Hierarchies Against Sublinear Advice

Citation for published version:
Fortnow, L & Santhanam, R 2014, 'Hierarchies Against Sublinear Advice' Electronic Colloquium on Computational Complexity (ECCC), vol. 21, pp. 171-182.

Document Version:
Peer reviewed version

Published In:
Electronic Colloquium on Computational Complexity (ECCC)

Electronic Colloquium on Computational Complexity, Report No. 171 (2014)

Hierarchies against Sublinear Advice

Lance Fortnow[1] and Rahul Santhanam[2]

[1] Georgia Institute of Technology `fortnow@cc.gatech.edu`
[2] University of Edinburgh `rsanthan@inf.ed.ac.uk`

Abstract. We strengthen the non-deterministic time hierarchy theorem of [5,15,18] to show that the lower bound holds against sublinear advice. More formally, we show that for any constants c and d such that $1 \leqslant c < d$, there is a language in $\mathsf{NTIME}(n^d)$ which is not in $\mathsf{NTIME}(n^c)/n^{1/d}$. The best known earlier separation [8] could only handle $o(\log(n))$ bits of advice in the lower bound.

We generalize our hierarchy theorem to work for other syntactic complexity measures between polynomial time and polynomial space, including alternating polynomial time with any fixed number of alternations. We also use our technique to derive an *almost-everywhere* hierarchy theorem for non-deterministic classes which use a sublinear amount of non-determinism, i.e., the lower bound holds on all but finitely many input lengths rather than just on infinitely many.

As an application of our main result, we derive a new lower bound for NP against NP-uniform non-deterministic circuits of size $O(n^k)$ for any fixed k. This result is a significant strengthening of a result of Kannan [12], which states that not all of NP can be solved with P-uniform circuits of size $O(n^k)$ for any fixed k.

1 Introduction

One of the fundamental questions in complexity theory is whether resource hierarchies exist, i.e., whether having more of a resource allows us to solve more computational problems. Hierarchies are known for many fundamental resources, including deterministic time [10,11], deterministic space [16] and non-deterministic time [5,15,18,7].

Hierarchy theorems yield the only unconditional separations we know against polynomial-time classes, and thus it is of interest to investigate how strong we can make these separations. Ideally, we would like the separations to work against *non-uniform* classes, not just uniform ones. The notion of *advice* allows us to interpolate between the uniform and the non-uniform settings, and then the question becomes how much advice we can handle in the lower bound when proving a hierarchy theorem.

This question is interesting for at least a couple of different reasons. First, the amount of non-uniformity in the lower bound is closely tied to the question of derandomization. If we could show that for any fixed k, there is a language in deterministic polynomial time which cannot be solved in deterministic time $O(n^k)$ with $O(n^k)$ bits of advice, we could conclude that every language in probabilistic polynomial time can be solved infinitely often in deterministic sub-exponential time, using the hardness-randomness tradeoffs of [13,3]. A similar derandomization result for the class MA follows from the assumption that there is a language in NP which cannot be solved in non-deterministic time $O(n^k)$ with $O(n^k)$ bits of advice.

Second, from a technical point of view, hierarchy theorems are used in many of the important separation results in complexity theory [2,6,17]. Improved hierarchy theorems open the way to stronger versions of these results.

The traditional proofs of hierarchy theorems yield only uniform lower bounds. However, the proof of the deterministic time hierarchy theorem [10,11] can easily be adapted to yield separations against $n - \omega(1)$ bits of advice. This adaptation exploits the closure of deterministic time under complementation.

The situation is very different for resources such as non-deterministic time which are not known to be closed under complementation. The best hierarchy theorem known for this case in terms of the advice handled by the lower bound is due to [9]. They adapt Zak's proof of the non-deterministic time hierarchy [18] to show that $\mathsf{NP} \not\subseteq \mathsf{NTIME}(n^c)/\log(n)^{1/2c}$ for any $c > 0$. Not much more can be expected of adaptations of classical proofs of the non-deterministic time hierarchy theorem [5,15,18]. Since such proofs consider exponentially

ISSN 1433-8092

many input lengths when diagonalizing against a single machine, they're incapable of handling advice more than $O(\log(n))$.

1.1 Our Results

Our main result is a significant improvement of the non-deterministic time hierarchy theorem in terms of the advice handled in the lower bound.

Theorem 1. *Let* $d \geqslant 1$ *and* $d' > d$ *be any constants, and let* t *be a time-constructible time bound such that* $t = o(n^d)$. *Then* $\mathsf{NTIME}(n^d) \not\subseteq \mathsf{NTIME}(t)/n^{1/d'}$.

Theorem 1 improves on known results handling advice in two respects. First, the amount of advice in the lower bound can be as high as $n^{\Omega(1)}$, in contrast to earlier results in which it was limited to be $O(\log(n))$. Second, the hierarchy is provably tight in terms of the time bounds, while earlier results handling advice could only separate $\mathsf{NTIME}(n^d)$ from $\mathsf{NTIME}(n^c)$ with advice, where $c < d$.

The ideas of the proof of Theorem 1 also enable us to make progress on another direction in which hierarchy theorems can be strengthened: showing that hierarchy theorems hold almost everywhere. By this we mean that the lower bound holds on all but finitely many input lengths, rather than just on infinitely many. While it is well-known that the deterministic time hierarchy theorem can be adapted to hold almost everywhere, it is a long-standing open problem whether this adaptation can be done for the non-deterministic hierarchy theorem. It is shown in [4] that any adaptation has to be non-relativizing.

We make progress on this question by showing that almost-everywhere hierarchies do hold for a very natural sub-class of non-deterministic time: non-deterministic time with bounded non-determinism. Given functions t and g, let $\mathsf{NTIMEGUESS}(t, g)$ denote the class of languages accepted by non-deterministic machines running in time $t(n)$ and using at most $g(n)$ non-deterministic bits on any input of length n. Note that most natural NP-complete problems, such as SAT and CLIQUE, belong to $\mathsf{NTIMEGUESS}(\mathsf{poly}(n), o(n))$. We show the following.

Theorem 2. *Let* $d > 1$ *be any constant, and let* t *be a time-constructible function such that* $t(n) = o(n^d)$. *Let* $g(n) = o(n)$ *be any function computable in time* $O(n)$. *Then* $\mathsf{NTIMEGUESS}(n^d, 2g) \not\subseteq \mathsf{i.o.NTIME}(t, g)$.

We are able to use Theorem 1 to derive a new circuit lower bound for NP, improving a 30-year old result of Kannan [12].

Theorem 3. *Let* $k > 1$ *be any constant.* NP *does not have* NP*-uniform non-deterministic circuits of size* $O(n^k)$.

Finally, we consider the question of whether Theorem 1 can be extended to complexity measures other than NTIME. We show that for a wide variety of complexity measures, including all the alternating time classes with a bounded number of alternations, the analogue of Theorem 1 holds. Since the statements of these results are somewhat technical, we refer the reader to Section 6.

1.2 Techniques

We now attempt to give some intuition for the ideas in our proofs.

Recall that we are attempting to give hierarchies for non-deterministic time where the upper bound is uniform, but the lower bound allows as large an amount of non-uniformity as possible. Tradtional proofs of uniform non-deterministic time hierarchy theorems [5, 15, 18] use the delayed diagonalization technique. We illustrate this technique through Zak's proof, which is arguably the simplest. Suppose we wish to define a non-deterministic machine M running in time n^d which diagonalizes against some non-deterministic machine M_i running in time $t = o(n^d)$. Rather than diagonalizing against M_i on some fixed input x depending on i as in the proof of the deterministic time hierarchy theorem [10, 11], we diagonalize against M_i on some interval I_i of input lengths, meaning that we are guaranteed M differs from M_i on some input of length in I_i. The interval I_i is of the form $[n_i, 2^{n_i^d}]$ for some n_i depending on i. The diagonalization proceeds via a "copying" mechanism. On an input x in I_i of length less than $2^{n_i^d}$, M on x simply simulates M_i on $x0$, accepting iff M_i accepts. On an input of the form $x0^{2^{n_i^d}-n_i}$, where $|x| = n_i$, M determines $M_i(x)$ by brute force search, accepting iff M_i rejects. By assumption on t and assuming n_i is large enough, M can be defined to run in time n^d on all inputs in I_i.

Assume, for the purpose of contradiction, that M and M_i define the same language. Then M and M_i agree on all inputs with lengths in I_i, which by the copying mechanism of M, implies that $M_i(x)$ agrees with $M_i(x0^j)$ for each x of length n_i and each $j \in [0, 2^{n_i^d} - n_i]$. But then M cannot agree with M_i on $x0^{2^{n_i^d}-n_i}$, as M on that input does the opposite of what M_i does on x. Note that we cannot guarantee that M differs from M_i on any specific input, merely that it differs from M_i on some input in I_i. Also note that the interval I_i is exponentially long. Intuitively, M bides its time for exponentially many input lengths, until it has enough resources to do the opposite of what M_i does on x.

With an appropriate choice of intervals I_i, the above argument yields a uniform hierarchy theorem. It was adapted by Fortnow, Santhanam and Trevisan [8] to show a hierarchy with advice, but the advice which the adaptation can handle is very low: $o(\log(n))$. The challenge in adapting Zak's argument to hierarchies against advice is that the argument uses exponentially many input lengths. This hurts us in two ways. First, using a naive copying argument requires an exponential amount of information (advice bits for all input lengths in the interval) to be encoded into the starting input x, which is impossible. This is dealt with in [8] by only using sub-logarithmically many polynomially spaced input lengths in an exponentially long interval, and "jumping" from one input length to a polynomially larger one during the copy phase. The cost paid for the way this issue is dealt with in [8] is that the time bounds in the hierarchy theorem are polynomially separated rather than just being asymptotically separated as in the proof of the uniform non-deterministic time hierarchy. There is also a second issue, which is that for Zak's form of delayed diagonalization to work, advice for the final input length in the interval must be encoded into x. This constrains the advice that can be handled in this argument to sub-logarithmic, as the final input length in the interval is exponentially larger than x.

This second issue is a bottleneck for all delayed diagonalization arguments using exponentially long intervals, which includes all the traditional arguments [5,15,18]. Recently, Fortnow and Santhanam [7] gave a new proof of the non-deterministic time hierarchy theorem, which unlike previous proofs, critically uses the definition of non-deterministic time using polynomial-time verifiability. This new argument has the benefit that it uses only a polynomially long interval, and is a natural starting point for an attempt to handle more advice in the non-deterministic time hierarchy.

Intuitively, rather than "copying along a line" as in Zak's argument, the Fortnow-Santhanam proof "copies down a tree". Suppose we wish to define a non-deterministic machine M running in time n^d which diagonalizes against some non-deterministic machine M_i running in time $t = o(n^d)$. We again define some interval I_i of input lengths for achieving this, but now $I_i = [n_i, n_i + n_i^d]$ is only polynomially long. For any input $y \in I$ of length less than $n_i + n_i^d$, M copies the behaviour of M_i on two different inputs of length one larger, by accepting iff both $M_i(x0)$ and $M_i(x1)$ accept. On input of the form $xw, |x| = n_i, |w| = n_i^d$, M simulates M_i on x with witness w and does the opposite. Thus this diagonalization phase actually use the non-deterministic nature of M_i, rather than simply doing brute force search. It is again easy to see that if I_i is chosen appropriately, M can be made to run in time n^d.

Now assume, for the purpose of contradiction, that M agrees with M_i on all inputs in M_i. If M_i accepts on x, then by the copying behaviour of M, M_i accepts on all inputs in the interval I. But this implies that for all candidate witnesses w of size n_i^d, M_i rejects on x with witness w, which is a contradiction, as M_i would then reject on x itself. The case where M_i rejects on x is argued similarly.

By using only a polynomially long interval, the argument above, which we term witness-based diagonalization, gives hope for handling a sub-polynomial amount of advice in the lower bound. However, there are again obstacles to adapting the argument to advice. Even if the argument uses a polynomially long interval, it still uses all input lengths within that interval. A naive adaptation of the argument would require advice for all these input lengths to be encoded into x, which would be impossible as the number of input lengths is larger than x.

We could try using jumps again, so that fewer input lengths within the interval are used. However, it is unclear how to do this with witness-based diagonalization, as every jump only contributes to one bit in the witness, and therefore with a small number of jumps, we are unable to build a witness which we can use in the diagonalization process at the last input length in the interval.

We solve the problem by hybridizing between delayed diagonalization and witness-based diagonalization. The idea is that witness-based diagonalization can be "simulated" within a single input length, namely the last input length in the interval. However, in order to perform this simulation, we need to copy from the first input length in the interval to the last one. This can be done using jumps again, but how we use jumps critically affects the parameters in the final hierarchy results. The fewer the jumps used, the more advice we can handle, but the larger the gap between the time upper bound and the time lower bound. We need to choose the jump mechanism appropriately to get an optimal tradeoff between the quality of the ensuing hierarchy theorem in terms of time bounds and the quality of the ensuing hierarchy theorem in terms of advice. This gets somewhat technical, but we are able to prove Theorem 1 using these ideas.

The proof of Theorem 1 still uses a polynomially long interval for diagonalization. Suppose we wish to prove an almost-everywhere hierarchy for non-deterministic time, i.e., a hierarchy theorem where the lower bound holds for almost all input lengths rather than for infinitely many lengths[3]. It is known [4] that this cannot be done in a relativizing way. We show in this paper that an almost-everywhere hierarchy *can* be obtained for a natural subclass of non-deterministic time, namely non-deterministic time with sub-linear witnesses. The key observation is that when the amount of non-determinism is sub-linear, a variant of the witness-based diagonalization argument can be carried out within a single input length, meaning that we can diagonalize against any fixed machine on *any* large enough input length. This yields an almost-everywhere hierarchy.

The proof of Theorem 3 is substantially different. It uses an indirect diagonalization technique due to [14], where the presumed existence of a simulation of a class C with weakly uniform circuits of fixed polynomial size is used multiple times to derive a simulation of C in a small amount of time with sub-linear advice, as long as the uniformity condition is in some sense stronger than the class C. We require a variant of this argument which uses a census technique, and then an application of Theorem 1 completes the proof.

For the extensions to other complexity measures, we abstract out the properties required of the complexity measure using the notion of *leaf languages*.

2 Preliminaries

2.1 Complexity Classes, Promise Problems and Advice

We assume a basic familiarity with complexity classes. The Complexity Zoo (which can be found at `http://qwiki.caltech.edu/wiki/ComplexityZoo`) is an excellent resource for basic definitions and statements of results.

We require some classes defined by simultaneous resource bounds. Let $t : \mathbb{N} \to \mathbb{N}$ be a time bound, and $g : \mathbb{N} \to \mathbb{N}$ be a bound on the amount of non-determinism used. The complexity class $\mathsf{NTIMEGUESS}(t, g)$ is the class of all languages L for which there is a non-deterministic machine M deciding L which runs in time $O(t(n))$ and uses at most $g(n)$ guess bits on any input of length n.

Given a complexity class C, coC is the class of languages L such that $\bar{L} \in \mathsf{C}$. Given a function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}(s)$ is the class of Boolean functions $f = \{f_n\}$ such that for each n, f_n has Boolean circuits of size $O(s(n))$. Given a language L and an integer n, $L_n = L \cap \{0,1\}^n$. Given a class C, $\mathsf{i.o.C}$ is the class of languages L for which there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for infinitely many length n.

In order to deal with promise classes in a general way, we take as fundamental the notion of a complexity measure. A complexity measure CTIME is a mapping which assigns to each pair (M, x), where M is a time-bounded machine (here a time function $t_M(x)$ is implicit) and x an input, one of three values "0" (accept), "1" (reject) and "?" (failure of CTIME promise). We distinguish between *syntactic* and *semantic* complexity

---

[3] Note that this notion of almost-everywhere separations is different from the related notion considered by [1], who give a negative relativization result

measures. Syntactic measures have as their range $\{0,1\}$ while semantic measures may map some machine-input pairs to "?". The complexity measures DTIME and NTIME are syntactic (each halting deterministic or non-deterministic machine either accepts or rejects on each input), while complexity measures such as BPTIME and MATIME are semantic (a probabilistic machine may accept on an input with probability $1/2$, thus failing the bounded-error promise). For syntactic measures, any halting machine defines a language, while for semantic measures, only a subset of halting machines define languages.

Let $t : \mathbb{N} \to \mathbb{N}$ be a time function, and $a : \mathbb{N} \to \mathbb{N}$ be an advice function. A language L is in $\mathsf{CTIME}(t)/a$ if there is a machine M halting in time $t(\cdot)$ taking an auxiliary *advice* string of length $a(\cdot)$ such that for each n, there is some advice string $b_n, |b_n| = a(n)$ such that M fulfils the CTIME promise for each input x with advice string b_n and accepts x iff $x \in L$.

For syntactic classes, a lower bound with advice translates to a lower bound for the class itself.

We will need standard notions of uniformity for circuits. The *direct connection language* for a sequence of circuits $C = \{C_n\}$, where C_n is on n input bits, is the language L_C consisting of all tuples of the form $\langle 1^n, g, h, r\rangle$, where g and h are indices of gates, r is the *type* of g (AND/OR/NOT/INPUT, and in case of INPUT, which of the n input bits g is, with an additional bit to specify whether g is the designated output gate), and h is a gate feeding in to g in case the type r is not INPUT. Other encodings of the direct connection language are of course possible, but our results are insensitive to the details of the encoding.

Given a class $\mathcal{C}$ of languages and a function $s : \mathbb{N} \to \mathbb{N}$, a language L is said to have $\mathcal{C}$-uniform circuits of size $s(n)$ if there is a size-$s(n)$ circuit family $\{C_n\}$ such that its *direct connection language* is computable in $\mathcal{C}$. By a *description of a circuit* C_n, we mean the list of tuples in L_C corresponding to gates in C_n.

3 Hierarchies for Non-deterministic Time against Sublinear Advice

In this section, we prove the following general theorem, and then show how it implies Theorem 1.

As described in the Introduction section, the proof involves a hybrid of delayed diagonalization and witness-based diagonalization. We think of the diagonalization as proceeding in two phases: the jump phase where copying occurs, and the witness-gathering phase where the witness is built and witness-based diagonalization is performed.

We need some preliminary notation. Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that $f(n)$ is computable in $O(\mathsf{polylog}(n))$ time and $f(n) > n$ for all n. We will use f to parameterize the jumps in the diagonalization. Given a time function t_1, for any n, let $g(n)$ be the minimum i such that $f^{(g(n))}(n) \geqslant n + 2t_1(n) + 2$. Note that for each $n, g(n)$ exists, using the monotonicity of f. For a string w of length r, we define $Enc(w)$ to be the $2r$-bit string whose even bits are all 0, and whose i'th odd bit is the i'th bit of w, for each $i \in [r]$.

Theorem 4. *Let t_1 and t_2 be increasing time-constructible functions, with $t_1, t_2 = \Omega(n)$. Let $f, g : \mathbb{N} \to \mathbb{N}$ be functions as defined above, and let $a : \mathbb{N} \to \mathbb{N}$ be an advice function such that $a(n)$ is computable in time $O(\mathsf{polylog}(n))$. Suppose $n = \sum_{l=0}^{g(n)} a(f^{(l)}(n)) + \omega(1)$, and $t_1(f(m)) + g(m)\mathsf{polylog}(m) = o(t_2(m))$. Then $\mathsf{NTIME}(t_2) \not\subseteq \mathsf{NTIME}(t_1)/a$*

Proof. Define a non-deterministic machine M as follows. On input x of length m, M first calculates $t_2(m)$. It then tries to decompose $x = 1^i01^j0z110^k$, where $i, j > 0$, $k \geqslant 0$, $z \in \{0,1\}^*$. Note that such a decomposition is unique if it exists. If M succeeds in finding such a decomposition, it sets $n = i + j + |z| + 4$, and checks if $m = f^l(n)$ for some $0 \leqslant l \leqslant g(n)$, and if $|z| \geqslant \sum_{l=0}^{g(n)} a(f^{(l)}(n))$. This check can be done in time at most $g(n)\mathsf{polylog}(n)$ and hence time at most $g(m)\mathsf{polylog}(m)$, by assumption on f and g. If this check doesn't succeed, M rejects. If it succeeds, there are two cases: $l < g(n)$ and $l = g(n)$. In the first case, M decomposes $z = z_0z_1 \ldots z_{l+1}z'$, where for each $i, 0 \leqslant i \leqslant l+1, |z_i| = a(f^{(i)}(n))$ and $z' \in \{0,1\}^*$. Note that by assumption on n and a, such a decomposition can be performed for n large enough - if it cannot be performed, M halts and rejects. M simulates M_i on $x0^{f(m)-m}$ with advice z_{l+1}, accepting iff M_i accepts. In the second case, where $l = g(n)$, M decomposes $z = z_0z_1 \ldots z_lz'$, where for each $i, 0 \leqslant i \leqslant l, |z_i| = a(f^{(i)}(n))$ and $z' \in \{0,1\}^*$. Note that by assumption on n and a, such a decomposition can be performed for n large enough - if it cannot be performed, M halts and rejects. It also calculates $q = k - 2t_1(n) - 2$. Note that q is non-negative by the assumptions on f and g. M simulates M_i on $1^i01^j0z11Enc(0^{t_1(n)}1)0^q$ with advice z_l, accepting iff M_i accepts. Throughout M maintains an internal clock, and if it detects that it has been running for more than $t_2(m)$ steps after the calculation of $t_2(m)$, it halts and rejects.

The operation of M above corresponds to the jump phase.

Now suppose M does not succeed in finding a decomposition as above. It then tries to decompose $x = 1^i01^j0z11Enc(0^s1w)0^q$, where $i, j > 0$, $s, q \geqslant 0$, $z, w \in \{0,1\}^*$ and moreover, setting $n = i + j + |z| + 4$, the conditions that $m = f^{(g(n)}(n)$ and $|z| \geqslant \sum_{l=0}^{g(n)} a(f^{(l)}(n))$ are satisfied. Note that such a decomposition is unique if it exists. If this decomposition attempt fails, M halts and rejects. If it succeeds, M decomposes $z = z_0z_1 \dots z_lz'$, where for each $i, 0 \leqslant i \leqslant l, |z_i| = a(f^{(i)}(n))$ and $z' \in \{0,1\}^*$. Note that by assumption on n and a, such a decomposition can be performed for n large enough - if it cannot be performed, M halts and rejects. Now there are two cases: $s > 0$ and $s = 0$. In the first case, M simulates M_i on $1^i01^j0z11Enc(0^{s-1}1w0)0^q$ with advice z_l and $1^i01^j0z11Enc(0^{s-1}1w1)0^q$ with advice z_l, accepting iff both computations accept. In the second case, M simulates M_i on 1^i01^j0z11 with non-deterministic sequence w and advice z_0, rejecting iff M_i accepts. Throughout M maintains an internal clock, and if it detects that it has been running for more than $t_2(m)$ steps after the calculation of $t_2(m)$, it halts and rejects.

The operation of M above corresponds to the witness-gathering phase.

By definition of M, it halts in time $O(t_2(m))$. Moreover, using the various assumptions on computability of f, a, t_1, t_2, all the checks and calculations of M, as well as the final simulation step, can be completed in time $O(t_2(m))$ for m large enough.

We now proceed to show that $L(M) \notin \mathsf{NTIME}(t_1(m))/a(m)$. Suppose, to the contrary, that M_i is a non-deterministic advice taking machine accepting $L(M)$ using $a(m)$ bits of advice. We derive a contradiction.

Choose j and n large enough so that all the checks, calculations and simulation of M can be completed in time $O(t_2(m))$ for any m such that there is an input of length m which can be successfully decomposed with the corresponding n and j, and so that $n > \sum_{l=0}^{g(n)} a(f^{(l)}(n))$. Let $z_0, z_1, \dots z_{g(n)}$ be the correct advice strings for M_i at lengths $n, f(n) \dots f^{g(n)}(n)$, and let $z = z_0z_1 \dots z_{g(n)}$. Consider the input $x = 1^i01^j0z11$. By assumption, M on x agrees with M_i on x with advice z_0 (since $|x| = n$). By the behaviour of M in the jump phase, we have that M on $x0^{f^i(n)-n}$ agrees with M_i on $x0^{f^i(n)-n}$ with advice z_i, for each $i \in [0, g(n)]$. By the behaviour of M in the witness-gathering phase, we have that M accepts $x0^{f^i(n)-n}$ iff M accepts $xEnc(0^s1w)0^q$ for each $s, 0 \leqslant s \leqslant t_1(n)$, w of length $t_1(n) - s$ and $q = m - n - 2t_1(n) - 2$ iff M_i accepts $xEnc(0^s1w)0^q$ with advice $z_{g(n)}$ for each $s, 0 \leqslant s \leqslant t_1(n)$, w of length $t_1(n) - s$ and $q = m - n - 2t_1(n) - 2$. But for each w of length $t_n(n)$, again by the behaviour of M in the witness-gathering phase, M accepts $xEnc(1w)0^q$, $q = m - n - 2t_1(n) - 2$ iff M_i rejects x with witness w and advice z_0. This happens iff M_i rejects x with advice z_0, which is a contradiction to the assumption that M on x agrees with M_i on x with advice z_0.

We now show how to derive Theorem 1 from the more general Theorem 4 above. This allows us to get the "best of both worlds" for non-deterministic time hierarchies with advice: time bounds only asymptotically separated, and advice in the lower bound which is $n^{\Omega(1)}$.

Proof of Theorem 1. Apply Theorem 4 with $t_1 = n^d, t_2 = t, f(n) = 2n, a(n) = n^{1/e}$. In this case, $g(n) = O(\log(n))$, and it can be checked easily that the conditions on f, g, a in terms of t_1, t_2, n all hold. The theorem follows. ☐

4 An Almost-everywhere Hierarchy Theorem

Ideally, we would like to prove almost-everywhere hierarchy theorems, i.e., show that reducing the amount of time available makes languages harder to compute on all but finitely many input lengths. Almost-everywhere hierarchy theorems are known for classes closed under complementation such as deterministic time and deterministic space, but not for non-deterministic time. It is shown in [4] that there is an oracle relative to which $\mathsf{NEXP} \subseteq \mathsf{i.o.NP}$, therefore non-standard techniques would be required even to show an almost-everywhere separation of NEXP from NP.

We consider non-deterministic classes with sub-linear non-determinism, i.e., the non-deterministic machine is allowed to use only $o(n)$ non-deterministic bits. These classes contain most commonly studied problems in NP including $SAT, CLIQUE, VC$ etc. when the input is encoded in the standard way. Thus showing an almost-everywhere hierarchy for such classes is of interest.

The following theorem immediately implies Theorem 2.

Theorem 5. *Let $g(n) = o(n)$ be any sub-linear function computable in time $O(n)$. Let t_1 and t_2 be time-constructible functions such that $n \leqslant t_1 = o(t_2)$. Then* $\mathsf{NTIMEGUESS}(t_2, 2g(n)) \not\subseteq \mathsf{i.o.NTIMEGUESS}(t_1, g(n))$.

Proof. Define a non-deterministic machine M as follows. On input x of length n, M first tries to decompose $x = 1^i01^k0z$, where $i, k \geqslant 0$. If x cannot be decomposed in this manner, or if it can but $|z| > g(n)$, M immediately rejects. If $|z| = g(n)$, M runs the non-deterministic Turing machine M_i on $1^i01^{n-i-2}0$ for at most $t_2(n)$ steps, using z as the sequence of guess bits for the simulation of the machine. If the machine M_i does not halt within time $t_2(n)$, or if it uses more than $g(n)$ guess bits, M rejects. Otherwise, it does the opposite of M_i, accepting if M_i rejects and rejecting otherwise.

If $|z| < g(n)$, M runs M_i on $x_1 = 1^i01^{k-1}00z$ and $x_2 = 1^i01^{k-1}01z$, accepting iff both simulations halt and accept within time $t_2(n)$, and each uses at most $g(n)$ guess bits.

M runs in time $O(t_2(n))$ and uses at most $2g(n)$ guess bits on any input of length n. We show that $L(M) \notin \mathsf{i.o.NTIMEGUESS}(t_1(n), g(n))$.

Suppose, to the contrary, that $L(M) \in \mathsf{i.o.NTIMEGUESS}(t_1(n), g(n))$, and let M_i be a non-deterministic machine running in time $ct_1(n)$ for some constant c, and with $g(n)$ guess bits, such that $L(M_i)$ coincides with $L(M)$ on infinitely many input lengths. Let I be an infinite set of input lengths such that $L(M_i)$ coincides with $L(M)$ on each input length in I. Choose $n \in I$ large enough such that M can complete its simulations of M_i on all inputs of length n of the form 1^i0y for some y. That such an n exists follows from the facts that $n \leqslant t_1(n) = o(t_2(n))$.

By the assumption that M agrees with M_i on length n, we have that M_i accepts $1^i01^{n-i-2}0$ iff M accepts $1^i01^{n-i-2}0$ iff M_i accepts $1^i01^{n-i-3}00$ and $1^i01^{n-i-3}10$... Continuing inductively, we have that M_i accepts $1^i01^{n-i-2}0$ iff M accepts all strings of the form $1^i01^{n-g(n)-i-2}0z$ iff M_i does not accept on $1^i01^{n-i-2}0$ for any guess sequence z of length $g(n)$. But then we have that M_i accepts $1^i01^{n-i-2}0$ iff M_i does not accept $1^i01^{n-i-2}0$, which is a contradiction.

By combining the ideas in the proof of Theorem 5 with the ideas of the proof of Theorem 4, we get the following almost-everywhere hierarchy against advice. We omit the proof because it contains no new ideas beyond those in the proofs of Theorem 5 and Theorem 4.

Theorem 6. *Let $a : \mathbb{N} \to \mathbb{N}$ be an advice function and $g : \mathbb{N} \to \mathbb{N}$ a guess function, both computable in time $O(n)$, such that $a(n) + g(n) = n - \omega(1)$. Then for any time-constructible functions t_1 and t_2 such that $n \leqslant t_1 = o(t_2)$,* $\mathsf{NTIMEGUESS}(t_2, 2g) \not\subseteq \mathsf{i.o.NTIMEGUESS}(t_1, g)/a$.

5 A Lower Bound against Weakly Uniform Circuits

While it is a major open problem to show that NP does not have linear size circuits, one could hope to show lower bounds when there is some uniformity condition on the circuits. A result of this form was shown by [12].

Theorem 7. *[12] For every k,* NP *does not have* P*-uniform circuits of size $O(n^k)$.*

We strengthen this lower bound in two ways. First, we allow the circuits to be NP-uniform rather than P-uniform. Second, we allow the circuits to be non-deterministic rather than deterministic. The following is a re-statement of Theorem 3.

Theorem 8. *For every $k > 1$,* NP *does not have* NP*-uniform non-deterministic circuits of size $O(n^k)$.*

Proof. Assume NP has NP-uniform non-deterministic circuits of size $O(n^k)$. Let $L \in \mathsf{NP}$ be arbitrary. We will show that L can be simulated in non-deterministic time n^{2k+2} with $n^{1/(4k)}$ bits of advice, which will yield a contradiction to Theorem 4 when $t_2 = n^{4k}$ and $t_1 = n^{2k+2}$.

By assumption, L has non-deterministic circuits of size $O(n^k)$, so there is a non-deterministic circuit family $\{C_n\}$ for L of size at most $c \cdot n^k$ for some constant c. Furthermore, by NP-uniformity, the direct connection language L_{dc} for $\{C_n\}$ (see Section 2 for the definition) is in NP. We consider a "succinct" version L_{succ} of the language L_{dc}, defined as follows. Letting $Bin(n)$ be the binary representation of n, define

$$L_{succ} = \{\langle Bin(n)01^{\lceil n^{1/(5k^2)} \rceil}, g, h, r\rangle \mid \langle 1^n, g, h, r\rangle \in L_{dc}\}.$$

Intuitively, L_{succ} is an "unpadded" version of L_{dc}.

Observe that $L_{succ} \in \mathsf{NP}$. Given an input y for L_{succ}, our non-deterministic polynomial-time algorithm first checks if y can be parsed as a "valid" tuple $\langle z, g, h, r\rangle$, where $z = Bin(n)01^{\lceil n^{1/(5k^2)} \rceil}$ for some positive integer n, g and h are valid gate indices between 1 and $c \cdot n^k$, and r is a valid gate type. If this check fails, *reject*. Otherwise, the algorithm runs the non-deterministic polynomial-time machine deciding L_{dc} on $\langle 1^n, g, h, r\rangle$, and *accepts* if and only if this machine accepts. Note that this algorithm for L_{succ} runs in time polynomial in $|y|$, since we only simulate the machine for L_{dc} when $n^{1/(5k^2)} \leqslant |y| \leqslant n$ and the machine for L_{dc} runs in time polynomial in n.

Now we apply the assumption that NP has NP-uniform circuits of size $O(n^k)$ for a second time. Since $L_{succ} \in \mathsf{NP}$, there is a non-deterministic circuit family $\{D_m\}$ of $O(m^k)$ size for L_{succ}. Given an integer n, let $m(n)$ be the least integer such the size of the tuple $\langle Bin(n)01^{\lceil n^{1/(5k^2)} \rceil}, g, h, r\rangle$ is at most $m(n)$ for any valid gate indices g and h for C_n and any valid gate type r. Using a standard encoding of tuples, we can assume, for large enough n, that $m(n) \leqslant n^{1/(4.5k^2)}$, since g, h, r can all be encoded with $O(\log n)$ bits each.

We now describe a simulation of L in time $O(n^{2k+2})$ with $n^{1(4k)}$ bits of advice. Let M be an advice-taking machine which operates as follows. On input x of length n, M receives an advice string of length $O(n^{1/4k})$. It interprets this advice as consisting of two parts: the description of a non-deterministic circuit D_m for the language L_{succ} on inputs of length $m(n) \leqslant n^{1/(4.5k^2)}$, and an $O(\log(n))$ bit string representing the census value, i.e., the number of inputs in L_{succ} of that length. For every possible pair of gate indices g and h of C_n and every possible gate type r, M simulates the circuit D_m on $\langle Bin(n)01^{\lceil n^{1/(5k^2)} \rceil}, g, h, r\rangle$ to decide whether gate h is an input to gate g and whether the type of gate g is r. Each such simulation can be done in time $O(n^{1/2k})$, as the size of D_m is $O(n^{1/4k})$. There are at most $O(n^{2k+1})$ such simulations that M performs, since there are at most that many relevant triples $\langle g, h, r\rangle$. Note that since the circuit D_m is non-deterministic, M cannot know for sure the answer to a given simulation. Instead, it performs all the simulations and then checks that the number of YES answers is equal to the census value encoded in the advice string. In such a case, it knows that the answers to all simulations are correct; otherwise, it rejects.

In the case where answers to all simulations are correct, M has a full description of the non-deterministic circuit C_n. It simulates C_n on x, and accepts if and only if $C_n(x)$ outputs 1. This simulation can be done in time $O(n^{2k})$ since the circuit C_n is of size $O(n^k)$. The total time taken by M is $O(n^{2k+2})$, and M uses $O(n^{1/4k})$ bits of advice. By our assumptions on C_n and D_m, the simulation is correct. Thus $L \in \mathsf{NTIME}(n^{2k+2})/O(n^{1/4k})$.

However, as $L \in \mathsf{NP}$ was chosen to be *arbitrary*, we have $\mathsf{NP} \subseteq \mathsf{NTIME}(n^{2k+2})/O(n^{1/4k})$, which for $k > 1$ contradicts Theorem 4.

The idea of using advice reduction in the proof of Theorem 8 is inspired by a result of Santhanam and Williams [14], who generalized Theorem 7 in a different direction, by showing that for any k, P does not have P-uniform circuits of size $O(n^k)$. The additional ingredients in the proof of Theorem 8 are the use of Theorem 4, as well as the use of a census technique to deal with NP-uniformity.

6 Generalizing to Other Syntactic Classes

In this section we show how to generalize Theorem 4. We first show how to generalize the robustly-often time hierarchy of [7], and then sketch how to use the ideas of the proof to generalize Theorem 4.

First, we define robustly-often simulations.

Let S be a subset of positive integers. S is robust if for each positive integer k, there is a positive integer $m \geqslant 2$ such that $n \in S$ for all $m \leqslant n \leqslant m^k$.

Let L be a language, C a complexity class, and S a subset of the positive integers. We say $L \in \mathsf{C}$ on S if there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for any $n \in S$.

Given a language L and complexity class C, $L \in \mathsf{r.o.C}$ if there is a robust S such that $L \in C$ on S. In such a case, we say that there is a robustly-often ($\mathsf{r.o.}$) simulation of L in C. We extend this notion to complexity classes in the obvious way - given complexity classes B and C, $\mathsf{B} \subseteq \mathsf{r.o.C}$ if there for each language $L \in \mathsf{B}$, $L \in \mathsf{r.o.C}$.

Now we describe a general framework in which we can show robustly-often hierarchies and hierarchies with sub-linear advice.

Let N be a nondeterministic polynomial-time Turing machine where on input x of length n, $N(x)$ has $2^{p(n)}$ computation paths indexed by strings $z \in \{0,1\}^{p(n)}$. We can also think of z as representing an integer between 1 and $2^{p(n)}$ in a standard way.

Define $\mathsf{OUTPUT}(N,x)$ to be the string w of length $2^{p(n)}$ such that zth bit of w is 1 if $N(x)$ accepts on the path indexed by z and 0 otherwise.

Let $A \subseteq \Sigma^*$. We define the class $\mathsf{LEAF}(A)$ as the class of languages L such that for some nondeterministic polynomial-time Turing machine N, $x \in L$ if and only if $\mathsf{OUTPUT}(N,x) \in A$. For example if A is the set of strings with at least one 1 then $\mathsf{LEAF}(A) = \mathsf{NP}$. We can also define $\mathsf{LEAFTIME}(A,t)$ where we restrict N to run in time $O(t)$.

We say a class $\mathcal{C}$ is closed under linear-time monotone 2-query transductions if for every language $L' \in \mathcal{C}$ and every deterministic linear-time oracle machine O making at most 2 queries to its oracle and outputting a monotone function of the answers to the queries, $L(O^{L'}) \in \mathcal{C}$. This definition might seem involved, but in fact any natural complexity arising from a leaf language satisfies this property, eg., the levels of the polynomial-time hierarchy.

We can generalize the robustly-often hierarchy for non-deterministic time [7] as follows.

Theorem 9. *Suppose A is computable by a family of $\mathsf{DLOGTIME}$-time uniform NC^1 circuits. If t_1 and t_2 are functions such that t_1 is time-constructible and*

- $t_1(n+1) = o(t_2(n))$,
- $n \leqslant t_1(n) \leqslant n^c$ *for some constant c, and*
- $\mathsf{LEAFTIME}(A, t_1(n))$ *is uniformly closed under linear-time monotone 2-query transductions,*

then $\mathsf{LEAFTIME}(A, t_2(n)) \not\subseteq \mathsf{r.o.LEAFTIME}(A, t_1(n))$.

Corollary 1. *Let $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ be functions such that t_1 is time-constructible and $t_1(n+1) = o(t_2(n))$. For every integer $k \geqslant 1$, $\Sigma_k - \mathsf{TIME}(t_2) \not\subseteq \mathsf{r.o.}\Sigma_k - \mathsf{TIME}(t_1)$, and $\Pi_k - \mathsf{TIME}(t_2) \not\subseteq \mathsf{r.o.}\Pi_k - \mathsf{TIME}(t_1)$.*

Proof (Proof of Theorem 9).

Without loss of generality assume A is computed by fan-in 2 circuits where every path has length $d \log n$ and negations are only on the inputs.

Let $M_1, M_2, \ldots$ be an enumeration of multitape nondeterministic machines that run in time $t_1(n)$. For an input x of length n, $\mathsf{OUTPUT}(M_i, x)$ will have length $2^{t_1(n)}$ and the circuit C used to determine if $\mathsf{OUTPUT}(M_i, x)$ is in A will have depth $dt_1(n)$. C has $2^{t_1(n))}$ inputs which we express as y_z for $z \in \{0,1\}^{t_1(n)}$.

Define a nondeterministic Turing machine M that on input 1^i01^m0w does as follows:

- If $|w| < dt_1(i+m+2)$ consider the gate g that is reached in C by following the path w
  - If g is an OR gate then accept if both $M_i(1^i01^m0w0)$ and $M_i(1^i01^m0w1)$ accepts.
  - If g is an AND gate then accept if either $M_i(1^i01^m0w0)$ or $M_i(1^i01^m0w1)$ accepts.
- If $|w| = dt_1(i+m+2)$ consider the input variable y_z.
  - If the variable is not negated then accept if $M_i(1^i01^m0)$ rejects on the path specified by z.
  - If the variable is negated then accept if $M_i(1^i01^m0)$ accepts on the path specified by z.

Since we can universally simulate $t(n)$-time nondeterministic multitape Turing machines on an $O(t(n))$-time 2-tape nondeterministic Turing machine and $\mathsf{LEAFTIME}(A, t_1)$ is closed under linear-time monotone 2-query transductions, $L(M) \in \mathsf{LEAFTIME}(A, O(t_1(n+1))) \subseteq \mathsf{LEAFTIME}(A, t_2(n))$.

Suppose $\mathsf{LEAFTIME}(A, t_2(n)) \subseteq \mathsf{r.o.LEAFTIME}(A, t_1(n))$. Pick a C such that $dt_1(n) \ll n^c$ for all n large enough. By the definition of $\mathsf{r.o.}$ there is some n_0 and a language $L \in \mathsf{LEAFTIME}(t_1(n))$ such that $L(M) = L$ on all inputs of length between n_0 and n_0^C. Fix i such that $L(x) = A(\mathsf{OUTPUT}(M_i, x))$ with $n_0 \leqslant |x| \leqslant n_0^C$. Then $z \in L(M_i) \Leftrightarrow z \in L(M)$ for all $z = 1^i01^{n_0}0w$ for $w \leqslant t_1(i + n_0 + 2)$.

By induction on the gates $M_i(1^i01^{n_0}0)$ accepts iff $C(\mathsf{OUTPUT}(M_i, 1^i01^{n_0}0))$ outputs false and thus iff $\mathsf{OUTPUT}(M_i, 1^i01^{n_0}0)$ is not in A. This contradicts our assumption that $L(1^i01^{n_0}0)) = A(\mathsf{OUTPUT}(M_i, 1^i01^{n_0}0))$.

We can combine the proofs of Theorem 9 and Theorem 4 to generalize Theorem 1 for $\mathsf{LEAFTIME}$.

Theorem 10. *Suppose A' is computable by $\mathsf{DLOGTIME}$-time uniform NC^1 circuits. Let $d \geqslant 1$ and $e > d$ be arbitrary constants. If t_1 is a time-constructible function such that*

- $t_1(n) = o(n^d)$,
- $\mathsf{LEAFTIME}(A', t_1(n))$ *is closed under linear time monotone 2-query transductions,*

then $\mathsf{LEAFTIME}(A', n^d) \not\subseteq \mathsf{LEAFTIME}(A', t_1(n))/n^{1/e}$.

Proof Sketch. We show how to modify the proof of Lemma 4 for LEAFTIME.

The jump phase will remain exactly the same. In the witness gathering phase, we need to change things a little. The string w obtained from a successful decomposition of the input x in the witness-gathering phase will now correspond to a path in the circuit C accepting the leaf language which determines the answer of M_i on x. Again, we will assume without loss of generality that C is a balanced logarithmic-depth circuit, where all input-output paths are of the same length. There are two cases: w encodes a maximum-length path in C, or it does not. In the former case, let g be the gate that is reached following the path described by w. If g is an OR gate, then M simulates M_i on $1^i01^j0z11Enc(0^{s-1}1w0)0^q$ with advice z_l and $1^i01^j0z11Enc(0^{s-1}1w1)0^q$ with advice z_l, accepting iff both computations accept. If g is an AND gate, M simulates M_i on $1^i01^j0z11Enc(0^{s-1}1w0)0^q$ with advice z_l and $1^i01^j0z11Enc(0^{s-1}1w1)0^q$ with advice z_l, accepting iff either computation accepts. If w encodes a maximum-length path, let y_z be the variable pointed to by w, where z is a witness for M on x. If y_z is un-negated, M does the opposite of M_i on x using witness z with advice z_0, and if y_z is negated, M does the same as M_i on x using witness z with advice z_0.

We now get a contradiction following an argument similar to the proof of Theorem 4. □

Corollary 2. *For any reals $1 \leqslant r < s$ and every integer $k \geqslant 1$, $\Sigma_k - \mathsf{TIME}(n^s) \not\subseteq \Sigma_k - \mathsf{TIME}(n^r)/n^{1/s}$ and $\Pi_k - \mathsf{TIME}(n^s) \not\subseteq \Pi_k - \mathsf{TIME}(n^r)/n^{1/s}$.*

7 Acknowledgments

We thank an anonymous referee for pointing out an error in a previous version of this paper.

References

1. Eric Allender, Richard Beigel, Ulrich Hertrampf, and Steven Homer. Almost-everywhere complexity hierarchies for nondeterministic time. *Theoretical Computer Science*, 115(2):225–241, 19 July 1993.
2. Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing*, 23(5):1026–1049, 1994.
3. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.
4. Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Proceedings of 36th International Colloquium on Automata, Languages and Programming*, pages 195–209, 2009.

5. Stephen Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192, Denver, Colorado, 1–3 May 1972.
6. Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52(6):833–865, 2005.
7. Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, pages 569–580, 2011.
8. Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Promise hierarchies. *Electronic Colloquium on Computational Complexity(ECCC)*, 11(98), 2004.
9. Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 2005.
10. Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.
11. Frederick Hennie and Richard Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.
12. Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.
13. Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.
14. Rahul Santhanam and Ryan Williams. On medium-uniformity and circuit lower bounds. In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity*, pages 15–23, 2013.
15. Joel Seiferas, Michael Fischer, and Albert Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, January 1978.
16. Richard Stearns, Juris Hartmanis, and Philip Lewis. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190. IEEE, 1965.
17. Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of 26th Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.
18. Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, October 1983.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de