presented in [AG94]. Since our results, like those of [CMTV96] and [AG94], concern uniform circuits, it is necessary to briefly discuss uniformity.

A circuit family $\{C_n\}$ consists of a circuit for each input length n. If C_n is "sufficiently easy" to construct from n, then the family $\{C_n\}$ is said to be *uniform*. Different notions of "sufficiently easy" give rise to different notions of uniformity, and the question of which notion of uniformity is the "right" one to use when studying classes of circuits is not always clear. For the circuit classes considered here, convincing arguments are presented in [BIS90], arguing that a very restrictive notion of uniformity called *Dlogtime-uniformity* is the correct notion to use. Briefly, a circuit family $\{C_n\}$ is Dlogtime-uniform if, given n, g, h, a deterministic Turing machine can, in time $O(|n, g, h|)$, determine if gate g is connected to gate h in circuit C_n, and determine what sort of gates g and h are. The name "Dlogtime-uniformity" comes from the fact that the length of the input n, g, h is logarithmic in the size of the circuit C_n. (Dlogtime-uniformity is essentially equivalent to what Ruzzo called U_E uniformity in

[Ruz81]; although he considered only circuits of fan-in two, and not the unbounded fan-in circuits considered here and in [BIS90].) Throughout the rest of this paper, all mention of uniform circuits refers to Dlogtime-uniform circuits.

In this paper, $\mathrm{ACC}^0(S(n))$ will denote the class of languages with *uniform* ACC^0 circuits of size $S(n)$. ACC^0 denotes $\mathrm{ACC}^0(n^{O(1)})$.

In contrast to our lack of lower bounds for nonuniform ACC^0 circuits for sets in $\mathrm{NTIME}(2^{n^{O(1)}})$, it was shown in [AG94] that exponential size (i.e., size at least 2^{n^ϵ}) is required to compute the permanent (and other problems complete for #P) on *uniform* ACC^0 circuits. Thus there are sets in $\mathrm{P}^{\#\mathrm{P}}$ that require exponential-sized uniform ACC^0 circuits.

The complexity class PP is closely related to #P (for instance, $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}^{\mathrm{PP}}$). Recall that a set A is in PP if there is a nondeterministic polynomial time machine M with the property that $x \in A$ if and only if the number of accepting paths of M on input x is greater than the number of rejecting paths. PP contains both NP and coNP [Gil77]. Another related complexity class is $\mathrm{C}_=\mathrm{P}$; a set A is in $\mathrm{C}_=\mathrm{P}$ if there is a nondeterministic polynomial time machine M with the property that $x \in A$ if and only if the number of accepting paths of M on input x is *equal to* the number of rejecting paths. $\mathrm{C}_=\mathrm{P}$ contains coNP but is not known to contain NP; PP is contained in $\mathrm{NP}^{\mathrm{C}_=\mathrm{P}}$ [Tor91].

One might expect that similar exponential lower bounds would hold for PP or $\mathrm{C}_=\mathrm{P}$ as hold for #P, but [AG94] was able only to show that sets complete for these classes require more than "sub-subexponential" size ACC^0 circuits, where a function t is said to be sub-subexponential if $t(t(n)) = 2^{n^{o(1)}}$. (Note that for all "natural" and interesting size bounds t, t is sub-subexponential according to this definition if and only if for each k, $t(t(n)^k) = 2^{n^{o(1)}}$. For the rest of this paper, this will be the definition of "sub-subexponential". Observe that size bounds such as $2^{\log^k n}$ and $2^{(\log n)^{k \log\log n}}$ are sub-subexponential.)

Another class of constant-depth circuits that has attracted interest uses threshold (or MAJORITY) gates instead of counters. Let $\mathrm{TC}^0(S(n))$ denote the class of sets accepted by uniform constant-depth threshold circuits of size $S(n)$; TC^0 will denote $\mathrm{TC}^0(n^{O(1)})$. TC^0 captures the complexity of important natural computational problems such as sorting, counting, and integer multiplication. It is also a good complexity-theoretic model for the "neural net" model of computation [Par90].

It is easy to observe that $\mathrm{ACC}^0 \subseteq \mathrm{TC}^0$, and thus we have even fewer lower bounds for the threshold circuit model than for ACC^0 circuits. It is an easy consequence of the space hierarchy theorem that PSPACE-complete sets require exponential size uniform TC^0 circuits, but there is still no smaller complexity class in PSPACE that is known to require exponential-size TC^0 circuits.

There are well-studied subclasses of PSPACE that correspond in a natural way to the complexity classes AC^0, ACC^0, and TC^0. The relationship between the polynomial hierarchy and AC^0 is well-known and was established by [FSS84]. One way to present this correspondence is to observe that, when one considers alternating Turing machines that make only $O(1)$ alternations, a polynomial running time yields the polynomial hierarchy, while a logarithmic running time yields uniform AC^0. The analogous subclasses of PSPACE corresponding to ACC^0 and TC^0 are ModPH, and the counting hierarchy, respectively.

ModPH is in some sense a generalization of the polynomial hierarchy and of $\oplus$P (formal definitions appear in Section 2). The counting hierarchy (defined in [Wag86] and studied by several authors) consists of the union of the complexity classes PP, $\mathrm{PP}^{\mathrm{PP}}$, $\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}$, ... (Note that this is equal to the union of the classes $\mathrm{C}_{=}\mathrm{P}$, $\mathrm{C}_{=}\mathrm{P}^{\mathrm{C}_{=}\mathrm{P}}$, $\mathrm{PP}^{\mathrm{C}_{=}\mathrm{P}^{\mathrm{C}_{=}\mathrm{P}}}$, ...) In Section 2, we present models of computation (similar to alternating Turing machines) such that polynomial time on this model characterizes ModPH (or the counting hierarchy) while logarithmic time characterizes ACC^0 (or TC^0, respectively).

1.2 Statement of the Main Results

A recent paper by Caussinus, McKenzie, Thérien, and Vollmer [CMTV96] shows that ACC^0 is properly contained in ModPH, and TC^0 is properly contained in the counting hierarchy. The proof given by [CMTV96] uses "leaf languages" as a tool, and does not explicitly present a lower bound for any language in ModPH or in the counting hierarchy. The present work began as an attempt to discover if these techniques could be used to present an explicit lower bound. This attempt was only partially successful. For each given language A in ModPH, it is *still* an open question if A has polynomial size uniform ACC^0 circuits. The proof in [CMTV96] shows only that there *exists* a set in ModPH that requires superpolynomial size ACC^0 circuits; the present work gives a very simple direct proof of this same separation, but with the improvement that "superpolynomial" is replaced by "sub-subexponential".

In contrast, we *are* able to give explicit lower bounds on the uniform threshold circuit size required for many problems in the counting hierarchy. Although we are able only to show that some set *exists* in the counting hierarchy that requires more than sub-subexponential size uniform threshold circuits, we *can* obtain explicit lower bounds if we weaken the size bound only slightly.

Recall that a function t is sub-subexponential if $t^{(2)}(n) = 2^{n^{o(1)}}$, where $t^{(k)}$ denotes t composed with itself k times. We obtain a smaller class of functions if we impose the harsher restriction that for *all* k, $t^{(k)}(n) = o(2^n)$, but there seem to be no "natural" functions of interest that satisfy the former condition but not the latter. In particular, functions t such as $2^{\log^k n}$ and $2^{(\log n)^k \log\log n}$ satisfy the condition that for all k, $t^{(k)}(n) = o(2^n)$.

The main result of this paper can now be stated:

Theorem 5.1. *Let A be hard for $\mathrm{C}_{=}\mathrm{P}$ under $\leq_{\mathrm{T}}^{\mathrm{TC}^0}$ reducibility, and let t be a function such that for all $k, t^{(k)}(n) = o(2^n)$. Then $A \notin TC^0(t(n))$.*

The notion of reducibility $\leq_{\mathrm{T}}^{\mathrm{TC}^0}$ is defined as follows. Let A and B be subsets of $\{0,1\}^*$. Then $A \leq_{\mathrm{T}}^{\mathrm{TC}^0} B$ if there is a uniform family of polynomial-size constant depth circuits with MAJORITY gates and oracle gates for B, accepting A. (This is a natural adaptation of the notion of AC^0 reducibility studied in [Wil90] and elsewhere.)

In particular, all sets that are currently known to be complete for PP require threshold circuits of this size, because all such sets currently known are in fact complete under many-one reductions computable in uniform AC^0. Similarly, as was shown in [Zan91] (see also comments in [AG94]), the set $\{(x,i,b) \mid$ the ith bit of $\textsc{Permanent}(x)$ is equal to $b\}$ is hard

for $C_=P$ under AC^0 reducibility (with only one query), and hence uniform threshold circuits for the permanent must also satisfy this lower bound. In contrast, some of the functions that are shown to be #P-complete in [Val79] are shown *only* to be complete under *poly-time Turing* reducibility; for example, we have not checked to see if the problem of counting the number of (possibly imperfect) matchings in a bipartite graph is hard for $C_=P$ under truth-table reducibility (although we suspect that this is the case) and until this is established, the lower bounds of this paper are not known to hold for this problem. Similarly, the functions that are shown by Toda in [Tod94] to be complete for $FP^{\#P}$ are not immediately known to require large threshold circuits; it would first need to be established that they are hard for $C_=P$ under TC^0 reductions.

2 Machine Models

We assume the reader is familiar with nondeterministic oracle Turing machines. Given natural number m and oracle A, $\mathrm{Mod}_m P^A$ is the class of languages B such that, for some nondeterministic polynomial-time Turing machine M, x is in B if and only if the number of accepting computations of M^A on input x is a multiple of m. Then the class ModPH is defined to be the smallest class of languages containing P and with the property that if A is in ModPH, then so are NP^A and $\mathrm{Mod}_m P^A$ for every natural m. ModPH has been studied by several authors. (See, for example, [GKR+95].)

It is useful to have a model of computation characterizing ACC^0 and ModPH, in the same way that alternating Turing machines characterize both AC^0 and the polynomial hierarchy. The appropriate model of computation was defined in [AG94] as a variant of alternating Turing machines. We refer the reader to that [AG94] for detailed definitions; for the purposes of this paper it will suffice for the reader who is familiar with alternating Turing machines to consider the most natural way of augmenting the usual existential and universal states of an alternating Turing machine with Mod_m states.

Let a *signature* σ be a finite string from $\{\forall, \exists, \mathrm{Mod}_2, \mathrm{Mod}_3, \mathrm{Mod}_4, \ldots\}^*$. For any alternating Turing machine making $O(1)$ alternations, each path in the alternating tree of the machine on any input x has a signature given by the sequence of types of states the machine enters. If M is an alternating machine such that on all inputs x, all paths have the same signature σ, then M is said to be a σ machine. For instance, the signature of a Σ_2 machine is $\exists\forall$, and the signature of a typical machine accepting a language in $NP^{\oplus P^{\mathrm{Mod}_7 P}}$ is $\exists \mathrm{Mod}_2 \mathrm{Mod}_7$. Let $\sigma\mathrm{time}(t(n))$ denote the class of languages accepted by σ machines running in time $t(n)$. The technical lemmas in [AG94] essentially prove the following proposition.

Definition 1 *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 $O(g(n))$.*

Proposition 2.1 *Let $t(n)$ be a constructible function, $t(n) = \Omega(\log n)$. Then Uniform $\mathrm{ACC}^0(2^{O(t(n))}) = \bigcup_\sigma \sigma time(O(t(n)))$.*

It will turn out to be useful to us to note that a "tape reduction theorem" holds for σ machines; if a set is accepted in time $t(n)$ by a σ machine with k worktapes, then it is also accepted in time $O(t(n))$ by a σ machine with two worktapes.

(*Proof sketch:* Given a k-tape σ machine, follow the construction in [AG94] and build an ACC circuit, such that σ is the sequence of types of gates encountered in a root-to-leaf path. In the construction given in [AG94], the deterministic linear-time machine that checks the uniformity condition needs k tapes. (Roughly, the gates of the circuit are labeled with configurations of the σ-machine at points in the computation when an alternation is made, (the output gate of the circuit is labelled with the start configuration of the σ machine) and the labels also include a sequence of bits denoting the path in the alternation tree that leads from the first configuration to the second. In order to determine what gates are connected, the "uniformity machine" needs only to simulate the σ-Turing along that path; if the σ-machine has k tapes, then the uniformity machine will have k tapes, too.) However, by changing the naming convention for the gates in the circuit in a way that makes use of the ideas in the original tape-reduction proof of [BG70] for nondeterministic machines, we can make do with a two-tape deterministic machine checking the uniformity condition. (That is, if M_1 is the k-tape uniformity machine for the original circuit family, and the original circuit has gates g and h, where there is an edge in the circuit from h to g (corresponding to a computation path of the σ machine from g to h), then the new circuit will have gates (g, u) and (h, uv) where v is a string of length $t(n)$ recording what each of the k heads of the uniformity machine M_1 are reading in the computation of length $t(n)$ that verifies that h is connected to g. Since there are only $O(1)$ alternations of the σ-machine, and hence the circuit has depth $O(1)$, the size of the labels is still $O(t(n))$ bits, and thus the circuit size is still $2^{O(t(n))}$.) Now given a uniform σ-circuit family where the uniformity condition is checked by a 2-tape machine, the construction in [AG94] yields a two-tape σ-machine accepting the original language.)

Similarly, we will find it very convenient to have a single model of computation that is sufficient for describing both TC^0 and the counting hierarchy. Such a model was described in [PS88]. In their model, which they call a "threshold Turing machine", TC^0 corresponds to $O(\log n)$ time and $O(1)$ uses of the "threshold" operation, and the counting hierarchy corresponds to polynomial time and $O(1)$ uses of the threshold operation. The characterization of the counting hierarchy in terms of threshold Turing machines is given in [PS88], but the corresponding characterization of TC^0 is *not* presented there (since [PS88] predates the uniformity considerations of [BIS90]), and it also does not seem to have been published anywhere else. Although [BIS90] *does* give many equivalent characterizations of TC^0, the threshold Turing machine model is not mentioned in [BIS90]. Nonetheless, the proof of the following proposition is quite standard and follows along the lines of related results in [PS88, BIS90]:

Proposition 2.2 *Let $t(n)$ be a constructible function, $t(n) = \Omega(\log n)$. Then Uniform threshold circuit depth($O(1)$), size($2^{O(t(n))}$) = Threshold Turing machine time($O(t(n))$), thresholds($O(1)$).*

As is the case with the σ machines considered above, the Threshold Turing machines also enjoy a tape-reduction property, proved in essentially the same way. If a set is accepted in time $t(n)$ by a k-tape Threshold Turing machine, then it is accepted in time $O(t(n))$ by a Threshold Turing machine with two tapes.

The lower bounds presented in this paper do not depend on this tape reduction, but the statement of Theorem 3.1 is simplified by taking advantage of the tape reduction.

3 Diagonalization

It is important to note that the techniques used to prove the nondeterministic time hierarchy (originally proved in [SFM78], although we will use the very simple and general version proved by Žák [Ž83]) can be used to prove analogous hierarchies for other computational models defined in terms of nondeterministic Turing machines (with a fixed bound on the number of worktapes). In particular, an essentially word-for-word translation of the proof in [Ž83] shows the following.

Theorem 3.1 *Let 2^T be constructible. Then there is a set B in $\sigma time(T(n))$ such that, for all t with $t(n+1) = o(T(n))$, B is not in $\sigma time(t(n))$. Also, there is a set in D in Threshold Turing machine time($O(T(n))$),thresholds(k) such that, for all t with $t(n+1) = o(T(n))$, D is not in Threshold Turing machine time($O(t(n))$),thresholds(k).*

Proof: For completeness, we present the main outline of the proof. Let $M_1, M_2, \ldots$ be an enumeration of 2-tape σ-machines (threshold machines, respectively). Let f be a rapidly-growing function such time $T(f(i,n,s))$ is enough time for a *deterministic* machine to compute the function

$$(i,n,s) \mapsto \begin{cases} 1 & \text{if } M_i \text{ accepts } 1^n \text{ in } \leq s \text{ steps} \\ 0 & \text{otherwise} \end{cases}$$

(Letting $f(i,n,s)$ be greater than $T^{-1}(2^{2^{i+n+s}})$ is sufficient; note that it is important in our setting to handle *sublinear* functions T.)

Now divide Σ^* into regions, so that in region $j = (i,y)$, we diagonalize against machine M_i, thus ensuring that each machine is considered infinitely often. The regions are defined by functions $start(j)$ and $end(j)$, defined as follows: $start(1) = 1$, $start(j+1) = end(j)+1$, where $end(j) = f(i,\ start(j),\ T(start(j)-1))$ (where $j = (i,y)$). The important point is that, on input $1^{end(j)}$, a deterministic machine can, in time T, determine whether M_i accepts $1^{start(j)}$ in $\leq T(start(j)-1)$ steps.

By picking f appropriately easy to invert, we can guarantee that, on input 1^n, we can in time $T(n)$ determine which region j contains n.

Now it is easy to verify that the following routine can be computed in time $T(n)$ by a σ-machine (or a threshold machine, respectively). (In the pseudo-code below, U is a "universal" σ-machine (or threshold machine) with 4 tapes which is therefore able to simulate one step of machine M_i in about i^3 steps.)

1. On input 1^n, determine which region j contains n. Let $j = (i, y)$.
2. If $n = end(j)$, then accept iff M_i does *not* accept $1^{start(j)}$ in $\leq T(start(j) - 1)$ steps.
3. Otherwise, accept iff U accepts $(i, 1^{n+1})$ in $\leq T(n)$ steps. (Here, it is important that we are talking about $T(n)$ steps of U, which may be only about $T(n)/i^3$ steps of M_i.)

Let us call the set defined by the preceding pseudo-code A. Clearly, A is in $\sigma\text{time}(T(n))$. We now claim that is is not in $\sigma\text{time}(t(n))$.

Assume otherwise, and let M_i be the σ machine accepting A in time $t(n)$. Let c be a a constant such that $i^3 t(n+1) < T(n)$ for all $n \geq c$. Let y be a string of length $\geq c$, and consider stage $j = (i, y)$. Then for all n such that $start(j) \leq n < end(j)$, we have $1^n \in A$ iff $1^{n+1} \in A$. However this contradicts the fact that $1^{start(j)} \in A$ iff $1^{end(j)} \notin A$. □

4 Nonconstructive Lower Bounds

Once the definitions are in hand, the proof is now quite straightforward.

Theorem 4.1 *Let t be a sub-subexponential constructible function. Then there exist sets A in* ModPH *requiring size greater than $t(n)$ to compute on uniform* ACC^0 *circuits.*

Proof: Let t be given. Let C be a set complete for P under Dlogtime-uniform projections. (For instance, the standard complete set $\{(i, x, 0^j) : M_i \text{ accepts } x \text{ in time } j\}$ is a good choice for C.)

The proof consists of two cases.

Case 1: [C requires size greater than $t(n)$ to compute on uniform ACC^0 circuits.] In this case, of course there is nothing to prove.

Case 2: [C can be computed by uniform ACC^0 circuits of size $t(n)$.]

Since t is constructible, let g be the function such that $t(n) = 2^{g(n)}$.

In this case, it must happen that there is some σ such that ACC^0 is in $\sigma\text{time}(g(n^{O(1)}))$, because uniform circuits for any set reducible to C can easily be constructed from the circuits for C.

Now standard translational techniques can be used to show that for any signature τ, $\tau\text{time}(g(n))$ is contained in $\sigma\text{time}(g(t(n)^{O(1)}))$. To see this, consider any language A in $\tau\text{time}(g(n))$. Let $A' = \{x10^j : j + |x| + 1 = t(|x|) \text{ and } x \in A\}$. Our constructibility assumptions on t assure that A' is in ACC^0, and hence is in $\sigma\text{time}(g(n^l))$ for some l. Let M be this $g(n^l)$-time-bounded σ machine accepting A'. The σ machine M' that, on input x, simulates M on input $x10^{t(|x|)-|x|-1}$ runs in time $g(t(n)^l)$.

Since t is sub-subexponential, $2^{n^\epsilon} > t(t(n)^l) = 2^{g(t(n)^l)}$ and thus $g(t(n)^l) = o(n)$. Thus it follows from Theorem 3.1 that there is a set B in $\sigma\text{time}(n)$ (and hence in ModPH) such that, for all l, B is not in $\sigma\text{time}(g(t(n)^l))$, and thus B is not in $\tau\text{time}(g(n))$ and thus B does not have uniform ACC^0 circuits of size $t(n)$. □

It is important to note that, because of the nonconstructive nature of the proof of this theorem, the proof offers no clue as to *what* set in ModPH has large ACC^0 circuits.

An essentially identical proof yields the following theorem.

Theorem 4.2 *Let t be a sub-subexponential constructible function. Then there exist sets A in the counting hierarchy requiring size greater than $t(n)$ to compute on uniform threshold circuits.*

5 Main Result

Theorem 5.1 *Let t be a constructible function such that for all k, $t^{(k)}(n) = o(2^n)$. Let A be any set that is hard for $C_=P$ under $\leq_T^{\text{TC}^0}$ reductions. Then A cannot be computed by uniform constant-depth threshold circuits of size $t(n)$.*

Proof: Assume otherwise. Then we will show that for every set B in the counting hierarchy, there is some k such that B has uniform constant-depth threshold circuits of size $T(n) = t^{(k)}(n)$. But since $T(T(n)) = 2^{n^{o(1)}}$, this contradicts Theorem 4.2 of the paper.

For the purposes of this proof, define CH_1 to be $\text{C}_=\text{P}$, and for $i > 1$, define CH_i to be $\text{C}_=\text{P}^{\text{CH}_{i-1}}$.

First note that, under the assumption, $\text{C}_=\text{P}$ has circuits of size $t(n^{O(1)})n^{O(1)} = O(t(t(t(n))))$. (The circuit consists of a poly-size TC^0 reduction from the $\text{C}_=\text{P}$ set to A, where the oracle gates are replaced by circuits for A.) (Here, we are assuming without loss of generality that $t(n) \geq n^{\log n}$. Otherwise, we can take t' to be the maximum of $t(n)$ and $n^{\log n}$.)

Now assume that all sets in CH_i have uniform constant-depth circuits of size $O(t^{(4i)}(n))$, and consider a set $A \in \text{CH}_{i+1}$. Thus there is some nondeterministic machine M and a set $D \in \text{CH}_i$ such that M^D has exactly as many accepting paths as rejecting paths on input x if and only if $x \in A$. The set $\{(x, C) : M$ has exactly as many accepting paths as rejecting paths on input x, when all oracle queries are answered according to the circuit $C\}$ is in $\text{C}_=\text{P}$, and by the basis case has circuits of size $t(t(t(|(x, C)|)))$. When we replace C by the circuit for A that exists by inductive hypotheses, we obtain a circuit of size $\leq t^{(3)}(n + t^{(4i)}(n)) \leq t^{(4(i+1))}(n)$.

□

We do not know how to prove an explicit lower bound for any problem in ModPH that would be analogous to Theorem 5.1. It is easy to observe by the same proof techniques, that there is a set that is complete either for NP or for Mod_pP for some prime p that requires large ACC^0 circuits. Thus in order to find a set that is not in ACC^0 one need not consider anything beyond one of the "bottom" levels of ModPH. However, unlike the counting hierarchy, there are infinitely many such "bottom" levels in ModPH.

6 More Separations

From the foregoing, we know that TC^0 is properly contained in $\mathrm{C}_=\mathrm{P}$ (and hence is properly contained in PP). Note however that $\mathrm{C}_=\mathrm{P}$ is not known (or expected) to have circuits of less than exponential size. It is natural to ask if exponential size is necessary in order to find a language that is not in TC^0. In this section we show that it is not necessary; smaller size is sufficient in order to define languages that are not in TC^0. (On the other hand, merely having superpolynomial size is *not* known to be sufficient.)

First we make a simple observation:

Proposition 6.1 *For all $\epsilon > 0$, ACC^0 is properly contained in*

$$(DTIME(n^\epsilon) \cup \bigcup_\sigma \sigma time(\log n \log^* n)).$$

Proof: By standard padding methods, it is easy to construct a set $A \in \mathrm{DTIME}(n^\epsilon)$ that is complete for P under projections. This set A is thus also hard for ACC^0 under projections. If A is not in ACC^0 then this yields the desired conclusion.

Otherwise A is in ACC^0 and is therefore in $\sigma\mathrm{time}(O(\log n))$ for some σ. Since $\sigma\mathrm{time}(O(\log n))$ is closed under projections, it follows that ACC^0 is equal to $\sigma\mathrm{time}(O(\log n))$. By diagonalization, we obtain that ACC^0 is properly contained in $\sigma\mathrm{time}(O(\log n \log^* n))$. □

An identical proof yields

Proposition 6.2 *For all $\epsilon > 0$, TC^0 is properly contained in*

$$(DTIME(n^\epsilon) \cup \mathrm{TC}^0(n^{O(\log^* n)}).$$

(By essentially the same argument, we obtain that TC^0 is properly contained in $\mathrm{NC}^1 \cup \mathrm{TC}^0(n^{O(\log^* n)})$.)

We immediately get the following corollaries, which seem only marginally better than the results of [CMTV96] showing proper inclusion in ModPH and the counting hierarchy:

Corollary 6.3 *Let ϵ be greater than 0. Then:*

ACC^0 *is properly contained in* $\mathrm{ACC}^0(2^{n^\epsilon})$.
TC^0 *is properly contained in* $\mathrm{TC}^0(2^{n^\epsilon})$.

But now we will use the technique of [ABHH93] to get a better separation.

Lemma 6.4 *Let S be a constructible function, $S(n) \geq n$.*
If $\mathrm{ACC}^0 = \mathrm{ACC}^0(S(n))$, *then* $\mathrm{ACC}^0 = \mathrm{ACC}^0(S(S(n)))$.

Proof: Let A be any set in $\sigma\mathrm{time}(O(\log S(S(n))))$. Since a constructible function $S(n)$ is of the form $2^{g(n)}$, this means that A is in $\sigma\mathrm{time}(O(g(S(n))))$. Let A' be the padded version $\{x10^{S(|x|)-|x|-1} : x \in A\}$. Our assumption implies that A' is in ACC^0, and thus is in $\sigma'\mathrm{time}(O(\log n))$ for some σ'. This in turn implies that A is in $\sigma'\mathrm{time}(O(\log(S(n))))$, and thus by assumption A is in ACC^0. □

Corollary 6.5 *Let T be a constructible function such that, for some k and all large n, $T^{(k)}(n) > 2^n$, where $T^{(k)}$ is T composed with itself k times. Then*

$$\mathrm{ACC}^0 \text{ is properly contained in } \mathrm{ACC}^0(T(n)).$$

Corollary 6.6 *Let T be a constructible function such that, for some k and all large n, $T^{(k)}(n) > 2^n$. Then*

$$\mathrm{TC}^0 \text{ is properly contained in } \mathrm{TC}^0(T(n)).$$

7 Conclusions and Open Problems

It is often harder to ask the right question than to answer that question. In [AG94] we presented lower bounds on the uniform circuit complexity of certain problems in PSPACE, and did not see any way to prove lower bounds on the ACC^0 circuit complexity of any given problem in ModPH. Given the inspiration of [CMTV96], it is easy to give a direct proof showing that there *exist* sets in ModPH having large ACC^0 circuit complexity, without giving lower bounds on any specific set in ModPH.

This same technique, taken one step further, provides explicit lower bounds for many specific problems in the counting hierarchy, including the complete sets for $\mathrm{C}_=\mathrm{P}$, PP, and several functions complete for #P.

An obvious question is whether the sub-subexponential lower bounds given here and in [AG94] can be improved to exponential lower bounds. The lower bounds presented here for $\mathrm{C}_=\mathrm{P}$, PP, and the permanent are incomparable with the bounds presented in [AG94]; the bounds presented here are for more powerful circuits (threshold circuits as opposed to ACC^0 circuits), but the size bounds presented here are not as large as in [AG94]. It seems unlikely that the bounds presented here are optimal; probably exponential size is required for all of these problems.

Of course, an even more desirable step would be to prove directly that MAJORITY requires exponential size for ACC^0 circuits. (The "natural proofs" framework of [RR94] indicates that many lower bound proofs may be quite difficult to obtain. However, since ACC^0 is a very limited class in many respects (and in particular it is not clear that one should expect pseudorandom generators to be computable in ACC^0), it it not clear that lower bounds for ACC^0 should be hard to obtain.)

Acknowledgments

I thank the authors of [CMTV96] for making their manuscript available to me. I thank Dieter van Melkebeek for helpful discussions.

From allender@imsc2.imsc.ernet.in Thu Feb 27 10:01:00 1997 Received: from imsc2.imsc.ernet.in (imsc2.imsc.ernet.in [202.41.95.3]) by macon.informatik.uni-tuebingen.de (8.8.4/8.8.3/AIX-4.1/WSI-1.0) with ESMTP id JAA24774 for ¡allender@informatik.uni-tuebingen.de¿; Thu,

27 Feb 1997 09:59:20 +0100 Received: (from allender@localhost) by imsc2.imsc.ernet.in (8.7.5/8.7.5) id OAA21014; Thu, 27 Feb 1997 14:27:47 +0530 (GMT+05:30) Date: Thu, 27 Feb 1997 14:27:47 +0530 (GMT+05:30) From: Eric Allender ¡allender@imsc.ernet.in¿ Message-Id: ¡199702270857.OAA21014@imsc2.imsc.ernet.in¿ To: allender@informatik.uni-tuebingen.de Subject: threshold.bbl Status: R

References

[ABHH93] E. Allender, R. Beigel, U. Hertrampf, and S. Homer. Almost-everywhere complexity hierarchies for nondeterministic time. *Theoretical Computer Science*, 115:225–242, 1993.

[AG94] E. Allender and V. Gore. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing*, 23:1026–1049, 1994.

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

[BG70] R. Book and S. Greibach. Quasi-realtime languages. *Mathematical Systems Theory*, 4:97–111, 1970.

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

[BT88] D. A. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35:941–952, 1988.

[CMTV96] H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. In *Proceedings, 11th Annual IEEE Conference on Computational Complexity*, pages 12–21, 1996.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6:675–695, 1977.

[GKR+95] F. Green, J. Köbler, K. Regan, T. Schwentick, and J. Torán. The power of the middle bit of a #P function. *Journal of Computer and System Sciences*, 50:456–467, 1995.

[Hås87] J. Håstad. *Computational Limitations for Small Depth Circuits.* MIT Press, Cambridge, MA, 1987.

[Par90] I. Parberry. A primer on the complexity theory of neural networks. In R. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Sourcebook*, volume 6 of *Studies in Computer Science and Artificial Intelligence*, pages 217–268. North-Holland, Amsterdam, 1990.

[PS88] I. Parberry and G. Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36:278–302, 1988.

[Raz87] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41:598–607, 1987. English translation in Mathematical Notes of the Academy of Sciences of the USSR **41:4**, 333-338.

[RR94] A. Razborov and S. Rudich. Natural proofs. In *Proceedings, 26th ACM Symposium on Theory of Computing*, pages 204–213, 1994.

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

[SFM78] J. Seiferas, M. Fischer, and A. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25:146–167, 1978.

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

[Tod94] S. Toda. Simple characterizations of p(#p) and complete problems. *Journal of Computer and System Sciences*, 49:1–17, 1994.

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38:753–774, 1991.

[Val79] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[Ž83] S. Žàk. A Turing machine hierarchy. *Theoretical Computer Science*, 26:327–333, 1983.

[Wag86] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[Wil90] C. Wilson. Decomposing NC and AC. *SIAM Journal on Computing*, 19:384–396, 1990.

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

[Zan91] V. Zankó. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2:77–82, 1991.