Electronic Colloquium on Computational Complexity, Report No. 51 (2006)

# Infinitely-Often Universal Languages and Diagonalization

Alan Nash
Department of Mathematics
anash@math.ucsd.edu

Russell Impagliazzo
Department of Computer Science and Engineering
russell@cs.ucsd.edu

Jeff Remmel
Department of Mathematics
remmel@math.ucsd.edu

University of California, San Diego, CA 92093

## Abstract

*Diagonalization is a powerful technique in recursion theory and in computational complexity [2]. The limits of this technique are not clear. On the one hand, many people argue that conflicting relativizations mean a complexity question cannot be resolved using only diagonalization. On the other hand, it is not clear that diagonalization arguments necessarily relativize. In [5], the authors proposed a definition of "separation by strong diagonalization" in which to separate class $\mathcal{A}$ from $\mathcal{B} \subseteq \mathcal{A}$ a proof is required that $\mathcal{A}$ contains a* universal language *for $\mathcal{B}$.*

*However, in this paper we show that such an argument does not capture every separation that could be considered to be by diagonalization. Therefore, we consider various weakenings of the notion of universal language and corresponding formalizations of separation by diagonalization. We introduce four notions of* infinitely-often universal language*. For each notion, we give answers or partial answers to the following questions:*

1. *Under what conditions does the existence of a variant of a universal language for $\mathcal{B}$ in $\mathcal{A}$ show $\mathcal{B} \neq \mathcal{A}$? More precisely, what closure properties are needed on $\mathcal{A}$ and $\mathcal{B}$?*
2. *Can any separation be reformulated as this kind of diagonalization argument? More precisely, are there complexity classes $\mathcal{B} \subseteq \mathcal{A}$ with nice closure properties, so that $\mathcal{A}$ has no such variant of a universal language for $\mathcal{B}$?*
3. *Are these variants of universal language different from the other notions we have defined?*

*The main examples of a separation by diagonalization are the time and space hierarchy theorems. We explore the following question: is any separation of a* P *from $\mathcal{A}$ where $\mathcal{A}$ is closed under polynomial-time Turing reducibility essentially a separation by the time hiearchy theorem?*

## 1. Introduction

This paper continues a line of research started in [5] where we explored some of the limits of the power of diagonalization to separate complexity classes. In general, to be able to obtain any limitative results on the power of diagonalization to separate classes of languages it is necessary to formalize what constitutes a separation by diagonalization. However, it is often hard to define what constitutes use of a specific proof technique in a way that is general enough to state limitative results of such a technique. A possible "structural" approach is to define a proof technique in terms of objects which are explicitly or implicitly shown to exist within the proof. For example, such structural approach was used by Razborov and Rudich [6] to define "natural proofs." They define a "natural combinatorial property" to be a certain set of Boolean functions, define the notion of "useful," and define a "natural proof" to be one which "contains, more or less explicitly, the definition of a natural combinatorial property $C_n$ which is useful against $\mathrm{P}/\mathrm{poly}$."

In [5] we provided such a structural definition of "separation by strong diagonalization" and initiated the study of the power of such separations. We took the point of view that the key object involved in such separations is a universal language. A *universal language $U$* for $\mathcal{C}$ is a language which enumerates all languages in $\mathcal{C}$. That is,

$$(\forall L \in \mathcal{C})\exists e \forall x \;\; U(e,x) = L(x).$$

We write $U \multimap \mathcal{C}$ if $U$ is a universal language for $\mathcal{C}$. If $U \multimap \mathcal{C}$ enumerates only the languages in $\mathcal{C}$ we say that $U$

ISSN 1433-8092

is *exact* for $\mathcal{C}$.[1] For a *separation by strong diagonalization* of $\mathcal{A}$ and $\mathcal{B}$ we require:

1. a definition of $U$,
2. a proof that $U \multimap \mathcal{B}$,
3. a proof that $U \in \mathcal{A}$, and
4. a proof[2] that $\mathcal{A}$ or $\mathcal{B}$ is closed under $\leqslant_m^{\text{lin}}$.

The separation follows by the following result, where $E$ is the empty language and $F$ its complement.

**Theorem 1 ([5]).** *If $\mathcal{C}$ is a set of computable languages closed under $\leqslant_m^{\text{lin}}$ and $\mathcal{C} \neq \{E\}$ and $\mathcal{C} \neq \{F\}$, then $\mathcal{C} \not\multimap \mathcal{C}$.*

In [5], we showed that in constrast to Kozen's result [3], not all complexity classes can be separated by strong diagonalization.

Of course, the question is whether our notion of strong diagonalization is broad enough to capture all known diagonalization arguments. The answer is no; we shall show that the following strong forms of the time and space hierarchy theorems (see, e.g, [1])—which are separations by what most people would agree is diagonalization—are not, in general, separations by strong diagonalization.

**Theorem 2.** *If $s$ and $s'$ are space-constructible and $s' \notin O(s)$, then* $\text{SPACE}[s'] \not\subseteq \text{SPACE}[s]$.

**Theorem 3.** *If $t$ and $t'$ are time-constructible and $t' \notin O(t \cdot \log t)$, then* $\text{TIME}[t'] \not\subseteq \text{TIME}[t]$.

That is, we show in Theorem 8 that we can find $s, s', t, t'$ satisfying the hypotheses of Theorems 2 and 3 for which

$$\text{SPACE}[s'] \not\multimap \text{SPACE}[s] \text{ and } \text{TIME}[t'] \not\multimap \text{TIME}[t].$$

This suggests that there are interesting formalizations of diagonalizations that lie between the notion of weak diagonalization implicit in [3] and strong diagonalization. The proof of Theorem 2 suggests one such form of diagonalization: it shows that that $\text{SPACE}[s']$ contains what we call an effectively io universal language for $\text{SPACE}[s]$ and the same holds for $\text{TIME}[t']$ and $\text{TIME}[t]$. An *effectively io universal language*[3] $U$ for $\mathcal{C}$, is a language which, for every index, agrees with a computable universal language $V$ for $\mathcal{C}$ on infinitely many lengths. That is,

$$\forall e \exists^{\infty} n (\forall x\colon |x| = n)\ \ U(e,x) = V(e,x).$$

To establish a separation, it is often enough to require the weaker condition of agreement with a computable universal language on infinitely many inputs; this gives an *effectively iv universal language* ('iv' stands for "infinitely (many) values") [4] for $\mathcal{C}$. Furthermore, we could drop the requirement of effectiveness and simply require agreement on infinitely many lengths or inputs with every language in $\mathcal{C}$, without regard to some computable enumeration of them. This gives four flavors of such weakenings of the notion of universal language: (1) effectively io, (2) io, (3) effectively iv, and (4) iv. We mention this last flavor for completeness only since it seems quite useless. Any universal language which enumerates the empty language and its complement is an iv universal language for every set of languages.

We show in Proposition 1 and Theorem 4 that each of these notions is strictly stronger than the next in case $\mathcal{C}$ has a computable universal language, except possibly for eio and io. We show in Theorem 11 that if $\mathcal{C}$ is closed under $\leqslant_T^{\text{lin}}$, then $\mathcal{C}$ does not contain an effectively iv universal language for itself and in Theorem 12 that this does not hold if we replace $\leqslant_T^{\text{lin}}$ with $\leqslant_m^{\text{lin}}$ even for effectively io universal languages.

This motivates the following definition. For a *separation by, respectively, (a) effectively io, (b) io, (c) effectively iv diagonalization* of $\mathcal{A}$ and $\mathcal{B}$ we require:

1. a definition of $U$,
2. a proof that $U$ is, respectively, an (a) effectively io, (b) io, (c) effectively iv universal language for $\mathcal{B}$,
3. a proof that $U \in \mathcal{A}$, and
4. a proof that $\mathcal{A}$ or $\mathcal{B}$ is closed under $\leqslant_T^{\text{lin}}$.

On one hand, we show in Theorem 14 that effectively iv diagonalization separates virtually everything. On the other hand, we show in Theorems 4 and 16 that there are distinct classes with strong closure properties that can not be separated by io diagonalization. In summary:

- iv universal languages do not provide separation of classes,
- eiv diagonalization is essentially "all separating," and
- io and eio diagonalization are both interesting, have different power, can separate more than strong diagonalization, yet can not separate everything.

Finally, we investigate the following question. If $g$ is superpolynomial and time constructible, then we know that $\text{TIME}[g] \multimap \text{P}$. Does the converse hold? That is given $U \multimap \text{P}$, is there always a superpolynomial, time-constructible function $g$ such that $\text{TIME}[g] \subseteq \text{P}^U$? We have been unable to settle this question. However, we shall show that if $U \multimap \text{P}$ and we can effectively enumerate the $U$-indexes of a certain class of languages in P, then we there is a superpolynomial, time-constructible function $g$ such that $\text{TIME}[g] \subseteq \text{P}^U$. Formally,we shall define a class of universal languages, which we call *graded*, for which the

[1] In [5] we called such universal languages *strict*.

[2] In [5] we required the more conservative "a proof that (a) $\mathcal{A}$ is closed under $\leqslant_T^{\text{lin}}$ or (b) $\mathcal{B}$ is closed under $\leqslant_m^{\text{lin}}$."

[3] 'io' stands for "infinitely often" (on lengths)

[4] i.e., infinitely often on inputs rather than lengths

answer to our question is yes. We shall also show that there are universal langues of P which are not graded.

To simplify the presentation of the results, we focus on many-one reducibilities $\leqslant_m^{\mathrm{lin}}$ (linear time) and $\leqslant_m^{\mathrm{P}}$ (polynomial time) and their corresponding Turing reducibilities $\leqslant_T^{\mathrm{lin}}$ (linear time) and $\leqslant_T^{\mathrm{P}}$ (polynomial time). Clearly most of our results can be easily adapted to a variety of different reducibilities.

The outline of this paper is as follows. In section 2 we establish the notation and introduce basic concepts. In section 3 we introduce our four new variants of universal languages: eio, io, eiv, and iv and show that each of these notions is strictly stronger than the next (except for io vs. eio). In section 4 we define separation by diagonalization in terms of these variants and analyze the power of such separations. In section 5 we present structural results on these variants of universal languages for a fixed class of languages. In section 6 we explore the relationship between universal languages of P and superpolynomial time classes.

## 2. Preliminaries

In this section we fix some notation. We view languages as either sets of strings or their corresponding characteristic functions, so we write $x \in L$ or $L(x) = 1$ depending on which is more convenient. We use $E$ for the empty language and $F$ for its complement. That is, $E(x) = 0$ and $F(x) = 1$ for all $x$.

We denote concatenation of strings by simple juxtaposition so, for example $0x1y$ corresponds to the string obtained by concatenating 0, $x$, 1, and $y$ in that order. We write $0^n$ for a string of $n$ zeros and similarly $1^n$ for a string of $n$ ones. We fix a correspondence between strings and positive integers as follows: a string $s$ corresponds to the number $m$ obtained by reading $1s$ as a binary number. This makes strings empty, '0', '1', '00', '01', '10', '11', ... correspond to the numbers 1, 2, 3, 4, 5, 6, 7, ... We look at inputs as either strings or numbers, again depending on convenience. It should be clear from context whether we refer to a string or the corresponding number.

In some proofs we use the function $\mathrm{lex}(x)$ which gives the $m$-th string where $m$ is the binary number encoded by $x$. For example, lex('01') = lex('001') = empty and lex('101') = lex('00101') = '01'. Notice that lex is linear-time computable: all it needs to do is strip the prefix $0^m1$ from its input string (we also set $\mathrm{lex}(0^m)$ = empty so that lex is defined on all inputs). The property of lex which we need is

$$\forall y \forall^\infty m (\exists x\colon |x| = m)(\mathrm{lex}(x) = y).$$

We fix a linear-time pairing function $\langle \bullet, \bullet \rangle$ and write $U(e, x)$ for $U(\langle e, x \rangle)$. We write $U_e$ for $U(e, \bullet)$ and $U \multimap \mathcal{C}$ in case $U$ is a universal language for $\mathcal{C}$, that is if

$$(\forall L \in \mathcal{C}) \exists e (U_e = L).$$

We set $[U] := \{U_e\colon e \in \mathbb{N}^+\}$; therefore $U \multimap \mathcal{C}$ iff $\mathcal{C} \subseteq [U]$. We write $\mathcal{A} \multimap \mathcal{B}$ if there exists $U \in \mathcal{A}$ such that $U \multimap \mathcal{B}$. We fix a second computable *onto* pairing function $(\bullet, \bullet)\colon \mathbb{N}^+ \times \mathbb{N}^+ \longrightarrow \mathbb{N}^+$, which we need for some of the proofs. We say that $g$ is superpolynomial if for every polynomial $p$, $\forall^\infty x (g(x) > p(x))$ and we say that $g$ is weakly superpolynomial if for every polynomial $p$, $\exists^\infty x (g(x) > p(x))$

We write

- $\varphi_e$ for the $e$th deterministic Turing machine,
- $\psi_e$ for the $e$th deterministic decision Turing machine,
- $\psi_{e,s}$ for the $e$th deterministic decision Turing machine restricted to $s$ steps,
- $\psi_{e,f}(x)$ where $f$ is a function for $\psi_{e,f(|x|)}(x)$, and
- $\psi_{e;k}(x)$ for $\psi_{e,|x|^k}(x)$.
- $\psi_{e,s}(x)\downarrow$ if the $e$th deterministic decision Turing machine terminates on input $x$ in $s$ steps or less,
- $A =^* B$ for $\forall^\infty x (A(x) = B(x))$,
- $A \stackrel{n}{=} B$ for $(\forall x\colon |x| = n)(A(x) = B(x))$,
- $A \stackrel{\mathrm{io}}{=} B$ for $\exists^\infty n (A \stackrel{n}{=} B)$, and
- $\mathrm{io}\mathcal{C}$ for $\{L\colon (\exists L' \in \mathcal{C})(L \stackrel{\mathrm{io}}{=} L')\}$.

We write $\mathrm{queries}(M^Q(x))$ for the set of queries that oracle Turing machine $M$ makes of $Q$ on input $x$.

We write $A \leqslant_m^{\mathrm{lin}} B$ if $A$ is linear-time many-one reducible to $B$, $A \leqslant_m^{\mathrm{P}} B$ if $A$ is polynomial-time many-one reducible to $B$, and $A \leqslant_T^{\mathrm{lin}} B$ and $A \leqslant_T^{\mathrm{P}} B$ for the corresponding case for Turing reducibility. We set $\langle \mathcal{A} \rangle_m^{\mathrm{lin}} := \{L\colon (\exists A \in \mathcal{A})(L \leqslant_m^{\mathrm{lin}} A)\}$ and similarly for the other reducibilities.

## 3. Variants of Universal Languages

In this section we introduce four variants of universal languages: eio, io, eiv, and iv. The strongest of these, eio universal languages, are sufficient to obtain the strong forms of the space and time hierarchy theorems (Theorems 6 and 7). In contrast, plain universal languages are not sufficient to obtain these hierarchy theorems (Theorem 8). These four variants, together with plain universal languages, form a hierarchy (Proposition 1). The hierarchy is strict, except for eio and io universal languages for which strict separation is open (Theorem 4) although we know that these notions do not coincide completely (Theorem 5). It is natural to wonder how the condition (1) $\mathcal{A} \stackrel{\mathrm{io}}{\multimap} \mathcal{B}$ is related to (2) $\mathrm{io}\mathcal{A} \multimap \mathcal{B}$. This question only makes sense for universal languages defined in terms of restricted pairing functions which we call *length-regular*. If we limit ourselves to such universal languages, (1) implies (2) (Theorem 9), but (2) does not imply (1) (Theorem 10).

**Definition 1.** *We say that $U$ is an* effectively io universal

language *for* $\mathcal{C}$ *which we write* $U \overset{\text{eio}}{\multimap} \mathcal{C}$ *if there exists a computable* $V \multimap \mathcal{C}$ *such that* $\forall e(U_e \overset{\text{io}}{=} V_e)$.

**Definition 2.** *We say that* $U$ *is an* io universal language *for* $\mathcal{C}$ *which we write* $U \overset{\text{io}}{\multimap} \mathcal{C}$ *if* $(\forall L \in \mathcal{C})\exists e(U_e \overset{\text{io}}{=} L)$.

**Definition 3.** *We say that* $U$ *is an* effectively iv universal language *for* $\mathcal{C}$ *which we write* $U \overset{\text{eiv}}{\multimap} \mathcal{C}$ *if there exists a computable* $V \multimap \mathcal{C}$ *such that* $\forall e \exists^\infty x(U_e(x) = V_e(x))$.

**Definition 4.** *We say that* $U$ *is an* iv universal language *for* $\mathcal{C}$ *which we write* $U \overset{\text{iv}}{\multimap} \mathcal{C}$ *if* $(\forall L \in \mathcal{C})\exists^\infty x(U_e(x) = L(x))$.

We include this last definition for completeness, but notice that if $U \multimap \{E, F\}$, then $U \overset{\text{iv}}{\multimap} \mathcal{C}$ for any set of languages $\mathcal{C}$. We have defined these to correspond to universal languages which are not necessarily exact. There are four additional flavors which correspond to exact universal languages, but we do not consider them here in the interest of brevity and because they are not relevant to separations.

**Proposition 1.** *If* $\mathcal{A}$ *is closed under* $\leqslant_m^{\text{lin}}$ *and there is a computable* $V \multimap \mathcal{B}$*, then:*

1. *If* $\mathcal{A} \multimap \mathcal{B}$*, then* $\mathcal{A} \overset{\text{eio}}{\multimap} \mathcal{B}$.
2. *If* $\mathcal{A} \overset{\text{eio}}{\multimap} \mathcal{B}$*, then* $\mathcal{A} \overset{\text{io}}{\multimap} \mathcal{B}$.
3. *If* $\mathcal{A} \overset{\text{io}}{\multimap} \mathcal{B}$*, then* $\mathcal{A} \overset{\text{eiv}}{\multimap} \mathcal{B}$.
4. *If* $\mathcal{A} \overset{\text{eiv}}{\multimap} \mathcal{B}$*, then* $\mathcal{A} \overset{\text{iv}}{\multimap} \mathcal{B}$.

*Proof.*

1. If $\mathcal{A}$ is a set of computable languages, it follows from the definitions that $\mathcal{A} \overset{\text{eio}}{\multimap} \mathcal{B}$. However, it may be that $U \in \mathcal{A}$, $U \multimap \mathcal{B}$, and $U$ is not computable, in which case we may have $U \overset{\text{eio}}{\not\multimap} \mathcal{B}$ (see Theorem 5).

   Assume $U \in \mathcal{A}$ and $U \multimap \mathcal{B}$. Set $W_e(x) := U_{\text{lex}(|x|)}(x)$. Since $\mathcal{A}$ is closed under $\leqslant_m^{\text{lin}}$, $W \in \mathcal{A}$. Now pick $e, n$. Since $U \multimap \mathcal{B}$, there is $i$ such that $U_i = V_e$ and $m \geqslant n$ large enough so that there exists $x$ with $|x| = m$ and $\text{lex}(|x|) = i$. Then

   $$W_e(x) \overset{(1)}{=} U_{\text{lex}(x)}(x) \overset{(2)}{=} U_i(x) \overset{(3)}{=} V_e(x)$$

   (a) by definition of $U$,
   (b) by choice of $x$, and
   (c) since $U_i = V_e$

   as desired.

2. Follows directly from the definitions without any conditions on $\mathcal{A}$ and $\mathcal{B}$.

3. Assume $U \in \mathcal{A}$ and $U \overset{\text{io}}{\multimap} \mathcal{B}$. Set $W_e(x) := U_{\text{lex}(x)}(x)$. Since $\mathcal{A}$ is closed under $\leqslant_m^{\text{lin}}$, $W \in \mathcal{A}$. Now pick $e, n$. Since $U \overset{\text{io}}{\multimap} \mathcal{B}$, there is $i$ and $m \geqslant n$ large enough so that $U_i \overset{m}{=} V_e$ and there exists $x$ with $|x| = m$ and $\text{lex}(x) = i$. Then

   $$W_e(x) \overset{(1)}{=} U_{\text{lex}(x)}(x) \overset{(2)}{=} U_i(x) \overset{(3)}{=} V_e(x)$$

   (a) by definition of $U$,
   (b) by choice of $x$, and
   (c) since $U_i \overset{m}{=} V_e$

   as desired.

4. Follows directly from the definitions without any conditions on $\mathcal{A}$ and $\mathcal{B}$.

□

**Lemma 1.** *There exists a computable universal language* $L$ *such that, for all* $e$, $L_e \in \text{P} - \text{TIME}[n^e]$.

**Theorem 4.** *For any set of languages* $\mathcal{C} \supseteq \text{P}$ *with computable* $V \multimap \mathcal{C}$*, there exists* $U^1$*,* $U^2$*, and* $U^3$ *such that*

1. $U^1 \overset{\text{eio}}{\multimap} \mathcal{C}$ *and* $\text{P}^{U^1} \not\multimap \mathcal{C}$.
2. $U^2 \overset{\text{eiv}}{\multimap} \mathcal{C}$ *and* $\text{P}^{U^2} \overset{\text{io}}{\not\multimap} \mathcal{C}$.
3. $U^3 \overset{\text{iv}}{\multimap} \mathcal{C}$ *and* $\text{P}^{U^3} \overset{\text{eiv}}{\not\multimap} \mathcal{C}$.

*Proof.* The proof uses Theorem 11 from the next section for $U^2$ and $U^3$.

1. We will construct $U^1$ in stages. At each stage $s$ we will have the characteristic function $U^{1,s}$, which will be zero outside of $D^s$. We will have $D^i \subseteq D^j$ for $i \leqslant j$ and $\bigcup_s D^s = \mathbb{N}^+$. We set $U^1(z) := \lim_{s\to\infty} U^{1,s}(z)$. It will be clear that this limit exists for all $z$.

   To ensure that $U^1 \overset{\text{eio}}{\multimap} \mathcal{C}$, we satisfy the following requirements for all $e, n$:

   $$P_{(e,n)}: \quad (\exists m \geqslant n)(U_e \overset{m}{=} V_e).$$

   To ensure that $\text{P}^{U^1} \not\multimap \mathcal{C}$, we satisfy the following requirements for all $e, i$:

   $$N_{(e,i)}: \quad \forall i \exists x(\psi_{e;e}^{U^1}(i, x) \neq L_e(x))$$

   for some computable universal language $L$ with the properties given in Lemma 1. We proceed as follows.

   At stage 0, we set $D^0 = \emptyset$ and $U^{1,s}(z) = 0$ for all $z$.

   At stage $s + 1$ with $s = 2(e, n) - 2$ we satisfy requirement $P_{(e,n)}$ as follows. We find the smallest $m$ such

that $D^s \cap S_{e,m} = \emptyset$ where $S_{e,m} := \{\langle e, x\rangle \colon |x| = m\}$. We set $D^{s+1} := D^s \cup S_{e,m} \cup \{z \colon |z| < s\}$ and

$$U^{1,s+1}(z) := \begin{cases} U^{1,s}(z) & \text{if } z \in D^s \\ V(z) & \text{if } z \in S_{e,m} \\ 0 & \text{otherwise} \end{cases}$$

Then $U_e^{1,s+1} \stackrel{m}{=} V_e$, as desired.

At stage $s+1$ with $s = 2(e,n) - 1$ we satisfy requirement $N_{(e,n)}$ as follows. We find the smallest $x$ such that $\psi_{e;e}^{U^1,s}(i,x) \neq L(x)$ and we set $U^{1,s+1} := U^{1,s}$ and $D^{s+1} := D^s \cup \text{queries}(\psi_{e;e}^{U^1,s}(i,x))$.

2. Define $U^2$ as follows: Set $U_e^2(1^{2^n}) = V_e(1^{2^n})$ if $e \leqslant n$ and set $U^2$ to zero elsewhere. Then $U^2 \stackrel{\text{eiv}}{\multimap} \mathcal{C}$, since for every $e$ and $n$, $x = 1^{2^n}$ satisfies $U_e^2(x) = V_e(x)$.

   Now assume, to get a contradiction, that $W \in \mathrm{P}^{U^2}$ and $W \stackrel{\text{io}}{\multimap} \mathcal{C}$. From $W$, we construct $Z \in \mathrm{P}$ such that $Z \stackrel{\text{eiv}}{\multimap} \mathcal{C} \supseteq \mathrm{P}$, which contradicts Theorem 11.

   We have $W = \psi_{k;k}^{U^2}$ for some $k$. On input $\langle e, x\rangle$, $\psi_{k;k}^{U^2}$ can only query $A$ on strings of length at most $|\langle e, x\rangle|^k$. Since we have assumed that $\langle e, x\rangle$ is computable in linear time from $e$ and $x$, we must have $|\langle e, x\rangle| \leqslant c(|e| + |x|)$ for some $c$.

   $U^2$ is very sparse. It contains at most $\log^2(n)$ strings of length $\leqslant n$. Therefore, $\psi_{k;k}^{U^2}$ can only query $A$ on $k^2 \log^2(c(|e| + |x|))$ such strings. Define

   $$U(w) := \{\langle e, 1^{2^n}\rangle \colon \text{ bit } \langle e, n\rangle \text{ of } w \text{ is one}\}.$$

   We use $U(w)$ to specify an initial segment of $U^2$. The calculation above shows that on input $\langle e, x\rangle$, it is sufficient to provide $w$ of length $k^2 \log^2(c(|e| + |x|))$ to guarantee $\psi_{k;k}^{U^2}(e,x) = \psi_{k;k}^{U(w)}(e,x)$.

   For every $j$, if $x = 0^{|e|}1ew$, we set

   $$Z(j,x) = \psi_{k;k}^{A(w)}(e,x)$$

   and we set $Z$ to zero elsewhere. We have $Z \in \mathrm{P}$ since we do not need oracle $U^2$ to compute $Z$. We show that

   $$\forall e, n (\exists x, |x| \geqslant n)(Z_e(x) = V_e(x))$$

   as follows. Pick $e, n$. Since $W \stackrel{\text{io}}{\multimap} \mathcal{C}$, there must be $i$ such that $W_i \stackrel{\text{io}}{=} V_e$. Pick $m \geqslant n$ such that $W_i \stackrel{m}{=} V_e$ and $k^2 \log^2(c(|i| + m)) \leqslant m - 2|e| - 1$.

   Then there is $w$ such that for $x = 0^{|i|}1iw$ with $|x| = m$ we have

   $$Z_e(x) \stackrel{(a)}{=} \psi_{k;k}^{A(w)}(i,x) \stackrel{(b)}{=} \psi_{k;k}^{U^2}(i,x) \stackrel{(c)}{=} W_i(x) \stackrel{(d)}{=} V_e(x)$$

   (a) by definition of $Z$,
   (b) by choice of $w$,
   (c) by choice of $k$, and
   (d) since $|x| = m$

   as desired.

3. Pick any $U^3 \in \mathrm{P}$ satisfying $[U] = \{E, F\}$ where $E$ is the empty language and $F$ its complement. For example, set $U_e^3(x) := e \bmod 2$. Then $U^3 \stackrel{\text{iv}}{\multimap} \mathcal{C}$ for any $\mathcal{C}$, yet $\mathrm{P}^{U^3} = \mathrm{P} \stackrel{\text{eiv}}{\not\multimap} \mathrm{P} \subseteq \mathcal{C}$ by Theorem 11. □

We have been unable to prove the corresponding strong separation of $\stackrel{\text{eio}}{\multimap}$ from $\stackrel{\text{io}}{\multimap}$, but we have the following.

**Theorem 5.** *There exists a (non-computable) $U \multimap \mathcal{C}$ (and therefore also $U \stackrel{\text{io}}{\multimap} \mathcal{C}$) such that $U \stackrel{\text{eio}}{\not\multimap} \mathcal{C}$.*

*Proof.* Pick any $V \multimap \mathcal{C}$ and set

$$U_e(x) := \begin{cases} V_{e/2}(x) & \text{if } e \text{ is even} \\ 1 - \psi_{(e+1)/2}(e,x) & \text{if } e \text{ is odd and} \\ & \psi_{(e+1)/2} \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

Now if $L \in \mathcal{C}$, then since $V \multimap \mathcal{C}$, $L = V_e$ for some $e$ and therefore $L = U_{2e}$. This shows that $U \multimap \mathcal{C}$. On the other hand, for every computable $W \multimap \mathcal{C}$, we must have $W = \psi_w$ for some $w$ and therefore for all $x$,

$$U_{2w-1}(x) = 1 - \psi_w(2w-1, x) = 1 - W_{2w-1}(x).$$

This shows that $W$ does not witness $U \stackrel{\text{eio}}{\multimap} \mathcal{C}$. □

The proofs of the strong forms of the hierarchy theorems mentioned in the introduction give the following.

**Theorem 6.** *If $s$ and $s'$ are space-constructible and $s' \notin O(s)$, then* $\mathrm{SPACE}[s'] \stackrel{\text{eio}}{\multimap} \mathrm{SPACE}[s]$.

**Theorem 7.** *If $t$ and $t'$ are time-constructible and $t' \notin O(t \cdot \log t)$, then* $\mathrm{TIME}[t'] \stackrel{\text{eio}}{\multimap} \mathrm{TIME}[t]$.

**Theorem 8.**

1. *For every space-constructible $s$ satisfying $s(n) \geqslant n^2$, there exists space-constructible $s'$ such that $s' \notin O(s)$ (and therefore* $\mathrm{SPACE}[s'] \not\subseteq \mathrm{SPACE}[s]$*) and* $\mathrm{SPACE}[s'] \not\multimap \mathrm{SPACE}[s]$.

2. *For every time-constructible $t$ satisfying $t(n) \geqslant n^2$, there exists time-constructible $t'$ such that $t' \notin O(t \cdot \log t)$ (and therefore* $\mathrm{TIME}[t'] \not\subseteq \mathrm{TIME}[t]$*) and* $\mathrm{TIME}[t'] \not\multimap \mathrm{TIME}[t]$.

*Proof.* We prove (2); the proof for (1) is similar.

We pick a language $L \in \mathrm{TIME}[t] - \mathrm{TIME}[\sqrt{t}]$, which we know exists by the time hierarchy theorem.

We will construct time-constructible $t'$ such that $t' \notin O(t \cdot \log t)$ and $\mathrm{TIME}[t'] \not\multimap \mathrm{TIME}[t]$ in stages using delayed diagonalization. At stage $n$ we define $t'(n)$ and attempt to satisfy a requirement given by $\rho(n)$. We will make sure that computing $\rho(n)$ and attempting to satisfy the requirement given by $\rho(n)$ takes no more time than $t'(n)$ to ensure that $t'$ is time constructible.

To ensure that $t' \notin O(t \cdot \log t)$ we satisfy the following requirements for all $n$:

$$P_e: \quad (\exists m \geqslant e)(t'(m) = t^2(m))$$

To ensure that $\mathrm{TIME}[t'] \not\multimap \mathrm{TIME}[t]$ we satisfy the following requirements for all $e, i$:

$$N_{(e,i)}: \quad \exists x\ \psi_{e,t'}(i,x) \neq L(x)$$

That is, the $e$th Turing machine running in time $t'$ fails to index $L$ at $i$.

If $\rho(n) = 2e - 1$, we set $t'(n) = t^2(n)$ which satisfies requirement $P_e$. If $\rho(n) = 2(e,i)$, we set $t'(n) = \sqrt{t(n)}$, which is an attempt to satisfy requirement $N_{(e,i)}$. We will succeed for large enough $n$, since $L \in \mathrm{TIME}[t] - \mathrm{TIME}[\sqrt{t}]$.

We compute $\rho(n)$ as follows. We use a total of $n$ steps to check what requirements have already been satisfied, one at a time, in order $P_1, N_1, P_2, N_2, \ldots$. To check that requirement $N_{(e,i)}$ has been satisfied, we compute $\psi_{e,t'}(i,x)$ and $L(x)$ for $x = 1, 2, 3, \ldots$ until we find a difference. To check that requirement $P_e$ has been satisfied, we compute $t'(e), t'(e+1), \ldots$ until we find $t'(m) = t^2(m)$ for $m \geqslant e$. If the last requirement which have so determined to be satisfied is $P_e$, we set $\rho(n) := 2e$. If the last requirement which have so determined to be satisfied is $N_{(e,i)}$, we set $\rho(n) := 2(e,i) + 1$. Otherwise, we set $\rho(n) := 1$. □

We have not assumed much from the pairing function $\langle \bullet, \bullet \rangle$ used to define universal languages. We say that a pairing function $\langle \bullet, \bullet \rangle$ is *length-regular* if it satisfies

$$\forall e \forall^{\infty} m \exists n \forall x (|\langle e, x \rangle| = m \iff |x| = n).$$

For example, the pairing function given by $\langle e, x \rangle := 0^{|e|}1ex$ is length-regular. We say that $U$ is *length-regular* if it is defined in terms of a length-regular pairing function.

**Theorem 9.** *If $V$ is length-regular then*

$$V \in \mathrm{io}\mathcal{A} \text{ and } V \multimap \mathcal{B} \Longrightarrow \mathcal{A} \overset{\mathrm{io}}{\multimap} \mathcal{B}.$$

*Proof.* Assume the hypotheses. Pick $U \in \mathcal{A}$ such that $U \overset{\mathrm{io}}{=} V$. Then $U \overset{\mathrm{io}}{\multimap} \mathcal{B}$ as follows. Pick any $e, n$. Find $m \geqslant |\langle e, 0^n \rangle|$ such that $U \overset{m}{=} V$. There must be such $m$ because $U \overset{\mathrm{io}}{=} V$. Then, since $V$ is regular, $U_e \overset{n'}{=} V_e$ for some $n' \geqslant n$, as desired. □

**Theorem 10.** *If $V \multimap \mathcal{B}$ is length-regular, then there exists length-regular $U$ such that*

$$U \overset{\mathrm{io}}{\multimap} \mathcal{B} \text{ and } \mathrm{io}U \not\multimap \mathcal{B}.$$

*Furthermore, if $V$ is computable then*

$$U \overset{\mathrm{eio}}{\multimap} \mathcal{B} \text{ and } \mathrm{io}U \not\multimap \mathcal{B}.$$

*Proof.* Pick disjoint infinite sets $S_0, S_1, \ldots$, for example by setting

$$S_e = \{(e,p) \colon p \in \mathbb{N}\}.$$

Set

$$U_e(x) := \begin{cases} V_e(x) & \text{if } |\langle e, x \rangle| \in S_e \\ 1 - V_{\mathrm{lex}(x)}(x) & \text{otherwise} \end{cases}$$

For every $e$ and $n$, there is $m \geqslant n$ such that $m \in S_e$ and therefore $U_e \overset{m}{=} V_e$. This shows that $U \overset{\mathrm{io}}{\multimap} \mathcal{B}$ and, if $V$ is computable, also shows that $U \overset{\mathrm{eio}}{\multimap} \mathcal{B}$.

Now assume $W \overset{\mathrm{io}}{=} U$. Set $C_{W,U} = \{m \colon W \overset{m}{=} U\}$. Pick any $\ell$ so that $C_{W,U} - S_\ell$ is infinite. There must be such $\ell$, because $C_{W,U}$ is infinite and if $C_{W,U} - S_e$ is finite, then for any $\ell \neq e$, $S_\ell \cap C_{W,U}$ is finite (because $S_\ell$ is disjoint from $S_e$) and therefore $C_{W,U} - S_\ell$ must be infinite.

Then $\forall e (W_e \neq V_\ell)$—which implies $W \not\multimap \mathcal{B}$—as follows. For any $e$, pick $m \in C_{W,U} - S_e$ large enough so that there is, by length-regularity of $V$, some $x$ satisfying $|\langle e, x \rangle| = m$ and $\mathrm{lex}(x) = \ell$.

For such $m$ and $x$,

$$W_e(x) \overset{(1)}{=} U_e(x) \overset{(2)}{=} 1 - V_{\mathrm{lex}(x)}(x) \overset{(3)}{=} 1 - V_\ell(x)$$

1. since $m \in C_{W,U}$,
2. by def. of $U$, since $m \notin S_e$, and
3. since $\mathrm{lex}(x) = \ell$.

and therefore $W_e \neq V_\ell$ as desired.

□

## 4. Separation by Diagonalization

We have seen in the previous section (Theorems 6, 7, and 8) that separation by strong diagonalization, as defined in [5] is insufficient to establish the strong forms of the space and time hierarchy theorems (Theorems 2 and 3). In this section we introduce three additional kinds of separations by diagonalization: eio, io, and eiv corresponding to three of the

four variants of universal languages introduced in the previous section. The fourth variant, iv universal languages, is too weak to obtain separations. For the other three variants, it is enough to require closure under $\leqslant_T^{\mathrm{lin}}$ to obtain a separation (Theorem 11). On the other hand, closure under $\leqslant_m^{\mathrm{lin}}$ is not enough even for the strongest kind, eio diagonalization (Theorem 12). Separation by eiv diagonalization separates most classes that can be separated by other means (Theorem 14). On the other hand, the structural result from the next section (Theorem 16) together with the results from the previous section (Theorem 4), show that there are distinct sets of languages $\mathcal{A}$ and $\mathcal{B}$ which can not be separated by eio or io diagonalization.

**Definition 5.** *We say that a* separation by, respectively, (a) effectively io, (b) io, or (c) effectively iv diagonalization *of $\mathcal{A}$ and $\mathcal{B}$ requires:*

1. *a definition of $U$,*
2. *a proof that $U$ is, respectively, an (a) effectively io, (b) io, (c) effectively iv universal language for $\mathcal{B}$,*
3. *a proof that $U \in \mathcal{A}$, and*
4. *a proof that $\mathcal{A}$ or $\mathcal{B}$ is closed under $\leqslant_T^{\mathrm{lin}}$.*

The separation follows from the following theorem and its easy corollary.

**Theorem 11.** *If $\mathcal{C}$ is closed under $\leqslant_T^{\mathrm{lin}}$, then $\mathcal{C} \overset{\mathrm{eiv}}{\not\multimap} \mathcal{C}$.*

*Proof.* Assume $U \overset{\mathrm{eiv}}{\multimap} \mathcal{C}$ witnessed by computable $V \multimap \mathcal{C}$. We construct $L \leqslant_T^{\mathrm{lin}} U$ by delayed diagonalization such that $L \notin \mathcal{C}$. It follows that if $\mathcal{C}$ is closed under $\leqslant_T^{\mathrm{lin}}$ then $U \notin \mathcal{C}$. We set $L(x) := 1 - U_{\rho(x)}(x)$ where $\rho(x)$ is computed by "looking back" and gives the smallest number $e$ for a language $V_e$ for which we have not yet found a difference with $L$. We compute $\rho(x)$ by using a total of $|x|$ steps, first to look for the smallest $x$ such that $L(x) \neq V_1(x)$, then for the smallest $x$ such that $L(x) \neq V_2(x)$ and so on. $\rho$ is not eventually constant since otherwise we would have $1 - U_e =^* L =^* V_e$ which contradicts $U \overset{\mathrm{eiv}}{\multimap} \mathcal{C}$ witnessed by $V$. Notice that we need $V$ to be computable in order to compute $\rho$. □

**Corollary 1.** *If $\mathcal{A}$ or $\mathcal{B}$ are closed under $\leqslant_T^{\mathrm{lin}}$ and $\mathcal{A} \overset{\mathrm{eiv}}{\multimap} \mathcal{B}$, then $\mathcal{A} \neq \mathcal{B}$.*

We can not weaken $\leqslant_T^{\mathrm{lin}}$ here to $\leqslant_m^{\mathrm{lin}}$ as we did in our definition of separation by strong diagonalization in [5] due to the following result, which stands in contrast to Theorem 1, where $E$ is the empty language and $F$ its complement.

**Theorem 12.** *There exists a set $\mathcal{C}$ of computable languages closed under $\leqslant_m^{\mathrm{lin}}$ such that $\mathcal{C} \neq \{E\}$, $\mathcal{C} \neq \{F\}$, and $\mathcal{C} \overset{\mathrm{eio}}{\multimap} \mathcal{C}$.*

*Proof.* We will construct computable $U$ in stages. At each stage $s$ we will have the characteristic function $U^s$, which will be zero outside of $D^s$. We will have $D^i \subseteq D^j$ for $i \leqslant j$ and $\bigcup_s D^s = \mathbb{N}^+$. We set $U(z) := \lim_{s\to\infty} U^s(z)$. It will be clear that this limit exists for all $z$.

We will set $\mathcal{C} := \langle U \rangle_m^{\mathrm{lin}}$. We pick a computable enumeration $f_1, f_2, \ldots$ of the linear-time functions and therefore $V$ given by $V_e(x) := U(f_e(x))$ will be computable and will satisfy $V \multimap \mathcal{C}$.

To ensure that $\mathcal{C} \overset{\mathrm{eio}}{\multimap} \mathcal{C}$ we satisfy the following requirements for all $e, n$:

$$P_{(e,n)}: \quad (\exists m \geqslant n)(U_e \overset{m}{=} V_e).$$

At stage 0 we set $U_1^0(1) = 1$, $U_2^0(2) = 2$, and $D^0 = \{\langle 1,1\rangle, \langle 2,2\rangle\}$. This ensures $\mathcal{C} \neq \{E\}$ and $\mathcal{C} \neq \{F\}$.

At stage $s+1$ with $s+1 = (e,n)$ we satisfy requirement $P_{(e,n)}$ as follows. We set $S_{e,m} := \{\langle e,x\rangle : |x| = m\}$ and find the smallest $m$ such that $S_{e,m} \cap D^s = \emptyset$. Now we define the graph $G = (V, E)$ with vertex set

$$V := S_{e,m} \cup \{f_e(x) : \langle e,x\rangle \in S_{e,m}\}$$

and edge set

$$E := \{(f_e(x), \langle e,x\rangle) : \langle e,x\rangle \in S_{e,m}\}.$$

We break this graph into connected components $C_1, \ldots, C_k$. Notice that there are no edges going into $V - S_{e,m}$. It follows that every component $C_i$ has at most one vertex outside of $S_{e,m}$ and therefore at most one vertex in $D^s$. For every component $C_i$ we pick a value $z_i$ as follows. If $C_i \cap D^s \neq \emptyset$, we set $z_i$ to the unique $z \in C_i \cap D^s$. Otherwise, we set $z_i$ to some $z \in D^s$. Now we set $D^{s+1} := D^s \cup V$ and

$$U^{s+1}(z) := \begin{cases} U^s(z) & \text{if } z \in D^s - V \\ U^s(z_i) & \text{if } z \in C_i \\ 0 & \text{otherwise} \end{cases}$$

If $|x| = m$ then $\langle e,x\rangle \in C_i$ for some $i$ and therefore

$$U_e^{s+1}(x) \overset{(1)}{=} U^s(z_i) \overset{(2)}{=} U^{s+1}(f_e(x)) \overset{(3)}{=} U(f_e(x)) \overset{(4)}{=} V_e(x)$$

1. by definition of $U^{s+1}$,
2. since $f_e(x) \in C_i$,
3. since $U$ and $U^{s+1}$ agree on $D^{s+1}$ and $f_e(x) \in D^{s+1}$, and
4. by definition of $V$.

Therefore, $U_e \overset{m}{=} V_e$ as desired. □

On the other hand the following result is a straightforward adaptation of Theorem 6 in [5], which is itself an adaptation of a result of Žák [7].

**Theorem 13.** *If $\mathcal{B}$ is a set of computable languages closed under $\leqslant_m^{\mathrm{lin}}$, $\mathcal{B} \neq \{E\}$, $\mathcal{B} \neq \{F\}$, $U \overset{\mathrm{eio}}{\multimap} \mathcal{C}$ with $\forall e(U_e \overset{\mathrm{io}}{=} V_e)$ for $V \multimap \mathcal{C}$ computable, and there is a linear-time function $f$ for which $U_e \overset{|x|}{=} V_e$ implies $U_e \overset{|f(x)|}{=} V_e$, then $U \notin \mathcal{C}$.*

The following result shows that eiv-diagonalization can essentially separate everything.

**Theorem 14.** *If $\mathcal{A}$ is closed under $\leqslant_T^{\mathrm{lin}}$, $\mathcal{B}$ is closed under finite variations, $\mathcal{B} \subset \mathcal{A}$, and $V \multimap \mathcal{B}$ is computable, then $\mathcal{A} \overset{\mathrm{eiv}}{\multimap} \mathcal{B}$.*

*Proof.* Pick $L \in \mathcal{A} - \mathcal{B}$ and, for all $e$, set $U_e(x) = 1 - L(x)$. Then $U \in \mathcal{A}$ because $\mathcal{A}$ is closed under $\leqslant_T^{\mathrm{lin}}$ and $U \overset{\mathrm{eiv}}{\multimap} \mathcal{B}$ as follows. For any $e$, $L \neq^* V_e \in \mathcal{B}$ because $\mathcal{B}$ is closed under finite variations and $V \multimap \mathcal{B}$ and therefore

$$\exists^\infty x(U_e(x) = 1 - L(x) = V_e(x)).$$

□

## 5. Structural Results

Here we give two results on the structure of io universal languages for a fixed set of languages $\mathrm{P}^B$. We show that $\mathrm{P}^B$ has infinitely descending chains of io universal languages (Theorem 15) but that these chains do not approach $\mathrm{P}^B$ arbitrarily close (Theorem 16).

**Theorem 15 (Density).** *If $U \overset{\mathrm{io}}{\multimap} \mathrm{P}^B$ and $W <_T^{\mathrm{P}} U$, then*

$$\exists V(W <_T^{\mathrm{P}} V <_T^{\mathrm{P}} U \text{ and } V \overset{\mathrm{io}}{\multimap} \mathrm{P}^B).$$

*Proof.* Assume that $U \overset{\mathrm{io}}{\multimap} \mathrm{P}^B$ witnessed by computable $Y \multimap \mathrm{P}^B$. By Ladner's density theorem [4], we know there is a language $L$ such that $W <_T^{\mathrm{P}} L <_T^{\mathrm{P}} U$.

We will construct $V$ in stages using delayed diagonalization. At each stage $n$ we define $V(z)$ for all $z$ satisfying $|z| = n$ and we attempt to satisfy requirement $\rho(n)$.

To ensure that $V \overset{\mathrm{io}}{\multimap} P^B$ we satisfy the following requirements for all $e$:

$$P_{(e,n)}\colon \quad (\exists m \geqslant n)(V_e \overset{m}{=} Y_e).$$

To ensure that $U \not\leqslant_T^{\mathrm{P}} V$ we satisfy the following requirements for all $e$:

$$N_e\colon \quad \exists z(\psi_{e;e}^V(z) \neq U(z)).$$

To ensure that $W <_T^{\mathrm{P}} V$ we ensure that $L \leqslant_T^{\mathrm{P}} V$ at stage 0. We will show that $V \leqslant_m^{\mathrm{lin}} U$ which implies $V \leqslant_T^{\mathrm{P}} U$.

To satisfy these requirements we set

$$V(z) := \begin{cases} L(e) & \text{if } z = \langle e, 1\rangle \\ U(z) & \text{if } \rho(|z|) \text{ is odd} \\ 0 & \text{if } \rho(|z|) \text{ is even} \end{cases}$$

If $\rho(n) = 2(e,n) - 1$, we attempt to satisfy requirement $P_{(e,n)}$. We succeed eventually since $U \overset{\mathrm{io}}{=} Y$. If $\rho(n) = 2e$, we attempt to satisfy requirement $N_e$. We succeed eventually since $L <_T^{\mathrm{P}} U$.

We compute $\rho(n)$ as follows. We use a total of $n$ steps to check what requirements have already been satisfied, one at a time, in order $P_1, N_1, P_2, N_2, \ldots$. To check that requirement $P_e$ has been satisfied, we look for the smallest $m \geqslant n$ such that $V_e \overset{m}{=} Y_e$. To check that requirement $N_{(e,i)}$ has been satisfied, we compute $\exists z(\psi_{e;e}^V(z) \neq U(z))$ for $z = 1, 2, 3, \ldots$ If, during the computation of $\psi_{e;e}^V(z)$, $V$ is queried for some string of length $> n$, we abort that computation and move on to the next $z$. If the last requirement which have so determined to be satisfied is $P_{(e,n)}$, we set $\rho(n) := 2(e,n)$. If the last requirement which have so determined to be satisfied is $N_e$, we set $\rho(n) := 2e + 1$. Otherwise, we set $\rho(n) := 1$. □

**Theorem 16 (Gap).** $\forall B \exists A(B <_T^{\mathrm{P}} A \text{ and } \mathrm{P}^A \overset{\mathrm{io}}{\not\multimap} \mathrm{P}^B)$.

*Proof.* Set $A' := \{1^{2^e}\colon \psi_{e;e}^B(1^{2^e}) = 0\}$ which ensures $A' \notin \mathrm{P}^B$. Set $A := A' \oplus B$. Now assume, to get a contradiction, that $U \in \mathrm{P}^A$ and $U \overset{\mathrm{io}}{\multimap} \mathrm{P}^B$. From $U$, we construct $V \in \mathrm{P}^B$ such that $V \overset{\mathrm{eiv}}{\multimap} \mathrm{P}^B$, which contradicts Theorem 11.

We have $U = \psi_{k;k}^A$ for some $k$. On input $\langle e, x\rangle$, $\psi_{k;k}^A$ can only query $A'$ on strings of length at most $|\langle e, x\rangle|^k$. Since we have assumed that $\langle e, x\rangle$ is computable in linear time from $e$ and $x$, we must have $|\langle e, x\rangle| \leqslant c(|e| + |x|)$ for some $c$. Since there are at most $n$ strings in $A'$ of the form $1^{2^i}$ and of length $\leqslant 2^n$, $\psi_{k;k}^A$ can only query $A'$ on $k \log(c(|e| + |x|)) \leqslant k \log c + k \log(|e| + |x|)$ such strings. Define $A(w) := \{1^{2^i}\colon \text{ bit } i \text{ of } w \text{ is one }\}$. We use $A(w)$ to specify an initial segment of $A'$. We have seen above that on input $\langle e, x\rangle$, it is sufficient to provide $w$ of length $k \log c + k \log(|e| + |x|)$ so that $\psi_{k;k}^A(e, x) = \psi_{k;k}^{A(w)\oplus B}$.

For every $j$, if $x = 0^{|e|}1ew$, we set $V(j, x) = \psi_{k;k}^{A(w)\oplus B}(e, x)$ and we set $V(z) = 0$ elsewhere. We have $V \in \mathrm{P}^B$ since we do not need oracle $A'$ to compute $V$. We show that

$$\forall e, n(\exists x, |x| \geqslant n)(V_e(x) = \psi_{e;e}^B(x))$$

as follows. Pick $e, n$. Since $U \overset{\mathrm{io}}{\multimap} \mathrm{P}^B$, there must be $i$ such that $U_i \overset{\mathrm{io}}{=} \psi_{e;e}^B$. Pick $m \geqslant n$ so that $(\forall x, |x| = m)(U_i(x) = \psi_{e;e}^B(x))$ and so that $k \log c + k \log(|i| + m) \leqslant m - 2|e| - 1$.

Then there is $w$ such that for $x = 0^{|i|}1iw$ with $|x| = m$ satisfying

$$V(e,x) \stackrel{(1)}{=} \psi_{k;k}^{A(w)\oplus B}(i,x) \stackrel{(2)}{=} \psi_{k;k}^{A}(i,x) \stackrel{(3)}{=} U_i(x) \stackrel{(4)}{=} \psi_{e;e}^{B}(x)$$

1. by definition of $V$,
2. by choice of $x$,
3. by choice of $k$, and
4. by choice of $m$.

□

## 6. Universal Languages for P

The proof of the time hierarchy theorem shows:

**Proposition 2.** *If $g$ is superpolynomial and time-constructible, then* $\mathrm{TIME}[g] \multimap \mathrm{P}$.

In this section, we investigate the converse:

**Question 1.** *If $U \multimap \mathrm{P}$ is computable, is there a superpolynomial, time-constructible function $g$ such that* $\mathrm{TIME}[g] \subseteq \mathrm{P}^U$*?*

If the converse were to always holds, then we would have that any separation of P from a larger class closed under $\leqslant_T^{\mathrm{P}}$ by strong diagonalization could also be obtained by the time hierarchy theorem.

We were unable to settle this question, but we give several partial results. First we prove a technical result which allows us to show that the converse fails if we do not require $U$ to be computable (Theorem 18). If we require some extra, computable information about $U$, then the converse does hold (Theorem 19). We call universal languages for which such information is available, *graded* (see definition below). We define graded time-constructible functions similarly. However, there are non-graded time-constructible superpolynomial functions and non-graded computable universal languages (Theorems 20 and 21), and therefore the question above remains open.

**Theorem 17.** *If $V$ and $W$ are universal languages satisfying $[V] \cap [W] = \emptyset$, $[V]$ is closed under joins ($\oplus$) and $\leqslant_T^{\mathrm{P}}$, then there is a universal language $U \multimap [V]$ so that $P^U \cap [W] = \emptyset$. If $V$ and $W$ are computable, then $U$ can be made computable.*

*Proof.* We will construct $U$ in stages. At each stage $s$ we will have the characteristic function $U^s$, which will be zero outside of $D^s$. We will have $D^i \subseteq D^j$ for $i \leqslant j$ and $\bigcup_s D^s = \mathbb{N}^+$. We set $U(z) := \lim_{s\to\infty} U^s(z)$. It will be clear that this limit exists for all $z$. Furthermore, we will guarantee that for any $s$, the set $R_s := \{e\colon \exists x(\langle e,x\rangle \in D^s)\}$ is finite.

To ensure that $U \multimap [V]$, we satisfy the following requirements for all $e$:

$$P_e\colon \quad \exists j (U_j = V_e).$$

To ensure that $P^U \cap [W] = \emptyset$, we satisfy the following requirements for all $i, j$:

$$N_{(i,j)}\colon \quad \psi_{i;i}^U \neq W_j.$$

At stage 0 we set $U^0(z) := 0$ for all $z$ and we set $D^0 = \emptyset$. At stage $s+1$ with $s = 2e-2$ we satisfy $P_e$ as follows. We look for the smallest $j$ such that $j \notin R_s$. Since $R_s$ is finite, there must be such $j$. We set $D^{s+1} := D^s \cup \{\langle j,x\rangle\colon x \in \mathbb{N}^+\} \cup \{z\colon |z| \leqslant s\}$ and

$$U^{s+1}(z) := \begin{cases} U^s(z) & \text{if } z \in D^s \\ V_e(x) & \text{if } z = \langle j,x\rangle \\ 0 & \text{otherwise} \end{cases}$$

At stage $s+1$ with $s = 2(i,j)-1$ we satisfy $N_{(i,j)}$ as follows. We look for the smallest $x$ such that $\psi_{i;i}^{U^s}(x) \neq W_j(x)$. Since $R^s$ is finite, $U^s$ is equivalent to the join of finitely many languages from $V$, and therefore, since $[V]$ is closed under joins, $U^s \in [V]$. Since $[V]$ is closed under $\leqslant_T^{\mathrm{P}}$, $\psi_{i;i}^{U^s} \in [V]$ and therefore, since $[V] \cap [W] = \emptyset$, $\psi_{i;i}^{U^s} \neq W_j$. We set $D^{s+1} := D^s \cup \mathrm{queries}(\psi_{i;i}^{U^s}(x))$ and $U^{s+1} := U^s$. □

**Theorem 18.** *There is a (non-computable) $U \multimap \mathrm{P}$ so that for any time-constructible superpolynomial function $g$,* $\mathrm{TIME}[g] \not\subseteq \mathrm{P}^U$.

*Proof.* Take $g_0, g_1, \ldots$ to be a (non-computable) enumeration of time-constructible superpolynomial functions. For each $e$, pick $W_e$ to be a language in $\mathrm{TIME}[g_e] - \mathrm{P}$. Then by Theorem 17, $\mathrm{P}^U \cap [W] = \emptyset$ so for every $e$, we have $L_e \in \mathrm{TIME}[g_e] - P^U$. □

The universal language $U$ in Theorem 18 is not computable and we can not apply Theorem 17 to obtain a computable universal language in Theorem 18 due to the following result.

**Proposition 3.** *There is no computable $W$ such that*

1. *for every time-constructible superpolynomial function $g$, there is some $e$ such that $W_e \in \mathrm{TIME}[g] - \mathrm{P}$, and*
2. *for every $e$, $W_e \notin \mathrm{P}$.*

*Proof.* Assume $W$ is computable and satisfies (2). We will show that $W$ fails by constructing a time-constructible superpolynomial function $g$ for which (1) fails in stages using delayed diagonalization. At stage $n$ we define $g(n)$ and attempt to satisfy the requirement given by $\rho(n)$. We will make sure that computing $\rho(n)$ and attempting to satisfy

the requirement given by $\rho(n)$ takes no more time than linear time to ensure that $g$ is time constructible.

To ensure that (1) fails we satisfy the following requirements for all $e$:

$$R_{(e,n)}: \quad \exists x(W_e(x) \notin \psi_{n,g}(x)).$$

We set $g(n) = n^{\rho(n)}$ to attempt to satisfy requirement $R_{(e,n)}$ where $\rho(n) = (e,n)$. We will eventually succeed since $W_e \notin \mathrm{P}$.

We compute $\rho(n)$ as follows. We use a total of $n$ steps to check what requirements have already been satisfied, one at a time, in order $R_1, R_2, R_3, \ldots$. To check that requirement $R_e$ has been satisfied, we compute $W_e(x)$ and $\psi_{n,g}(x)$ for $x = 1, 2, 3, \ldots$ until we find a difference. If the last requirement which have so determined to be satisfied is $R_{(e,n)}$, we set $\rho(n) := (e,n) + 1$. Otherwise, we set $\rho(n) := 1$. □

**Definition 6.** *We say that* $U \dashrightarrow \mathrm{P}$ *is* graded *if there is a* grading *function* $h$ *satisfying* $U_{h(k)}(e,x) = \psi_{e;k}(x)$ *with* $h$ *computable. In particular,* $U_{h(k)} \multimap \mathrm{TIME}[n^k]$.

**Definition 7.** *We say that a superpolynomial function* $f$ *is* graded *if there is a computable* grading *function* $h$ *satisfying*

$$\forall k(\forall n \geqslant h(k))(n^k \leqslant f(n)).$$

If $f'$ is computable, nondecreasing, and unbounded, then $f$ given by $f(n) := n^{f'(n)}$ is graded superpolynomial.

**Theorem 19.** *If* $\mathcal{A}$ *is closed under* $\leqslant_m^{\mathrm{lin}}$*, then the following are equivalent:*

1. *There is a computable, graded* $U \multimap \mathrm{P}$ *in* $\mathcal{A}$
2. *There is a graded time-constructible superpolynomial function* $g$ *such that* $\mathrm{TIME}[g] \subseteq \mathcal{A}$.

*Proof.* If (2) holds, then the proof of the time hierarchy theorem shows that (1) holds.

Conversely, assume (1) holds and $U$ is graded by $h$. Define $g$ as follows. On input $n$, compute $h(1), h(2), \ldots$ for a total of $n$ steps. Assume the largest value so computed is $\ell = h(k)$. Set $g'(n) = k$. By the definition of $g'$, we can compute $h \circ g'(n)$ in $O(n)$ steps since $h \circ g'(n) = h(k) = \ell$. Since $g'$ is nondecreasing and unbounded, $g$ given by $g(n) = n^{g'(n)}$ is graded, time-constructible, and superpolynomial.

Now assume $L \in \mathrm{TIME}[g]$. Then, for some $e$, $\forall x(L(x) := \psi_{e,g(|x|)}(x))$ and so

$$L(x) \overset{(1)}{=} \psi_{e,g(|x|)}(x) \overset{(2)}{=} \psi_{e;g'(|x|)}(x) \overset{(3)}{=} U_{h \circ g'(|x|)}(e,x)$$

1. by choice of $e$,
2. since $g(n) = n^{g'(n)}$, and
3. since $U$ is graded by $h$.

Therefore, $L \leqslant_m^{\mathrm{lin}} U$ is witnessed by the reduction $x \mapsto \langle h \circ g'(|x|), e, x\rangle$. This shows that (2) holds. □

**Theorem 20.** *There exists a time-constructible non-graded superpolynomial function.*

*Proof.* We set $g(n) = \rho(n)$. We compute $\rho(n)$ in quadratic time and ensure that $\rho(n) \geqslant 2$; therefore, $g$ is time-constructible. Furthermore, we make sure that for every $k$, the set $\{n\colon \rho(n) = k\}$ is finite; therefore, $g$ is superpolynomial. To compute $\rho(n)$ we proceed as follows. First we compute

$$E_n := \{e\colon \psi_{e,n}(e+1)\downarrow \text{ and } e \leqslant n\}$$

which we can do in $O(n^2)$ steps. Next we use at most $n^2$ steps to compute an approximation $R'_n$ of

$$R_n := \{e \in E_n\colon (\exists m < n)(\psi_e(e{+}1) < m \wedge g(m) < m^{e+1})\}.$$

Finally, we set

$$\rho(n) := \begin{cases} \mu e(e \in E_n - R'_n) & \text{if } E_n - R_n \neq \emptyset \\ n+2 & \text{otherwise} \end{cases}$$

Notice that if $m \leqslant n$ then $E_m \subseteq E_n$, $R_m \subseteq R_n$, and $R'_m \subseteq R'_n$. Furthermore, if $e \in E_n$ then there is $n'$ such that $e \in R'_{n'}$ since otherwise consider the first $e$ for which this fails. For such $e$, we have $\forall^\infty n(g(n) = n^e)$, yet we know that $\psi_e(e+1)\downarrow$ and $m^{e+1}$ dominates $m^e$, so for some $n''$, $e \in R_{n''}$ and therefore for some $n'$, $e \in R'_{n'}$, which is a contradiction. Since

$$\{n\colon \rho(n) = k\} \subseteq \{k\} \cup \{n\colon k \in E_n - R'_n\}$$

and the the right hand set is finite, the left hand side set is also finite, as desired. □

**Lemma 2.** *If* $g$ *is a time-constructible non-graded superpolynomial function,* $f$ *is time-constructible and superpolynomial, and* $\mathrm{TIME}[f] \subseteq \mathrm{TIME}[g]$*, then* $f$ *is not graded.*

**Theorem 21.** *There exists a computable* $U \multimap \mathrm{P}$ *such that* $\langle U\rangle_m^{\mathrm{lin}}$ *does not contain a graded universal language for* P.

*Proof.* Pick a time-constructible non-graded superpolynomial function $g$, guaranteed to exist by Theorem 20. The proof of the time hierarchy Theorem shows that there exists $U \in \mathrm{TIME}[g]$ such that $U \multimap \mathrm{P}$. Now assume, to get a contradiction, that $V \in \langle U\rangle_m^{\mathrm{lin}}$ is a graded universal language for P. Then, since $\langle U\rangle_m^{\mathrm{lin}} \subseteq \mathrm{TIME}[g]$, $V \in \mathrm{TIME}[g]$. But then, by Theorem 19, we would have a graded time-constructible superpolynomial function $f$ satisfying $\mathrm{TIME}[f] \subseteq \langle U\rangle_m^{\mathrm{lin}} \subseteq \mathrm{TIME}[g]$, which contradicts Lemma 2. □

## References


[1] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. 2nd edition, 1995.

[2] L. Fortnow. Diagonalization. *Bulletin of the European Association for Theoretical Computer Science*, 71:102–112, 2000.

[3] D. Kozen. Indexings of subrecursive classes. *Theoretical Comput. Sci.*, 11:277–301, 1980.

[4] R. E. L. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[5] A. Nash, R. Impagliazzo, and J. Remmel. Universal languages and the power of diagonalization, 2003.

[6] A. A. Razborov and S. Rudich. Natural proofs. In *ACM Symposium on Theory of Computing (STOC)*, pages 204–213, 1994.

[7] S. Žàk. A turing machine time hierarchy. *Theoretical Comput. Sci.*, 26(3):327–333, 1983.



ECCC ISSN 1433-8092

http://eccc.hpi-web.de/