LANGUAGES TO DIAGONALIZE AGAINST ADVICE CLASSES

Chris Pollett

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. Let poly_k denote those functions in $O(n^k)$. These variants imply that $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/\mathsf{poly}_k$ does not contain P^{NE}, $\mathsf{DTIME}(2^{n^{k'}})/\mathsf{poly}_k$ does not contain EXP, $\mathsf{SPACE}(n^{k'})/\mathsf{poly}_k$ does not contain PSPACE, uniform $\mathsf{TC}^0/\mathsf{poly}_k$ does not contain CH, and uniform $\mathsf{ACC}/\mathsf{poly}_k$ does not contain ModPH. Consequences for selective sets are also obtained. In particular, it is shown that $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{NP\text{-}sel})$ does not contain P^{NE}, $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{P\text{-}sel})$ does not contain EXP, and that $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{L\text{-}sel})$ does not contain PSPACE. Finally, a circuit size hierarchy theorem is established.

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

1. Introduction

One way to characterize nonuniform complexity classes is in terms of advice functions. A set A is in $\mathcal{C}/\mathcal{F}$, where $\mathcal{F}$ is a collection of functions from $\mathbb{N}$ to $\mathbb{N}$, if there is an $h : \mathbb{N} \to \Sigma^*$, a $f \in \mathcal{F}$, and a $B \in \mathcal{C}$ such that for all n, we have $|h(n)| = f(n)$ and we have $\forall x \in \Sigma^*$, $x \in A$ if and only if $\langle x, h(|x|)\rangle \in B$. Languages with polynomial size circuits can be characterized as P/poly, where poly is the class of polynomially bounded functions from $\mathbb{N}$ to $\mathbb{N}$, and those with p-size branching programs as L/poly. Despite many years of study it is open whether $\mathsf{NEXP} \subseteq \mathsf{P}/\mathsf{poly}$ or even $\mathsf{NEXP} \subseteq \mathsf{L}/\mathsf{poly}$. 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 (15) which exhibits sets in $\mathsf{NEXP}^{\mathsf{NP}}$ requiring super-polynomial sized circuits. Kannan also gives sets

in Σ_2^{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 (14) show that $\mathsf{EXP} \not\subseteq \mathsf{DTIME}(2^{O(n^{c_1})})/n^{c_2}$ where c_1 and c_2 are fixed. Here the advice string is of length exactly n^{c_2} rather than $O(n^{c_2})$. Fu (7), 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 $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{P\text{-}sel})$ 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, hard strings. In this paper three languages useful for diagonalizing 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 $\mathcal{C}/\mathcal{F} \not\supseteq \Sigma_2\text{-}\mathsf{TIME}([t(n)]^{O(1)})^{\mathcal{C}'}$ where $\mathcal{F} \subseteq o(t(n))$, $\mathcal{C}$ and $\mathcal{C}'$ are recursively presentable classes of languages, and $\mathcal{C}'$ contains a "universal predicate" for the class $\mathcal{C}$. The proof idea comes from a constructive version of Kannan's result in Cai and Watanabe (5). 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 $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/\mathsf{poly}_k \not\supseteq \mathsf{P}^{\mathsf{NE}}$, $\mathsf{SPACE}(n^{k'})/\mathsf{poly}_k \not\supseteq \mathsf{PSPACE}$, and $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{L\text{-}sel}) \not\supseteq \mathsf{PSPACE}$ for fixed $k, k' > 0$. Here poly_k is the class of the functions in $O(n^k)$.

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 $\mathsf{DLOGTIME}$ uniform if one can in $\mathsf{DLOGTIME}$ 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 whether TC^0, the class of languages computed by nonuniform constant depth threshold circuits, contains all of the counting hierarchy CH (the union of P, P^{PP}, $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$, ...). However, it is known from Caussinus, et al. (6) that $\mathsf{DLOGTIME}$ uniform 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; 27). In this paper, we show our variant of Kannan's theorem implies $u\mathsf{TC}^0/\mathsf{poly}_k \not\supseteq \mathsf{CH}$. That is, one can still separate these classes after some nonuniformity has been added back. One also gets that the class ACC, the class of uniform, constant depth, unbounded fan-in, AND, OR, MOD_m gate (for any m) circuits, where n^k advice is "added back" does not contain ModPH. Here ModPH is the generalization of the polynomial time hierarchy which allows modular counting quantifiers. These results appear to be the best known to date. Recently, Vinodchandran (24) has shown that PP is not contained in circuits of size n^k for any fixed k. It should be pointed that this result does not immediately imply that $u\mathsf{TC}^0/\mathsf{poly}_k \not\supseteq \mathsf{CH}$ as the class $u\mathsf{TC}^0$ contains threshold circuits of arbitrary polynomial degree size, even if they are uniform.

Another application of an advice based approach to separating nonuniform classes, is the possibility of using one of the three diagonalization languages constructed in this paper 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. As an example of this idea, Corollary 7 shows for bounded fan-in, AND, OR, NOT circuits that $\mathsf{SIZE}(\alpha(n)s(n)\log^2 s(n)) \supsetneq \mathsf{SIZE}(s(n))$ where $\alpha(n)$ is a nondecreasing unbounded function which is $o(s(n))$.

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 (15). 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

The books by Balcázar, Díaz, and Gabarró (2; 3), by Papadimitriou (17), by Hemaspaandra and Ogihara (11), and by Vollmer (25) 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 $\{0,1\}$. The notation $\{0,1\}^{\leq n}$ is used to denote the strings over $\{0,1\}$ of length less than or equal to n. Both vw and $v \frown w$ will be used to indicate concatenation of strings. The sequence of values $\langle x_1, \ldots, x_n \rangle$ is defined to be the string obtained by replacing 0's and 1's in x_i's by 00 and 10 respectively and by inserting a 01 in between numbers. We will often use quantifiers such as

the $(\exists y \in \{0,1\}^{\leq n})$ or $(\forall z \leq n)$. In the first case we view y as a string and we typically intend it to be implemented by nondeterministically guessing its bits; in the second case, we view z as a number and intend it to be implemented by using a counter and cycling through values. For a number z such that $|z| \leq i$, we use the notation $0^i + z$ to mean the string of length i which consists of a prefix of $i - |z|$ many zeros followed by z written as a binary string.

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**, and we allow an operation to quickly decompose sequences from the input. To begin it is allowed that there may be one main input tape t_0 on which the input is initially written and several auxiliary input tapes, $t_1, \ldots, t_k$. On an input $\langle x_1, \ldots, x_k \rangle$, a machine can enter a special state and in one time step have x_1 decoded from this tuple and written to tape t_1, x_2 decoded from this tuple and written to tape t_2, etc. This operation saves the linear time scan needed to read out each of the x_i's. For reading an input tape, a machine computes i on a work tape, enters a query state that specifies which tape t_j to read from, and in one time step enters one of a fixed, finite set of states according to the symbol on the ith tape square of this tape. For writing, a machine computes a pair $\langle i, b \rangle$ where $b \in \Sigma$ (in the case of this paper, $\Sigma = \{0,1\}$), enters a special state for a desired tape t_j, then b in one step becomes the ith symbol of the input tape t_j. 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. Making queries to an oracle A other than an input is also slightly non-standard: $\langle t_1, \ldots t_k \rangle$, where now the t_i's can be input or work tapes, is written and a query state is entered. The oracle A receives $\langle x_1, \ldots, x_k \rangle$ 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 $A(x)$, $A(x_1, \ldots, x_k)$ denotes $A(\langle x_1, \ldots, x_k \rangle)$. Frequently, the distinction between a set A and its characteristic function, which will be written as $A(x)$, is 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 $\mathcal{C} = \{L(M_i) \,|\, i > 0\}$. The complexity classes P, NP, PP, PSPACE, etc. all have such recursive presentations. The main results about such recursive presentations are given in Balcázar, Díaz, and Gabarró (2). It will be assumed that each language gets enumerated infinitely often in a recursive 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 from $\mathbb{N}$ to $\mathbb{N}$, and (3) $t = t(n)$ is a non-decreasing,

time constructible function on $\mathbb{N}$.

DEFINITION 1. Let $\mathcal{C}/f$ denote those languages of the form $\{x|\langle x, h(|x|)\rangle \in L\}$, where L is in $\mathcal{C}$, h is a function from $\mathbb{N}$ to $\{0,1\}^*$, and f is a nondecreasing function from $\mathbb{N}$ to $\mathbb{N}$ such that for all n, $|h(n)| = f(n)$. Given a class $\mathcal{F}$ of nondecreasing functions from $\mathbb{N}$ to $\mathbb{N}$, let $\mathcal{C}/\mathcal{F}$ denote class of languages L which belong to $\mathcal{C}/f$ for some f in $\mathcal{F}$.

As an example, the notation $\mathcal{C}/n^k$ is used when the function f in the above definition is $f(n) := n^k$. Some common classes of functions we will consider are:

$$\begin{aligned}
\mathsf{log} &:= \{f|f(n) \in O(\log n)\} \\
\mathsf{lin} &:= \{f|f(n) \in O(n)\} \\
\mathsf{quadratic} &:= \{f|f(n) \in O(n^2)\} \\
\mathsf{poly}_k &:= \{f|f(n) \in O(n^k)\}
\end{aligned}$$

Finally, $\mathsf{poly} := \cup_k \mathsf{poly}_k$.

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

$\mathsf{PRTIME}(t(n))$ denotes those languages decidable by a NTM in time $O(t(n))$ 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:

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

Let Σ_k (Π_k) -$\mathsf{TIME}(t(n))$ denote those languages recognized by any alternating TM with at most k alternations the outermost being existential (universal) running in time $O(t)$. Write Δ_k-$\mathsf{TIME}(t(n))$ for $\mathsf{DTIME}(t(n))^{\Sigma_k - \mathsf{TIME}(t(n))}$. The shorter notations Σ_k^{p}, Π_k^{p}, and Δ_k^{p} are used when the time bound is polynomial.

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 definitions are needed for the main results.

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

The word versal was invented for this paper as a convenient way to write 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 universal set 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 the enumeration of $\mathcal{C}'$. This machine M_U stops on all of its 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. Suppose $|z| \leq |x_i|$. The class $\mathcal{C}$ is *clearable* if $P(x_1, \ldots, x_n)$ being a predicate in $\mathcal{C}$ implies that $P(x_1, \ldots, 0^{|x_i|} + z, \ldots, x_n)$ is also a predicate in $\mathcal{C}$.

To see the class P is clearable, consider a predicate $P(x_1, \ldots, x_i, \ldots, x_n)$ in P. Given an input string $\langle x_1, \ldots, x_i, \ldots, x_n, z\rangle$ where $|z| \leq |x|$ we can in linear time compute the string $\langle x_1, \ldots, 0^{|x_i|} + z, \ldots, x_n\rangle$ and then compute the predicate P using this string. For sub-linear time machines the operation $0^{|x_i|} + z$ might be hard to do. As we will sometimes want to consider such machines with access to an oracle, such as in Lemma 3 below, we want to "push" this operation into the oracle itself so that the sub-linear time machine does not need to worry about this operation. If the class of oracles we are considering is clearable then we can do this without going out of this class.

3. Main Result

Three ways to diagonalize against advice classes are now explored. The first technique comes from Schöning's proof (20) of the result of Kannan (15) that $\mathsf{EXPSPACE} \not\subseteq \mathsf{P/poly}$. The basic idea of this proof is to enumerate polynomial time machines. Stage i diagonalizes against the machine M_i and advice strings of length less than $i^{\log i}$. 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 at least halves the number of remaining correctly answering advice strings. After $i^{\log i}+1$ 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).

Recall we are assuming t is a non-decreasing, time constructible function on $\mathbb{N}$. In order to define a language, $L(\mathcal{C}, t)$, which is hard for $\mathcal{C}$ using the idea above, we first define auxiliary languages, $Aux(\mathcal{C}, t)_{i,j}$. These languages capture the process of producing the language that diagonalizes against M_i, the machine for the ith language in $\mathcal{C}$ according to some fixed enumeration. We define:

$$Aux(\mathcal{C}, t)_{i,-1} := \{0,1\}^{\leq t(i)}, \quad \text{and for } j \geq 0,$$

$$Aux(\mathcal{C}, t)_{i,j} := \begin{cases} acc_{M_i}(0^i + j, Aux(\mathcal{C}, t)_{i,j-1}), \\ \qquad \text{if } acc_{M_i}(0^i + j, Aux(\mathcal{C}, t)_{i,j-1}) \text{ has fewer} \\ \qquad \text{elements than } rej_{M_i}(0^i + j, Aux(\mathcal{C}, t)_{i,j-1}); \\ \\ rej_{M_i}(0^i + j, Aux(\mathcal{C}, t)_{i,j-1}), \text{ otherwise.} \end{cases}$$

From these sets define:

$$L(\mathcal{C}, t)_{i,-1} := \emptyset, \text{ and for } j \geq 0,$$

$$L(\mathcal{C}, t)_{i,j} := \begin{cases} L(\mathcal{C}, t)_{i,j-1} \cup \{0^i + j\}, \\ \qquad \text{if } Aux(\mathcal{C}, t)_{i,j} = acc_{M_i}(0^i + j, Aux(\mathcal{C}, t)_{i,j-1}); \\ \\ L(\mathcal{C}, t)_{i,j-1}, \text{ otherwise.} \end{cases}$$

Finally, define $L(\mathcal{C}, t)$ as $\cup_i \cup_j L(\mathcal{C}, t)_{i,j}$.

LEMMA 1. Assume $\mathcal{F} \subseteq o(t(n))$ and $t(n) < 2^n$. 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, we can assume i is such that $t(i) > f(i)$, since $\mathcal{F} \subseteq o(t)$. There are at most $1 + 2 + \cdots + 2^{t(i)} = 2^{t(i)+1} - 1$ advice strings of length less than or equal to $t(i)$. 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 the strings 0^i, $0^i + 1$, $0^i + 2$, ..., $0^i + t(i)$ are in $L(\mathcal{C}, t)$. As $t(i) < 2^i$, each of these strings has length i, so for each of them M_i would receive the same advice string $f(i)$. Making use of w, M_i must answer 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 advice strings of length less than or equal to $t(i)$ answer incorrectly on 0^i, so w cannot be among these. Of those that answer correctly at least half answer incorrectly on $0^i + 1$, and so on. After $t(i) + 1$ iterations there are no advice strings left that can successfully decide each of the strings 0^i, $0^i + 1$, $0^i + 2$, ..., $0^i + t(i)$. Therefore, $L(\mathcal{C}, t) \neq L(M_i)/f$. □

Let w_i be the string of length $t(i) + 1$ which has a 1 in bit position j if and only if $0^i + j \in L(\mathcal{C}, t)$. Lemma 1 shows that M_i on the inputs 0^i, $0^i + 1$, $0^i + 2$, ..., $0^i + t(i)$ together with any fixed advice w' of length less than or equal to $t(i)$ differs in at least one position from w_i. Such a w_i is called a *hard string*.

Checking that M_i accepts or rejects an input for a majority of advice strings is a probabilistic rather than bounded error probabilistic operation. One might try to show that $L(\mathcal{C}, t)$ could be recognized by a deterministic time $t(n)$ reduction to an appropriate probabilistic class able to carry out the above construction. However, as probabilistic operations are powerful as evidenced by the fact (23) 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 (15).

Define $f_M(n, w', s)$ on input n, w', and s to output $b_0 \cdots b_s$ where b_j is either 1 (resp. 0) depending on whether M on the jth 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 $t(n) + 1$ such that w is not equal to $f_M(n, w', t(n))$ for any string w' of length less than or equal to $t(n)$. The set of strings of length $t(n) + 1$ that are being minimized over is non-empty by the argument in the proof of Lemma 1.

Let $BIT(j, w)$ be the function which returns the jth bit of w. Given $w := \mu_{M_i}(n, t)$, we could diagonalize against M_i by letting $0^i + j$ for $0 \leq j \leq t(i)$ be

in the language if and only if $BIT(j, w) = 1$. In general, checking if a string is of the form $0^i + j$ for $0 \leq j \leq t(i)$ is computationally prohibitive for sub-linear time classes. Nevertheless, we would like to construct hard languages that are contained in sub-linear time classes. To do this given x, let j be the number corresponding to the $|t(i)|$ least significant bits of x. We still let $w := \mu_{M_i}(n, t)$, but now say x is in the language if $BIT(j, w) = 1$ for this j obtained from x. This discussion motivates the next definition.

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:

$$\left\{ x \;\middle|\; \begin{array}{l} |x| = i,\ x = wj, \text{where } |j| = |t(i)| \\ \text{and } BIT(j, \mu_{M_i}(i, t)) = 1 \end{array} \right\}.$$

The next lemma should be clear.

LEMMA 2. Assume $\mathcal{F} \subseteq o(t(n))$ and $t(n) < 2^n$. Then $\mathcal{C}/\mathcal{F}$ does not contain $L_\mu(\mathcal{C}, t)$.

The condition that $t(n) < 2^n$ is for the same reason as in Lemma 1. 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 $t(n) \in \Omega(\log n)$ and $t(n) < 2^n$. Assume $\mathcal{C}'$ is versal for $\mathcal{C}$ and clearable. Then $L_\mu(\mathcal{C}, t)$ is in $\Delta_3\text{-}\mathsf{TIME}(t(n))^{\mathcal{C}'}$.

PROOF. Let $U(e, x)$ show $\mathcal{C}'$ is versal for $\mathcal{C}$. For the remainder, assume $e = |x|$. This $|x|$ can be found from x in $O(\log |x|)$ time: One starts by writing 1, 10, 100, etc. on a work tape and querying the auxiliary input tape with x on it until one finds the first blank symbol. Then one deletes the last 0 and moves the work tape back to the left hand side. Thereafter, one moves to the right again changing 0's to 1's and querying the input for a blank. If it is a blank, one changes the 1 back to 0; otherwise, one leaves it a 1. When one has arrived at the right end of the work tape again the length in binary will be written.

Consider now the $\text{co-}\mathsf{NTIME}(t(|x|))^{\mathcal{C}'}$ predicate $SOMEDIFF(x, w)$:

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

In English, for $SOMEDIFF(x, w)$ to hold it must be the case that for any advice y of length less than or equal to $t(|x|)$, the zth bit of w disagrees with

M_e on $\langle 0^{|x|} + z, y\rangle$ for at least one $z \leq t(|x|)$. Following our convention from the preliminaries, $U(e, 0^{|x|} + z, y)$ is $U(\langle e, 0^{|x|} + z, y\rangle)$, so the effect is that U computes what M_e would upon receiving the arguments $\langle 0^{|x|} + z, y\rangle$. As $e = |x|$, e is not treated as a free variable in the above. The predicate $SOMEDIFF(x, w)$ is in $\mathsf{co\text{-}NTIME}(t(|x|))^{\mathcal{C}'}$ as guessing y is in $\mathsf{co\text{-}NTIME}(t(|x|))$ and as searching over the $z \leq t(|x|)$ is in $\mathsf{DTIME}(t(|x|))^{\mathcal{C}'}$. Notice we are using that $\mathcal{C}'$ is clearable here so that $U'(e, x, z, y) := U(e, 0^{|x|} + z, y)$ will be a predicate in $\mathcal{C}'$. This is needed for sub-linear time classes, as one would not expect to be able to have the time to write the $|x|$ many bits of $clear(x, z)$ needed to prepare the input if we used U itself as the oracle. Observe also that in the sub-linear case we are using the fact that our machine model allows the encoding of tuples to be decomposed in one time step onto auxiliary tapes so that we can quickly figure out what is x and what is w from the input tuple $\langle x, w\rangle$ to $SOMEDIFF(x, w)$. Finally, one should also pay attention to the fact that the querying of U' for different values of z is making use of our machine's ability to write a sequence of tape numbers, enter a query state, and have the contents of those tapes converted to a tuple which is passed to the oracle.

Given $SOMEDIFF(x, w)$, 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 Σ_2-$\mathsf{TIME}(t(|x|))^{\mathcal{C}'}$ predicates:

$$(\exists y \in \{0,1\}^{\leq t(|x|)+1})[|vy| = t(|x|) + 1 \wedge SOMEDIFF(x, vy)].$$

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 $t(|x|) + 1$ bits of w. Finally, M accepts if and only if $BIT(z, w) = 1$ holds. □

Taking Lemma 2 and Lemma 3 together gives:

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

To go from the $\Delta_3\text{-}\mathsf{TIME}(t(n))^{\mathcal{C}'}$ to the $\Sigma_2\text{-}\mathsf{TIME}(t(n)^{O(1)})^{\mathcal{C}'}$ result, the idea of the proof in Cai and Watanabe (5), that there is a Σ_2^p -set that requires circuits of size n^k, will be used. We will again want to define a predicate which diagonalizes against machines M_e for each machine e in our enumeration. As in the case of the proof of Lemma 3, we will define a predicate whose computation on an input x is hard for a machine M_e where e is some slow growing but unbounded function of x. Let $(w)_i$ return the ith block of $t(|x|) + 1$ bits from a string w. To begin, consider the following $\Sigma_3\text{-}\mathsf{TIME}(t(n)^{O(1)})^{\mathcal{C}'}$ variant of the algorithm used in Lemma 3 to give a language not in $\mathcal{C}/\mathcal{F}$. $SIG3MU(x)$:

1. Guess a string w of length at most $(t(|x|)+1)^2$.

2. For each $i \leq t(|x|)+1$ check that:

   (a) The first i bits, u, of $(w)_i$ satisfies $EXISTSDIFF(x,u)$.

   (b) The first i bits of $(w)_i$ and of $(w)_{i+1}$ are the same.

   (c) For all strings v of length $t(|x|)+1$ with the same initial $i-1$ bits as $(w)_i$, if the ith bit of v is 0 and of $(w)_i$ is 1, then $\neg SOMEDIFF(x,v)$.

Recall $EXISTSDIFF(x,u)$ and $SOMEDIFF(x,u)$ in the proof of Lemma 3 are used to diagonalize against the machine M_e where e is written in their defining formulas, but where we said $e = |x|$. In what follows we will assume e is actually a even slower growing function of x. To simplify the discussion, let $PREEQUAL(j,v,v')$ hold if the first j bits of v and v' are the same. To make $SIG3MU$ into a Σ_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 $t(2(n+2t(n)+|t(n)|+4))$ (the reason for this size is described below) and it is assumed that $t(2(n+2t(n)+|t(n)|+4))$ is less than $t(n)^k$ for some fixed k. Provided t is monotone, this assumption implies that $t(n)$ must be in $O(n^{O(1)})$, since if $t(n)$ grows faster than this then $t(2(n+2t(n)+|t(n)|+4)) > t(t(n))$ will grow faster than $t(n)^k$ for any fixed k. Now it might be possible that with this advice y the versal predicate $U(e,x,j,u,v,y)$ holds if and only if u extends to a length j string v witnessing $\neg SOMEDIFF(x,v)$. More formally, one wants an advice y such that $U(e,x,j,u,v,y)$ holds if and only if the following predicate $PREFIXNOT(j,u,x,v)$ holds:

$$(\exists u' \leq \{0,1\}^j)[PREEQUAL(|u|,u,u') \wedge (\forall z \leq t(|x|)+1)(U(e,0^{|x|}+z,u') \Leftrightarrow BIT(z,v)=1)].$$

For our definition of $SIG2MU$, it will be assumed such a y exists. The final language that will eventually be constructed will be the union of $SIG2MU$ and another language handling hardness for when such a y does not exist at a given length. Again, the convention that $U(e,x,j,u,v,y)$ means $U(\langle e,x,j,u,v,y\rangle)$ is being used. As we have said above, in the current setting e is a different function of x then in Lemma 3 and is fixed at the end of the proof. If such a y exists then $\neg SOMEDIFF(x,v)$ is replaceable by the $\mathsf{DTIME}(t(|x|))^{\mathcal{C}'}$-predicate $(\exists j \leq t(|x|))U(e,x,j,\epsilon,v,y)$. 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:

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

This is in co-$\mathsf{NTIME}([t(n)]^{O(1)})^{\mathcal{C}'}$ and guarantees that $PREFIXNOT(j,u,x,v)$ implies $U(e,x,j,u,v,y)$. For the other direction, to ensure that $U(e,x,j,u,v,y)$ implies $PREFIXNOT(j,u,x,v)$, one checks that (1)

$$(\forall j \leq t(n))(\forall u \in \{0,1\}^{j})(\forall v \in \{0,1\}^{\leq t(n)+1})[U(e,x,j,u,v,y) \rightarrow$$
$$(\forall z \leq t(n)+1)(U(e,0^{|x|}+z,u) \Leftrightarrow BIT(z,v)=1)]$$

and that (2)

$$(\forall j \leq t(n))(\forall u \in \{0,1\}^{\leq j-1})(\forall v \in \{0,1\}^{\leq t(n)+1})[$$
$$U(e,x,j,u,v,y) \rightarrow U(e,x,j,u \frown 0,v,y) \vee U(e,x,j,u \frown 1,v,y)]$$

both hold where, again, $n = |x|$. The $j = 0$ case of (2) we view as trivially satisfied as the quantification over u is checking if u is in $\{0,1\}^{-1}$ which we view as the empty set. Both (1) and (2) are co-$\mathsf{NTIME}(t(n))^{\mathcal{C}'}$ checks, so $SIG2MU(x)$ is a Σ_2-$\mathsf{TIME}([t(n)]^{O(1)})^{\mathcal{C}'}$ predicate that is hard for e on advice of length less than or equal to $t(n)$ provided y exists. However, if y does not exist, then $PREFIXNOT(j,u,x,v)$ itself is hard for e and advice of length less than or equal to $t(2(n+2t(n)+|t(n)|+4))$. The expression $2(n+2t(n)+|t(n)|+4)$ bounds the maximum length of the string coding the tuple $\langle j,u,x,v\rangle$ ($|j| \leq |t(n)|$, $n = |x|$, $|v| \leq t(n)+1$, $|u| \leq t(n)$, and one has three commas in the code). That is, if $PREFIXNOT(w)$ is considered where w codes $\langle j,u,x,v\rangle$, then $PREFIXNOT(w)$ is hard for e for advice of length less than or equal to $t(|w|)$. If $t \in \Omega(\log n)$ and $t(n) < 2^n$ (which it will be for all but finitely many n if $t(n) \in O(n^{O(1)})$), then $\log^* t(2(n+2t(n)+|t(n)|+4))$ and $\log^* n$ differ by at most 2, as $[\log^*(\log n)]+2 = \log^*(2^n)$. So if e is set to $\log^*(|x|)$, then in the $PREFIXNOT(w)$ case a fixed adjusting factor can be used to calculate e from w. Let $L := \{y \frown 0 \mid SIG2MU(y)\} \cup \{y \frown 1 \mid PREFIXNOT(y)\}$. Note by the above construction L is in Σ_2-$\mathsf{TIME}([t(n)]^{O(1)})^{\mathcal{C}'}$ and not in $\mathcal{C}/\mathcal{F}$. It will be in Σ_2-$\mathsf{TIME}([t(n)]^{O(1)})^{\mathcal{C}'}$ for sub-linear time t's by essentially the same arguments as in Lemma 3. This discussion establishes the next result:

THEOREM 2. Let $t \in \Omega(\log n)$ be monotone. Assume that $t(n) \in O(n^{O(1)})$ and that $\mathcal{C}'$ is versal for $\mathcal{C}$ and clearable. Further, assume $\mathcal{F} \subseteq o(t(n))$. Then $\mathcal{C}/\mathcal{F} \not\supseteq \Sigma_2$-$\mathsf{TIME}([t(n)]^{O(1)})^{\mathcal{C}'}$.

Although Theorem 2 has a lower number of alternations in its conclusion, it requires more prerequisites and the time class is greater 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 $t(n) < 2^n$. 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 clearable and that $\mathcal{F} \subseteq o(t(n))$. Then the following relationships hold:

(1) $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t(n)\cdot s'(n))^{\mathcal{C}'}$.

(2) $R_T^{\mathsf{DTIME}(s(n))}(\mathcal{C}/\mathcal{F}) \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t(n)\cdot s'(n))^{\mathcal{C}'}$.

(3) $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Sigma_2\text{-}\mathsf{TIME}([t(n)\cdot s'(n)]^{O(1)})^{\mathcal{C}'}$.

(4) $R_T^{\mathsf{DTIME}(s(n))}(\mathcal{C}/\mathcal{F}) \not\supseteq \Sigma_2\text{-}\mathsf{TIME}([t(n)\cdot s'(n)]^{O(1)})^{\mathcal{C}'}$.

For (3) and (4), we assume additionally that $t(n)$ is monotone and $t(n) \in O(n^{O(1)})$.

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 $\mathsf{DTIME}(s(n))^{\mathcal{C}}/\mathcal{F} \not\supseteq \Delta_3\text{-}\mathsf{TIME}(t(n))^{\mathsf{DTIME}(s'(n))^{\mathcal{C}'}}$ from which (1) follows. □

5. Time, Space, and Counting Implications

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

COROLLARY 1. For each $k \in \mathbb{N}$, there is a Σ_2^{p}-set that requires circuits larger than size n^k.

PROOF. It is known (see Vollmer (25)) that $\mathsf{SIZE}(s(n))$ is contained in the class $\mathsf{DTIME}([s(n)]^2)/O(s(n)\log s(n))$, and that for $t(n) \geq n$, $\mathsf{DTIME}(t(n)) \subseteq \mathsf{SIZE}(t(n)\log t(n))$. So $\mathsf{SIZE}(n^k) \subseteq \mathsf{DTIME}(n^{2k})/O(n^k \log n)$. 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 n^{k+1} is in $O(n^{O(1)})$, from Theorem 2, it follows $\Sigma_2^{\mathsf{p}}(\mathsf{DTIME}(n^{2k+1}))$ contains a language L not in $\mathsf{SIZE}(n^k)$. But a $\mathsf{DTIME}(n^{2k+1})$ oracle adds no power to a Σ_2^{p} machine, so L is also in Σ_2^{p}. □

COROLLARY 2. For each $k, k' \in \mathbb{N}$, the following relationships between complexity classes can be established:

(1) Neither $\mathsf{R}_T^{\mathsf{DTIME}(n^{k'})}(\mathsf{NE}/\mathsf{poly}_k)$ nor $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/\mathsf{poly}_k$, contains P^{NE}.

(2) Neither $\mathsf{R}_T^{\mathsf{DTIME}(n^{k'})}(\mathsf{E}/\mathsf{poly}_k)$ nor $\mathsf{DTIME}(2^{n^{k'}})/\mathsf{poly}_k$ contains EXP.

(3) $\mathsf{SPACE}(n^{k'})/\mathsf{poly}_k \supseteq \mathsf{L}/\mathsf{poly}_k$ does not contain PSPACE.

(4) For $i \geq 1$, $\mathsf{C}^i\mathsf{PRTIME}(n^{k'})/\mathsf{poly}_k$ does not contain CH.

PROOF. As argued in the previous section, if $\mathcal{C}$ is either NE or E then $\mathsf{R}_T^{\mathsf{DTIME}(n^{k'})}(\mathcal{C}/\mathsf{poly}_k) \subseteq \mathsf{DTIME}(n^{k'})^{\mathcal{C}}/\mathsf{poly}_k$, 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, $\mathsf{DTIME}(n^{k'})^{\mathsf{NE}}/\mathsf{poly}_k$ does not contain $\Sigma_2^{\mathsf{p}}(\mathsf{NEXP})$. By the collapse of the strong exponential hierarchy (9), this latter class is P^{NE}. The remaining parts (2), (3), and (4), each essentially follow from Theorem 2, as this theorem implies that the given advice class does not contain $\Sigma_2^p(\mathcal{C}')$, where 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 (14). 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 $\mathsf{DLOGTIME}$ uniform circuits. The circuit class ACC is defined as $\cup_{m>0}\mathsf{AC}^0[m]$ where $\mathsf{AC}^0[m]$ consists of those languages decided by polynomial-sized circuit families of constant-depth with unbounded fan-in gates of type AND, OR, NOT, or MOD_m. As mentioned in the introduction, 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 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 (18) 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 applications of the threshold operation; whereas, CH is precisely the languages decided by TTM's in polynomial time with constantly many applications of the threshold operation. Similarly, Allender (1) defines a notion of a σ-machine and shows $u\mathsf{ACC}$ 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 $u\mathsf{ACC}$. 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, gives a proof of the next corollary:

Corollary 3. For each $k \in \mathbb{N}$, we have:

(1) $u\mathsf{TC}^0/\mathsf{poly}_k$ does not contain CH.

(2) $u\mathsf{ACC}/\mathsf{poly}_k$ does not contain ModPH.

6. Selective Set Implications

Consequences of Theorem 2 for selective sets are now explored. Selman (22) 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 books by Hemaspaandra and Ogihara (11) and Hemaspaandra and Torenvliet (12) provide good introductions to these sets and their literature.

Definition 6. A language L is in $\mathcal{C}\text{-}\mathsf{sel}$ if and only if there is a $R(x, y) \in \mathcal{C}$ such that: if $x \in L$, but $y \notin L$, then $R(x, y)$ holds; and, if $x \notin L$, but $y \in L$, then $R(x, y)$ does not hold. It is also required that for all distinct strings x and y exactly one of $R(x, y)$ and $R(y, x)$ must hold and that $R(x, x)$ should always hold.

P-sel is usually defined in terms of polynomial computable functions $f(x, y)$ that output x or y so that, if only one of the two strings is in the language, then that one is output. With a little more effort, one can define classes like NP-sel using functions (10). These definitions turn out to be equivalent to Definition 6 where $\mathcal{C}$ is P or NP. Definition 6 will be slightly more convenient for us, as it is defined entirely in terms of language classes as opposed to function classes. To see in the case of P that the two definitions are equivalent, suppose one has a polynomial time $f(x, y)$ selector for some language L. It can be assumed that f is symmetric in its arguments by defining $f'(x, y)$ to be $f(\min(x, y), \max(x, y))$. This can be verified to still be a selector for L. For this f', define $R(x, y)$ to be $f'(x, y) = x$. Notice if x is in L but y is not, then $R(x, y)$ holds but $R(y, x)$ does not. Similarly, if x is not in L but y is, then $R(x, y)$ does not hold but $R(y, x)$ does hold. Lastly, notice if both x and y are in or both not in the language, then as one $f'(x, y) = x$ and $f'(x, y) = y$ holds, one also has one of $R(x, y)$ and $R(y, x)$ holds. We have defined f' so that $f'(x, y) = f'(y, x)$ so only one of these two cases can be the case. For the other direction, suppose now one has a polynomial time $R(x, y)$ that selects as in Definition 6, one can define $f(x, y)$ to output x if $R(x, y)$ holds and output y otherwise. This can easily be checked to be a selector.

Ko (16) shows that P-sel is contained in P/quadratic. One proof of this is as follows: Let L be in P-sel via relation $R(x, y)$. Recall a tournament is a digraph without self loops such that between any two vertices $x \neq y$ there is exactly one edge among (x, y) and (y, x). Construct on the strings of L of length n, a tournament, $T_R(n)$, by directing an edge from y to x if $R(x, y)$ holds and $x \neq y$. Definition 6 guarantees for each pair of strings x, y in L that exactly one of (y, x) and (x, y) will be an edge in the resulting graph. This condition makes $T_R(n)$ a tournament of size at most 2^n. From the theory of tournaments, 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 for each length n string x it holds that x is in the language if and only if for some z_i, $R(x, z_i)$ holds. To see this notice, if x is not in L, then as z_i is in L, and by the the selecting property of R, $R(x, z_i)$ will not hold. On the other hand, if x is in L, by the the tournament property, there will be a z_i such that $R(x, z_i)$ holds. We remark that if there are no strings of length n in L then the set of strings is the empty set, so there is no element z_i in this set to make $R(z, z_i)$ hold. So in this case all strings x of length n would be rejected. Thus, in all cases, given at most quadratic advice, one can decide sets in P-sel. This argument also establishes:

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

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 $k \in \mathbb{N}$. 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 clearable.

COROLLARY 4. For each $k \in \mathbb{N}$, the following relationships hold:

(1) $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{NP\text{-}sel}) \not\supseteq \mathsf{P}^{\mathsf{NE}}$.

(2) $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{P\text{-}sel}) \not\supseteq \mathsf{EXP}$.

(3) $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{L\text{-}sel}) \not\supseteq \mathsf{PSPACE}$.

(4) $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(u\mathsf{TC}^0\text{-}\mathsf{sel}) \not\supseteq \mathsf{CH}$.

(5) $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(u\mathsf{ACC\text{-}sel}) \not\supseteq \mathsf{ModPH}$.

PROOF. (1) Theorem 4 gives that $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{NP\text{-}sel}) \not\supseteq \Sigma_2^{\mathsf{p}}(\mathsf{NEXP})$. By the strong exponential hierarchy collapse (9), this latter is P^{NE}. (2) Theorem 4 gives $\mathsf{R}_T^{\mathsf{DTIME}(n^k)}(\mathsf{P\text{-}sel}) \not\supseteq \Sigma_2^p(\mathsf{EXP}) = \mathsf{EXP}$. (3), (4), and (5) follow similarly. □

Fu (7), 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. Set $x \in L$ 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 $t(n) \geq n$ be such that $t(n) < 2^n$ and let $\mathcal{F} \subseteq o(t)$. Then $L_\mu(\mathcal{C}, t) \in \mathsf{DTIME}(\log t(n))/t(n) + 1$ and so $\mathsf{DTIME}(\log t(n))/t(n) + 1 \not\subseteq \mathcal{C}/\mathcal{F}$.

PROOF. Let $\mu_{M_n}(n, t)$ be the advice string on inputs of length n. The $\mathsf{DTIME}(\log t(n))$ machine copies the low-order $\lceil \log(t(n) + 1) \rceil$ 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. □

A slightly different statement of the result $\mathsf{DTIME}(\log t(n))/t(n)+1 \not\subseteq \mathcal{C}/\mathcal{F}$ appears in Balcázar, Hermo, Mayordomo (4) and Hermo, Mayordomo (13). Both of these papers use Komolgorov complexity arguments for their proofs. We next present some interesting consequences of this result. Let PREC denote the primitive recursive languages. PREC is recursively presentable so the next corollary follows from the above theorem.

COROLLARY 5. Fix $k \in \mathbb{N}$. Then $\mathsf{DLOGTIME}/\mathsf{poly}_{k+1} \not\subseteq \mathsf{PREC}/\mathsf{poly}_k$. Hence, one also has that $\mathsf{DLOGTIME}/\mathsf{poly}_{k+1}$ contains a language not in any of the classes $\mathsf{NEXP}/\mathsf{poly}_k$, $\mathsf{P}/\mathsf{poly}_k$, and $\mathsf{L}/\mathsf{poly}_k$.

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

Let $\mathsf{SIZE}(t(n))$ denote those languages computed by fan-in 2, AND, OR, NOT circuits of size $O(t(n))$. The use of $O(t(n))$ rather than $t(n)$ ensures that the class would remain stable if we chose a different basis rather than AND, OR, NOT. Let AC^0-$\mathsf{SIZE}(t(n))$ denote those languages computed by constant-depth, unbounded fan-in AND, OR, NOT circuits of size $O(t(n))$. Let $\alpha(n)$ be a nondecreasing, unbounded function which is $o(s(n))$. As we will see below, Theorem 5 also entails $\mathsf{SIZE}(s(n)) \subsetneq \mathsf{SIZE}(\alpha(n)s(n)\log^2 s(n))$. Such circuit hierarchies have been noted before. Kannan (15) shows by counting that there is a circuit of size $3n^{2k+2}$ not computed by any circuit of size n^k, a result which is weaker than ours and is basis specific since it does not incorporate a big-O in the definition of size. Also, in the fixed basis, non big-O setting, Paterson and Wegener (19) show if $s(n)$ is of growth rate less than $C_{\{\wedge,\vee,\neg\}}(B_{n-1})$, then there is a circuit of size $s(n)+1$ that computes a boolean function not computable by a circuit of size $s(n)$. Here $C_{\{\wedge,\vee,\neg\}}(B_{n-1})$ is the AND, OR, NOT circuit size of the boolean function on $n-1$ variables which requires the greatest circuit size. They also show if $s(n)$ is smaller than $C_{\{\wedge,\vee,\neg\}}(B_n)$, there there is a circuit of size $s(n)+n$ that computes a boolean function not computable by a circuit of size $s(n)$. Using the first result for $t(n) < n$ and the second for larger $t(n)$'s, one can thus show if $s(n)$ is $o(t(n))$ and $t(n)$ is less than $C_{\{\wedge,\vee,\neg\}}(B_n)$ then $\mathsf{SIZE}(s(n))$ is strictly contained in $\mathsf{SIZE}(t(n))$. So Patterson and Wegener's results are stronger than what we will show. Nevertheless, as our result is easily obtained from the work so far, and generalizes easily to other nonuniform models, we present it anyway.

COROLLARY 7. Let $\alpha(n)$ be as above and assume $\mathsf{SIZE}(s(n))$ does not contain all languages. For $k > 0$, $\mathsf{SIZE}(\alpha(n) \cdot s(n)\log^2 s(n)) \supsetneq \mathsf{SIZE}(s(n))$ and AC^0-$\mathsf{SIZE}(\alpha(n) \cdot s(n)\log^2 s(n)) \not\subseteq \mathsf{SIZE}(s(n))$.

PROOF. First, note $\mathsf{SIZE}(2^n/n)$ does contain all languages, so we do need the assumption on $\mathsf{SIZE}(s(n))$. As mentioned earlier, it is known (see Vollmer (25)) that for $s(n) > n$, $\mathsf{SIZE}(s(n)) \subseteq \mathsf{DTIME}([s(n)]^2)/O(s(n)\log s(n))$. Next we note as $\alpha(n)$ is nondecreasing and unbounded, $\mathsf{DTIME}([s(n)]^2)/O(s(n)\log s(n))$ is contained in $\mathsf{DTIME}([s(n)]^2)/(\alpha(n)s(n)\log s(n) - 1)$. So by Theorem 5 there is a language in $\mathsf{DTIME}(\log(\alpha(n)\cdot s(n)\log s(n)))/\alpha(n)s(n)\log s(n)$ that is not in $\mathsf{SIZE}(s(n))$. Basically, the machine just queries the appropriate bit of the advice string. We could do this with a DNF circuit as follows: For each bit position of the advice string which is a '1', we AND together the $\log(\alpha(n)\cdot s(n)\log s(n)) + 1$ inputs which would be used in this query, and then OR over the at most $\alpha(n)\cdot s(n)\log s(n)$ such 1 values. This gives a circuit of size $O(\alpha(n)s(n)\log s(n)[\log\alpha(n) + \log s(n) + \log\log s(n) + 1])$. All of the terms in the square brackets are $O(\log s(n))$, so one has a circuit of size $O(\alpha(n)s(n)\log^2 s(n))$ as desired. As our circuit is a DNF, it also implies the second result of the corollary. □

The above proof only needs: (1) a primitive recursive procedure to evaluate a circuit of size $s(n)$ given the $O(s(n)\log s(n))$ sized encoding of it and an input, and (2) a circuit exists of size $O(\alpha(n)\cdot s(n)\log^2 s(n))$ which extracts the appropriate information from the advice string. 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 $s(n)$, denoted $ACC(s(n))$, (b) constant depth threshold circuits of size $s(n)$, denoted $\mathsf{TC}^0(s(n))$, (c) $\log^k s(n)$ depth, fan-in less than or equal to two, AND, OR, NOT circuits of size $s(n)$, denoted $\mathsf{NC}^k(s(n))$, (d) $\log^k s(n)$ depth, unbounded fan-in, AND, OR, NOT circuits of size $s(n)$, denoted $\mathsf{AC}^k(s(n))$, and (e) size $s(n)$ branching programs, denoted $\mathsf{BP}(s(n))$. Thus, one gets:

COROLLARY 8. Let $\mathcal{C}(s(n))$ denote one of $\mathsf{AC}^0(s(n))$, $\mathsf{ACC}(s(n))$, $\mathsf{TC}^0(s(n))$, $\mathsf{NC}^k(s(n))$, $\mathsf{AC}^k(s(n))$, $\mathsf{BP}(s(n))$. Let $\alpha(n)$ be as above and assume $\mathcal{C}(s(n))$ does not contain all languages. For $k > 0$, $\mathcal{C}(\alpha(n)s(n)\log^2 s(n)) \supsetneq \mathcal{C}(s(n))$.

8. Acknowledgements

The author thanks Eric Allender, Steve Homer, Nicholas Tran, Bin Fu, Lance Fortnow, and Ingo Wegener for useful discussions/e-mail exchanges. The author would also like to thank the anonymous referees and the editors for their valuable feedback.

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] J. Balcázar, M. Hermo, E. Mayordomo. Characterizations of logarithmic advice complexity classes. Proceedings of the IFIP 12th World Computer Congress (IFIP'92). J. Van Leeuwen IFIP transactions A-12. pp. 315–321. 1992.

[5] Jin-Yi Cai, Osamu Watanabe. On Proving Circuit Lower Bounds Against PH and Some Related Lower Bounds for Constant Depth Circuits. SIAM Journal of Computing. Vol. 33 Iss. 4. pp. 984–1009. 2004.

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

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

[8] 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.

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

[10] L. Hemaspaandra, A. Hoene, A. Naik, M. Ogiwara, A. Selman, T. Thierauf, and J. Wang. Nondeterministically Selective Sets. *International Journal of Foundations of Computer Science.* Vol. 6 No. 4. pp. 403–416. 1995.

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

[12] L. Hemaspaandra and L. Torenvliet. Theory of Semi-feasible Algorithms. Springer-Verlag. 2003.

[13] M. Hermo, E. Mayordomo. A note on polynomial size circuits with low resource-bounded Kolmogorov complexity. Mathematical Systems Theory. Vol. 27. pp. 247–356. 1994.

[14] 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.

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

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

[17] C. Papadimitriou. *Computational Complexity.* Addison-Wesley. 1994.

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

[19] M. S. Paterson, I. Wegener. Nearly optimal hierarchies for network and formula size. Acta Informatica. Vol. 23. 1986. pp. 217–221.

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

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

[22] 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.

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

[24] N. V. Vinodchandran A note on the circuit complexity of PP. Electronic Colloquium on Computational Complexity (ECCC). Vol. 56. 2004.

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

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

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

Manuscript received

Chris Pollett