Electronic Colloquium on Computational Complexity, Report No. 14 (2004)

# Languages to diagonalize against advice classes

Chris Pollett
214 MacQuarrie Hall,
Department of Computer Science,
San Jose State University,
One Washington Square, San Jose CA 95192.
Email: pollett@cs.sjsu.edu

## Abstract

*Variants of Kannan's Theorem are given where the circuits of the original theorem are replaced by arbitrary recursively presentable classes of languages that use advice strings and satisfy certain mild conditions. These variants imply that* $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/n^k$ *does not contain* $\mathsf{P}^{\mathsf{NE}}$, $\mathsf{DTIME}(2^{n^{k'}})/n^k$ *does not contain* $\mathsf{EXP}$, $\mathsf{SPACE}(n^{k'})/n^k$ *does not contain* $\mathsf{PSPACE}$, *uniform* $\mathsf{TC}^0/n^k$ *does not contain* $\mathsf{CH}$, *and uniform* $\mathsf{ACC}/n^k$ *does not contain* $\mathsf{ModPH}$. *Consequences for selective sets are also obtained. In particular, it is shown that* $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{NP\text{-}sel})$ *does not contain* $\mathsf{P}^{\mathsf{NE}}$, $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{P\text{-}sel})$ *does not contain* $\mathsf{EXP}$, *and that* $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{L\text{-}sel})$ *does not contain* $\mathsf{PSPACE}$. *Finally, a circuit size hierarchy theorem is established.*

*Categories and Subject Descriptors: F.1.3 [***Theory of Computation***]: Relations among complexity classes*
*General Terms: Theory*
*Additional Key Words and Phrases: advice classes, EXP, NEXP, NE, CH, ModPH, p-selective*

## 1 Introduction

One way to characterize nonuniform complexity classes is in terms of advice functions. A set $A$ is in $\mathcal{C}/\mathcal{F}$ if there is a $f : \mathbb{N} \to \Sigma^* \in \mathcal{F}$ and a $B \in \mathcal{C}$ such that $x \in A$ if and only if $\langle x, f(|x|)\rangle \in B$. Languages with p-size circuits can be characterized as $\mathsf{P/poly}$ and those with p-size branching programs as $\mathsf{L/poly}$. Despite many years of study it is open whether $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ or even $\mathsf{NEXP} \subseteq \mathsf{L/poly}$. In this paper the advice string characterization of nonuniform classes rather than their combinatorial characterization is examined in more detail in an attempt to both simplify existing proofs as well as shed some insight on these hard problems.

The starting point of the present work is Kannan [11] which exhibits sets in $\mathsf{NEXP}^{\mathsf{NP}}$ requiring super-polynomial sized circuits. Kannan also gives sets in $\Sigma_2^{\mathsf{p}}$ requiring circuits of size greater than $n^k$ for any fixed $k$. The idea in these results is to guess a minimal circuit of a somewhat larger size and verify that no smaller circuit can compute it. That a circuit of this larger size works follows by a counting argument. This counting argument is done for a specific computational model so if one wants to transfer this result to other models, one has to come up with a new counting argument.

Another approach to lower bounds for nonuniform classes is via Komolgorov complexity. Using this approach, Homer and Mocas [10] show that $\mathsf{EXP} \not\subseteq \mathsf{DTIME}(2^{O(n^{c_1})})/n^{c_2}$. Fu [6], also by this approach, shows that $EXP$ is not contained in the sets reducible in deterministic time $n^k$ to a $p$-selective set. Here $k$ is fixed and this paper uses the notation $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{P\text{-}sel})$ for this class of sets. The Komolgorov complexity notions used in these results are based on time, so to generalize them to space, counting, or probabilistic classes requires a reworking of the argument.

If a nonuniform class has an advice characterization, however, then the advice strings themselves can be used as both combinatorial objects to diagonalize against and as a source of random larger strings to serve as counterexamples. In this paper three languages useful to diagonalize against an advice class $\mathcal{C}/\mathcal{F}$ are presented. Using these languages, two advice versions of Kannan's Theorem are proven. In terms of alternations, the slightly stronger variant is that $\mathcal{C}/\mathcal{F} \not\supseteq \Sigma_2\text{-}\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ where $\mathcal{F} \subseteq o(t)$ and $\mathcal{C}$ and $\mathcal{C}'$ are recursively presentable and the latter contains a "universal predicate" for the former. The proof idea comes from a constructive version of Kannan's result in Cai and Watanabe [4]. The current paper's result can be used to not only get the results of Homer and Mocas and Fu mentioned above, but also allows one to show new results like

ISSN 1433-8092

$\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/n^k \not\supseteq \mathsf{P}^{\mathsf{NE}}$, $\mathsf{SPACE}(n^{k'})/n^k \not\supseteq \mathsf{PSPACE}$, and $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{L\text{-}sel}) \not\supseteq \mathsf{PSPACE}$.

A common technique for making nonuniform complexity classes uniform is to require that some property of a combinatorial object in the nonuniform class be of low complexity. For instance, a circuit family $\{C_n\}$ is DLOGTIME uniform if one can in DLOGTIME in the size of $C_n$ determine if two gates in $C_n$ are connected and if so by what gate type. It is unknown if the languages computed by nonuniform constant depth threshold circuits, $\mathsf{TC}^0$, contain all of the counting hierarchy CH (the union of P, $\mathsf{P}^{\mathsf{PP}}$, $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$, ...); however, it is known from Caussinus, et al. [5] that DLOGTIME uniform $\mathsf{TC}^0$, $u\mathsf{TC}^0$, does not contain CH. Allender [1] gives a threshold machine diagonalization proof of this fact based on the padded diagonalization techniques used in the proof of the nondeterministic time hierarchy theorem [21, 16]. In this paper, we show our variant of Kannan's theorem implies $u\mathsf{TC}^0/n^k \not\supseteq \mathsf{CH}$. That is, one can still separate these classes after some nonuniformity has been added back. One also gets that uniform, constant depth unbounded fan-in, AND, OR, $\mathrm{mod}_m$ for any $m$ gate circuits, ACC, where $n^k$ advice is "added back" does not contain ModPH. Here ModPH is the generalization of polynomial hierarchy which allows modular counting quantifiers.

Another application of an advice based approach to separating nonuniform classes, is the possibility to use one of the languages mentioned above to separate advice classes from other advice classes whose advice strings are shorter. Using this idea, size hierarchies for many combinatorial classes can be shown. In this paper, it is shown that for bounded fan-in, $AND$, $OR$, $NOT$ circuits that $\mathsf{SIZE}(s\log^{2+\epsilon} s) \supsetneq \mathsf{SIZE}(s)$ for any $\epsilon > 0$.

This paper is organized as follows: Section 2 summarizes the notations used in this paper. Section 3 presents three classes of languages useful to diagonalize against advice classes. It also presents advice based variants of Kannan [11]. Section 4 studies the power of languages computed by reductions to advice based classes. Corollaries of these two sections are then given. Section 6 considers implications of earlier results to selective sets. Section 7 concerns separating advice classes from other advice classes.

## 2 Preliminaries

Balcázar, Díaz, and Gabarró [2, 3], Papadimitriou [13], Hemaspaandra and Ogihara [8], and Vollmer [19] have more on advice classes and circuit complexity. This section contains only what is needed in the following.

For convenience sake, the alphabet of machines considered in this paper is $\{0,1\}$. $\{0,1\}^{\leq n}$ denotes the strings over $\{0,1\}$ of length $\leq n$. Both $vw$ and $v \frown w$ denote concatenation of strings. $\langle x_1, \ldots x_n\rangle$ is defined to be the string obtain by replacing 0's and 1's in $x_i$'s by 00 and 10 respectively and by inserting a 01 in between numbers. The string $x + y$ is calculated by viewing $x$ and $y$ as numbers with the left digit being the high-order bit, adding these two numbers, and padding high order 0's to ensure the length of the result is at least $\max(|x|, |y|)$.

In this paper, sub-linear time machines operate in a slightly non-standard manner: the input tape is treated as an oracle **for both reading and writing**. For reading, a machine computes $i$ on a work tape, enters a query state, and in one time step enters one of a fixed finite set of states according to the symbol on the $i$th tape square of the input. For writing, a machine computes a pair $\langle i, b\rangle$ where $b \in \Sigma$, enters a special state, then $b$ in one step becomes the $i$th symbol of the input tape. This operation is useful, if the machine also has access to another oracle set $A$: The machine can make changes to the input and then query the changed version of the input to $A$. Querying any $A$ other than the input is also slightly non-standard: $\langle t_1, \ldots t_k\rangle$ is written and a query state is entered. The oracle $A$ receives $\langle x_1, \ldots, x_n\rangle$ where $x_i$ is the contents of tape $t_i$. Based on whether this is in $A$, the machine enters the appropriate state. These changes to the machine model give at most a linear speed-up over the usual model. Given a predicate $A(x)$, $A(x_1, \ldots, x_n)$ denotes $A(\langle x_1, \ldots, x_n\rangle)$. Frequently, the distinction between a set $A$ and it characteristic function, which will be written $A(x)$ glossed over.

A recursive presentation of $\mathcal{C}$ is an effective enumeration $M_1$, $M_2$,$\cdots$ of DTM's which halt on all their inputs, and such that $\mathcal{C} = \{L(M_i) \,|\, i > 0\}$. P, NP, PP, PSPACE, etc. all have such a presentation. Balcázar, Díaz, and Gabarró [2] has more details on such presentations. It will be assumed that each language gets enumerated infinitely often in the presentation. For the remainder of this paper assume: (1) $\mathcal{C}$, $\mathcal{C}'$ are recursively presentable, (2) $\mathcal{F}$ is a class of nondecreasing functions on $\mathbb{N}$, and (3) $t$ is a nondecreasing, time constructible function on $\mathbb{N}$.

**Definition 1** *$\mathcal{C}/\mathcal{F}$ denotes the languages $L/\{f\} := \{x|\langle x, f(|x|)\rangle) \in L\}$ for some $L \in \mathcal{C}$ and $f \in \mathcal{F}$.*

$\mathcal{C}/n^k$ is used for $\mathcal{C}/\cup_c \{c \cdot n^k\}$. When $n^k$ where $c = 1$ is of interest, $\mathcal{C}/\{n^k\}$ is used. log, lin, and poly are used for the classes $\cup_c c\cdot\log n$, $\cup_c c\cdot n$, and $\cup_{k,c}(c\cdot n^k)$, respectively.

$\mathcal{C}/\mathcal{F}$ is called an advice class. The most interesting advice classes are $\mathsf{P}/\mathsf{poly}$ and $\mathsf{L}/\mathsf{poly}$. Here P denotes polynomial time and L denotes log-space. $\mathsf{P}/\mathsf{poly}$ are the languages recognized by p-sized circuits and $\mathsf{L}/\mathsf{poly}$ are the languages recognized by p-sized branching programs.

$\mathsf{PRTIME}(t)$ denotes those languages decidable by a NTM in time $O(t)$ where the acceptance condition is that more than half of the paths accept when it is in the language. PP is $\cup_k \mathsf{PRTIME}(n^k)$. This paper uses the following standard names for complexity classes:

$$
\begin{aligned}
\mathsf{E} &:= \mathsf{DTIME}(2^{\mathsf{lin}}) \\
\mathsf{NE} &:= \mathsf{NTIME}(2^{\mathsf{lin}}) \\
\mathsf{EXP} &:= \mathsf{DTIME}(2^{\mathsf{poly}}) \\
\mathsf{NEXP} &:= \mathsf{NTIME}(2^{\mathsf{poly}}) \\
\mathsf{C}^0\mathsf{PRTIME}(t) &:= \mathsf{DTIME}(t) \\
\mathsf{C}^{i+1}\mathsf{PRTIME}(t) &:= \mathsf{PRTIME}(t)^{\mathsf{C}^i\mathsf{PRTIME}(t)} \\
\mathsf{CH}(t) &:= \cup_i \mathsf{C}^i\mathsf{PRTIME}(t) \\
\mathsf{C}^i\mathsf{P} &:= \mathsf{C}^i\mathsf{PRTIME}(\mathsf{poly}) \\
\mathsf{CH} &:= \mathsf{CH}(\mathsf{poly})
\end{aligned}
$$

Let $\Sigma_k$ ($\Pi_k$) -$\mathsf{TIME}(t)$ denote those languages recognized by any alternating TM with at most $k$ alternations the outermost being existential (universal) running in time $O(t)$. Write $\Delta_k$-$\mathsf{TIME}(t)$ for $\mathsf{DTIME}(t)^{\Sigma_k-\mathsf{TIME}(t)}$. When the time bound is polynomial, $\Sigma_k^{\mathsf{p}}$, $\Pi_k^{\mathsf{p}}$, and $\Delta_k^{\mathsf{p}}$ are used. In this case, if an oracle set $A$ is also in use then $\Sigma_k^{\mathsf{p}}(A)$, $\Pi_k^{\mathsf{p}}(A)$, and $\Delta_k^{\mathsf{p}}(A)$ are written to avoid subscripts and superscripts. Finally, co-$\mathcal{C}$ denotes the class $\{\bar{L} \mid L \in \mathcal{C}\}$.

The next two definitions are needed for the main results.

**Definition 2** *$\mathcal{C}'$ is* universal *(*co-universal*) for $\mathcal{C}$ if for some fixed enumeration of $\mathcal{C}$, $U := \{\langle e, x\rangle \mid$ machine $e$ in $\mathcal{C}$ accepts $x\} \in \mathcal{C}'$ ($U \in$ co-$\mathcal{C}$). $\mathcal{C}'$ is* versal *if it either universal or co-universal.*

The reader can check that NEXP is co-universal for co-NE, PSPACE is universal and co-universal for L. One common place where the distinction between a set and its predicate will be ignored is for this set universal $U$; the notation $U(e, x)$ will frequently be used for the predicate corresponding to this set. The next remark shows how to go from a $U$ which shows versality to a recursive presentation of $\mathcal{C}$.

**Remark 1** *Notice if $\mathcal{C}'$ is recursively presentable and versal for $\mathcal{C}$ by predicate $U(e, x)$ then $\mathcal{C}$ is recursively presentable. This is because a machine $M_U$ for $U$ must appear at some point in $\mathcal{C}'$'s enumeration. This machine $M_U$ stops on all of it inputs. So in the case where $\mathcal{C}'$ is universal, $\mathcal{C}$ can be presented by listing out the machines $M_e$ based on $M_U$ where the value of $e$ has been hard coded. If $\mathcal{C}'$ is co-universal then a presentation is obtained from these $M_e$'s by interchanging the accept and reject states.*

**Definition 3** *Let $clear(x, z) := 0^{|x|} + z$ if $|z| \leq |x|$ and $clear(x, z) := 0^{|x|}$ otherwise. $\mathcal{C}$ is $t$-clearable if whenever $P(x_1, \ldots, x_n)$ is a predicate in $\mathcal{C}$ then so is $P(x_1, \ldots, clear(x_i, z), \ldots, x_n)$ for $z \leq t$.*

The reader should verify that P is $p$-clearable for any polynomial $p$.

## 3 Main Result

Three ways to diagonalize against advice classes are now explored. The first technique is based on Schöning's proof [15] based on Kannan [11] that EXPSPACE $\not\subseteq$ P/poly. The basic idea of this proof is to enumerate $p$-time machines. Stage $i$ diagonalizes against the machine $M_i$ and advice strings of length less than $i^{i \log i}$. This is done in substeps the first of which is to run $M_i$ on the input $0^i$ for each advice string of less than this length. The string $0^i$ is put into the language iff the majority of the time $M_i$ rejects. The process is then repeated on the input $0^i + 1$ and the advice strings that answered correctly in the first substep. Taking the majority again further at least halves the number of correctly answering advice strings. After $i^{\log i} + 2$ substeps no advice strings that answer correctly are left and the diagonalization against $M_i$ is complete. The idea of this argument is now abstracted so that a general result can be obtained.

**Definition 4** *Let $acc_M(x, A)$ ($rej_M(x, A)$) denote the set of strings $y \in A$ such that $M$ on input $\langle x, y\rangle$ accepts (resp. rejects).*

Define $L(\mathcal{C}, t)$ as $\cup_i L(\mathcal{C}, t)_i$ where in turn $L(\mathcal{C}, t)_i := \cup_j L(\mathcal{C}, t)_{i,j}$ and the $L(\mathcal{C}, t)_{i,j}$'s are defined iteratively in conjunction with the languages $Aux(\mathcal{C}, t)_{i,j}$ as follows: $Aux(\mathcal{C}, t)_{i,-1} := \{0, 1\}^t$, and for $i \geq 0$, $Aux(\mathcal{C}, t)_{i,j}$ is $acc_{M_i}(0^i, Aux(\mathcal{C}, t)_{i,j-1})$ if $acc_{M_i}(0^i, Aux(\mathcal{C}, t)_{i,j-1})$ has fewer elements than $rej_{M_i}(0^i, Aux(\mathcal{C}, t)_{i,j-1})$ and is $rej_{M_i}(0^i, Aux(\mathcal{C}, t)_{i,j-1})$ otherwise. Here $M_i$ is the machine for the $i$th language in $\mathcal{C}$ according to some fixed enumeration. From these sets define $L(\mathcal{C}, t)_{i,-1} := \emptyset$ and for $j \geq 0$, $L(\mathcal{C}, t)_{i,j} := L(\mathcal{C}, t)_{i,j-1} \cup \{0^i + j\}$ if $Aux(\mathcal{C}, t)_{i,j} = acc_{M_i}(0^i, Aux(\mathcal{C}, t)_{i,j-1})$ and $L(\mathcal{C}, t)_{i,j} := L(\mathcal{C}, t)_{i,j-1}$ otherwise.

**Lemma 1** *Assume $\mathcal{F} \subseteq o(t)$. Then $\mathcal{C}/\mathcal{F}$ does not contain $L(\mathcal{C}, t)$.*

**Proof.** Suppose $L(\mathcal{C}, t)$ were in $\mathcal{C}/\mathcal{F}$. Then $L(\mathcal{C}, t) = L(M_i)/\{f\}$ for some machine $M_i$ in the enumeration of $\mathcal{C}$ and for some $f$ in $\mathcal{F}$. As from the preliminaries it is assumed that in a presentation a machine accepting the same language as $M_i$ is enumerated infinitely often, so $M_i$ can be chosen so that $t(i) > f(i)$ since $\mathcal{F} \subseteq o(t)$. There are at most $1 + 2 + \cdots + 2^t = 2^{t+1} + 1$ advice strings of length $\leq t(i)$. One of these strings, say $w$, of length $f(i)$ must be the string used to show $L(\mathcal{C}, t) = L(M_i)/\{f\}$. Now consider which of strings $0^i$, $0^i + 1$, $0^i + 2$, …, $0^i + t(i) + 1$ are in $L(\mathcal{C}, t)$. Making use of $w$, $M_i$ must answers correctly for each of these strings whether or not it is in $L(\mathcal{C}, t)$. However, given the definition of $L(\mathcal{C}, t)_{i,0}$ at least 1/2 of all advices strings of length $\leq t(i)$ answer incorrectly on $0^i$, so $w$ cannot be among these. Of those that answer correct at least half answer incorrectly on $0^i + 1$, and so on. After $t(i) + 2$ iterations there are no advice strings left that can

successfully decide each of the strings $0^i$, $0^i+1$, $0^i+2$, ..., $0^i+t(i)+1$. Therefore, $L(\mathcal{C},t) \neq L(M_i)/\{f\}$. $\square$

Lemma 1 shows that for each $e$ there is a $w$ of length $t+2$ so that $M_e$ on the inputs $0^i$, $0^i+1$, $0^i+2$, ..., $0^i+t(i)+1$ together with any fixed on any fixed advice $w'$ of length $\leq t$ differs in at least one position from $w$. Such a $w$ is called a *counterexample advice*. It is not hard to show the above language could be recognized by a deterministic time $t$ reduction to a PP-like class in $\mathcal{C}'$. However, as PP-like operations are powerful as evidenced by the fact [18] that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}$, a stronger result will be sought after. The next goal, instead, is an advice version of Kannan [11].

Define $f_M(n,w,s)$ on input $n$ to output $x_0 \cdots x_s$ where $x_i$ is either 1 (resp. 0) depending on whether $M$ on the $i$th string lexicographically of length $n$ using advice $w$ accepts (resp. rejects). Define $\mu_M(n,t)$ to be the lexicographically least string $w$ of length $t(n)+2$ such that such that $w$ is not equal to $f_M(n,w',t+1)$ for any string $w'$ of length $\leq t$. The set of strings of length $t(n)+2$ that are being minimized over is non-empty by the argument in the proof of Lemma 1.

**Definition 5** *Let $L_\mu(\mathcal{C},t) := \cup_i L_\mu(\mathcal{C},t)_i$, where $L_\mu(\mathcal{C},t)_i$ is defined as:*

$$\left\{ v \;\middle|\; \begin{array}{l} |v| = i,\ v = v'w, \text{where } |v'| = \lceil \log(t(i)+2) \rceil, \\ \text{and bit } v' \text{ of } \mu_{M_i}(i,t) \text{ is on} \end{array} \right\}.$$

The idea is that strings $x$ which match $0^{|x|}$, $0^{|x|}+1$,... $0^{|x|}+t(|x|)+1$ on their the low-order $\lceil \log(t(|x|)+2) \rceil$ bits will belong to the set if and only if their corresponding $0^{|x|}+y$ string is. The latter string belongs to the set if and only if the corresponding bit of the least counterexample advice is on. The next lemma should be clear.

**Lemma 2** *Assume $\mathcal{F} \subseteq o(t)$. Then $\mathcal{C}/\mathcal{F}$ does not contain $L_\mu(\mathcal{C},t)$.*

An upper bound on the complexity of $L_\mu(\mathcal{C},t)$ is next calculated. Notice our definition of $L_\mu(\mathcal{C},t)$ depends implicitly on what enumeration is being used for $\mathcal{C}$. For the remainder, it is assumed that this enumeration is given by some $U$ which establishes versality via Remark 1.

**Lemma 3** *Let $t \in \Omega(\log n)$ and $t(n)+1 < 2^n$. Assume $\mathcal{C}'$ is versal for $\mathcal{C}$ and $t$-clearable. Then $L_\mu(\mathcal{C},t)$ is in $\Delta_3$-$\mathsf{TIME}(t)^{\mathcal{C}'}$.*

**Proof.** Let $U(e,x)$ show $\mathcal{C}'$ is versal for $\mathcal{C}$. For the remainder, assume $e = |x|$. This can be found from $x$ in $\log |x|$ time by binary search. Consider the co-$\mathsf{NTIME}(t)^{\mathcal{C}'}$ predicate $SOMEDIFF(x,w)$:

$$\begin{array}{l} \forall y \in \{0,1\}^{\leq t}[\exists z \leq t(|x|)+1 \\ \quad (\neg(U(e,clear(x,z),y) \Leftrightarrow BIT(z,w)=1))]. \end{array}$$

Here $BIT(z,w)$ returns the $z$th bit of $w$. $SOMEDIFF(x,w)$ asserts that for any advice $y$ of length $\leq t$, the $z$th bit of $w$ disagrees with $M_e$ on $0^{|x|}+z, y$ for at least one $z \leq t(|x|)+1$. It is needed that $|0^{|x|}+z| \leq n$. Also, as $e = |x|$, $e$ is not treated as a free variable. This predicate is in co-$\mathsf{NTIME}(t)^{\mathcal{C}'}$ as guessing $y$ is in co-$\mathsf{NTIME}(t)$ and as searching over the $z \leq t+1$ is in $DTIME(t)^{\mathcal{C}'}$. To compute $L_\mu(\mathcal{C},t)$, it suffices to find a least $w$ such that $SOMEDIFF(x,w)$ holds. Let $EXISTSDIFF(x,v)$ be the $\Sigma_2$-$\mathsf{TIME}(t)^{\mathcal{C}'}$ predicates:

$$\begin{array}{l} (\exists w \in \{0,1\}^{\leq t+2})[w = vy \wedge \\ \quad |w| = t+2 \wedge SOMEDIFF(e,x,w). \end{array}$$

Now let $M$ compute $L_\mu(\mathcal{C},t)$ by checking if $EXISTSDIFF(x,0)$ holds and if so continuing to compute additional bits. If not, $M$ changes the 0 to 1 and computes additional bits. $M$ continues until it gets all $t+2$ bits of $w$. Finally, $M$ accepts if and only if $BIT(z,w)=1$ holds. $\square$

Taking Lemma 2 and Lemma 3 together gives:

**Theorem 1** *Let $t \in \Omega(\log n)$ and $t(n)+1 < 2^n$. Assume $\mathcal{C}'$ is versal for $\mathcal{C}$ and $t$-clearable. Assume $\mathcal{F} \subseteq o(t)$. Then $\mathcal{C}/\mathcal{F} \not\supseteq \Delta_3$-$\mathsf{TIME}(t)^{\mathcal{C}'}$.*

To go from $\Delta_3$-$\mathsf{TIME}(t)^{\mathcal{C}'}$ to the $\Sigma_2$-$\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ result, the idea of the proof in Cai and Watanabe [4] that there is a $\Sigma_2^p$ set that requires circuits of size $n^k$ will be used. To begin, consider the following $\Sigma_3$-$\mathsf{TIME}(t)^{\mathcal{C}'}$ variant, $SIG3MU(x)$, of the previous algorithm:

$$\begin{array}{l} \exists w \in \{0,1\}^{\leq (t+2)^2} \forall i \leq t+2[\forall j < i \\ \quad (BIT(j,(w)_i) = BIT(j,(w)_{i+1})) \\ \quad \wedge SOMEDIFF(x,(w)_i) \wedge (BIT(j+1,(w)_{i+1}) = 1 \supset \\ \quad \forall v \in \{0,1\}^{t+2}(BIT(j+1,v) = 0 \wedge \\ \quad PREEQUAL(j,(w)_{i+1},v) \supset \neg SOMEDIFF(x,v))]. \end{array}$$

In the current setting, $e$ is a different function of $x$ then in Lemma 3 and is fixed at the end of the proof. Here $(w)_i$ returns the $i$ block of $t+2$ bits from $w$ (the start position of a block can be calculated as $i \cdot t$ in $t^2$ time) and here $PREEQUAL(j,v,x)$ holds if the first $j$ bits of $v$ and $x$ agree. To make $SIG3MU$ into a $\Sigma_2$-$\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ predicate ($SIG2MU(x)$) the implicit existential quantifier in $\neg SOMEDIFF(x,v)$ needs to be eliminated. So after guessing $w$, in $SIG2MU$ an advice string $y$ is also guessed of length $t(2(n+2t+|t|+1))$ (the reason for this size is described below) and it is assumed that $t(2(n+2t+|t|+1))$ is less than $t^k$ for some fixed $k$. It is hoped that using $y$, $U(e,x,j,u,v,y)$ holds if and only if

$$\begin{array}{l} PREFIXNOT(j,u,x,v) := \exists u' \leq \{0,1\}^{\leq t+1} \\ \quad |u'| = j \wedge PREEQUAL(|u|,u,u') \\ \quad \wedge \forall z \leq t+2U(e,clear(x,z),u') \Leftrightarrow BIT(z,v) \end{array}$$

holds. So $PREFIXNOT$ holds if $u$ extends to a length $j$ string witnessing $\neg SOMEDIFF(x,v)$. If such a $y$ exists then $\neg SOMEDIFF(x,v)$ is replaceable by the $\mathsf{DTIME}(t)^{\mathcal{C}'}$-predicate $\exists j \leq tU(e,x,j,\epsilon,v,y)$. For correctness, checks must be added in $SIG2MU(x)$ to ensure $U(e,x,j,u,v,y)$ indeed calculates $PREFIXNOT(j,u,x,v)$. One check is that:

$$\forall u \in \{0,1\}^{\leq t+1} \forall v \in \{0,1\}^{\leq t+2} \forall j \leq t \\ PREFIXNOT(j,u,x,v) \supset U(e,x,j,u,v,y).$$

This is in $\mathsf{co\text{-}NTIME}(t)^{\mathcal{C}'}$ and guarantees that $PREFIXNOT(j,u,x,v)$ implies $U(j,e,x,u,v,y)$. For the other direction, that $U(e,x,j,u,v,y)$ implies $PREFIXNOT(j,u,x,v)$, it is checked that (1)

$$\forall u \in \{0,1\}^{\leq t+1} \forall v \in \{0,1\}^{\leq t+2} \forall j \leq t \\ |u| = j \wedge U(e,x,j,u,v,y) \supset \\ \forall z \leq t + 2U(e, clear(x,z), u) \Leftrightarrow BIT(z,v)$$

and (2)

$$\forall u \in \{0,1\}^{\leq t+1} \forall v \in \{0,1\}^{\leq t+2} \forall j \leq t |u| < j \wedge \\ U(e,x,j,u,v,y) \supset U(e,x,j,u0,v,y) \vee U(e,x,j,u1,v,y)$$

both hold. As these are $\mathsf{co\text{-}NTIME}(t)^{\mathcal{C}'}$ checks, $SIG2MU(x)$ is a $\Sigma_2\text{-}\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ predicate that is hard for $e$ on advice of length $\leq t$ provided $y$ exists. However, if $y$ does not exist, $PREFIXNOT(j,u,x,v)$ is hard for $e$ and advice of length $\leq t(2(n+2t+|t|+1))$. If $PREFIXNOT(w)$ is considered where $w$ codes $j$,$u$,$x$,$v$ then $PREFIXNOT(w)$ is hard for $e$ for advice of length $\leq t(|w|)$. If $t \in \Omega(\log n)$ and $t(n)+1 < 2^n$, then $\log^* t(2(n+2t+|t|+1))$ and $\log^* n$ differ by a fixed constant. So if $e$ is set to $\log^* n$, then in the $PREFIXNOT(w)$ case a fixed adjusting factor can be used in calculating $e$. Let $L := \{y1 \mid SIG2MU(y)\} \cup \{y2 \mid PREFIXNOT(y)\}$. Note from the above $L$ is in $\Sigma_2\text{-}\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ and not in $\mathcal{C}/\mathcal{F}$. It will be in $\Sigma_2\text{-}\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$ for sub-linear time $t$'s because of the ability described in the preliminaries for the machines in this paper to write back to the input tape (and so after reading it to clear the last symbol of the input). This discussion establishes the next result:

**Theorem 2** *Let $t \in \Omega(\log n)$ and $t(n)+1 < 2^n$. Assume that $t(2(n+2t+|t|+3))$ is in $O(t^{O(1)})$ and that $\mathcal{C}'$ is versal for $\mathcal{C}$ and $t$-clearable. Assume $\mathcal{F} \subseteq o(t)$. Then $\mathcal{C}/\mathcal{F} \not\supseteq \Sigma_2\text{-}\mathsf{TIME}(t^{O(1)})^{\mathcal{C}'}$.*

Theorem 2 needs more conditions and more time than Theorem 1. Thus, both results are of interest and not strictly comparable.

## 4 Reductions to advice classes

Let $\mathsf{R}_T^{\mathcal{FC}}(\mathcal{C}')$ denote those languages Turing reducible to a language in $\mathcal{C}'$ where the reduction was computed by a function in $\mathcal{FC}$. The next result applies the theorems of the last section to get results about reductions.

**Theorem 3** *Let $t \in \Omega(\log n)$ and $t(n)+1 < 2^n$. Let $s(n), s'(n) \in \Omega(n)$ and $s(n)\log s(n) \in o(s'(n))$. Assume $\mathcal{C}'$ is versal for $\mathcal{C}$ and $t$-clearable. Assume $\mathcal{F} \subseteq o(t)$. Then (1) $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t \cdot s')^{\mathcal{C}'}$. (2) $R_T^{\mathsf{DTIME}(s(n))}(\mathcal{C}/\mathcal{F}) \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t \cdot s')^{\mathcal{C}'}$. (3) $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Sigma_2\text{-}\mathsf{TIME}((t \cdot s')^{O(1)})^{\mathcal{C}'}$. (4) $R_T^{\mathsf{DTIME}(s(n))}(\mathcal{C}/\mathcal{F}) \not\supseteq \Sigma_2\text{-}\mathsf{TIME}((t \cdot s')^{O(1)})^{\mathcal{C}'}$.*

**Proof.** (3) and (4) follow by the proofs of (1) and (2) but using Theorem 2. (1) implies (2) since in (1) the $\mathsf{DTIME}(s(n))$ can both use the advice string as well as send it along to the oracle from $\mathcal{C}$ when it makes a query. For (1) note that the condition $s(n)\log s(n) \in o(s'(n))$ guarantees $\mathsf{DTIME}(s'(n))^{\mathcal{C}'}$ is versal for $\mathsf{DTIME}(s(n))^{\mathcal{C}}$. Applying Theorem 1 then gives $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t)^{\mathsf{DTIME}(s')^{\mathcal{C}'}}$ from which (1) follows. □

## 5 Time, Space, and Counting Implications

Corollaries of the results of the last two sections are now given.

**Corollary 1** *There is a $\Sigma_2^{\mathsf{p}}$-set that requires circuits larger than size $n^k$.*

**Proof.** It is known (see Vollmer [19]) that $\mathsf{SIZE}(s)$ is contained in the class $\mathsf{DTIME}(s^2)/s\log s$ and that for $t \geq n$, $\mathsf{DTIME}(t) \subseteq \mathsf{SIZE}(t\log t)$. So $\mathsf{SIZE}(n^k) \subseteq \mathsf{DTIME}(n^{2k})/n^k \log n$. From the proof of the time hierarchy theorem, $\mathsf{DTIME}(n^{2k+1})$ will be versal for $\mathsf{DTIME}(n^{2k})$. As $O(n^k \log n) \subseteq o(n^{k+1})$ and as $(2(n+2n^{k+1}+(k+1)|n|+1))^k$ is in $O(t^{O(1)})$, from Theorem 2, $\Sigma_2^{\mathsf{p}}(\mathsf{DTIME}(n^{2k+1}))$ contains a language $L$ not in $\mathsf{SIZE}(n^k)$. But a $\mathsf{DTIME}(n^{2k+1})$ oracle adds no power to a $\Sigma_2^{\mathsf{p}}$ machine, so $L$ is also in $\Sigma_2^{\mathsf{p}}$. □

**Corollary 2** *(1) Neither $\mathsf{R}_T^{DTIME(n^{k'})}(\mathsf{NE}/n^k)$ nor $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/n^k$, contains $\mathsf{P}^{\mathsf{NE}}$. (2) Neither $\mathsf{R}_T^{DTIME(n^{k'})}(\mathsf{E}/n^k)$ nor $\mathsf{DTIME}(2^{n^{k'}})/n^k$ contains EXP. (3) $\mathsf{SPACE}(n^{k'})/n^k \supseteq \mathsf{L}/n^k$ does not contain PSPACE. (4) $\mathsf{C}^i PRTIME(n^{k'})/n^k$ does not contain CH.*

**Proof.** As argued in the previous section, if $\mathcal{C}$ is either NE or E then $\mathsf{R}_T^{DTIME(n^{k'})}(\mathcal{C}/n^k) \subseteq \mathsf{DTIME}((n^{k'})^{\mathcal{C}})/n^k$, so only the latter class result needs to be considered. For (1), note then that $n^{k'} \log n \in o(n^{k'+1})$, so by Theorem 3, $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}})/n^k$ does not contain $\Sigma_2^{\mathsf{p}}(NEXP)$. By the collapse of the strong exponential hierarchy [9], this latter class is $\mathsf{P}^{\mathsf{NE}}$. The remaining part of (2) and (3) and (4) each essentially follow from Theorem 2 as it gives that the given advice class does not contain $\Sigma_2^p(\mathcal{C}')$ where $\mathsf{C}'$ is EXP, PSPACE, or CH. For each of these classes, though, $\Sigma_2^p(\mathcal{C}') = \mathcal{C}'$. □

Item (2) of Corollary 2 above was previously shown by Homer and Mocas [10]. Many interesting variations on item (4) can be given. We present here some variants connected to circuit complexity. Recall from the introduction, a $u$ in front of a circuit class means the class restricted to DLOGTIME uniform circuits. The circuit class ACC consists of those languages decided by polynomial-sized circuits of constant-depth with unbounded fan-in gates of type AND, OR, NOT, or $\mathrm{MOD}_m$ for $m > 0$. As mentioned in the introduction, $\mathsf{TC}^0$ is used to denote those languages decided by constant depth threshold circuits. The class ModPH is the smallest class of languages containing P such that if $A$ is in ModPH so are $\mathsf{P}^A$ and $\mathsf{Mod}_m\mathsf{P}^A$ for some $m$. A language $B$ is in $\mathsf{Mod}_m\mathsf{P}^A$ if there is some nondeterministic oracle machine $M$ with oracle $A$ such that for all $x$, $x$ is in $B$ iff the number of paths on which $M$ is accepts $x$ is a multiple of $m$.

Parberry and Schnitger [14] define the notion of a threshold Turing machine (TTM). The class $u\mathsf{TC}^0$ is known is to be equal to the languages decided in logarithmic time on a TTM with constantly many application of the threshold operation; whereas, CH is precisely the languages decided by TTM's in polynomial time with c onstantly many applications of the threshold operation. Similarly, Allender [1] defines a notion of a $\sigma$-machine and shows $u$ACC corresponds to log-time on such machines and ModPH to polynomial time on such machines. In both cases, Allender argues these machines enjoy the tape reduction property and in his diagonalization proof argues there is a universal machine $U$ in both these models that simulates one step of the machine $M_i$ (in one of these models) in about $i^3$ steps. By affixing a linear number of steps count-down clock to such a universal machine, one gets that CH is versal for a class that contains $u\mathsf{TC}^0$ and similarly that ModPH is versal for a class that contains $u$ACC. Noting that $\Delta_3\text{-}\mathsf{TIME}(n^k)^{\mathsf{CH}} = \mathsf{CH}$ and $\Delta_3\text{-}\mathsf{TIME}(n^k)^{\mathsf{ModPH}} = \mathsf{ModPH}$ as well as recalling Theorem 1, one has a proof of the next corollary:

**Corollary 3** *For $k > 0$, (1) $u\mathsf{TC}^0/n^k$ does not contain* CH *and (2) $u$ACC$/n^k$ does not contain* ModPH.

## 6 Selective Set Implications

Consequences of Theorem 2 for selective sets are now explored. Selman [17] defines the P-selective sets based on the semi-recursive sets from computability theory. These latter sets had previously been used to study semi-membership algorithms. P-selective sets model an aspect of semi-feasible computation, and have also been extensively studied. The book by Hemaspaandra and Ogihara [8] provides a good introduction to these sets and their literature.

**Definition 6** *A language $L$ is in $\mathcal{C}$-*sel *if and only if there is a $R(x,y) \in \mathcal{C}$ such that if $x$ is in $L$ but $y$ is not then $R(x,y)$ holds and if $x$ is not in $L$ but $y$ is then $R(x,y)$ does not hold. The value of $R(x,y)$ when both $x$ and $y$ are both in the language or both not is arbitrary.*

P-sel is usually defined using polynomial $f(x,y)$'s that output $x$ or $y$ so that if only one of the two strings is in the language then that one is output. For $\mathcal{C} = \mathsf{P}$, Definition 6 is equivalent and more convenient to use.

Ko [12] shows that P-sel is contained in $\mathsf{P}/n^2$. One proof of this is as follow: First, redefine $R(x,y)$ as $R(\min(x,y), \max(x,y))$ to make it symmetrical. Then observe that on inputs of length $n$, a P-sel set induces a tournament, $T_R(n)$, (a digraph without self loops such that between any two vertices $x$, $y$ there is exactly one edge among $(x,y)$ and $(y,x)$) on the strings of length $n$ by directing an edge from $x$ to $y$ if $R(x,y)$ and from $y$ to $x$, otherwise. From the theory of tournaments, in this tournament of size $2^n$ there is a set of at most $n+1$ vertices, $z_0, \ldots, z_n$ such that for all $z$ in $T_R(n)$, there is some $z_i$ such that $(z_i, z)$ (i.e., $R(z, z_i)$ holds) is an edge in $T_R(n)$. Given this set of $n+1$ strings, each $n$-bits long, then $x$ is in the language if for some $z_i$, $R(x, z_i)$ holds. So given quadratic advice one can decide sets in P-sel. This argument also establishes:

**Lemma 4** $\mathcal{C}$-sel *is contained in* $\mathsf{DTIME}(n^2)^{\mathcal{C}}/n^2$.

As $\mathsf{DTIME}(n^3)^{\mathcal{C}'}$ is versal for $\mathsf{DTIME}(n^2)^{\mathcal{C}}$ and by Theorem 3, one has:

**Theorem 4** *Let $k \geq 0$. Then $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathcal{C}\text{-sel}) \not\supseteq \Delta_3\text{-}\mathsf{TIME}(n^{k+4})^{\mathcal{C}'}$ and $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathcal{C}\text{-sel}) \not\supseteq \Sigma_2^p(\mathcal{C}')$, provided $\mathcal{C}'$ is versal for $\mathcal{C}$ and $n^2$-clearable.*

**Corollary 4** *(1)* $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{NP\text{-}sel}) \not\supseteq \mathsf{P}^{\mathsf{NE}}$. *(2)* $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{P\text{-}sel}) \not\supseteq \mathsf{EXP}$. *(3)* $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{L\text{-}sel}) \not\supseteq \mathsf{PSPACE}$. *(4)* $\mathsf{R}_T^{DTIME(n^k)}(u\mathsf{TC}^0\text{-}\mathsf{sel}) \not\supseteq \mathsf{CH}$. *(5)* $\mathsf{R}_T^{DTIME(n^k)}(u\mathsf{ACC\text{-}sel}) \not\supseteq \mathsf{ModPH}$.

**Proof.** (1) Theorem 4 gives that $\mathsf{R}_T^{DTIME(n^k)}(\mathsf{NP\text{-}sel}) \not\supseteq \Sigma_2^{\mathsf{p}}(NEXP)$. By the strong exponential hierarchy collapse [9], this latter is $\mathsf{P}^{\mathsf{NE}}$. (2) Theorem 4 gives

$\mathsf{R}_T^{DTIME(n^k)}(\mathsf{P\text{-}sel}) \not\supseteq \Sigma_2^p(\mathsf{EXP}) = \mathsf{EXP}$. (3), (4), and (5) follow similarly. □

Fu [6], using Komolgorov complexity, had previously shown result (2).

## 7 On Avoiding Advice and Size Hierarchies

Given any of the complexity classes $\mathcal{C}$ considered in this paper, it is straightforward to construct a nonrecursive set in $\mathcal{C}/1$: Set the advice on inputs of length $n$ to be 1 if $n$ is in the halting set and 0 otherwise. Let $x \in L$ if and only if the advice for length $|x|$ is 1. Then $L$ is in $\mathcal{C}/1$ and clearly not recursive. One could hope that a result like $\mathsf{L/poly}$ is contained in $\mathsf{L}/lin$ is possible and from this get that $\mathsf{L/poly} \not\supseteq \mathsf{PSPACE}$ using the languages of this paper. Unfortunately, $L_\mu(\mathcal{C}, t)$ itself provides a counterexample showing this is impossible.

**Theorem 5** *Let $t \geq n$ be such that $t(n) + 3 < 2^n$ and let $\mathcal{F} \subseteq o(t)$. Then $L_\mu(\mathcal{C}, t) \in \mathsf{DTIME}(\log t)/\{t+2\}$ and so $\mathcal{C}/\mathcal{F} \not\supseteq \mathsf{DTIME}(\log t)/\{t+2\}$.*

**Proof.** Let $\mu_{M_n}(n, t)$ be the advice on inputs of length $n$. The $\mathsf{DTIME}(\log t)$ machine copies the low-order $\lceil \log(t(|x|) + 2) \rceil$ bits of the input to the the advice query tape and queries that bit of the advice string. It then accepts if that bit of the advice string is on and rejects otherwise. □

Let $\mathsf{PREC}$ denote the primitive recursive languages. $\mathsf{PREC}$ is recursively presentable so the next corollaries follow from the above theorem.

**Corollary 5** $\mathsf{DLOGTIME}/n^{k+1} \not\subseteq \mathsf{PREC}/n^k$. *So,* $\mathsf{DLOGTIME}/n^{k+1}$ *contains a language not in any of the classes* $\mathsf{NEXP}/n^k$, $\mathsf{P}/n^k$, *and* $\mathsf{L}/n^k$.

**Corollary 6** *Neither* $\mathsf{L/poly}$ *nor* $\mathsf{P/poly}$ *contains* $\mathsf{DTIME}(\log^2 n)/2^{\log^2 n}$.

Let $\mathsf{SIZE}(t)$ denote those languages computed by fan-in 2 AND, OR, NOT circuits of size $O(t)$. Let $\mathsf{AC}^0\text{-}\mathsf{SIZE}(t)$ denote those languages computed by constant-depth, unbounded fan-in AND, OR, NOT circuits of size $O(t)$. Theorem 5 also entails $\mathsf{SIZE}(s) \subsetneq \mathsf{SIZE}(s \log^{2+\epsilon} s)$. Kannan [11] gave a DNF formula of size $3n^{2k}$ not computed by circuits of size $n^k$, but the author knows of no tighter bound prior to this paper.

**Corollary 7** *Let $s \geq n$ and let $\epsilon > 0$. For $k > 0$,* $\mathsf{SIZE}(s \log^{2+\epsilon} s) \supsetneq \mathsf{SIZE}(s)$ *and* $\mathsf{AC}^0\text{-}\mathsf{SIZE}(s \log^{2+\epsilon} s) \not\subseteq \mathsf{SIZE}(s)$.

**Proof.** As mentioned earlier, it is known (see Vollmer [19]) that for $s > n$, $\mathsf{SIZE}(s) \subseteq \mathsf{DTIME}(s^2)/s \log s$. By Theorem 5 there is a language in $\mathsf{DLOGTIME}/s \log^{1+\epsilon} s \subseteq \mathsf{DLINTIME}/s \log^{1+\epsilon} s$ that is not in $\mathsf{SIZE}(s)$. The linear time is in the sum of the input and advice length. As $\mathsf{DTIME}(t) \subseteq \mathsf{SIZE}(t \log t)$, by hard-coding the bits of the advice as inputs in these circuits, one gets a language in at most $\mathsf{SIZE}((s \log^{1+\epsilon} s) \cdot \log(s \log^{1+\epsilon} s)) = \mathsf{SIZE}(O(s \log^{2+\epsilon} s)) = \mathsf{SIZE}(s \log^{2+\epsilon} s)$ that is not in $\mathsf{SIZE}(s)$. The $\mathsf{AC}^0\text{-}\mathsf{SIZE}(s \log^{2+\epsilon} s)$ result follows similarly as $\mathsf{DLOGTIME}$ is contained in $\mathsf{AC}^0$. □ The above proof only needs: (1) a polynomial time deterministic procedure to evaluate a circuit of size $s$ given the $s \log s$ sized encoding of it and an input, and (2) that the $\mathsf{DLOGTIME}$ algorithm from Theorem 5 can be evaluated in the circuit class with at most another $s \log s$ blow up. These conditions hold for a variety of classes such as: (a) constant depth unbounded fan-in $AND$, $OR$, $NOT$, $MOD_k$ (for all $k > 0$) circuits of size $s$ ($ACC(s)$), (b) constant depth threshold circuits of size $s$ ($\mathsf{TC}^0(s)$), (c) $\log^k s$ depth, fan-in $\leq 2$ $AND$, $OR$, $NOT$ circuits of size $s$ ($\mathsf{NC}^k(s)$), (d) $\log^k s$ depth, unbounded fan-in $AND$, $OR$, $NOT$ circuits of size $s$ ($\mathsf{AC}^k(s)$), and (e) size $s$ branching programs ($\mathsf{BP}(s)$). Thus, one gets:

**Corollary 8** *Let $\mathcal{C}(s)$ denote one of* $\mathsf{AC}^0(s)$, $\mathsf{ACC}(s)$, $\mathsf{TC}^0(s)$, $\mathsf{NC}^k(s)$, $\mathsf{AC}^k(s)$, $\mathsf{BP}(s)$. *Let $s \geq n$ and let $\epsilon > 0$. For $k > 0$, $\mathcal{C}(s \log^{2+\epsilon} s) \supsetneq \mathcal{C}(s)$.*

## 8 Acknowledgements

The author thanks Steve Homer, Nicholas Tran, Bin Fu, and Lance Fortnow for encouraging discussions/e-mail exchanges.

## References

[1] Eric Allender. The Permanent Requires Large Uniform Threshold Circuits. Chicago Journal of Theoretical Computer Science. Volume 1999. Article 7.

[2] J. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag. Second Edition. 1995.

[3] J. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity II.* Springer-Verlag. 1990.

[4] Jin-Yi Cai, Osamu Watanabe. On Proving Circuit Lower Bounds Against PH and Some Related Lower Bounds for Constant Depth Circuits. Tokyo Institute of Technology, Department of Math and Computing Sciences Technical Report. C-167. February 2003.

[5] Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic $\mathsf{NC}^1$ computation. Journal of Computer and System Sciences. Vol. 57 pp. 200–212. 1998.

[6] B. Fu. On P-selective Sets and EXP Hard Sets. Manuscript. 1997.

[7] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pp. 6–20, 1987.

[8] L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* Springer-Verlag. 2002.

[9] Lane A. Hemachandra The Strong Exponential Hierarchy Collapses. *Journal of Computer and System Sciences* Vol. 39 Issue 3. pp. 299–322. 1989.

[10] S. Homer and S. Mocas. Nonuniform Lower Bounds for Exponential Time Classes. In *Proceedings from the 20th International Symposium on Mathematical Foundations of Computer Science August, 1995*. LNCS #969, Springer-Verlag.

[11] R. Kannan. Circuit-Size Lower Bounds and Non-reducibility to Sparse Sets. *Information and Control*. Vol. 55. pp. 40–56. 1982.

[12] K. Ko. On self-reducibility and weak-P-selectivity. *Journal of Computer and System Sciences*. Vol. 26. Iss. 2 pp. 209–221. 1983.

[13] C. Papdimitriou. *Computational Complexity.* Addison-Wesley. 1994.

[14] I. Parberry, G. Schnitger. Parallel Computations with Threshold Functions. Journal of Computer and System Sciences. Volume 36. Issue 3. 1988. pp. 278–302.

[15] U. Schöning. *Complexity and Structure.* LNCS #211. Springer-Verlag. 1986.

[16] J. Seiferas, Michael J. Fischer, Albert R. Meyer. Separating Nondeterministic Time Complexity Classes. Journal of the ACM. Volume 25 , Issue 1. January 1978. pp. 146–167.

[17] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory* Vol. 13 Iss. 1. pp.55–65. 1979.

[18] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Jounral of Computing*. Vol. 20, pp. 865–877. 1991.

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

[20] Ingo Wegener. *The Complexity of Boolean Functions.* Wiley. 1987.

[21] S. Žák. A Turing machine hierarchy. Theoretical Computer Science Vol. 26. 1983. pp. 327–333.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'