Electronic Colloquium on Computational Complexity, Report No. 59 (2020)

$\Pr-\mathsf{ZSUBEXP} \not\subseteq \Pr-\mathsf{RP}$

Gonen Krak * Noam Parzanchevski * Amnon Ta-Shma *

Abstract

We unconditionally prove there exists a promise problem in promise $\mathsf{ZSUBEXP}$ that cannot be solved in promise RP. The proof technique builds upon Kabanets' easy witness method [Kab01] as implemented by Impagliazzo et. al [IKW02], with a separate diagonalization carried out on each of the two alternatives in the win-win argument. We remark that even though the easy witness method is a key component in many celebrated results in derandomization, we are not aware of any previous unconditional separation like the one we show.

The result relativizes. We could not prove a similar result for *total* functions, nor for functions in $\mathsf{ZTime}(T(n))$ for $T(n)$ below a half-exponential function (i.e., T such that $T(T(n)) < 2^n$).

1 Introduction

Understanding the power of probabilistic computation is a cornerstone problem of theoretical computer science. A famous problem of this type is whether $\mathsf{BPP} \overset{?}{=} \mathsf{P}$. Some widely believed assumptions imply derandomization (e.g., [Yao82, NW94, IW97]) and, in consequence, it is widely believed that probabilistic computation can be efficiently simulated by deterministic computation and $\mathsf{BPP} = \mathsf{P}$. In particular, uniform lower bounds on non-uniform classes imply derandomization. Remarkably, partial converse results also apply in the other direction [KI04], and, morally, probabilistic computation derandomizes iff uniform lower bounds on circuits exist. However, in spite of the extensive study of the problem, there are almost no unconditional results in the area.[1] In particular, it might be possible, e.g., that $\mathsf{BPP} = \mathsf{NEXP}$.

Probabilistic computation can appear in many variants, and, in particular, a probabilistic algorithm might have two-sided, one-sided and zero-sided error. In two-sided error we allow both false positive and false negative results, in one-sided error we allow just false negatives, and in zero-sided error we require that the algorithm either gives the correct answer or outputs "don't know". The corresponding probabilistic classes are denoted by BPTime (for two-sided error), RTime (for one-sided error) and ZTime (for zero-sided error). See Def 6 for the formal definitions.

As before, not much is unconditionally known about these classes, other than the trivial containments. Early results from the 80's explain this lack of knowledge by showing, e.g., that

- There is an oracle O under which $\mathsf{BPP}^O = (\mathsf{EXP}^{\mathsf{NP}})^O$ [Hel86],
- There is an oracle O for which $\mathsf{NP}^O \neq \mathsf{RP}^O$ [Rac82]
- There is an oracle O for which $\mathsf{BPP}^O \neq \mathsf{RP}^O$ [MV96]

*The Blavatnik School of Computer Science, Tel-Aviv University, Israel 69978. Supported by the Israel Science Foundation grant no. 952/18.

[1]The situation for randomized *space bounded* computation is very different, but in this paper we focus on probabilistic time bounded computation.

ISSN 1433-8092

Thus, if we want to separate BPP from $\mathsf{EXP}^{\mathsf{NP}}$ (or NEXP), or want to show BPP is contained in NP or collapses to RP, we need to do that in an unrelativized way. While we do know today several non-relativizing techniques and results (such as $\mathsf{IP} = \mathsf{PSPACE}$) we still do not know how to prove that $\mathsf{BPP} \neq \mathsf{NEXP}$ or $\mathsf{BPP} = \mathsf{ZPP}$.

In this paper we take a small step, building on previous work, and prove an *unconditional* separation. We prove:

Theorem 1. $\Pr-\mathsf{ZSUBEXP} \nsubseteq \Pr-\mathsf{RP}$

The promise quantifier $\Pr-$ that appears before the class, means that we talk about a class of promise problems. A promise problem is a partition of all inputs to yes instances, no instances and those we do not care about. We say an algorithm solves a promise problem if it solves correctly all yes and no instances, and we put no restriction on the behaviour of the algorithm on inputs outside the promise. Promise problems abound in theoretical computer science, e.g., in approximation algorithms, PCPs and cryptography.

The theorem compares two classes that offhand might be incomparable: $\Pr-\mathsf{ZSUBEXP}$ and $\Pr-\mathsf{RP}$. A priori, it is possible that $\Pr-\mathsf{ZSUBEXP}$ contains languages not in $\Pr-\mathsf{RP}$, because we allow more time in $\Pr-\mathsf{ZSUBEXP}$, and indeed this is the case if we are right and probabilistic computation perfectly derandomizes, i.e., $\Pr-\mathsf{BPP} = \Pr-\mathsf{RP} = \Pr-\mathsf{ZPP} = \Pr-\mathsf{P}$. It is also possible that $\Pr-\mathsf{RP}$ contains languages not in $\Pr-\mathsf{ZSUBEXP}$ because in $\Pr-\mathsf{RP}$ we allow one-sided error and in $\Pr-\mathsf{ZTime}$ just zero-sided error computation. We can say for sure, though, that $\Pr-\mathsf{RP} \neq \Pr-\mathsf{ZSUBEXP}$, because otherwise $\Pr-\mathsf{RP}$ is closed under complement, which implies $\Pr-\mathsf{RP} = \Pr-\mathsf{ZPP}$ and $\Pr-\mathsf{ZPP} = \Pr-\mathsf{ZSUBEXP}$ contradicts a known hierarchy (see Theorem 19). However, the statement $\Pr-\mathsf{RP} \neq \Pr-\mathsf{ZSUBEXP}$ does not reveal which of the inclusions is wrong. In fact, often we can prove two classes are different without resolving which inclusion is wrong, e.g., it is easy to prove that $\mathsf{Space}(n) \neq \mathsf{P}$, but it is open to prove a specific containment is false. Theorem 1 states that unconditionally there is a problem in $\Pr-\mathsf{ZSUBEXP}$ that *does not belong* to $\Pr-\mathsf{RP}$.

We repeat again that the result we prove, proves the expected thing. If $\Pr-\mathsf{BPP} = \Pr-\mathsf{P}$ (as we believe) then it simply states that $\Pr-\mathsf{SUBEXP} \nsubseteq \Pr-\mathsf{P}$, which immediately follows from the time hierarchy theorem. Similarly, proving $\mathsf{BPP} \neq \mathsf{EXP}$, would be consistent with the way we expect things to be. However, the main point of the paper is that this result can be unconditionally proven.

The result relativizes, i.e., for every oracle O, $\Pr-\mathsf{ZSUBEXP}^{O} \nsubseteq \Pr-\mathsf{RP}^{O}$, and indeed, this result is not ruled out by previous work on relativized separations.

Next, we turn our attention to the technique and explain how this work builds upon previous work in derandomization.

1.1 Hardness vs. Randomness and the easy witness method

Nisan and Wigderson [NW94] developed the *Hardness vs. Randomness paradigm*: If one has access to a function $w : [M] \to \{0,1\}$ that no circuit of size s can compute, then one can create a *pseudo-random-generator* (PRG) converting $O(\log M)$ truly uniform bits to about s bits that are almost indistinguishable from uniform to all circuits of size about s. Nisan [Nis92] first used this idea together with the provable average-case hardness of the parity function against AC^0 circuits, to construct a PRG against AC^0, and later on Nisan and Wigderson [NW94] explored the implications of this method against polynomial-size circuits. A sequence of follow-up papers culminated with [IW97] showing that if there exists a function in EXP not having sub-exponential circuits then $\mathsf{BPP} = \mathsf{P}$.

In 2001 Kabanets [Kab01] suggested the following *win-win argument*. View a string $w \in \{0,1\}^M$ as the truth table of a function $w : [M] \to \{0,1\}$ and say the string w has *hardness* s, if no circuit of size s can compute the function. Now consider the set

$$\mathsf{EASY} \quad = \quad \left\{ w \in \{0,1\}^M \mid \mathsf{Hard}(w) \leq s \right\},$$

where $\mathsf{Hard}(w)$ is the size of the smallest circuit C computing w (i.e., $C(i) = w_i$ for all $i \in [M]$).

Now, intuitively, we have a win-win situation:

- Either EASY is a hitting set generator (HSG) against RP, meaning that for any $L \in \mathsf{RP}$, solvable by an algorithm $A(x, y)$, and for every $x \in L$, there exists an element $w \in \mathsf{EASY}$ such that $A(x, w) = 1$.
- Or else EASY is not a HSG against RP, implying that for some $L \in \mathsf{RP}$, solvable by $A(x, y)$, there exists infinitely many inputs $x_i \in L$ for which $A(x_i, w) = 0$ for all $w \in \mathsf{EASY}$.

Kabanets argues that this is a win-win situation. If EASY is a HSG against RP, then we can determine RP by trying all the possible *easy* witnesses, and because there are relatively few easy witnesses we get a non-trivial derandomization of RP. If on the other hand EASY is not a HSG against RP, and if we manage to put our hands on an input x such that $\Pr_y[A(x_i, y) = 1] \geq \frac{1}{2}$ but $A(x, w) = 0$ for all $w \in \mathsf{EASY}$, then we have a powerful tool to find hard strings with *certified* hardness: simply choose w at random and check that $A(x, w) = 1$. With probability half we choose w such that $A(x, w) = 1$ and then we know *with certainty* that w is hard. Once we have a certified hard string w, we can use w to derandomize probabilistic computations using the hardness vs. randomness paradigm. Thus, in such a case Kabanets argues we should be able to solve BPP with zero-sided error probabilistic algorithm, i.e., in ZSUBEXP.

However, there is still one missing part in the argument: If EASY is not a HSG it is infinitely often wrong, but we still need an efficient way to find elements on which the HSG fails. To remedy this Kabanets argues that

- Either EASY is a HSG against *efficient refuters*, i.e., no efficient program can find with a good probability an input on which EASY fails as a HSG, or,
- There is an efficient refuter that finds inputs on which EASY fails as a HSG, and then the above argument works and $\mathsf{BPP} = \mathsf{ZPP}$.

As a result Kabanets [Kab01] proves that either $\mathsf{RP} \subseteq \text{i.o.pseudo}-\mathsf{SUBEXP}$ or $\mathsf{BPP} = \mathsf{ZPP}$. The infinitely often quantifier (i.o.) is natural and also appears in the above intuitive discussion. The $\text{pseudo}-$ quantifier means that RP has a sub-exponential time algorithm that *looks correct* to efficient players, i.e., no efficient player (also called refuter) is able to find with a good probability an input on which the derandomization is wrong.

One immediate consequence is:

Theorem 2. *(Informal version of [Kab01, Corollary 17])* $\mathsf{RP} \subseteq \text{i.o.pseudo}-\mathsf{ZSUBEXP}$.

In many ways this is the inclusion we would like to get. However, the main weakness in Theorem 2 is that it only gives an inclusion in a heuristic world, i.e., solving languages in RP with a ZSUBEXP algorithm that "looks" correct to efficient refuters.

1.2 Going beyond efficient refuters

The main result of [IKW02] is that $\mathsf{NEXP} \subseteq \mathsf{P} \mid \mathsf{Poly}$ implies $\mathsf{NEXP} = \mathsf{MA}$. A central idea underlying the proof is an adaptation of the easy witness method to non-deterministic computation, while avoiding the heuristic world. The paper is technically complex, and contains many other results (we will discuss some soon). Here, we want to focus on a central idea that is often obscured by the many details in the IKW paper. To demonstrate this idea we avoid the non-deterministic case (that is the focus of [IKW02]) and instead focus on RP (that is the focus of [Kab01]). Our goal is to avoid the heuristic refuters.

The IKW suggestion to avoid the heuristic refuters is to give the inputs x_i (that have only hard witnesses) as advice. Let us now look at this suggestion in more detail. Suppose $R(x, w)$ is a deterministic machine solving an $\mathsf{RTime}(t(n))$ language. Suppose x is such that x is in the language (so that $\Pr_w(R(x, w) = 1) \geq \frac{1}{2}$) and furthermore all witnesses w are s-hard. As explained before an algorithm that is given such an x as advice, can guess a witness w and use its hardness to derandomize probabilistic computation. How long is the advice length? For simplicity let us assume the algorithm can use the s hardness to create s truly random bits (this is not correct, but also not far from being true, see Theorem 14). Thus, if the algorithm wants to derandomize an algorithm in $\mathsf{BPTime}(n^c)$, then it needs s to be n^c, which implies the length of w (dictated by how much hardness we are guaranteed in w) which in turn implies the length of x (because $|x| = t^{-1}(|w|)$).

The argument of [Kab01] says that either RP has easy witnesses, or BPP can be derandomized with zero error. Doing the calculation in this setting shows that if RP does not have easy witnesses then BPP is in $\mathsf{ZPP}|\mathsf{Poly}$, which is trivial as $\mathsf{BPP} \subset \mathsf{P} \mid \mathsf{Poly}$. However, if instead of asking whether RP always has easy witnesses, we ask whether REXP always has easy witnesses, we get much shorter advice length because the advice length $|x|$ is logarithmic in the witness length $|w|$.

While these ideas are central in the [IKW02] paper, they are mostly *implicit* in the paper. Furthermore, they are mostly worked out in conjunction with other ideas that are required for proving $\mathsf{NEXP} \subseteq \mathsf{P} \mid \mathsf{Poly}$ implies $\mathsf{NEXP} = \mathsf{MA}$. In Section 3 we single out this argument and do it in full. The calculation reveals that essentially the two alternatives we get are:

- Either we get a derandomization of RTime into deterministic time with half-exponential cost, or,
- We get a derandomization of BPTime to ZTime with half-exponential cost and logarithmic advice,

where a function H is half-exponential if $H(H(n)) = 2^n$ (see Definition 7). Specifically we get:

Theorem 3. *There exists constants $c' > c > 1$ such that for $T(n) = H(2^{c'n})$ either:*

- $\Pr\!-\!\mathsf{RTime}(T(n)) \subseteq \text{i.o.}\Pr\!-\!\mathsf{DTime}(\text{poly}(H(T(n))))$*, or,*
- $\Pr\!-\!\mathsf{BPTime}(n) \subseteq \Pr\!-\!\mathsf{ZTime}(\text{poly}(H(n^c)))\ /\ \log n$*.*

Theorem 3 is also true if the $\Pr-$ quantifier is removed from all clauses, because a total language is in particular a promise language, see Remark 24.

The parameters are, at first, difficult to digest. One way to read it is that either (roughly) $\mathsf{RTime}(T(n))$ has a non-trivial derandomization to $\mathsf{DTime}(H(T(n)))$ (the trivial derandomization is $2^{T(n)}$ time), or else $\mathsf{BPTime}(n)$ can be solved in a non-trivial way with a ZTime algorithm (the trivial derandomization is 2^n time) and some advice. While the second alternative in the Theorem is in the usual range of parameters and implies that $\mathsf{BPP} \subseteq \mathsf{ZSUBEXP}$, the first alternative is a bit unusual and talks about classes high up, namely, $\mathsf{RTime}(H(n))$ because of the reasons explained above.

While Theorem 3 does not appear in [IKW02] it lies entirely within the set of techniques and ideas used in [IKW02]. In particular, Section 6 of [IKW02] contains other "gap" theorems. However, we believe it is important to work out the idea of reducing the advice length by moving to higher-up classes on its own. Furthermore, doing this reveals the half-exponential time function as a natural barrier in these arguments.

One way that we find useful to look at this (and the way we choose to present the result in the technical sections) is the following. Originally, the easy witness method explores whether the set of all easy strings is a HSG against all linear-size circuits. If it is, then RTime can be non-trivially derandomized. If it isn't, then a provably hard string can be found in ZTime with non-uniform advice, albeit, sometimes too large. Another way to phrase this is that the easy witness method explores whether the set of all easy strings is a HSG against all linear-size circuits that have a *small Kolmogorov complexity*, i.e., circuits that can be described by a much shorter string then the circuit size. We therefore define $\mathsf{Circ}(n, s, a)$ to be the set of all circuits on n bits of size at most s and *description size* at most a (see Definition 16) and we explore whether the easy witnesses form a hitting set against this class. This point of view also appears in several recent works, e.g., Hirahara's work on the MinKT problem [Hir18] and [SCR$^+$20].

To summarize, there are several technical differences in the way the easy witness method is used in [Kab01] and [IKW02]. Some of these differences are due to the fact that [Kab01] considers what happens if RTime does not have easy witnesses, whereas [IKW02] ask what happens if NTime does not have easy witnesses. However, the most important difference from our point of view is both simple and surprising. In [Kab01] the main difficulty was how to obtain the inputs x which belong to L but have only hard witnesses. In [Kab01] this was solved by moving to a heuristic world with efficient refuters, these refuters, in turn, can be used to produce the infinite sequence of desired inputs. In [IKW02] the problem is bypassed by moving to exponentially long witnesses and giving the input x_i (which is much shorter) as advice. In NEXP the length of the input is logarithmically small compared to the length of the witness. The hardness s in the witness should be about the same magnitude as the length of the random string we wish to derandomize. In particular this means that the advice length $|x|$ can be much smaller than s and $|w|$, and, this translates to logarithmic advice length.

1.3 Going beyond "gap" theorems

The win-win analysis gives two alternatives. One can view these alternatives as "gap" theorems: e.g., one can interpret Theorem 3 as saying that either RTime (hence also ZTime) is weak and can be derandomized, or else ZTime is strong and contains two-sided error computations. However, ultimately, we would like to get one single unconditional containment, rather than one of two alternatives.

We already saw one attempt at getting such a single containment: We saw Kabanets [Kab01] proved $RP \subseteq \text{i.o.pseudo}-\mathsf{ZSUBEXP}$. Another celebrated result is William's unconditional result:

Theorem 4. *[Wil14]* $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$

The proof technique uses many ideas, and, in particular, the beautiful idea of using non-trivial algorithms (for statisfiablility of certain circuit classes) for proving lower bounds. Yet, one of its underlying basic building blocks is the easy witness method. Roughly speaking, the idea is that $\mathsf{NEXP} \subset \mathsf{ACC}^0$ also implies NEXP has easy witnesses (this time easy is a non-trivial ACC^0 circuit) and then a fast (i.e., non-trivial time) satisfiability algorithm for ACC^0 breaks the non-deterministic hierarchy. The proof that $\mathsf{NEXP} \subset \mathsf{ACC}^0$ implies NEXP has easy witnesses, is inspired by and uses [IKW02].

Another unconditional containment is given in [Wil16]. In the paper Williams shows that $\mathsf{NEXP} \subseteq \mathsf{P} \mid \mathsf{Poly}$ is, in fact, equivalent to NEXP having easy witnesses. In that paper Williams unconditionally shows, building on Kabanets' easy witness method, that

Theorem 5. *[Wil16]*

$$\mathsf{RP} \quad \subseteq \quad \text{i.o.}\mathsf{ZSUBEXP}|n^c, \tag{1}$$

While it is known that $\mathsf{BPP} \subseteq \mathsf{P} \mid \mathsf{Poly}$, the above result shows that RP is contained in ZSUBEXP with fixed polynomial advice.

The result of this paper is another consequence of the easy-witness method. It is a separation result, rather than a containment. Containments usually imply separations. For example BPP in SUBEXP would also imply $\mathsf{EXP} \not\subseteq \mathsf{BPP}$ because of the strict time hierarchy. However, the containments of Theorems 2 and 5 do not allow a separation result. For the first containment, it is true that $\mathsf{DTime}(T) \not\subseteq [\text{io-pseudo}]\mathsf{DTime}(t)$ for $T \gg t$, but a similar containment is not known for ZTime. For the second containment, we cannot use the ZTime hierarchy because of the larger than linear advice and the i.o. quantifier.

The main result of this paper is an unconditional proof that $\mathrm{Pr}-\mathsf{ZSUBEXP} \not\subseteq \mathrm{Pr}-\mathsf{RP}$. We achieve this separation by looking at each of the two alternatives of Theorem 3 separately, and showing that the separation result is implied by each of them (and in a relativized way). For that:

- The separation result used in the first alternative of Theorem 3 is simply the deterministic time hierarchy.
- For the second alternative we need a hierarchy for $\mathrm{Pr}-\mathsf{ZTime}$ against $\mathrm{Pr}-\mathsf{ZTime}$ with short advice. While $\mathrm{Pr}-\mathsf{ZTime}$ is closed under complement (which makes diagonalization simpler) it is a *semantic* class, i.e., the $\mathrm{Pr}-\mathsf{ZTime}$ promise may not be respected by some machines M on some inputs x, and given (M, x) it is not easy to determine if the promise is respected or not.

  Fortnow et. al. [FST05, Thm 12] diagonalized NTime against NTime with very short advice. We adapt the argument of [FST05] from NTime (which is a syntactic class not closed under complement) to ZTime (which is a semantic class closed under complement). The adaptation is almost one-to-one and we give it in Section 4. We remark that there is a more sophisticated result that generalizes [FST05, Thm 12] to larger (but still sub-linear) advice (see [FS14]) but we do not require these techniques.

The separation result gets rid of the i.o. quantifier and the advice (because the diagonalization argument can handle sub-linear advice) and is stated at the usual setting of parameters because separations go down.[2]

Two remarks are in place:

- We do not know how to show that $\mathsf{ZSUBEXP} \not\subseteq \mathsf{RP}$, because it is not known not how to diagonalize ZTime again ZTime with small advice (it is not even known how to prove a pure ZTime hierarchy (see, e.g., [Bar02, FS04])).
- Morally (but not precisely), we show that $\mathrm{Pr}-\mathsf{ZHalf} \not\subseteq \mathrm{Pr}-\mathsf{RP}$ where $\mathsf{ZHalf} = \mathsf{ZTime}(H(n))$ for a half-exponential function $H(n)$, see Theorem 31 for precise details.

We remark that it is intriguing whether the current results can be combined with other non-relativizing techniques to yield further unconditional statements,

[2]In fact, we do not know a corresponding separation result for the classes up.

2 Preliminaries

We use deterministic, non-deterministic, and the probabilistic classes in the standard way. We give below the definition of $\mathrm{Pr}{-}\mathsf{ZTime}$ with advice to stress how the non-uniform advice interacts with the i.o. quantifier:

Definition 6. *Let $L = (L_{\mathsf{YES}}, L_{\mathsf{NO}})$ be a promise problem. We say*

$$L \in \text{i.o.Pr}{-}\mathsf{ZTime}(T(n)) \;/\; a(n)$$

if there exists an infinite set $I \subset \mathbb{N}$, a sequence $\left\{a_n \in \{0,1\}^{a(n)}\right\}_{n \in I}$ and a probabilistic machine $M(x, y, a_n)$ running in time $T(n)$, such that for all $n \in I$, $x \in (L_{\mathsf{YES}} \cup L_{\mathsf{NO}}) \cap \{0,1\}^n$, $y \in \{0,1\}^{T(n)}$,

- *$M(x, y; a_n)$ is either "don't know" or $L(x)$.*
- *$\Pr_y[M(x, y, a_n) = L(x)] \geq \frac{1}{2}$.*

We say x is the input, y is the randomness of the machine and a_n us the advice string.

Notice that we require the ZTime guarantee (of either don't know or the correct answer, and the few "don't know") only on inputs in the promise for which the non-uniform advice is correct, and only for the correct advice. We do not require the ZTime guarantee for incorrect advice strings.

Definition 7. *Functions $S : \mathbb{N} \to \mathbb{N}$ and $H : \mathbb{N} \to \mathbb{N}$ are called:*

- sub-exponential *if for any constant $\varepsilon > 0$ and sufficiently large n, $S(n) \leqslant 2^{n^{\varepsilon}}$,*
- half exponential *if $H(H(n)) = 2^n$, and* above half exponential *if $H(H(n)) > 2^n$ for sufficiently large n.*

Definition 8. *A function $f : \mathbb{N} \to \mathbb{N}$ is called a* resource function *if:*

- *f is non-decreasing,*
- *There is a polynomial time algorithm which, on input 1^n, outputs the value $f(n)$.*

2.1 Hardness vs. Randomness

We represent a circuit C by a string w containing: the number of vertices of C, for each vertex its label (input, output,internal), for internal gates their type (And,Or,Not) and the vertices to which they are connected. We say w *represents* C and we say C has size a if it has a representation w with $|w| \leq a$.

Definition 9. *(Hardness) The* hardness *of a string $w \in \{0,1\}^M$ is the size s of the smallest circuit C such that w is the truth table of C, i.e., the circuit C has $\log M$ inputs and $C(i) = w_i$ for every $i \in [M]$. We write $\mathsf{Hard}(w) = s$.*

Definition 10. *[Kab01](The easy witness generator) Given two numbers $\log M \leq s \leq M$,*

$$\mathsf{EW}_{s \to M} = \left\{x \in \{0,1\}^M \mid \mathsf{Hard}(x) \leq s\right\}.$$

Remark 11. *One can enumerate all strings in $\mathsf{EW}_{s \to M}$ in time $\mathrm{poly}(2^s \cdot M \cdot s) = 2^{O(s)}$.*

Definition 12. *(Hitting Set Generator) A family of functions $H = \left\{H_n : \{0,1\}^{\ell(n)} :\to \{0,1\}^n\right\}$ is called a hitting set generator (HSG) against a class of circuits $\mathcal{C}$ if for every circuit $C \in \mathcal{C}$ that accepts at least $\frac{1}{2}$ of the inputs of length n, there exists a seed $z \in \{0,1\}^{\ell(n)}$ such that $C(H_n(z)) = 1$.*

Definition 13. *(Pseudo Random Generator) A family of functions $G = \left\{G_n : \{0,1\}^{\ell(n)} :\to \{0,1\}^n\right\}$ is called an ε pseudo random generator (PRG) against a class of circuits $\mathcal{C}$ if for every circuit $C \in \mathcal{C}$*

$$\left| \Pr_{x\in\{0,1\}^n}[C(x) = 1] - \Pr_{s\in\{0,1\}^{\ell(n)}}[C(G(s)) = 1] \right| < \varepsilon$$

Theorem 14. *[Uma02, Theorem 1] There exists a constant $\alpha > 0$ such that the following holds. Suppose $w \in \{0,1\}^{2^m}$ and $\mathsf{Hard}(w) \geq s$. Then there exists a function $\mathsf{PRG}_w : \{0,1\}^{O(m)} \to \{0,1\}^T$ that is a T^{-1}-PRG against size T^2 circuits, for*

$$T = s^\alpha$$

Moreover, PRG *is computable in time* $\mathrm{poly}(|w|)$ *with oracle calls to* w.

As we explained in the introduction, we wish to use the PRGs against the class $\mathcal{C}$ of linear-size circuits and tiny description size. We define:

Definition 15. *(Time-bounded Kolmogorov complexity) The Kolmogorov complexity of a string $w \in \{0,1\}^*$ is the size of the smallest TM that outputs w in time $O(|w|)$ when run on the empty string.*

Definition 16. *We let $\mathsf{Circ}(n, s, a)$ be the set of all circuits on n input bits, of size at most s such that the representation of C has time-bounded Kolmogorov complexity at most a.*

2.2 Diagonalization Theorems

Definition 17. *Let $T, t : \mathbb{N} \to \mathbb{N}$ be two constructible functions. We say $T(n) \gg t(n)$ if the universal Turing machine can simulate in time $T(n)$ a Turing machines running in time $t(n)$.*

We will state two diagonalization theorems. The first is the standard time hierarchy for i.o. deterministic time:

Theorem 18. *Let $T, t : \mathbb{N} \to \mathbb{N}$ be two constructible functions and assume $T(n) \gg t(n)$. Then:*

$$\mathsf{DTime}(T(n)) \not\subseteq \text{i.o.}\mathsf{DTime}(t(n))$$

For completeness we give a proof in section 4.1.

Additionally, we state a ZTime hierarchy against ZTime with short advice. We follow the techniques of [FST05, Thm 12] that were originally applied for NTime. We mention that the more sophisticated techniques of [FS14] allow diagonalization against longer advice, but we do not require these stronger results. We could not find the particular statement that we need in the literature and we therefore prove theorem 19 following [FST05, Thm 12].

Theorem 19. *(Following [FST05, Thm 12]) Let $T, t : \mathbb{N} \to \mathbb{N}$ be two time-constructible, monotone functions such that $t(n) \geq n$, $T(\frac{n}{2}) \geq t(n) \log t(n)$ and $T(n) > 2^{3\cdot t(2\cdot \log^3 n)}$ then:*

$$\Pr\text{−}\mathsf{ZTime}(T(n)) \quad \not\subseteq \quad \Pr\text{−}\mathsf{ZTime}(t(n)) \ / \ \log^2 n$$

We prove this theorem in section 4.2.

3 The easy witness method revisited

In this section we redo the easy witness argument. We follow the same argument as in [Kab01, IKW02] except for the following two changes:

- In one of the alternatives we have that $\mathsf{RTime}(T(n))$ has easy witnesses for a very large function T,
- We introduce a parameter measuring the Kolmogorov complexity of a family of circuits, and we record how the two alternatives interact with this parameter.

In the following theorem we are given three functions $s, \beta, T : \mathbb{N} \to \mathbb{N}$. $s(M)$ is the hardness we need in the string. By [Uma02] given a string with hardness s, we can create a PRG outputting $out(M) = s^{\alpha}$ bits, for some constant α, see Theorem 14. Throughout the section we fix this constant α. $\beta(M)$ is the measuring the Kolmogorov complexity of the circuit class we work with (or against) and T is the running time of the RTime algorithm we wish to derandomize. In what follows we explore how the large T affects the other parameters, and β and the advice size in particular.

Theorem 20. *Fix α as in Theorem 14. Let $s, \beta, T : \mathbb{N} \to \mathbb{N}$. Denote $out(M) = s(M)^{\alpha}$. We assume:*

- $\log M \le s(M) \le M$, $n \le T(n)$, $\beta(T(n)) > 10n$,
- *T, out, β are monotone increasing and tend to infinity and T is a resource function.*

Then:

- *If there exists an infinite subset $I \subseteq \mathbb{N}$ such that for all $M \in I$, $\mathsf{EW}_{s(M) \to M}$ is a HSG against $\mathsf{Circ}(M, M^2, \beta(M))$, then*

$$\mathrm{Pr}\!-\!\mathsf{RTime}\,(T(n)) \subseteq \mathrm{i.o.Pr}\!-\!\mathsf{DTime}\left(2^{O(s(T(n+1)))}\right).$$

- *If there exists some $M_{start} \in \mathbb{N}$ such that for all $M \geqslant M_{start}$, $\mathsf{EW}_{s(M) \to M}$ is not a HSG against $\mathsf{Circ}(M, M^2, \beta(M))$, then there exists a constant d such that:*

$$\mathrm{Pr}\!-\!\mathsf{BPTime}\,(n) \subseteq \mathrm{Pr}\!-\!\mathsf{ZTime}\left((out^{-1}(n))^{d}\right) \;/\; \beta(out^{-1}(n)).$$

Proof.

The first item: For the first part of the lemma, let $L \in \mathrm{Pr}\!-\!\mathsf{RTime}\,(T(n))$. Denote by A the probabilistic algorithm solving L. We construct a deterministic algorithm D as follows: On input $x \in \{0,1\}^n$, D accepts iff $A(x,z) = 1$ for some $z \in \mathsf{EW}_{s(M) \to M}$ for some integer $M \in [T(n), T(n+1))$.

Running time: For a given M, D can enumerate $\mathsf{EW}_{s(M) \to M}$ in $2^{O(s(M))}$ time. There are at most $T(n+1)$ possible values for M, and given M, z the simulation takes at most $\mathrm{poly}(T(n+1))$ time. Thus, altogether, D runs in time $2^{O(s(M))}\mathrm{poly}(T(n+1)) = 2^{O(s(T(n+1)))}$.

Soundness: For $x \in \mathsf{NO}(L)$, for every y, $A(x,y) = 0$, hence the algorithm rejects.

Completeness: By assumption I is infinite. Hence, there are infinitely many input lengths n for which $[T(n), T(n+1)] \cap I \neq \emptyset$. Fix such an input length n. Fix $M \in [T(n), T(n+1)] \cap I$. Fix $x \in \{0,1\}^n \cap \mathsf{YES}(L)$. Define the circuit C_x where $C_x(y) = A(x,y)$.

- $\Pr_y[C_x(y)=1] = \Pr_y[A(x,y)=1] \geq \frac{1}{2}$,
- By standard translation methods we can assume that the size of C_x is at most $T(n)^2$ (see, for example, [Sip96, Theorem 9.30]).
  It is easy to see, therefore, that $C_x \in \mathsf{Circ}(O(T(n), T(n)^2, 10n) \subseteq \mathsf{Circ}(M, M^2, \beta(M))$.

Hence, by the HSG property, for some $z \in \mathsf{EW}_{s(M)\to M}$ we have $A(x,z)=1$ and the algorithm accepts.

Hence $L \in \text{i.o.Pr}-\mathsf{DTime}\left(2^{O(s(T(n+1)))}\right)$.

The second item: We now proceed to the second item. By assumption, for all $M > M_{start}$ there exists a circuit $\mathsf{DIST}_M \in \mathsf{Circ}(M, M^2, \beta(M))$ such that $\Pr_{y\in\{0,1\}^M}[\mathsf{DIST}_M(y)=1] \geq \frac{1}{2}$ and $\mathsf{DIST}_M(w)=0$ for all $w \in \mathsf{EW}_{s(M)\to M}$.

Let $L \in \text{Pr}-\mathsf{BPTime}(n)$ solvable by $A(x,y)$ with $|y|=n$. For every $n \geq M_{start}$ set M to be the first integer such that

$$out(M) \geq n.$$

Notice that $M \geq n \geq M_{start}$. For input length n we set the advice to be DIST_M. The advice size is the description size of DIST_M, which is $\beta(M) = \beta(out^{-1}(n))$. We denote $\gamma(n) = \beta(out^{-1}(n))$.

Define the circuit $C_x(y) = A(x,y)$ which has size n^2.

We now describe a zero-error probabilistic algorithm solving L given the advice. Let $x \in \{0,1\}^n$.

1. Choose uniformly $y \in \{0,1\}^M$,
2. If $\mathsf{DIST}_M(y)=0$ quit, otherwise continue. If we continue we know for sure that
$$\mathsf{Hard}(y) \geq s(M)$$
3. Denote by $\mathsf{PRG}_y : \{0,1\}^{O(\log M)} \to \{0,1\}^{out(M)}$ the PRG given by theorem 14. Notice that $out(M) \geq n$. Run $C_x(\mathsf{PRG}_y(z))$ over all $z \in \{0,1\}^{O(\log M)}$ and accept iff for the majority of z, $C_x(\mathsf{PRG}_y(z))=1$.

Running time: We note:

- Drawing $y \in \{0,1\}^M$ requires M time
- Constructing DIST_M from its description requires $\text{poly}(M)$ time
- Given y, z, we can compute $\mathsf{PRG}_y(z)$ in time $\text{poly}(|y|) = \text{poly}(M)$
- Given $x, \mathsf{PRG}_y(z)$ we compute $C_x(\mathsf{PRG}_y(z))$ in time n,
- We do the above for each $z \in \{0,1\}^{O(\log M)}$

Thus, the running time of the algorithm is

$$\text{poly}(M) \cdot n = \text{poly}(out^{-1}(n)))$$

Correctness: By construction, $\Pr_{y\in\{0,1\}^M}[\mathsf{DIST}_M(y)=1] \geq \frac{1}{2}$ and so with probability at least half we get $y \in \{0,1\}^M$ such that we know for sure that $\mathsf{Hard}(y) \geq s(M)$. Hence,

$$\mathsf{PRG}_y : \{0,1\}^{O(m)} \to \{0,1\}^T$$

is a T^{-1}-PRG against circuits of size $T^2 \geqslant n^2$, for $T = s(M)^\alpha = out(M)$. By the PRG property we have that for most z, $C_x(\mathsf{PRG}_y(z))$ is correct, and the algorithm is correct.

□

3.1 Instantiating parameters

Theorem 20 measures complexity with $\beta(M)$ that measures description size and $T(n)$ that corresponds to the running time of the RTime machine. From our point of view it is more natural to measure things with respect to the advice length $a(n)$ and the required hardness $s(M)$ in the witness. We claim:

Theorem 21. *Let $a, s : \mathbb{N} \to \mathbb{N}$ be two resource functions, $\alpha \log n < a(n) \le n$, $\log n \le s(n) \le n$. Set $T(n) = s^{-1}(a^{-1}(10n))$. Then either*

- $\mathrm{Pr-}\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.Pr-}\mathsf{DTime}(\mathrm{poly}(2^{a^{-1}(10n)}))$*, or else,*
- $\mathrm{Pr-}\mathsf{BPTime}(n) \subseteq \mathrm{Pr-}\mathsf{ZTime}(\mathrm{poly}(s^{-1}(n^{1/\alpha})))\ /\ a(n^{1/\alpha})$.

Proof. Set

$$\begin{aligned} \beta(m) &= a(s(m)), \\ out(M) &= s(M)^{\alpha} \end{aligned}$$

Clearly, $\beta, T, out, a^{-1}, s^{-1}, \beta^{-1}, T^{-1}, out^{-1}$ are monotone increasing. $n \le s^{-1}(n) \le 2^n$, $n \le a^{-1}(n) \le 2^{\frac{1}{\alpha}n}$, $T(n) \ge n$. Also, $\beta(T(n)) = a(s(T(n)) = a(s(s^{-1}(a^{-1}(10n)))) = 10n$. Then, by Theorem 20, at least one of the two alternatives of Theorem 20 happens.

- The first alternative of Theorem 20 gives the first alternative of our theorem, because $s(T(n+1)) = a^{-1}(10(n+1)) = O(a^{-1}(10n)) = O(s(T(n)))$.
- For the second alternative, we first notice that $out^{-1}(n) = s^{-1}(n^{1/\alpha})$ because

  $$out(s^{-1}(n^{1/\alpha})) \quad = \quad (s(s^{-1}(n^{1/\alpha}))^{\frac{1}{\alpha}} = n.$$

  Finally, $\beta(out^{-1}(n)) = \beta(s^{-1}(n^{1/\alpha})) = a(s(s^{-1}(n^{1/\alpha}))) = a(n^{1/\alpha})$.

□

We now fix any monotone increasing above half-exponential time-constructible function H.

Theorem 22. *Let a be as in Theorem 21. Set $T(n) = H(a^{-1}(10n))$. Either*

- $\mathrm{Pr-}\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.Pr-}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$*, or,*
- $\mathrm{Pr-}\mathsf{BPTime}(n) \subseteq \mathrm{Pr-}\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))\ /\ a(n^{1/\alpha})$.

Proof. Set $s = H^{-1}$. By Theorem 21, either

- $\mathrm{Pr-}\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.Pr-}\mathsf{DTime}(\mathrm{poly}(2^{a^{-1}(10n))}) \subseteq \mathrm{i.o.Pr-}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$,
- Or else, $\mathrm{Pr-}\mathsf{BPTime}(n) \subseteq \mathrm{Pr-}\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))\ /\ a(n^{1/\alpha})$.

□

We aim for logarithmic advice. We therefore set

$$a(n) = \alpha \log n.$$

This implies $a^{-1}(n) = 2^{\frac{1}{\alpha}n}$ and $a(n^{1/\alpha}) = \log n$. Plugging a into Theorem 22 we get:

Theorem 23. *Let α be as in Theorem 14. Set $T(n) = H(2^{\frac{10}{\alpha}n})$. Either*

- $\mathrm{Pr}{-}\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.Pr}{-}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$*, or,*
- $\mathrm{Pr}{-}\mathsf{BPTime}(n) \subseteq \mathrm{Pr}{-}\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))\ /\ \log n$.

This proves Theorem 3.

Remark 24. *We note that Theorem 23 is also true if we remove the* $\mathrm{Pr}-$ *quantifier from all clauses in the theorem. To see that notice that:*

In the first alternative:

$$\mathrm{Pr}{-}\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.Pr}{-}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$$

implies

$$\mathsf{RTime}(T(n)) \subseteq \mathrm{i.o.}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$$

because if $L \in \mathsf{RTime}(T(n))$, then, in particular, $L \in \mathrm{Pr}{-}\mathsf{RTime}(T(n))$, hence

$$L \in \mathrm{i.o.Pr}{-}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$$

i.e., there exists a TM M that i.o. solves L and then the machine M puts L in $\mathrm{i.o.}\mathsf{DTime}(\mathrm{poly}(H(T(n)))$*. Also,*

In the second alternative:

$$\mathrm{Pr}{-}\mathsf{BPTime}(n) \subseteq \mathrm{Pr}{-}\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))/\log n$$

implies

$$\mathsf{BPTime}(n) \subseteq \mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))/\log n$$

because

$$\mathsf{BPTime}(n) \subseteq \mathrm{Pr}{-}\mathsf{BPTime}(n)$$

and a total language L in $\mathrm{Pr}{-}\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))/\log n$ *is in* $\mathsf{ZTime}(\mathrm{poly}(H(n^{1/\alpha})))/\log n$*.*

4 Diagonalization Theorems

4.1 Diagonalizing i.o. deterministic time

We recall Theorem 18 which is the standard time hierarchy for i.o. deterministic time, and give a proof for completeness.

Theorem 18. *Let $T, t : \mathbb{N} \to \mathbb{N}$ be two constructible functions and assume $T(n) \gg t(n)$. Then:*

$$\mathsf{DTime}(T(n)) \nsubseteq \mathrm{i.o.}\mathsf{DTime}(t(n))$$

Proof. Let M_i be a recursive enumeration of deterministic Turing Machines running in time $t(n)$. We define a Turing machine N that on input $x \in \{0,1\}^n$ does the following: If x is of the form 0^i1^{n-i}, N simulates $M_i(x)$ for $t(n)$ time, and if it reaches the end of the simulation, N answers the opposite answer. Otherwise, N rejects the input. Assume towards contradiction that there exists an index j such that the machine M_j agrees with N on infinitely-many input lengths. In particular, there exists $n > j$ such that $M_j(x) = N(x)$ for all $x \in \{0,1\}^n$. But by the construction of N, $N(0^j1^{n-j}) \neq M_j(0^j1^{n-j})$ which is a contradiction. $\square$

4.2 Diagonalizing ZTime with small advice

We recall Theorem 19 and prove it following [FST05, Thm 12].

Theorem 19. *(Following [FST05, Thm 12]) Let $T, t : \mathbb{N} \to \mathbb{N}$ be two time-constructible, monotone functions such that $t(n) \geq n$, $T(\frac{n}{2}) \geq t(n) \log t(n)$ and $T(n) > 2^{3 \cdot t(2 \cdot \log^3 n)}$ then:*

$$\text{Pr}-\mathsf{ZTime}(T(n)) \quad \not\subseteq \quad \text{Pr}-\mathsf{ZTime}(t(n)) \;/\; \log^2 n$$

Proof. We prove by delayed diagonalization, following the argument of [FST05]. We construct a TM N that works on inputs of the form

$$w_{i,m,k}^{a_1,\ldots,a_m} = 1^i 0 1^m 0 1^{2^k} 0 a_1, \ldots, a_m$$

where $i, k \leqslant m$ and $a_1, \ldots, a_m$ are arbitrary strings each of length $(m+1)^2$. For $i, k, m, a_1, \ldots, a_m$ denote:

$$n_{i,m,k} \quad = \quad |w_{i,m,k}^{a_1,\ldots,a_m}| \; = \; i + m + 2^k + m(m+1)^2 + 3.$$

In particular, for m large enough,

$$\begin{aligned} n_{i,m,1} &\leqslant 2m^3, \\ n_{i,m,k+1} &\leq 2n_{i,m,k}, \text{ and,} \\ n_{i,m,m} &\in [2^m, 2^{m+1}). \end{aligned}$$

The machine N works as follows:

- On input of the form $w_{i,m,m}^{a_1,\ldots,a_m}$, for each y, N simulates $M_i(w_{i,m,1}^{a_1,\ldots,a_m}, y)$ for at most $t(|w_{i,m,1}^{a_1,\ldots,a_m}|)$ steps. If for some y the simulation takes longer, N returns 0. Otherwise, N determines the probability

  $$p = \Pr_y[M_i(w_{i,m,1}^{a_1,\ldots,a_m}, y) = 1]$$

  when using the first $\log^2 n_{i,m,m} < (m+1)^2$ bits from a_m as advice. If $p > \frac{1}{2}$, N outputs 0 with probability 1 and otherwise N outputs 1 with probability 1.

- On input of the form $w_{i,m,k}^{a_1,\ldots,a_m}$ for $1 \leqslant k < m$, N simulates M_i on input $w_{i,m,k+1}^{a_1,\ldots,a_m}$ using the first $\log^2 n_{i,m,k+1} < (m+1)^2$ bits of a_{k+1} as advice, and answers the same answer.

We put inside the promise of N exactly all the inputs s on which N behaves as a zero-sided error machine.

Claim 25. *For all $i, m, a_1, \ldots, a_m$, the input $w_{i,m,m}^{a_1,\ldots,a_m}$ belongs to the promise of N.*

Proof. Set $n_1 = n_{i,m,1}$ and $n_m = n_{i,m,m}$. N simulates M_i at most $2^{t(n_1)}$ times (for each $y \in \{0,1\}^{t(n_1)}$) and each such simulation takes $t(n_1)\log(t(n_1))$ time. Since $n_m \geqslant 2^m$ and $n_1 \leqslant 2m^3$ we have

$$T(n_m) \geq T\left(2^{\left(\frac{n_1}{2}\right)^{1/3}}\right) \geq 2^{3 \cdot t(n_1)} \geq 2^{t(n_1)} \cdot t(n_1) \log t(n_1),$$

thus N has enough time to finish the simulation, and therefore returns a definite answer with probability one. Hence $w_{i,m,m}^{a_1,\ldots,a_m}$ belongs to the promise. □

Claim 26. *Suppose $M_i \in \mathrm{Pr}{-}\mathsf{ZTime}(t(n)) \ / \ \log^2 n$ solves the promise language of N. Let $m \geq i$ large enough and let a_k be the advice string for inputs of length $n_{i,m,k}$, completed arbitrarily to strings of length $(m+1)^2$. Then the inputs $w_{i,m,k}^{a_1,\dots,a_m}$, for all $1 \leq k \leq m$, belong to the promise of N (and therefore also to the promise of M_i).*

Proof. Claim 25 guarantees that $w_{i,m,m}^{a_1,\dots,a_m}$ belongs to the promise of N. Hence $w_{i,m,m}^{a_1,\dots,a_m}$ also belongs to the promise of M_i.

For any $1 < k \leqslant m$ set $n_k = n_{i,m,k}$. Now suppose $w_{i,m,k}^{a_1,\dots,a_m}$ belongs to the promise of N and M_i, and let us prove that $w_{i,m,k-1}^{a_1,\dots,a_m}$ belongs to the promise of N, and hence also to the promise of M_i. On input $w_{i,m,k-1}^{a_1,\dots,a_m}$ N simulates M_i on $w_{i,m,k}^{a_1,\dots,a_m}$ and answers the same. The time it takes N to prepare the input $w_{i,m,k}^{a_1,\dots,a_m}$ and do the simulation is $t(n_k)\log t(n_k) \leq t(2n_{k-1})\log t(2n_{k-1})) \leq T(n_{k-1})$. Hence N has enough time to finish the simulation and return the same answer. It follows that since $w_{i,m,k}^{a_1,\dots,a_m}$ belongs to the promise of M_i, $w_{i,m,k-1}^{a_1,\dots,a_m}$ belongs to the promise of N. □

Assume towards contradiction that there exists i such that the machine M_i calculates the same promise language as N. Let $m, a_1, \dots, a_m$ be as in Claim 26. By Claims 25 and 26 the inputs $w_{i,m,k}^{a_1,\dots,a_m}$ belong to the promise of both M_i and N for all $1 \leq k \leq m$. As M_i with the correct advice is correct on the promise of N, M_i and N answer the same on all inputs $w_{i,m,k}^{a_1,\dots,a_m}$ for all $1 \leq k \leq m$. Hence, by construction,

$$\begin{aligned} N(w_{i,m,m}^{a_1,\dots,a_m}) &= \neg M_i(w_{i,m,1}^{a_1,\dots,a_m}) = \neg N(w_{i,m,1}^{a_1,\dots,a_m}), \text{ and,} \\ N(w_{i,m,k+1}^{a_1,\dots,a_m}) &= M_i(w_{i,m,k}^{a_1,\dots,a_m}) = N(w_{i,m,1}^{a_1,\dots,a_m}), \text{for all } k = 0,\dots,m-1. \end{aligned}$$

This implies $N(w_{i,m,1}^{a_1,\dots,a_m}) = \neg N(w_{i,m,1}^{a_1,\dots,a_m})$ - a contradiction. □

5 The separation result

5.1 Another look at half-exponential functions

We recall Definition 7: A function $S : \mathbb{N} \to \mathbb{N}$ is above half exponential if $H(H(n)) > 2^n$ for sufficiently large n. We first claim that a half exponential function is sandwiched between iterated quasi-poly and iterated quasi-subexp as we now explain.

Definition 27. *(Iterated Log and iterated Exp) We define $\mathsf{Log}\,(k,n)$ and $\mathsf{Exp}\,(k,n)$ inductively on k:*

- *For $k = 0$: $\mathsf{Log}\,(0,n) = n$ and $\mathsf{Exp}\,(0,n) = n$*
- *For $k \geqslant 1$: $\mathsf{Log}\,(k,n) = \log(\mathsf{Log}\,(k-1,n))$ and $\mathsf{Exp}\,(k,n) = 2^{\mathsf{Exp}(k-1,n)}$*

All logarithms are to the base 2.

Let

$$\begin{aligned} B_k(n) &= \mathsf{Exp}\,(k, 2\mathsf{Log}\,(k,n)), \\ U_k(n) &= B_k^{-1}(2^n). \end{aligned} \tag{2}$$

For example $B_1(n) = 2^{2\log n} = n^2$, $B_2(n) = 2^{2^{2\log\log n}} = 2^{\log^2 n} = n^{\log n}$. In general the B_k sequence gets larger with k. B_1 is polynomial, B_2 quasi-polynomial, and we might say B_3 is quasi-quasi polynomial.

Also,it is easy to check that

$$U_k(n) \quad = \quad \mathsf{Exp}\left(k, \frac{1}{2}\mathsf{Log}\,(k-1, n)\right).$$

Thus, $U_1(n) = 2^{n/2}$, $U_2(n) = 2^{2^{\frac{1}{2}\log n}} = 2^{\sqrt{n}}$. The U_k sequence is monotonically decreasing with k, with U_1 being exponential, U_2 sub-exponential and so forth. B_k and U_k are kind of inverse to each other using Eq (2) showing that $B_k(U_k(n)) = 2^n$.

In Appendix A we prove:

Claim 28. *For any constant $k \geqslant 3$, $U_k(n)$ is sub-exponential and* above half-exponential.

Because of small overheads we have in the reductions we do, we will also require a modification of U_k. We define

$$W_k(n) \quad \stackrel{\text{def}}{=} \quad \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\,(k-1, n)\right). \tag{3}$$

We prove in Appendix A:

Claim 29. *For any constant $k \geq 3$, the function W_k is sub-exponential and* above half-exponential.

and:

Claim 30. *For any constant $k \geqslant 4$ and sufficiently large n,*

$$\begin{aligned} W_k(m) \quad &\geq \quad U_k(m^{\log m}), \\ \log W_k(m) \quad &\geq \quad W_k(\log^4 m). \end{aligned}$$

5.2 The separation

We now prove our main theorem:

Theorem 31. *(Main) With the notation above, for any constant $k \geqslant 4$,*

$$\mathrm{Pr-}\mathsf{ZTime}\,(\mathrm{poly}(W_k(n))) \nsubseteq \mathrm{Pr-}\mathsf{RP}.$$

Proof. Fix a constant $k \geqslant 4$. Let $H(n) = U_k(n)$ and $T(n) = U_k(2^{\frac{10}{\alpha}n})$. As U_k is above half-exponential, there exists some constant $c \geqslant 1$ such that one of the two alternatives of Theorem 23 happens. We consider both cases of Theorem 23:

If alternative 1 in Theorem 23 holds : By our assumption and Remark 24

$$\mathsf{RTime}(T(n)) \quad \subseteq \quad \text{i.o.}\mathsf{DTime}(U_k^c(T(n))).$$

By Theorem 18 there exists $c' > c$ such that

$$\mathsf{DTime}(U_k^{c'}(T(n))) \quad \not\subseteq \quad \mathsf{DTime}(U_k^{c}(T(n)))$$

Hence,

$$\mathsf{DTime}(U_k^{c'}(T(n))) \quad \not\subseteq \quad \mathsf{RTime}(T(n)).$$

Choose $m = m(n)$ such that $m^{\log m} = T(n)$. Then, using padding and that separations go down we get:

$$\mathsf{DTime}(U_k^{c'}(m^{\log m})) \quad \not\subseteq \quad \mathsf{RTime}(m^{\log m}).$$

However, by Claim 30, $U_k(m^{\log m}) \leq W_k(m)$. Hence,

$$\mathsf{DTime}(W_k^{c'}(m)) \quad \not\subseteq \quad \mathsf{RTime}(m^{\log m}),$$

and, in particular, $\mathsf{DTime}(\text{poly}(W_k)) \not\subseteq \mathsf{RP}$. It now follows by definition that $\text{Pr}{-}\mathsf{DTime}(\text{poly}(W_k)) \not\subseteq \text{Pr}{-}\mathsf{RP}$ as demonstrated by the total language in $\mathsf{DTime}(\text{poly}(W_k))$ not in RP.

If alternative 2 in Theorem 23 holds : By our assumption, for some $c \geqslant 1$,

$$\text{Pr}{-}\mathsf{BPTime}(n) \quad \subseteq \quad \text{Pr}{-}\mathsf{ZTime}(U_k^c(n^{1/\alpha}))/\log n,$$

By a padding argument

$$\begin{aligned} \text{Pr}{-}\mathsf{BPP} \quad &\subseteq \quad \text{Pr}{-}\mathsf{ZTime}\left(\bigcup_d U_k(n^{d/\alpha})^c\right) \ / \ d \cdot \log n \\ &\subseteq \quad \text{Pr}{-}\mathsf{ZTime}(U_k(n^{\log n})) \ / \ \log^2 n \end{aligned}$$

Define

$$\begin{aligned} t(n) \quad &= \quad U_k(n^{\log n}) \\ T(n) \quad &= \quad W_k(n) \end{aligned}$$

We have

$$\begin{aligned} T(n) \quad &= \quad W_k(n) && \text{By definition of } T \\ &\geq \quad 2^{W_k(\log^4 n)} && \text{By Claim 30} \\ &\geq \quad 2^{U_k((\log^4 n)^{4\log\log n})} && \text{By Claim 30} \\ &= \quad 2^{t(\log^4 n)} && \text{By definition of } t \\ &\geq \quad 2^{3t(2\log^3 n)}. \end{aligned}$$

Hence by Theorem 19 we have that $\text{Pr}{-}\mathsf{ZTime}(W_k(n)) \not\subseteq \text{Pr}{-}\mathsf{RP}$.

□

Combining claim 29 and theorem 31 we see that $W_4(n)$ is subexponential and that

$$\text{Pr}{-}\mathsf{ZTime}(\text{poly}(W_4(n))) \not\subseteq \text{Pr}{-}\mathsf{RP}$$

since subexponential functions are closed under polynomial growth we have proved Theorem 1.

References

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 194–208. Springer, 2002. 6

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 316–324. IEEE, 2004. 6

[FS14] Lance Fortnow and Rahul Santhanam. Hierarchies against sublinear advice. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 171, 2014. 6, 8

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 348–355. ACM, 2005. 6, 8, 13

[Hel86] Hans Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71(3):231–243, 1986. 1

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within np. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 247–258. IEEE, 2018. 5

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002. 1, 4, 5, 9

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 220–229, New York, NY, USA, 1997. ACM. 1, 2

[Kab01] Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001. 1, 2, 3, 4, 5, 7, 9

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 1

[MV96] Andrei A Muchnik and Nikolai K Vereshchagin. A general method to construct oracles realizing given relationships between complexity classes. *Theoretical Computer Science*, 157(2):227–258, 1996. 1

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. 2

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994. 1, 2

[Rac82] Charles Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the ACM (JACM)*, 29(1):261–268, 1982. 1

[SCR+20] R Santhanam, L Chen, N Rajgopal, Ján Pich, IC Oliveira, and S Hirahara. Beyond natural proofs: Hardness magnification and locality. *Leibniz International Proceedings in Informatics*, 151, 2020. 5

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996. 10

[Uma02] Christopher Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 627–634, New York, NY, USA, 2002. ACM. 8, 9

[Wil14] Ryan Williams. Nonuniform acc circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):1–32, 2014. 5

[Wil16] R Ryan Williams. Natural proofs versus derandomization. *SIAM Journal on Computing*, 45(2):497–529, 2016. 5, 6

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982. 1

A Quasi-poly and Quasi-subexp functions

Let us define

$$F_{k,c}(n) \quad = \quad \mathsf{Exp}\left(k, c \cdot \mathsf{Log}\left(k-1, n\right)\right),$$

so that $U_k = F_{k,\frac{1}{2}}$ and $W_k = F_{k,\frac{2}{3}}$. We claim:

Claim 32. *For constants $k \geq 3$, $0 < c < 1$, $F_{k,c}$ is sub-exponential and above half-exponential.*

This proves Claims 28 and 29.

Proof. We first notice that $F_{k+1,c}(n) < F_{k,c}(n)$ for sufficiently large n. To see that notice that:

$$\begin{aligned} F_{k,c}(n) &= \mathsf{Exp}\left(k, c \cdot \mathsf{Log}\left(k-1, n\right)\right), \text{ and} \\ F_{k+1,c}(n) &= \mathsf{Exp}\left(k+1, c \cdot \mathsf{Log}\left(k, n\right)\right) = \mathsf{Exp}\left(k, 2^{c \cdot \mathsf{Log}(k,n)}\right) = \mathsf{Exp}\left(k, \left(\mathsf{Log}\left(k-1, n\right)\right)^c\right). \end{aligned}$$

and the claim follows, because for every fixed k, $(\mathsf{Log}\,(k-1,n))^c \leq c \cdot \mathsf{Log}\,(k-1,n)$ for large enough n.

$F_{3,c}(n) = 2^{2^{2^{c \cdot \log\log n}}} = 2^{2^{\log^c n}}$ is sub-exponential because for any $\varepsilon > 0$ we have $2^{\log^c n} < n^{\varepsilon} = 2^{\varepsilon \log n}$ for n large enough. Hence for any $k \geq 3$, $F_{k,c}$ is sub-exponential because $F_{k,c} \leq F_{3,c}$ for n large enough. Also,

$F_{k,c}(F_{k,c}(n)) \geqslant 2^n$ **for any constant** $k \geqslant 2$: We have:

$$\begin{aligned} F_{k,c}(F_{k,c}(n)) &= F_k(\mathsf{Exp}\left(k, c \cdot \mathsf{Log}\left(k-1, n\right)\right)) \\ &= \mathsf{Exp}\left(k, c \cdot \mathsf{Log}\left(k-1, \mathsf{Exp}\left(k, c \cdot \mathsf{Log}\left(k-1, n\right)\right)\right)\right) \\ &= \mathsf{Exp}\left(k, c \cdot \mathsf{Exp}\left(1, c \cdot \mathsf{Log}\left(k-1, n\right)\right)\right) \\ &= \mathsf{Exp}\left(k, c \cdot \left(\mathsf{Log}\left(k-2, n\right)\right)^c\right). \end{aligned}$$

Also,

$$2^n = \mathsf{Exp}\,(k, \mathsf{Log}\,(k-1, n))\,.$$

The claim follows since $c \cdot (\mathsf{Log}\,(k-2, n))^c \geq \mathsf{Log}\,(k-1, n)$ for large enough n as $c \cdot x^c \geq \log x$ for $c > 0$ and large enough x.

□

A.1 W_k

The sequence W_k was defined in Eq (3) by

$$W_k(n) \overset{\text{def}}{=} \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\,(k-1, n)\right). \tag{3}$$

We first prove Claim 30 which we restate now:

Claim 30. *For any constant $k \geqslant 4$ and sufficiently large n,*

$$\begin{aligned} W_k(m) &\geq U_k(m^{\log m}), \\ \log W_k(m) &\geq W_k(\log^4 m). \end{aligned}$$

Proof. For the first item,

$$\begin{aligned} U_k(m^{\log m}) &= \mathsf{Exp}\left(k, \frac{1}{2}\mathsf{Log}\left(k-1, m^{\log m}\right)\right) = \mathsf{Exp}\left(k, \frac{1}{2}\mathsf{Log}\left(k-2, (\log m)^2\right)\right) \\ &= \mathsf{Exp}\left(k, \frac{1}{2}\mathsf{Log}\,(k-3, 2\log\log m)\right) = \mathsf{Exp}\left(k, \frac{1}{2}\mathsf{Log}\,(k-4, \log\log\log m + 1)\right) \\ &\leq \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\,(k-4, \log\log\log m)\right) \\ &= \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\,(k-1, m)\right) = W_k(m). \end{aligned}$$

For the second item,

$$\begin{aligned} \log W_k(m) &= \log \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\,(k-1, m)\right) = \mathsf{Exp}\left(k-1, \frac{2}{3}\mathsf{Log}\,(k-1, m)\right) \\ &= \mathsf{Exp}\left(k-1, \frac{2}{3}\mathsf{Log}\,(k-3, \log\log m)\right) \\ &\geqslant \mathsf{Exp}\left(k-1, (4\mathsf{Log}\,(k-3, \log\log m))^{2/3}\right) \\ &> \mathsf{Exp}\left(k-1, (\mathsf{Log}\,(k-3, 4\log\log m))^{2/3}\right) \\ &= \mathsf{Exp}\left(k-1, (\mathsf{Log}\left(k-2, \log^4 m\right))^{2/3}\right) \\ &= \mathsf{Exp}\left(k, \frac{2}{3}\mathsf{Log}\left(k-1, \log^4 m\right)\right) = W_k(\log^4 m), \end{aligned}$$

where the first inequality is because $\frac{2}{3}x \geqslant (4x)^{2/3}$ for x large enough, and the second inequality is because $\mathsf{Log}\,(k', 4x) < 4\mathsf{Log}\,(k', x)$ for any fixed $k' \geqslant 1$ and sufficiently large x. □