Electronic Colloquium on Computational Complexity, Report No. 54 (2005)

Time Hierarchies for Computations with a Bit of Advice

Konstantin Pervyshev
Department of Mathematics and Mechanics
St. Petersburg State University
E-mail: tpc@mail.ru

19th May 2005

Abstract

A polynomial time hierarchy for **ZPTime** with one bit of advice is proved. That is for any constants a and b such that $1 < a < b$, $\mathbf{ZPTime}[n^a]/1 \subsetneq \mathbf{ZPTime}[n^b]/1$.

The technique introduced in this paper is very general and gives the same hierarchy for **NTime**$\cap$**coNTime**, **UTime**, **MATime**, **AMTime** and **BQTime**. It also significantly simplifies the previously known proofs of hierarchies for **BPTime** and **RPTime** with advice.

1 Introduction

1.1 Time Hierarchies

It is believed that time is the most important computational resource. But is that so? Does a computer given more time solve harder problems?

One can easily construct a function $f : \{0,1\}^* \to \{0,1\}^*$ computable in time $O(n^3)$ not computable in time $O(n^2)$ where n is the length of an argument. Just let the value of the function $f(x)$ be 1^{n^3}, a binary string formed of n^3 symbols "1", where n is the length of x. The obtained result, however, is not satisfactory because it simply tells that a computer performs more operations in time $O(n^3)$ than in time $O(n^2)$.

Let us restate the question. Is that true that there exists a predicate $\chi_L : \{0,1\}^* \to \{0,1\}$ computable in time $O(n^3)$ not computable in time $O(n^2)$? Is there a language L recognizable in time $O(n^3)$ not recognizable in $O(n^2)$? An answer to the question stated in this way, no matter positive or negative, would be essential.

In the 1960s Hartmanis and Stearns [HS65] showed that for any constants a and b such that $1 < a < b$, $\mathbf{DTime}[n^a] \subsetneq \mathbf{DTime}[n^b]$ where $\mathbf{DTime}[n^d]$ is a class of the languages decidable by multi-tape deterministic Turing machines operating within $O(n^d)$ steps. Consequently,

ISSN 1433-8092

there exists a language L recognizable in cubic time not recognizable in quadratic time.

After that, in the 1970s Cook [Coo72] proved a time hierarchy for nondeterministic computations also. It was shown that $\mathbf{NTime}[n^a] \subsetneq \mathbf{NTime}[n^b]$. Diagonalization, a mathematical technique proposed by Cantor, was used in the both papers [HS65] and [Coo72] to prove time hierarchies.

But this technique fails to work with probabilistic classes, in particular, with **BPTime** (the languages recognized by probabilistic algorithms with two-sided error), **RPTime** (the languages recognized by probabilistic algorithms with one-sided error) and **ZPTime** (the languages recognized by probabilistic algorithms with correct answers only). The main obstacle is that these classes impose some restrictions on the error probability of their machines, and, thus, not every probabilistic machine is appropriate for recognizing languages.

So there are many long-standing open questions concerning time as a computational resource. One of them is whether the classes $\mathbf{BPTime}[n]$ and $\mathbf{BPP} = \bigcup_d \mathbf{BPTime}[n^d]$ are different. The same for **RPTime** and **ZPTime** is of big interest.

1.2 Computations with Advice

In 2002 Barak [Bar02] suggested a new technique for proving time hierarchies that uses an *optimal algorithm* introduced by Levin [Lev73]. This technique was developed later by Fortnow and Santhanam [FS04] who showed that there exists a polynomial time hierarchy for **BPTime** with one-bit advice, that is for the class of the languages recognizable by probabilistic algorithms with two-sided error that use one bit of advice for every input length. Formally, it was proved that $\mathbf{BPTime}[n^a]/1 \subsetneq \mathbf{BPTime}[n^b]/1$.

The idea was to take some hard, **PSPACE**-complete language A and show that it is decidable by some optimal **BPTime**-algorithm in time $T(n)$ but no other **BPTime**-algorithm can decide it "noticeably" faster. To achieve this optimality effect, the algorithm simulates all possible probabilistic machines and uses an *instance checker* for the **PSPACE**-complete problem A [BFL91, TV02] to verify the answers of the machines.

Unfortunately, no enumeration of all and only **BPTime**-machines operating within a certain amount of time is known. The classes of computations like **BPTime** are called *semantic classes.* Among them one can name also **RPTime**, **ZPTime** and many others. They are opposed to so-called *syntactic classes.* The examples of them are **DTime** and **NTime**.

Therefore the optimal **BPTime**-algorithm needs *a small advice* to get over the semantic difficulties when simulating the machines. It was proved for the language A that

$$A \in \mathbf{BPTime}[T(n)]/\log\log T(n)$$
$$A \notin \mathbf{BPTime}[T(n)^{\epsilon}]/\log\log T(n)$$

where $\log \log T(n)$ is the advice length. Most probably, the value of $T(n)$ grows faster than any polynomial. So a time translation is used and a language B such that $B \in \mathbf{BPP}/1$ and $B \notin \mathbf{BPTime}[n^d]/1$ is obtained from A. Consequently, $\mathbf{BPTime}[n^d]/1 \subsetneq \mathbf{BPP}/1$. The last implies $\mathbf{BPTime}[n^a]/1 \subsetneq \mathbf{BPTime}[n^b]/1$ for any constants a and b such that $1 < a < b$ [Bar02]. The same result holds for **RPTime** [FST05] also.

Fortnow, Santhanam and Trevisan [FST05] used the idea of optimal algorithm later on to prove a time time hierarchy for a wide range of computations. In particular, they obtained a *quasipolynomial time* hierarchy for **ZPTime** with one bit of advice:

$$\mathbf{ZPTime}[n^{(\log n)^a}]/1 \subsetneq \mathbf{ZPTime}[n^{(\log n)^b}]/1$$

But a longer advice was needed to prove a *polynomial time* hierarchy. Namely, it was proved that $\mathbf{ZPTime}[n^d]/l(n) \subsetneq \mathbf{ZPP}/l(n)$ where $l(n)$ is some function such that $l(n) = O(\log n \log \log n)$. At the same time, the most interesting is a polynomial time hierarchy with only one bit of advice.

1.3 New Approach

A new approach for proving time hierarchies is proposed in this paper. It is based on the following idea which is very natural. Given more time, a computer is able to recognize some hard language "better" than before. That is, for any constant $d > 1$ there exists an *optimal* polynomial-time algorithm $\mathcal{M}$ of some kind, **BPTime** for example, that recognizes the hard language "better" than any algorithm of the same kind within time $O(n^d)$. To some extent, our approach is inspired by the paper of Fortnow, Santhanam and Trevisan [FST05].

The kind of optimality that the algorithm $\mathcal{M}$ possesses is different from the optimality introduced by Levin [Lev73] where $\mathcal{M}$ would recognize some hard language *faster* than any other machine. In contrast to Levin's optimality, our algorithm $\mathcal{M}$ simply recognizes the hard language *better* than others do and, thereby, does not have to "precisely" recognize that *hard language*. Thus, $\mathcal{M}$ can even be implemented as a polynomial-time algorithm.

So we have a flexibility that results in the proof of time hierarchies for a very broad range of computations with a shorter length of advice than in [FST05]. In particular, we show that $\mathbf{ZPP}/l(n) \nsubseteq \mathbf{ZPTime}[n^d]/(l(n)+1)$ for some function $l(n) \leq \log n$. This time hierarchy for $\mathbf{ZPTime}[n^d]$ with advice of length $O(\log n)$ enables us to apply a known result of Goldreich, Sudan and Trevisan [FST05, Lemma 10] and obtain the following theorem:

Theorem 1.1. *For any constant* $d > 1$, $\mathbf{ZPTime}[n^d]/1 \subsetneq \mathbf{ZPP}/1$.

Using a translation [Bar02], we have a polynomial time hierarchy for **ZPTime** with one-bit advice as a corollary of Theorem 1.1:

Corollary 1.1. *For any constants* a *and* b *such that* $1 < a < b$, $\mathbf{ZPTime}[n^a]/1 \subsetneq \mathbf{ZPTime}[n^b]/1$.

1.4 Contributions of this Paper

The technique introduced in this paper brings a polynomial time hierarchy with one bit of advice for a very broad range of classes, both syntactic and semantic. Among them, there are **ZPTime**, **NTime** $\cap$ **coNTime**, **UTime** (**NTime** with unambiguous accepting paths), **MATime** (Merlin-Arthur games with time-bounded Arthur), **AMTime** (Arthur-Merlin games with time-bounded Arthur) and **BQTime** (bounded-error quantum). Before, only time hierarchies with $O(\log n \log\log n)$ bits of advice were known for them [FST05].

Furthermore, the proof of polynomial time hierarchies for **RPTime** and **BPTime** with one-bit advice obtained using the new technique is significantly simpler than the previously known proofs [Bar02, FS04, FST05]. In particular, no instance checker is used in this paper and the constructed optimal algorithm is a polynomial-time algorithm. It makes a time hierarchy proof much easier.

2 Preliminaries

2.1 Notation and Definitions

$\mathbf{DTime}[n^d]/l(n)$ is the class of the languages recognized by deterministic multi-tape Turing machines with advice of length $l(n)$ in time $O(n^d)$. Given input of length n, such machines read an advice string a_n of the length $l(n)$ located at an "advice tape". Notice that any other advice $\{a_i\}_{i \neq n}$ is unknown when processing an input of the length n.

A property $P(n)$ holds *infinitely often* (*i.o.*) iff $P(n)$ is true for infinitely many numbers n. A property $P(n)$ holds *almost always* (*a.a.*) iff $P(n)$ is true for every n starting with some n_0. Let $L|_n$ denotes $L \cap \Sigma^n$ where L is a language over an alphabet Σ. Let us define a class of the languages that are "infinitely often" recognized by deterministic machines:

$$\mathbf{i.o.DTime}[n^d] = \{L \in \Sigma^* : \exists L' \in \mathbf{DTime}[n^d] : \ \textit{i.o.}\ L|_n = L'|_n\}$$

Since the result presented in this paper holds for a broad range of computations, we state and prove it for some hypothetic kind of computations called **CTime**. It has very natural properties which are stated bellow.

It may be useful to think of the abstract **CTime**-computations as of some particular kind of computations, **ZPTime** for instance. The class **ZPTime** is one of the numerous examples of the computations that were unknown before to have a polynomial time hierarchy with only one bit of advice.

A probabilistic machine N is a **ZPTime**-machine iff for every input x it produces an answer in $\{0, 1, \perp\}$, and for some language L_N the following holds:

- $x \in L_N \implies Pr\{N(x) = 1\} > 1/2$ and $Pr\{N(x) = 0\} = 0$
- $x \notin L_N \implies Pr\{N(x) = 0\} > 1/2$ and $Pr\{N(x) = 1\} = 0$

The language L_N is *recognized* by that machine N. Obviously, the **ZPTime**-machine N recognizes only one language. Notice that not every probabilistic machine is a **ZPTime**-machine. **ZPTime** is one of the semantic classes.

$\mathbf{ZPTime}[n^d]/l(n)$ is a class of the languages recognized by **ZPTime**-machines with advice of length $l(n)$ in time $O(n^d)$. A class $\mathbf{ZPP}/l(n)$ is defined as $\bigcup_d \mathbf{ZPTime}[n^d]/l(n)$.

2.2 Hypothetic CTime

Assume that there exist some hypothetic **CTime**-machines. Also let $\{M_i\}_{i=1}^{\infty}$ be an effective enumeration of all but not only **CTime**-machines. $\mathbf{CTime}[n^d]/l(n)$ is the class of the languages recognized by **CTime**-machines within time $O(n^d)$ with advice of length $l(n)$. A class $\mathbf{CP}/l(n)$ is defined as $\bigcup_d \mathbf{CTime}[n^d]/l(n)$. A class **i.o.CTime** is defined in the following way:

$$\mathbf{i.o.CTime}[n^d] = \{L \in \Sigma^* : \exists L' \in \mathbf{CTime}[n^d] : \ \textit{i.o.}\ L|_n = L'|_n\}$$

Sometimes we use the notation $\mathbf{CTime}[n^d]/l(n)$ not only for the class of languages but also for the **CTime**-machines operating within time $O(n^d)$ with advice of length $l(n)$. The usage of this notation is clear from the context.

Now we declare the properties required from **CTime**-computations:

Computability: For any constant a, $\mathbf{CTime}[n^a] \subseteq \mathbf{DTime}[2^{n^{2a}}]$.

Soundness: A polynomial-time machine N from **DTime** is a polynomial-time machine from **CTime**.

Composition: A machine C_1 from $\mathbf{DTime}[n^a]$ that finishes its execution by running a machine C_2 from $\mathbf{CTime}[n^b]$ on an input of length $m(n)$ belongs to $\mathbf{CTime}[n^a + (m(n))^b]$.

Simulation: A polynomial-time machine S from **DTime** that finishes by simulating a machine $M_{i(x)}$ from **CTime** that operates within time $c \cdot n^b$ on the input x is a polynomial-time machine from **CTime**.

1. S can simulate different machines on different inputs.
2. S simulates only one machine $M_{i(x)}$ on a particular input x.
3. S simulates only **CTime**-machines.
4. S simulates only those machines that operate within $c \cdot n^b$ steps where c and b are fixed for the machine S.

Given an input x, the machine S determines what **CTime**-machine $M_{i(x)}$ to simulate. Assume that this choice is made by writing the number $i(x)$ on a special "simulation tape". Mention that S can simulate only those machines that operate within strictly $c \cdot n^b$ steps, not $O(n^b)$ steps. Otherwise we cannot guarantee that S is a *polynomial-time* machine.

The declared properties are naturally extended to the case of computations with advice. The machines C_1 and S are responsible for providing the machines C_2 and $M_{i(x)}$ with a correct advice.

All the results that we obtain for the class **CTime** are valid for any kind of computations that reveal the properties of **CTime**-computations. One can see that the semantic classes **ZPTime**, **BPTime**, **RPTime**, **NTime** $\cap$ **coNTime**, **UTime**, **MATime**, **AMTime** and **BQTime** have such properties. The syntactic classes **DTime** and **NTime** do also have the properties of **CTime**.

Let L_N denote the language recognized by the machine N. This language depends on a particular class of computations, **ZPTime** for example, the machine N belongs to.

3 Time Hierarchy for Classes with Advice

3.1 Main Theorem

Theorem 3.1. *For any constant* $d > 1$, $\mathbf{CTime}[n^d]/1 \subsetneq \mathbf{CP}/1$.

Theorem 3.1 is a straight-forward implication from the two following lemmas which we prove in this section:

Lemma 3.1. *For any constant* $d > 1$ *there exists a function* $l(n) \leq \log n$ *such that* $\mathbf{CP}/l(n) \nsubseteq \mathbf{CTime}[n^d]/(l(n)+1)$.

Lemma 3.2. *For any constant* $d > 1$, *if* $\mathbf{CP}/l(n) \nsubseteq \mathbf{CTime}[n^{2d}]/(l(n)+1)$ *for some function* $l(n) \leq \log n$, *then* $\mathbf{CP}/1 \nsubseteq \mathbf{CTime}[n^d]/1$.

When proving Lemma 3.1 one can take $\lceil d \rceil$ instead of d. Therefore we assume that d is a natural number.

The most part of this section is devoted to the proof of Lemma 3.1. The idea of its proof is exposed in Subsection 3.2.

The generalized statement of Lemma 3.2 is presented in the paper of Fortnow, Santhanam and Trevisan [FST05, Lemma 10] who refer to the ideas of Goldreich, Sudan and Trevisan. For the sake of completeness we give the proof of Lemma 3.2 in Subsection 3.6.

Using translation from [Bar02] we obtain a polynomial time hierarchy with one bit of advice as a corollary of Theorem 3.1:

Corollary 3.1. *For any constants* a *and* b *such that* $1 < a < b$, $\mathbf{CTime}[n^a]/1 \subsetneq \mathbf{CTime}[n^b]/1$.

3.2 Proof Idea

To illustrate the proof idea, we assume for a moment that we want to prove $\mathbf{CP} \nsubseteq \mathbf{CTime}[n^d]$. Let us omit some details during the exposition.

In the beginning, we take some hard language A and "rarefy" it to obtain a language $\mathcal{R}(A)$ with some nice properties. After that, we construct an optimal algorithm $\mathcal{M}$ from $\mathbf{CP}$[1] that recognizes the language

[1]Dependently on a context, we use notation $\mathbf{CTime}[n^d]/l(n)$ both for languages and for machines.

$\mathcal{R}(A)$ "better" than *any* machine N from $\mathbf{CTime}[n^d]$. That is for some input length n, it holds that $L_{\mathcal{M}}|_n = \mathcal{R}(A)|_n$[2] and $L_N|_n \neq \mathcal{R}(A)|_n$, where $L_{\mathcal{M}}$ and L_N are the languages recognized by the machines $\mathcal{M}$ and N correspondingly. Therefore, $L_{\mathcal{M}} \neq L_N$ for any language L_N from $\mathbf{CTime}[n^d]$.

The kind of optimality that the algorithm $\mathcal{M}$ possesses is different from Levin's optimality [Lev73] where $\mathcal{M}$ is to recognize some language faster than any other machine. In our case, $\mathcal{M}$ does not have to recognize $\mathcal{R}(A)$ "everywhere". But the trick used to obtain the optimality is the same. The optimal algorithm $\mathcal{M}$ simulates all the machines from $\mathbf{CTime}[n^d]$ in order to recognize the language $\mathcal{R}(A)$ better than these machines can do on their own.

Assume $\eta(i,k)$ is an injective increasing function. The machine $\mathcal{M}$ "competes" against a machine $N = \mu(i)$ on the input lengths $\{\eta(i,k)\}_{k=1}^{\infty}$. The language $\mathcal{R}(A)$, the field of completion, is constructed in the following way:

$$\mathcal{R}(A)|_{\eta(i,0)} = A|_{\eta(i,0)} \tag{1}$$

$$\mathcal{R}(A)|_{\eta(i,k+1)} = \{0^l x : x \in \mathcal{R}(A)|_{\eta(i,k)},\ l = \eta(i,k+1) - \eta(i,k)\} \tag{2}$$

Due to the padding, the hardness of the input lengths $\eta(i,k)$ decreases as the number k increases. Since A is hard enough, it is typical that $L_N|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$ for any number k, or $L_N|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$ only starting with some $k_0 > 1$. In the first case, $\mathcal{M}$ defeats N, because $\mathcal{M}$ can easily solve input lengths $\eta(i,k)$ where k is big enough.

In the latter case, the optimal algorithm $\mathcal{M}$, given an input x of the length $n = \eta(i, k_0 - 1)$, constructs a string $y = 0^{\bar{n}-n}x$ of the length $\bar{n} = \eta(i,k_0)$ and simulates the machine $N = \mu(i)$ on it. Thus $L_{\mathcal{M}}|_n = \mathcal{R}(A)|_n \neq L_N|_n$. Certainly, $\mathcal{M}$ needs more than n^d steps to simulate $\bar{n}^d$ steps of the machine N, because $\bar{n} > n$.

Where Advice Appears. Have we came up with a proof of a polynomial time hierarchy for **CTime**? Unfortunately, we have not, because no enumeration of all and only **CTime**-machines is known. Since we need $\mu(i)$ to enumerate all **CTime**-machines, we have to let it enumerate not only **CTime**-machines. But then $\mathcal{M}$ may leave the class of **CTime**-computations when simulating some $N = \mu(i)$ that is not a **CTime**-machine. Therefore $\mathcal{M}$ needs an advice on whether $\mu(i)$ is a **CTime**-machine indeed.

Have we achieved a polynomial time hierarchy for **CTime** with one bit of advice, that is $\mathbf{CP}/1 \not\subseteq \mathbf{CTime}[n^d]/1$? Not yet, because now $\mathcal{M}$ has to compete with not only **CTime**-machines, but also with any possible correct advice for them. In the case of **BPTime**, $\mathcal{M}$ could simulate a machine $N = \mu(i)$ with any possible advice ("0" and "1") and use an instance checker to choose the correct answer, and still fulfill the requirements on probability of error. But in general case, **CTime** does not have an instance checker.

[2] $L|_n = L \cap \{0,1\}^n$

The solution is to provide $\mathcal{M}$ directly with a correct advice for the machine $N = \mu(i)$. Given an input of the length $n = \eta(i, k-1)$, $\mathcal{M}$ receives an advice on (1) whether $\mu(i)$ is a **CTime**-machine operating within time $O(n^d)$ with some advice $\{w_n\}_{n=1}^{\infty}$, and (2) what is a correct advice w for $\mu(i)$ for the input length $\bar{n} = \eta(i, k)$. Therefore if the advice length is $l(n)$, then $l(n) = 1 + l(\bar{n})$ and $l(n)$ decreases. It seems to be an obstacle because it implies $l(n) = \infty$.

However, $\mathcal{M}$ needs no advice for the input lengths $\eta(i, k)$ where k is big enough and, thus, $\mathcal{M}$ directly solves that instances. So we manage to keep $l(n) \leq \log n$ while proving $\mathbf{CP}/l(n) \nsubseteq \mathbf{CTime}[n^d]/(l(n)+1)$. That enables us to use the result of Goldreich, Sudan and Trevisan [FST05, Lemma 10] to obtain $\mathbf{CP}/1 \nsubseteq \mathbf{CTime}[n^d]/1$.

3.3 Rarefied Language

Let $\{p_i\}_{i=1}^{\infty}$ be the set of all prime numbers (where $p_1 = 2$, $p_2 = 3$, $p_3 = 5, \dots$).

Definition 3.1 (Rarefied Language).

$$\eta(i, k) = p_i^{c^k}, \text{where } c = 2^{6d},\ i, k \in \mathbb{Z},\ i \geq 1,\ k \geq 0 \tag{3}$$

$$\pi(x) = \begin{cases} z & \text{when } |x| = \eta(i, k),\ x = 0^l z,\ |z| = p_i \\ \perp & \text{otherwise} \end{cases} \tag{4}$$

$$\mathcal{R}(L) = \{x : \pi(x) \neq \perp,\ \pi(x) \in L\} \tag{5}$$

The mapping $\eta(i, k)$ is an injection. For a given n let i_n and k_n be such numbers that $n = \eta(i_n, k_n)$, if possible. Also let $m_n = p_{i_n}$.

In fact, m_n is the "real" length of an instance x where $n = |x|$. In order to solve the instance x, a polynomial-time algorithm can simply check whether the word $\pi(x)$ of the length m_n is in L. A polynomial-time algorithm can spend time $poly(n) = poly(m_n^{c^k})$ for the instance $x = 0^l \pi(x)$. So the instances of the length $\eta(i, k)$ becomes easier as k increases.

Lemma 3.3 (Classification). *For any two languages L and A, for any infinite set $I \subseteq \mathbb{N}$, at least one of the following items holds:*

1. *For almost all $i \in I$, for every k, $L|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$*
2. *For infinitely many $i \in I$, for almost all k, $L|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$*
3. *For infinitely many $i \in I$, for some k, $L|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$ and $L|_{\eta(i,k+1)} = \mathcal{R}(A)|_{\eta(i,k+1)}$*

Proof. Assume that item (1) does not hold. Then for infinitely many $i \in I$, for some k, we have $L|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$. Therefore at least one of items (2) and (3) holds. □

We will use the above lemma to classify the languages recognized by **CTime**-machines. Consider a **CTime**-machine N and let $I_N = \{i : N = \mu(i)\}$. When item (2) or (3) holds, the optimal algorithm $\mathcal{M}$ has an opportunity to recognize the language $\mathcal{R}(A)$ better than the machine N can. In the case of item (2), $\mathcal{M}$ directly solves an "easy" input length $\eta(i,k)$ where k is big enough, and defeats N. In the case of item (3), $\mathcal{M}$ simulates N on the input length $\eta(i,k+1)$ to solve the input length $\eta(i,k)$, and also defeats the machine N.

In the same time, it is possible to exclude item (1) by choosing the language A to be hard:

Lemma 3.4 ([FST05, Lemma 5]). *For any two constants a and b such that $0<a<b$,* $\mathbf{DTime}[2^{n^b}] \nsubseteq \mathbf{i.o.DTime}[2^{n^a}]/(n-\log n)$

Let us take a language $A \in \mathbf{DTime}[2^{n^{3d}}] \setminus \mathbf{i.o.DTime}[2^{n^{2d}}]/(n-\log n)$. It is recognized by some deterministic machine M_A operating within $C_A \cdot 2^{n^{3d}}$ steps.

Lemma 3.5. *For any language $L \in \mathbf{CTime}[n^d]/(l(n)+1)$ where $l(n) \leq \log n$, item (1) of Classification Lemma 3.3 relatively to $\mathcal{R}(A)$ does not hold.*

Proof. Assume that item (1) holds for some language $L \in \mathbf{CTime}[n^d]/(l(n)+1)$. Then for almost all $i \in I$, for every k, $L|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$. In particular, we have $L|_{p_i} = \mathcal{R}(A)|_{p_i}$ for almost all i, because $p_i = \eta(i,0)$. Keeping in mind that $\mathcal{R}(A)|_{p_i} = A|_{p_i}$, we see that $A \in \mathbf{i.o.CTime}[n^d]/(l(n)+1)$. Therefore $A \in \mathbf{i.o.DTime}[2^{n^{2d}}]/(l(n)+1)$, a contradiction. $\square$

3.4 Optimal Algorithm

There is an enumeration of all **CTime**-machines $\{M_i\}_{i=1}^{\infty}$. Some of the enumerated machines are not from **CTime**. Let us construct a new enumeration μ from the old one, where every **CTime**-machine has infinitely many numbers:

$$\mu(2^l + r) = M_r, \text{ where } 0 \leq r < 2^l. \tag{6}$$

Let us describe the optimal algorithm $\mathcal{M}$. Given an input x of length n and an advice (a_n, b_n), which is defined later, $\mathcal{M}$ performs the following:

1. If $\pi(x) = \perp$, then print 0 and stop.
2. Compute the values i_n and k_n such that $n = \eta(i_n, k_n)$.
3. Run the machine M_A on input $\pi(x)$ for $C_A \cdot n$ steps. If it gives an answer r, print r and stop.
4. If $a_n = 0$, then print 0 and stop.
5. Let $N = \mu(i_n)$, $\bar{n} = \eta(i_n, k_n + 1)$, $y = 0^{\bar{n}-n}x$.
6. Simulate the **CTime**-machine N on the input y of the length $\bar{n}$ with the advice b_n for $\bar{n}^{d+1}$ steps.

$\mathcal{M}$ computes the values i_n, k_n and p_{i_n} in time *poly*(n) where n is the input length.

A word x of the length $n = (p_{i_n})^{c^{k_n}} = (m_n)^{c^{k_n}}$ is "long", if $k_n \geq \frac{1}{2}\log m_n$. "Long" words are directly recognized by $\mathcal{M}$ with the help of the deterministic machine M_A, because:

$$c^{k_n} = (2^{6d})^{k_n} \geq (2^{6d})^{\frac{1}{2}\log m_n} = m_n^{3d} \tag{7}$$

$$n = (p_{i_n})^{c^{k_n}} \geq 2^{c^{k_n}} \geq 2^{m_n^{3d}} \tag{8}$$

$$C_A \cdot n \geq C_A \cdot 2^{m_n^{3d}} \tag{9}$$

Let us define the advice length function $l(n)$:

$$l(n) = \begin{cases} \log m_n - 2k_n & \text{when } k_n < \frac{1}{2}\log m_n \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

In particular, for all the input lengths n that are not representable as $n = p_i^{c^k}$, the value of $l(n)$ is equal to 0. Indeed, $\mathcal{M}$ needs no advice for such input lengths. The machine $\mathcal{M}$ also succeeds without advice when recognizing "long" words which have $k_n \geq \frac{1}{2}\log m_n$.

Now consider "short" words with $k_n < \frac{1}{2}\log m_n$. Let $\bar{n} = \eta(i_n, k_n + 1)$. If a **CTime**-machine $N = \mu(i_n)$ with some advice w of the length $l(\bar{n})+1$ (1) solves the $\mathcal{R}(A)$-instances of the length $\bar{n}$, and (2) completes within $\bar{n}^{d+1}$ steps, then let $a_n = 1$ and $b_n = w$, otherwise let $a_n = 0$. Thus $l(n) = l(\bar{n})+2$ and, therefore, $l(n) = \log m_n - 2k_n$. Consequently, we have $l(n) \leq \log m_n \leq \log n$.

Property 3.1. $L_{\mathcal{M}} \in \mathbf{CP}/l(n)$

Proof. By the definition of the algorithm $\mathcal{M}$ and by the choice of the advice $\{a_n, b_n\}_{n=1}^{\infty}$, $\mathcal{M}$ is a **CTime**-machine. Furthermore, it is a polynomial-time machine, because $\bar{n} = n^c$ and $\mu(i_n) \leq i_n \leq n$. Hence, $L_{\mathcal{M}} \in \mathbf{CP}/l(n)$ □

Property 3.2. For any i, for almost all k, $L_{\mathcal{M}}|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$.

Proof. For any i, for all the numbers $k \geq \frac{1}{2}\log p_i$, we have $L_{\mathcal{M}}|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$ by means of running M_A. □

Property 3.3. Let $I_N = \{i : \mu(i) = N\}$. For any machine N from $\mathbf{CTime}[n^d]/(l(n)+1)$, for almost all $i \in I_N$, for any k, if $L_N|_{\eta(i,k+1)} = \mathcal{R}(A)|_{\eta(i,k+1)}$ then $L_{\mathcal{M}}|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$.

Proof. Consider a machine N from $\mathbf{CTime}[n^d]/(l(n) + 1)$ operating within $C_N \cdot n^d$ steps. For any $i \in I_N$ such that $i > C_N$, for any k, we have $\eta(i, k) > C_N$. Let $n = \eta(i, k)$ and $\bar{n} = \eta(i, k + 1)$.

Now assume that $L_N|_{\bar{n}} = \mathcal{R}(A)|_{\bar{n}}$. It means that the machine $N = \mu(i)$ with some advice w of the length $l(\bar{n}) + 1$ (1) solves the $\mathcal{R}(A)$-instances of the length $\bar{n}$, and (2) completes within $C_N \cdot \bar{n}^d < \bar{n}^{d+1}$ steps. Then, by the definition of the advice $\{a_n, b_n\}_{n=1}^{\infty}$, we have $a_n = 1$ and $b_n = w$. Consequently, by the definition of the algorithm $\mathcal{M}$, $L_{\mathcal{M}}|_n = \mathcal{R}(A)|_n$. □

3.5 Hierarchy with Logarithmic Advice

We prove Lemma 3.1 bellow:

Lemma. *For any constant $d > 1$, there exists a function $l(n) \leq \log n$ such that* $\mathbf{CP}/l(n) \nsubseteq \mathbf{CTime}[n^d]/(l(n)+1)$.

Proof. By Property 3.1 we have $L_{\mathcal{M}} \in \mathbf{CP}/l(n)$ where $L_{\mathcal{M}}$ is the language recognized by the optimal algorithm $\mathcal{M}$. It remains to prove that

$$L_{\mathcal{M}} \notin \mathbf{CTime}[n^d]/(l(n)+1)$$

Consider any language $L_N \in \mathbf{CTime}[n^d]/(l(n)+1)$. It is recognized by some **CTime**-machine N within time $O(n^d)$ with advice of the length $l(n)+1$. To prove that $L_N \neq L_{\mathcal{M}}$, we classify L_N relatively to the language $\mathcal{R}(A)$ by Lemma 3.3 with $I_N = \{i : \mu(i) = N\}$.

Accordingly to Lemma 3.5, item (1) of the classification does not hold for any language $L_N \in \mathbf{CTime}[n^d]/(l(n)+1)$.

Assume that item (2) holds. Then for infinitely many $i \in I_N$, for almost all k, $L_N|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$. But by Property 3.2, for any i, for almost all k, we have $L_{\mathcal{M}}|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)}$. Therefore for infinitely many $i \in I_N$, for almost all k, $L_{\mathcal{M}}|_{\eta(i,k)} \neq L_N|_{\eta(i,k)}$.

Assume that item (3) holds. Then for infinitely many $i \in I_N$, for some k, $L_N|_{\eta(i,k)} \neq \mathcal{R}(A)|_{\eta(i,k)}$ and $L_N|_{\eta(i,k+1)} = \mathcal{R}(A)|_{\eta(i,k+1)}$. By Property 3.3, for infinitely many $i \in I_N$, for some k, we have $L_{\mathcal{M}}|_{\eta(i,k)} = \mathcal{R}(A)|_{\eta(i,k)} \neq L_N|_{\eta(i,k)}$. □

3.6 Advice Translation

We prove Lemma 3.2 bellow:

Lemma. *For any constant $d > 1$ if* $\mathbf{CP}/l(n) \nsubseteq \mathbf{CTime}[n^{2d}]/(l(n)+1)$ *for some function $l(n) \leq \log n$, then* $\mathbf{CP}/1 \nsubseteq \mathbf{CTime}[n^d]/1$.

Proof. Let us take a language $L \in \mathbf{CP}/l(n) \setminus \mathbf{CTime}[n^{2d}]/(l(n)+1)$. It is recognized by some machine M from $\mathbf{CP}/l(n)$ with advice $\{a_n\}_{n=1}^{\infty}$ of length $l(n)$. The idea of the proof is construct a language L' from the language L where the advice a_n would be coded into input length, so that $L' \in \mathbf{CP}/1 \setminus \mathbf{CTime}[n^d]/1$.

Let $b(a_n)$ denote the number that has the binary representation a_n. Evidently, we have $0 \leq b(a_n) \leq n-1$, because the advice length is not greater than $\log n$. Let $s_n = \sum_{i=1}^{n} i = O(n^2)$. Then we construct a language $L' = \{0^k 1x : x \in L,\ |x| = n,\ k = s_{n-1} + b(a_n)\}$. Therefore, every word $x \in L$ of length n has an "image" in L' that is $0^k 1x$. The image length is determined solely by the length n and the corresponding advice a_n. So the image length of a word x of length n is:

$$m(n) = s_{n-1} + b(a_n) + 1 + n \tag{11}$$

Every two words of different lengths have images of different lengths also:

$$m(n) < s_n + b(a_{n+1}) + 1 + (n+1) = m(n+1) \tag{12}$$

To show that $L' \in \mathbf{CP}/1$, we construct a **CTime**-machine M' operating within polynomial time with one-bit advice $\{a'_m\}_{m=1}^{\infty}$. Let $a'_m = 1$ iff there exists a number n such that $m = m(n)$. M' decides on whether a word $y = 0^k 1x$ of length m belongs to the language L' in the following way. The machine M' (1) computes the number n from m, (2) verifies that $|x| = n$, (3) checks whether $a'_m = 1$, and, if it is so, (4) runs the machine M on the input x with the advice a_n that can be easily reconstructed from the length m.

For a contradiction, assume that $L' \in \mathbf{CTime}[n^d]/1$. Then L' is recognized by some machine M' from $\mathbf{CTime}[n^d]/1$ with one-bit advice $\{a'_m\}_{m=1}^{\infty}$. To obtain a contradiction with the choice of the language L, let us construct a machine M'' from $\mathbf{CTime}[n^{2d}]/(l(n)+1)$ that recognizes the language L. We provide M'' with the advice $\{a_n, a'_{m(n)}\}_{n=1}^{\infty}$. Given an input x of length n, the machine M'' (1) lets $y = 0^k 1x$ where $k = s_{n-1} + b(a_n)$, and (2) runs M' with the one-bit advice $a'_{m(n)}$ on the input y of the length $m(n)$. One can easily see that $L_{M''} = L$. Since $|y| = O(|x|^2)$, we have $L \in \mathbf{CTime}[n^{2d}]/(l(n)+1)$, a contradiction. □

4 Conclusions

The technique introduced in this paper brings a polynomial time hierarchy with one bit of advice for a very broad range of classes, both syntactic and semantic. Before, only two of the semantic classes, namely **BPTime** and **RPTime**, were proved to have a time hierarchy with one-bit of advice.

Now, we see that these time hierarchies are due to the properties of the computations with advice rather than the properties of the classes **BPTime** and **RPTime**. However, the question whether semantic classes, in particular **BPTime**, have polynomial time hierarchies without advice remains open.

Acknowledgments

I am grateful to my scientific advisor Edward A. Hirsch for introducing me to the area and for valuable comments, and to Dimitri Grigoriev for helpful discussions.

References

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3-40, 1991.

[Bar02] Boaz Barak. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-Uniform" Algorithms. *Lecture Notes in Computer Science*, 2483:194-208, 2002.

[Coo72] Stephen A. 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.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy Theorems for Probabilistic Polynomial Time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004.

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Promise Hierarchies. In *Proceedings of the 37th ACM Symposium on the Theory of Computing*, 2005.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285-306, 1965.

[Lev73] Leonid Levin. Universal search problems (in Russian). *Problemy Peredachi Informatsii*, 9(3):265-266, 1973.

[TV02] Luca Trevisan and Salil Vadhan. Pseudorandomness and Average-case Complexity via Uniform Reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, volume 17, 2002.