INTRINSIC COMPLEXITY OF RECURSIVE FUNCTIONS ON NATURAL NUMBERS WITH STANDARD ORDER

NIKOLAY BAZHENOV, DARIUSZ KALOCIŃSKI, AND MICHAL WROCLAWSKI

Abstract. Intrinsic complexity of a relation on a given computable structure is captured by the notion of its degree spectrum—the set of Turing degrees of images of the relation in all computable isomorphic copies of that structure. We investigate the intrinsic complexity of unary total recursive functions on nonnegative integers with standard order. According to existing results, possible spectra of such functions include three sets consisting of precisely: the computable degree, all c.e. degrees and all $\Delta_2$ degrees. These results, however, fall far short of the full classification. In this paper, we obtain a more complete picture by giving a few criteria for a function to have intrinsic complexity equal to one of the three candidate sets of degrees. Our investigations are based on the notion of block functions and a broader class of quasi-block functions beyond which all functions of interest have intrinsic complexity equal to the c.e. degrees. We also answer the questions raised by Wright [16] and Harrison-Trainor [8] by showing that the division between computable, c.e. and $\Delta_2$ degrees is insufficient in this context as there is a unary total recursive function whose spectrum contains all c.e. degrees but is strictly contained in the $\Delta_2$ degrees.

1. Introduction

In mathematics we study structures of various sorts like rings, fields or linear orders. In computability theory we investigate complexity of countable objects. A combination of the two—computable structure theory—examines the relationship between complexity and structure in the above sense [11, 11]. One of the main research programs in computable structure theory consists in the study of how complexity of a relation on a given structure behaves under isomorphisms (see, e.g., [12, 11, 9, 6]). Recall that a structure is computable if its domain and basic relations are uniformly computable. Complexity of a relation might be captured by a measure such as Turing degrees. This leads to the notion of the degree spectrum (of a computable relation on a computable structure)—the set of Turing degrees assumed by the images of that relation in all computable isomorphic copies of that structure. This notion captures what might otherwise be called the intrinsic complexity of a relation.

A natural motivation for investigating intrinsic complexity comes from treating computable copies of a structure as notations: we regard the elements of the copy as names for the members of the structure, with the underlying isomorphism acting as a naming function. A computable copy of a structure is thus a notation in which
all the basic relations are computable (meaning that their images within the copy are computable). This is essentially Shapiro’s idea, as studied, though in a very restricted sense, in [14]. But this analogy goes further. Shapiro insisted, not without reason, that computations are not performed directly on numbers but rather on their names (using the terminology of computable structure theory: computations are not performed on the underlying structure but on isomorphic copies). This intuition transfers to all computation-dependent notions, including complexity. In the end, the intricate notion of intrinsic complexity boils down to the study of how difficult it is to compute the relation in notations in which all the basic relations are computable.

Following Downey et al. [4] and Wright [16], we investigate degree spectra on the most common ordering: non-negative integers with the standard less than relation, denoted by \((\omega, <)\). We study this question in the restricted setting of specific binary relations of general interest—graphs of unary total computable functions. As an example of how isomorphism might influence the complexity of a such a function, consider the successor. By a well-known result (see, e.g., Example 1.3 in [2]), there is an isomorphic copy of \((\omega, <)\) in which the image of the successor computes the halting problem.

Several results from the literature partially characterize degree spectra of such functions. Moses [12] provided a syntactical characterization of intrinsically computable (i.e. having only the computable degree in their spectrum) \(n\)-ary relations on \((\omega, <)\). These results imply that a total unary recursive function is intrinsically computable if and only if it is almost constant or almost identity (see, Proposition 3.1). In [4], Downey, Khoussainov, Miller and Yu examined degree spectra of unary relations on \((\omega, <)\). Their results show, among others, that the spectrum of any infinite coinfinite computable unary relation on \((\omega, <)\) contains all c.e. degrees (Theorem 1.1 in [4]). Wright extended their results by showing the following.

**Theorem 1.1** (Wright [16]). The spectrum of a computable \(n\)-ary relation which is not intrinsically computable contains all c.e. degrees.

He was also able to show that a computable unary relation which is not intrinsically computable has \(\Delta_2\) degrees as a spectrum (see, also, [10]).

Wright asked in [16] whether the computable, the c.e. and the \(\Delta_2\) degrees exhaust possible degree spectra for computable \(n\)-ary relations on \((\omega, <)\). Roughly at about the same time, Harrison-Trainor posed a related question in [8] where he showed that either

1. there is a computable relation \(R\) on \((\omega, <)\) whose degree spectrum strictly contains the c.e. degrees but does not contain all of the \(\Delta_2\) degrees, or
2. there is a computable relation \(R\) on \((\omega, <)\) whose degree spectrum is all of the \(\Delta_2\) degrees but does not have this degree spectrum uniformly.

Harrison-Trainor conjectured that (1) holds for the relation he constructed. We construct a unary total computable function (hence, a computable binary relation) witnessing (1). This also answers Wright’s question.

Results of this paper are heavily based on certain structural characteristics of functions, which we refer to as the block and (a weaker) quasi-block property. Intuitively, each block function on \((\omega, <)\) is defined by multiple sub-functions where each sub-function applies to a different finite \(<\)-interval of \(\omega\) (Definition 3.5). A quasi-block function is one for which there are increasingly long initial \(<\)-segments
such that no number from within the segment is sent outside it. The usefulness of these properties is clear in view of the observation that any computable non-quasi-block function has exactly all c.e. degrees as a spectrum (Theorem 5.15). One of the main contributions of the paper consists in the complete characterization of degree spectra of block functions which have at most finitely many isomorphism types of their elementary sub-functions (Theorem 5.10). The second main contribution is Theorem 3.20 which answers Wright’s and Harrison-Trainor’s questions.

2. Definitions

Definition 2.1. $(\omega, <)$ is a computable copy of $(\omega, <)$ if $<$ is a computable ordering on $\omega$ and $(\omega, <)$ and $(\omega, <)$ are isomorphic.

Definition 2.2. Let $R$ be a relation on $(\omega, <)$, i.e. $R \subseteq \omega^k$, for some $k \in \omega$, and let $A$ be a computable copy of $(\omega, <)$. If $\varphi$ is an isomorphism from $(\omega, <)$ to $A$, we write $R_A$ for the image of $R$ under $\varphi$.

Definition 2.3. Let $R$ be a relation on a computable copy of $(\omega, <)$. The degree spectrum or spectrum of $R$ on $(\omega, <)$, in symbols $DgSp(\omega, <)(R)$, is the set of Turing degrees of $R_A$ over all computable copies $A$ of $(\omega, <)$.

Throughout the article, we use abbreviated forms: spectrum of $R$ and $DgSp(R)$.

Definition 2.4. Let $R$ be a relation on $(\omega, <)$. The relation $R$ is intrinsically computable if $DgSp(R)$ contains only the computable degree.

Let $A = (A, <_A)$ be a linear order. If $a \leq_A b$, then $[a;b]_A$ and $(a;b)_A$ denote the intervals $\{x : a \leq_A x <_A b\}$ and $\{x : a <_A x \leq_A b\}$, respectively. If the order $A$ is clear from the context, then we omit the subscript $A$. Succ is the successor function on $(\omega, <)$. $\langle \cdot , \cdot \rangle$ is the pairing function. Computability-related notation is standard and follows [15]. For example, $\leq_T$ denotes the Turing reduction.

If $X \subseteq \omega$ is a $\Delta_2$ set, then one can choose its computable approximation $\xi(k,s)$, i.e. a $\{0,1\}$-valued computable function such that $\lim_s \xi(k,s) = X(k)$, for all $k$. We often use notation $X_s(k)$ for $\xi(k,s)$.

3. Results

The following two statements will be useful.

Proposition 3.1. Let $f$ be a unary total computable function. Then $f$ is intrinsically computable if and only if either $f$ is almost constant, or $f$ is almost identity.

Proof. The result of Moses (Theorem 2 in [12]) implies that a function $f$ is intrinsically computable if and only if there is a finite tuple $\bar{a}$ from $\omega$ and a quantifier-free formula $\theta(x,y,\bar{a})$ such that

\[
    f(x) = y \iff (\omega, <) \models \theta(x,y,\bar{a}).
\]

If $f$ is almost constant or almost identity, then one can easily find $\bar{a}$ and $\theta(x,y,\bar{a})$ satisfying equation (1).

Now suppose that $f$ is intrinsically computable, and $f$ is not equal to almost identity. Choose a tuple $\bar{a} = a_0 < a_1 < a_2 < \ldots < a_n$ and a formula $\theta$ satisfying (1). Choose a number $x_0 > a_n$ such that $f(x_0) \neq x_0$. 
We argue that $f(x_0) < x_0$. Indeed, assume that $f(x_0) > x_0$. Then, since $\theta$ is quantifier-free and $x_0 > a_n$, it is clear that

$$(\omega, <) \models \forall y [y > x_0 \rightarrow \theta(x_0, y, \bar{a})].$$

This contradicts the fact that $f$ is a function. Now we have that $f(x_0) < x_0$ and $\theta(x_0, f(x_0), \bar{a})$ is satisfied. We deduce that

$$(\omega, <) \models \forall x [x > x_0 \rightarrow \theta(x, f(x_0), \bar{a})].$$

Thus, for almost all $x$, the value $f(x)$ equals $f(x_0)$, i.e., $f$ is almost constant.}

\textbf{Proposition 3.2} (see, e.g., Example 1.3 in \cite{2}). \textit{The spectrum of successor is equal to the c.e. degrees.}

\textbf{Theorem 3.3.} Let $f$ be a unary computable function with finite range. If $f$ is not intrinsically computable then its spectrum is equal to the $\Delta_2$ degrees.

\textbf{Proof.} The proof is based on the ideas from Theorem 1.2 of \cite{10}. We provide a detailed exposition, so that a reader could familiarize with the proof techniques.

We fix $c_0 \neq c_1$ such that $f^{-1}(c_1)$ is infinite. Without loss of generality, one may assume that $c_0 = 0$ and $c_1 = 1$.

Let $X \subseteq \omega$ be an arbitrary $\Delta_2$ set. We build a computable isomorphic copy $\mathcal{A} = (\omega, <_\mathcal{A})$ of the order $\langle \omega, < \rangle$ such that $f_\mathcal{A}$ is Turing equivalent to the set $X$.

Our construction will ensure that the following two conditions hold:

(i) $k \in X$ if and only if $f_\mathcal{A}(2k) = 1$, for all $k$;
(ii) the restriction of $f_\mathcal{A}$ to the set of odd numbers (i.e., $f_\mathcal{A} \upharpoonright \{2k + 1 : k \in \omega\}$) is computable.

It is clear that these conditions imply $f_\mathcal{A} \equiv_T X$.

Let $M$ be a large enough natural number such that

$$(\forall x > M)[\text{the } f\text{-preimage of } f(x) \text{ is infinite, and } x \notin \text{range}(f)].$$

Beforehand, we use odd numbers to copy the initial segment $[0; M]$ of $\langle \omega, < \rangle$. More formally, we put $2k + 1 <_\mathcal{A} 2l + 1$ for all $k < l \leq M$. In addition, any newly added (to the copy $\mathcal{A}$) number will be strictly $\mathcal{A}$-greater than $2M + 1$.

Our construction satisfies the following requirements:

\begin{equation}
(\mathcal{R}_e)
\begin{align*}
eq X & \iff f_\mathcal{A}(2e) = 0. \\
eq X & \iff f_\mathcal{A}(2e) = 0. \\
\end{align*}
\end{equation}

As usual, this will be achieved by working with a computable approximation $X_s(e)$.

By $\mathcal{A}_s$ we denote the finite structure built at a stage $s$. At each stage $s$, there is a natural isomorphic embedding $h_s$ from $\mathcal{A}_s$ into $\langle \omega, < \rangle$. If $\mathcal{A}_s$ consists of $a_0 <_\mathcal{A} a_1 <_\mathcal{A} a_2 <_\mathcal{A} \ldots <_\mathcal{A} a_n$, then we assume that $h_s(a_i) = i$, for all $i \leq n$.

This convention allows one to talk about values $f_{\mathcal{A}_s}(x)$ for elements $x \in \mathcal{A}_s$. We simply assume that

$$f_{\mathcal{A}_s}(a_i) = h_s^{-1} \circ f \circ h_s(a_i).$$

Our construction will ensure that $f_{\mathcal{A}_s}(x) = \lim_s f_{\mathcal{A}_s}(x)$, for all $x$. Sometimes (when the usage context is unambiguous), we write just $f_{\mathcal{A}_s}(x)$ in place of $f_{\mathcal{A}_s}(x)$.

\textit{Strategy $\mathcal{R}_s$ in isolation.} Suppose that $(s_0 + 1)$ is the first stage of work for this strategy. Then we add $2e$ to the end of $\mathcal{A}$. Since we want to ensure that
\[ f_{A_{s+1}}(2e) = X_{s+1}(e) \], we also add (if needed) finitely many fresh odd numbers in-between \( A_s \) and \( 2e \), i.e., we set
\[
a <_A 2k + 1 <_A 2e,
\]
for \( a \in A_{s_0} \) and newly added numbers \( 2k + 1 \).

We say that \( R_e \) requires attention at a stage \( s \) if the current value \( f_{A_e}(2e) \) is not equal to \( X_s(e) \). In order to deal with \( R_e \), we introduce the following important ingredient of our proof techniques. For the sake of future convenience, we give a general description of the module.

**Pushing-to-the-right module (PtR-module).**

We split the (current finite) structure \( A_s \) into three intervals: \( B <_A C <_A D \), where, say, we have \( B = [a; b]_A \), \( C = \{ c^0 <_A c^1 <_A \ldots <_A c^n \} \), and \( D = \{ d^0 <_A d^1 <_A \ldots <_A d^m \} \). Informally speaking, the module aims to achieve the following goal: while preserving all values \( f_A(x) \) for \( x \in B \cup D \), we want to change the function \( f_A \upharpoonright C \) in such a way that \( f_A \) satisfies a particular requirement. In addition, we require that \( C \) remains an interval inside \( F \).

More formally, we extend the structure \( A_s \) to a finite structure \( F \) (which is intended to be an initial segment of \( A_{s+1} \)) with the following properties:

- every element \( x \in F \setminus A_s \) is a fresh odd number, and each such \( x \) satisfies either \( B <_A x <_A C \) or \( x >_A C \);
- \( f_F(d^i) = f_{A_s}(d^i) \) for all \( i \leq n \);
- the new values \( f_F(c^j) \) satisfy some target condition.

In the future, when we talk about a particular instance of the module, we will always explicitly specify the desired target condition.

Roughly speaking, our module keeps the interval \( B \) fixed, while all elements from \( C \cup D \) are pushed to the right (with the help of newly added odd numbers). In addition, the elements of \( C \) stick together.

Going back to \( R_e \): if \( R_e \) requires attention at a stage \( s \), then we implement the following actions.

*The PtR-module for the strategy \( R_e \).* In our \( R_e \)-setting, we choose the middle interval \( C \) as the singleton \( \{2e\} \). The desired target condition is a natural one: we aim to satisfy \( f_A(2e) = X_s(e) \).

We build a finite structure \( F \) extending \( A_s \) as dictated by the PtR-module. Then we declare that \( F \) is the output of our module, and proceed further. This concludes the description of the \( R_e \)-strategy.

**Construction.** At a stage \( s + 1 \), we work with strategies \( R_e \), for \( e \leq s \). So, a strategy \( R_e \) starts working at the stage \( e + 1 \). For each \( R_e \) (in turn), our actions follow the description given above. After \( R_i \) finished its work, the PtR-module of the next strategy \( R_{i+1} \) works with the finite structure produced by \( R_i \). Since the described PtR-module preserves \( f_A \upharpoonright (B \cup D) \), our strategies do not injure each other. We define \( A = \bigcup_{x \in \omega} A_x \), where \( A_{s+1} \) is the final content of our structure produced by the PtR-module of \( R_s \) at the end of stage \( s + 1 \).

**Verification.** First, we show that in the construction, every application of a PtR-module is successful (i.e., one can always build a desired structure \( F \)).
In order to prove this, we consider our structures from a different angle: The structure \((\omega, <, f)\) can be treated as an infinite string \(\beta\) over a finite alphabet \(\Sigma = \text{range}(f)\), where the \(i\)-th symbol \(\beta(i)\) of the string is equal to \(f(i), i \in \omega\).

Then the construction of \(\mathcal{F}\) in the PtR-module can be re-interpreted as follows. We are given three finite strings: \(\sigma, \tau\) (of length one), and \(\rho\) (of length \(n+1\)) — for the intervals \(B, C = \{2e\}\), and \(D\) correspondingly. Our task is to find finite strings \(\tau', \rho_0, \rho_1, \ldots, \rho_n\) with the following property:

\[\sigma \tau' a \rho_0 \rho(0) \rho_1 \rho(1) \ldots \rho_n \rho(n),\]

where \(a = X_s(e)\), is an initial segment of \(\beta\).

This task can be always implemented successfully — this is a consequence of the following simple combinatorial fact.

**Remark 3.4.** Let \(\Sigma\) be a finite alphabet, and let \(\alpha \in \Sigma^\omega\) be an infinite string over \(\Sigma\). Suppose that every symbol from \(\Sigma\) occurs infinitely often in \(\alpha\). Then for every finite string \(\sigma \in \Sigma^{<\omega}\) of length \(m > 0\), one can find finite strings \(\tau_0, \tau_1, \ldots, \tau_{m-1}\) such that

\[\tau_0 \sigma(0) \tau_1 \sigma(1) \ldots \tau_{m-1} \sigma(m - 1)\]

is an initial segment of \(\alpha\).

So, we deduce that all applications of a PtR-module are successful. Hence, if \(e \leq s\), then by the end of the stage \(s + 1\) we have \(f_{\mathcal{A}, s+1}(2e) = X_s(e)\). This implies that every requirement \(\mathcal{R}_e\) is satisfied.

Each element \(a \in \mathcal{A}\) moves (to the right) only finitely often. Indeed, there are only finitely many even numbers \(2e\) such that \(2e \leq \alpha\). Consider a stage \(s^*\) such that the values \(X_s(e)\) (for these \(2e\)) never change after \(s^*\). Clearly, the element \(a\) never moves after the stage \(s^*\).

We deduce that the structure \(\mathcal{A}\) is a computable copy of \((\omega, <)\). For every \(k\), after the value \(f_{\mathcal{A}, s}(2k + 1)\) is defined for the first time, this value never changes (since the PtR-module always preserves the restriction \(f_{\mathcal{A}} \upharpoonright (B \cup D)\)). Therefore, our structure \(\mathcal{A}\) satisfies Conditions (i) and (ii) defined above. Theorem 3.3 is proved. \(\square\)

### 3.1. Block functions

From now on, we study some natural subclasses of unary total recursive functions with infinite range.

**Definition 3.5.** We say that a total function \(f: \omega \to \omega\) is a block function if for every \(a \in \omega\), there is a finite interval \(I\) of \((\omega, <)\) with the following properties:

- \(a \in I;\)
- \(I\) is closed under \(f\) (i.e., for all \(x \in I, f(x) \in I\));
- \(I\) is closed under \(f^{-1}\) (i.e., for all \(x \in I, f^{-1}(x) \subseteq I\)).

Suppose that \(I\) is the least such interval (with respect to set-theoretic inclusion). Then we say that the structure \((I, <, f \upharpoonright I)\) is an \(f\)-block (of the element \(a\)).

If \((I, <, f \upharpoonright I)\) is an \(f\)-block, we refer to its isomorphism type as an \(f\)-type (or a type).

**Remark 3.6.** For any computable block function \(f\) there is a 1-1 computable enumeration of its types. \(f\) can be represented by the infinite string \(\alpha_f \in [0; N]^\omega\), where \([0; N]\) is the domain of the enumeration, for some \(N \in \omega \cup \{+\infty\}\). For example, if \(I_0, I_1, \ldots, I_N\) are all (isomorphism types of) \(f\)-blocks, then \((\omega, <, f)\) can be treated as an infinite string \(\alpha_f \in \{n : 0 \leq n \leq N\}^\omega\), e.g. a string 012012012….
corresponds to a disjoint sum of the following form: \( I_0 + I_1 + I_2 + I_0 + I_1 + I_2 + I_0 + I_1 + I_2 + \ldots \)

**Example 3.7.** \( f(n) = 2 \cdot \left\lfloor \frac{n}{2} \right\rfloor \) is a block function. Its spectrum consists of all \( \Delta_2 \) degrees by Theorem 3.10 below.

**Example 3.8.** Consider finite structures \( J_n \) from Figure 1. Let \( g \) be the involution such that \( (\omega, <, g) \cong J_0 + J_1 + J_2 + \ldots \). Clearly, \( g \) is a block function. Proposition 3.9 shows its degree spectrum is all of the c.e. degrees.

**Proposition 3.9.** The degree spectrum of \( g \), as defined in Example 3.8, is all of the c.e. degrees.

**Proof.** Observe that \( \deg(c_g) = 0 \). By Proposition 3.19, Theorem 1.1 and Proposition 3.1, the spectrum of \( g \) is all of the c.e. degrees. For the sake of illustration, below we give a separate proof.

We show that the RS-module introduced in Section 3.2 can be successfully applied to \( (\omega, <, g) \) which implies that the spectrum of \( g \) is all of the c.e. degrees.

Let \( I \) be an initial segment of a computable copy \( A \) of \( (\omega, <) \). Let \( R \) be the condition saying that \( I \cong J_0 + J_1 + \ldots + J_k \), for some \( k \in \omega \). We show that \( R \) is decidable in \( g_A \).

Observe that given any \( a \in I \), one can computably in \( g_A \) recreate the \( g \)-block of \( a \) in \( A \). The idea is that, initially, the recreated fragment \( G := \{a\} \). We use the following procedure of one argument \( x \), starting from \( x = a \).

(\text{\bullet}) We compute \( g_A(x) \) and, if \( g_A(x) = x \), then we know there is precisely one number \( y \) such that \( x \) is between \( y_1 := y \) and \( y_2 := g_A(y) \), whereas if \( g_A(x) \neq x \), we know that there are precisely two numbers \( y_1, y_2 \) between \( x \) and \( g_A(x) \). In either case, we find \( y_1, y_2 \) accordingly, and for each \( y \in \{y_1, y_2\} \), if \( y \) is not yet in \( G \), we add it to \( G \) and we run (\text{\bullet}) on input \( y \).

One can prove that the above algorithm, when run with \( a \) on input, stops and after stopping, \( G \) consists of precisely the elements of the \( g \)-block of \( a \) in \( A \).

The above fact allows us to decide whether, for each \( a \in I \), \( I \) contains the whole \( g \)-block of \( a \). This is sufficient, since \( I \) is an initial segment and, therefore, if it consists of full blocks only, then it must have the desired form.

It remains to show that given any initial \( \prec_A \)-segment \( I_t \cong J_0 + J_1 + \ldots + J_t \), we can computably in \( g_A \) extend it to the \( \prec_A \)-segment \( I_{t+1} \cong J_0 + J_1 + \ldots + J_{t+1} \). We can proceed in stages as follows. At stage 0 we declare that all numbers outside \( I_1 \) are unused. At any stage \( s \), we take the least unused number and, computably
in $g_A$, we recreate its $g$-block in $A$. All numbers that are in this block join used elements. If the block is isomorphic to $J_{t+1}$ (which we check computably in $g_A$) then we already have $I_{t+1}$ which becomes $I_t$ extended by the $g$-block of $a$. Otherwise, we go to the next stage.

Notice that in the above argument, we have used the fact that $g$-types are pair-wise non-embeddable. Otherwise, we would not be sure, after finding some block isomorphic to $J_{t+1}$ whether we actually have a copy of $J_{t+1}$ or maybe a copy of some other type in which $J_{t+1}$ is embedded.

**Theorem 3.10.** Let $f$ be a computable block function such that it has only finitely many $f$-types and $f$ is not almost identity. Then the spectrum $DgSp(f)$ consists of all $\Delta_2$ degrees.

**Proof.** Let $I_0, I_1, \ldots, I_N$ be all (isomorphism types of) $f$-blocks. We represent the structure $B = (\omega, <, f)$ by $\alpha_f$ according to Remark 3.6 As in Theorem 3.3 we fix a $\Delta_2$ set $X$. Our goal is to construct a computable copy $A = (\omega, <, \alpha_f)$ of $(\omega, <)$ such that $f_A \equiv_T X$. In general, we follow the notations of Theorem 3.3 (e.g., $f_A, (x)$ is defined in the same way as in the previous proof).

Beforehand, we choose a large enough number $M$ such that:
- $M$ lies at the right end of its $f$-block (inside $B$),
- for every $x > M$, the isomorphism type of its $f$-block occurs infinitely often in $B$.

As in the proof of Theorem 3.3, we copy the interval $[0; M]$ into our structure $A$, and all new elements will be added to the right of this interval.

The proof is split into three cases which depend on the properties of the string $\alpha_f$ (each of the cases requires a separate construction):

- (s) There are two different finite strings $\sigma$ and $\tau$ such that:
  - the lengths of $\sigma$ and $\tau$ are the same;
  - $\tau$ can be obtained via a permutation of $\sigma$, i.e., there is a permutation $h$ of the set $\{0, 1, \ldots, |\sigma| - 1\}$ such that $\tau(i) = \sigma(h(i))$, for all $i < |\sigma|$;
  - both $\sigma$ and $\tau$ occur infinitely often in $\alpha_f$.
- (b) There is only one block $I_k$ such that $k$ occurs infinitely often in $\alpha_f$.
- (c) Neither of the previous two cases holds.

**Case (a).** For the sake of simplicity, we give a detailed proof for the case when $\sigma = 01$ and $\tau = 10$. After that, we explain how to deal with the general case.

Our construction satisfies the following requirements:

- $e \in X \iff 2e \text{ belongs to a block isomorphic to } I_1$,
- $e \notin X \iff 2e \text{ belongs to a block isomorphic to } I_0$.

Suppose that $|I_0| + |I_1| = q + 1$.

**Strategy $R_e$ in isolation.** When $R_e$ starts working at a stage $s_0 + 1$, we proceed as follows. Assume that $X_{s_0}(e) = 1$ (the other case is treated similarly). We choose $q$ fresh odd numbers $c_1, c_2, \ldots, c_q$ and declare them the companions of $2e$. We add the chain

$$2e <_A c_1 <_A c_2 <_A \ldots <_A c_q$$

to the right of $A_{s_0}$. If needed, we add finitely many fresh odd numbers in-between $A_{s_0}$ and $2e$. This procedure ensures that (at the moment) the finite structure $([2e; c_q]_A, <_A, f_A)$ is isomorphic to the disjoint sum $I_1 + I_0$. 

□
The strategy \( R_e \) requires attention at a stage \( s \) if inside the current \( A_s \), the number \( 2e \) belongs to a copy of \( I_{1-X_s(e)} \). When \( R_e \) requires attention, we apply a PtR-module.

The PtR-module for \( R_e \). We choose the middle interval \( C \) as the set containing \( 2e \) and all its companions, i.e. \( C = \{2e <_A e_0 |_A <_A e_1 |_A \ldots <_A e_q \} \). Our target condition is defined as follows: inside the resulting structure \( \mathcal{F} \), the structure \( (C, <_F, f_F | C) \) is isomorphic to the disjoint sum \( I_{X_s(e)} + I_{1-X_s(e)} \). As in Theorem 3.3, the structure \( \mathcal{F} \) is treated as output of the module.

The construction is arranged similarly to that of Theorem 3.3. Verification. We need to show that every application of a PtR-module is successful. This follows from two observations:

1. If we want to “transform”, say, \( I_0 + I_1 \) into \( I_1 + I_0 \), then this can be achieved by an appropriate pushing to the right, since the string \( \tau = 10 \) occurs infinitely often in \( \alpha_f \).
2. Remark 3.4 guarantees that one can also safely push the interval \( D \) (from the PtR-module): notice that if some block \( I_r \) occurs in \( D \), then \( r \) occurs infinitely often in \( \alpha_f \).

Since pushing to the right is always successful, every requirement \( R_e \) is satisfied. Note that given \( f_A \) as an oracle, one can recover the \( f_A \)-block of \( 2e \). This fact (together with \( R_e \)-requirements) implies that \( X \leq_T f_A \).

Every element \( e \in A \) is pushed to the right only finitely often. Therefore, the structure \( A \) is a computable copy of \(( \omega, <)\).

Given an odd number \( x = 2k + 1 \), one can computably determine which of the following two cases holds:

1. \( 2k + 1 \) is a companion \( e^t_p \) of some even number \( 2e \) (in this case, the indices \( e \) and \( t \) are also computed effectively), or
2. \( 2k + 1 \) is added as a “filler” by some action of an \( R_e \)-strategy (either by its initial actions, or by an application of a PtR-module).

In the second case, the value \( f_A(x) \) never changes (after being defined for the first time). In the first case, the oracle \( X \) can tell us whether \( x = e^t_p \) belongs to (a copy of) \( I_0 \) or \( I_1 \), and \( X \) can also compute the image \( f_A(x) \). In a similar way, \( X \) computes the images \( f_A(2e^t) \), for \( e \in \omega \). Hence, we obtain that \( f_A =_T X \). This concludes the case when \( \sigma = 01 \) and \( \tau = 10 \).

The case of arbitrary \( \sigma \) and \( \tau \) follows a similar proof outline. We illustrate this by considering \( \sigma = 012301 \) and \( \tau = 013021 \). Then our construction will switch between finite structures

\[
\mathcal{F}_\sigma = I_0 + I_1 + I_2 + I_3 + I_0 + I_1 \quad \text{and} \quad \mathcal{F}_\tau = I_0 + I_1 + I_3 + I_0 + I_2 + I_1.
\]

Since both \( \sigma \) and \( \tau \) occur infinitely often in \( \alpha_f \), an appropriate PtR-module can always “transform” \( \mathcal{F}_\sigma \) into \( \mathcal{F}_\tau \), and vice versa.

During the construction, an even number \( 2e \) will always belong to the third block from the left inside \( \mathcal{F}_0 \) (i.e., either \( I_2 \) in \( \mathcal{F}_\sigma \), or \( I_3 \) in \( \mathcal{F}_\tau \)). The third block is chosen because it corresponds to the first position, where \( \sigma \) and \( \tau \) differ. The rest of the corresponding copy of \( \mathcal{F}_0 \) consists of companions of \( 2e \). In the final structure \( A \), we will achieve the following: if \( e \in X \), then \( 2e \) lies in a copy of \( I_2 \); otherwise, \( 2e \) belongs to a copy of \( I_3 \). This concludes the discussion of Case (a).
CASE (B). Without loss of generality, we assume that \( I_k = I_0 \). We satisfy the following requirements:

\[
\begin{align*}
\mathcal{R}_e & \quad e \in X \iff 2e \text{ lies at the right end of a copy of } I_0, \\
\mathcal{R}_e & \quad e \notin X \iff 2e \text{ lies at the left end of a copy of } I_0.
\end{align*}
\]

Suppose that \( |I_0| = q + 1 \). Notice that \( q \geq 1 \), since \( f \) is not almost identity.

*Strategy \( \mathcal{R}_e \) in isolation.* \( 2e \) will have finitely many odd numbers as its companions. In contrast to Case (a), these companions could be added stage-by-stage.

When \( \mathcal{R}_e \) starts working at a stage \( s_0 + 1 \), we proceed as follows. Suppose \( X_{s_0}(e) = 1 \) (the other case is similar). Then we choose \( q \) fresh odd numbers \( c_1, \ldots, c_q \), and declare that they are companions of \( 2e \). We set \( c_1 <_A \ldots <_A c_q <_A 2e \) (these elements are added to the right of \( A_{s_0} \)). We ensure that the structure \( ([c_1;2e],<_A,<_A,f_A) \) is isomorphic to \( I_0 \) (if needed, one adds fresh odd numbers in-between \( A_{s_0} \) and \( c_1 \)).

We also ensure that by the end of each stage \( s \), \( 2e \) and its (current) companions form an interval inside \( A_s \), and this interval can be treated as a sum of blocks (in \( A_s \)).

The strategy \( \mathcal{R}_e \) requires attention at a stage \( s \) if inside the current \( A_s \), the corresponding requirement is not satisfied (e.g., if \( X_s(e) = 0 \) and \( 2e \) lies at the right end of \( I_0 \)). When \( \mathcal{R}_e \) requires attention, we apply a PtR-module.

The PtR-module for \( \mathcal{R}_e \). We choose the middle interval \( C \) as the set containing \( 2e \) and all its current companions. We consider the following two subcases.

**Subcase 1.** Assume that right now, \( X_s(e) = 1 \) and \( 2e \) lies at the left end of a copy of \( I_0 \). Then our target condition is defined as follows: inside the resulting output structure \( F \), the number \( 2e \) should belong to the right end of a copy of \( I_0 \).

In order to achieve this condition, we add precisely \( q \) fresh odd numbers in-between \( B \) and \( C \), and only one fresh odd number in-between \( C \) and \( D \). This guarantees that \( 2e \) “moves” to the right end of a block.

**Subcase 2.** Otherwise, suppose that \( X_s(e) = 0 \) and \( 2e \) lies at the right end of a copy of \( I_0 \). Then we pursue the following condition: inside the output \( F \), \( 2e \) should “move” to the left end of a block \( I_0 \).

In order to do this, we add one fresh number in-between \( B \) and \( C \), and \( q \) fresh numbers in-between \( C \) and \( D \).

In both subcases, we declare that the newly added odd numbers belong to the set of companions of \( 2e \).

The construction is arranged similarly to the previous ones.

Verification. Since almost every block from \( A_f \) is isomorphic to \( I_0 \), every application of a PtR-module is successful. In addition, the actions of the PtR-module for \( \mathcal{R}_e \) does not injure other strategies.

We deduce that all requirements \( \mathcal{R}_e \) are satisfied. Given \( f_A \) as an oracle, one can recover the position of \( 2e \) inside its \( f_A \)-block. This implies that \( X \leq_T f_A \). In addition, a standard argument shows that \( A \) is a computable copy of \( (\omega,<_f) \).

Notice the following. Since \( 2e \) and its companions always stick together as an interval, there are only two possible variants of the final \( f_A \)-block of \( 2e \): either it contains \( q \) companions of \( 2e \) added at the very beginning of the work of the \( \mathcal{R}_e \)-strategy, or it contains \( q \) closest (inside \( A \)) companions of \( 2e \) added by the first application of the PtR-module for \( \mathcal{R}_e \).

As in the previous case, given an odd number \( x = 2k + 1 \), one can determine which of the following two cases holds:
Let \( \tau \) be decomposed as
\[
\tau = \xi b \delta c^k b \quad \text{for some } k \geq 1 \quad \text{and finite strings } \xi, \delta. \quad \text{The string } \tau_b = \xi b \delta c^k b \text{ occurs infinitely often in } \alpha_f. \quad \text{In turn, this implies that both } b \delta c^k \text{ and } \delta c^k b \text{ occur infinitely often in } \alpha_f. \quad \text{Therefore, } \alpha_f \text{ satisfies Case (a), which gives a contradiction.}
\]
Hence, we have \( \tau = \rho b^k \) for some \( k \geq 1 \) and finite string \( \rho \), and the string \( \tau_b = \rho b^{k+1} \) occurs infinitely often in \( \alpha_f \). Note that \( #(\tau_b) = #(\tau) \). This implies that by applying induction, one can show that for every \( l \geq 1 \),
\[
(3) \quad \rho b^l \text{ occurs infinitely often in } \alpha_f.
\]
Since \( \alpha_f \) does not satisfy Case (b), there are at least two different symbols occurring infinitely often in \( \alpha_f \). This fact and (3) imply that for every \( n \in \omega \), there exist \( m > n \) and two symbols \( d' \) and \( e' \) such that \( d' \neq b, e' \neq b \), and \( d' b^m e' \) occurs in \( \alpha_f \). After that, we apply the pigeonhole principle to finish the proof of the lemma. \( \square \)

By Lemma 3.11, we may assume that for every \( n \in \omega \), there exists \( m > n \) such that, say, \( 10^m 2 \) occurs in \( \alpha_f \). We satisfy the same requirements as in Case (b):
\[
(\mathcal{R}_e) \quad \begin{cases} 
 e \in X & \iff 2e \text{ lies at the right end of a copy of } I_0, \\
 e \notin X & \iff 2e \text{ lies at the left end of a copy of } I_0.
\end{cases}
\]
In general, our notations also follow those of Case (b).

**Strategy \( \mathcal{R}_e \) in isolation.** When \( \mathcal{R}_e \) starts working at a stage \( s_0 + 1 \), we proceed as follows. Suppose \( X_{s_0}(e) = 0 \). We find a large enough number \( m \) such that \( 10^m 2 \) occurs in \( \alpha_f \).
occurs in \( \alpha_f \), and the corresponding sequence of \( f \)-blocks \( I_1 + I_0 + I_0 + \ldots + I_0 + I_2 \) does not intersect with the image of \( A_{s_0} \) inside \( (\omega, \omega) \).

We add \( 2e \) and fresh odd numbers into \( A \) ensuring that the newly added elements form a sequence of \( f_{A_s} \)-blocks:

\[
I_1 + I_0 + \ldots + I_0 + I_2;
\]

if needed, fresh odd numbers are also added in-between \( A_{s_0} \) and this sequence. The number \( 2e \) lies at the left end of the leftmost block \( I_0 \). The elements forming \( I_1 \) and \( I_2 \) are declared boundary companions of \( 2e \). The odd numbers forming the inner sequence of \( I_0 \)-s are declared non-boundary companions of \( 2e \).

As usual, \( R_e \) requires attention at a stage \( s \) if inside the current \( A_s \), the corresponding requirement is not satisfied. When \( R_e \) requires attention, we apply a PtR-module.

The PtR-module for \( R_e \). We choose the middle interval \( C \) as the set containing \( 2e \) and all its companions. Assume that right now, \( X_s(e) = 0 \) and \( 2e \) lies at the right end of a copy of \( I_0 \) (the other subcase is treated in a similar way). Then the target condition is defined as follows: inside \( F \), the number \( 2e \) belongs to the left end of a copy of \( I_0 \).

Suppose that right now, the companions of \( 2e \) form a sequence of \( f_{A_s} \)-blocks corresponding to a finite string \( 10^m2 \).

We always assume the following: if a fresh number \( x \) is added between some companions of some \( 2j \), then it is declared a non-boundary companion of \( 2j \). In addition, every such \( x \) is put between the \( I_1 \)-block and the \( I_2 \)-block containing the boundary companions of \( 2e \). Moreover, we require that inside the resulting structure \( F \), the element \( x \) becomes a part of a copy of \( I_0 \).

In order to achieve the target condition, we proceed as follows. First, we find a large enough \( m' > m \) such that \( 10^m2 \) occurs in \( \alpha_f \), and this occurrence of \( 10^m2 \) lies to the right of the image of \( A_s \) inside \( (\omega, \omega) \). We add fresh odd numbers in such a way that:

- The companions of \( 2e \) (including newly added companions) form a sequence of \( f_{A_s} \)-blocks corresponding to \( 10^m2 \) (inside \( \alpha_f \)). This is achieved by adding numbers in-between \( B \) and \( C \), and by adding fresh \( I_0 \)-blocks between the \( I_1 \)-block and the \( I_2 \)-block containing the boundary companions of \( 2e \).
- Similarly to Case (b), this procedure must ensure that \( 2e \) moves to the left end of an \( I_0 \)-block.

Second, we carefully push the companions of \( 2j \), where \( e < j < s \), to the right. Consider each such \( j \) (in turn). Suppose that the companions of \( 2j \) form a sequence of \( f_{A_s} \)-blocks corresponding to a finite string \( 10^{m_j}2 \). We choose a large enough \( m'_j > m_j \) (again, with \( 10^{m_j}2 \) occurring in \( \alpha_f \) to the right of the image of the current (preliminary) version of \( F \)). We add fresh numbers in such a way that:

- The companions of \( 2j \) (including new ones) form a sequence of \( f_{x} \)-blocks corresponding to \( 10^{m_j}2 \) inside \( \alpha_f \).
- If \( x \) is a new companion of \( 2j \), then it belongs to a new \( I_0 \)-block which corresponds to one of the underlined zeros in the following decomposition:

\[
10^{m_j}2 = 10^{m_j}00 \ldots 02.
\]
This careful pushing allows to ensure that the PtR-module does not injure strategies \( R_j \), for \( j \neq e \). Indeed, after the pushing, the value \( f_A(2j) \) does not change.

The construction is arranged in a similar way as before.

Verification. The fact that \( \alpha_f \) contains occurrences of \( 10^m2 \) for arbitrarily large \( m \) implies that every application of a PtR-module is successful. We deduce that all requirements \( R_e \) are satisfied. The rest of the verification is similar to that of Case (b). This concludes the proof of Theorem \([3.10]\) \( \square \)

### 3.2. Quasi-block functions.

The notion of a quasi-block function is a generalization of the notion of a block function. Unlike blocks which are disjoint and follow each other, quasi-blocks are increasingly larger and they are initial segments of \( \omega \).

**Definition 3.12.** We say that \( f : \omega \rightarrow \omega \) is a quasi-block function if there are arbitrarily long finite initial segments of \( \omega \) closed under \( f \). If \( f \) is a quasi-block function but not a block function, we call \( f \) a proper quasi-block function.

**Example 3.13.** Euler’s function is a function \( \varphi \) such that if \( n > 0 \), then \( \varphi(n) \) is the number of such \( m \leq n \) that \( m \) and \( n \) are relatively prime. \( \varphi \) is a proper quasi-block function.

Since \( \varphi \) has a computable non-decreasing lower bound \( \frac{\sqrt{n}}{2} \) diverging to \( \infty \), the spectrum of \( \varphi \) is equal to the c.e. degrees by Theorem \([3.10]\).

**Example 3.14.** The function \( nd : \omega \rightarrow \omega \) assigning to each \( n > 0 \) the number of its divisors is a proper quasi-block function.

Below we describe a method used to show that the degree spectrum of a certain unary recursive function \( f \) consists exactly of c.e. degrees.

**Retrieving the Successor module (RS):** on \( (\omega, <) \), for \( f \) recursive, is a scheme of algorithm which, for any computable copy \( A \) of \( (\omega, <) \) and an initial segment \( I_t \) of \( A \) satisfying some condition \( R \) (to be specified in a concrete implementation) computes, uniformly in \( t \) and relative to \( f_A \), a longer initial segment \( I_{t+1} \) of \( A \) satisfying \( R \), which enables us to construct an increasing sequence of initial segments \( I_0 \subseteq I_1 \subseteq ... \).

Suppose that there exists a concrete implementation of the RS-module for \( (\omega, <, f) \). We wish to show that the degree spectrum of \( f \) on \( (\omega, <) \) consists of exactly c.e. degrees. To this aim, we want to show that \( Succ_A \) is Turing-reducible to \( f_A \). We also observe that the reduction in the other direction works. We conclude that \( Succ_A \equiv_T f_A \), hence \( DgSp(Succ) = DgSp(f) \), i.e. they consist of all c.e. degrees.

This conclusion is based on Proposition \([3.2]\).

Suppose that the initial segment of \( \omega \) up to \( n \) (according to \( < \)) has already been determined, along with its isomorphic image \( I_t \) in \( (\omega, <) \). In this description we adopt a convention that the isomorphic image of each number \( i \) is \( k_i \). Observe that for each number \( i \) such that \( k_i < k_n \) we know how to determine its successor in \( (\omega, <) \). In an application of the RS-module, given \( k_n \)—the rightmost element of \( I_t \)—we get some \( k_m \) and \( m \) such that \( k_n < k_m \) and \( [k_0; k_m]_A \) satisfies \( R \). We know that in \( A \) there are exactly \( m - n - 1 \) elements between \( k_n \) and \( k_m \). Since the ordering \( < \) is recursive, we can check elements one by one until we determine what elements (and in what order) are between \( k_n \) and \( k_m \). Thus we extended the initial segment \( I_t \) of \( A \) to a larger initial segment \( I_{t+1} \) satisfying \( R \) and we are able to retrieve more values of the successor in this structure.

**Theorem 3.15.** The spectrum of any unary total computable non-quasi-block function is equal to the c.e. degrees.
Proof. We show that the RS module can be used for \((\omega, <, f)\). Given a computable copy \(A\) of \((\omega, <)\), we set \(I_0\) as the image of some initial segment of \((\omega, <)\) such that for every \(n\) outside \(I_0\) there is \(m < n\) such that \(f(m) > n\). The condition \(R\) states that there is \(j \in I_1\) such that \(f(j) > n\). Then if we already know \(I_t\) and want to determine \(I_{t+1}\), we calculate both \(f(j)\) and \(f_A(k_j)\) from the condition \(R\), obtaining some values of these functions \(m\) and \(k_m\), each of them somewhere behind \(n\) and \(k_n\) in their sequences. \(\square\)

**Theorem 3.16.** If \(f\) is a recursive proper quasi-block function with a computable non-decreasing lower bound diverging to \(+\infty\), then its spectrum consists of exactly c.e. degrees.

Proof. We claim that there exist only finitely many quasi-blocks closed under both \(f\) and \(f^{-1}\). Observe that if there were infinitely many such quasi-blocks, then \(f\) would be a block function. Observe also that if \(f\) is as above, then we are able to calculate how many times each of its values is assumed.

We utilise the RS module. The segment \(I_0\) is any initial segment such that none of its super-quasi-blocks is closed under both \(f\) and \(f^{-1}\). Assume we already have a segment \(I_t\) of \(A\) retrieved. \(I_t\) satisfies the condition \(R\) stating that it is a initial segment which is not closed under both \(f\) and \(f^{-1}\).

We wish to algorithmically construct \(I_{t+1}\) satisfying the same condition \(R\). If there is \(n \in I_t\) such that \(f(n) > I_t\), we set \(I_{t+1}\) as the segment consisting of all elements up to \(f(n)\). If not, then there must be \(m \in I_t\) such that for some \(n > I_t\), \(f(n) = m\). What is more, for every such \(m\) there are only finitely many arguments satisfying this identity and we are able to determine what they are. If \(M\) is the largest of these elements, then we set \(I_{t+1}\) as the segment until \(M\). \(\square\)

**Theorem 3.17.** There exists a recursive quasi-block function \(f\) with a non-decreasing lower bound diverging to \(+\infty\) but with no such computable bound with all c.e. degrees as a spectrum.

Proof. Consider a set \(A \subseteq \omega\) which is \(\Delta_2\) but not computable. Observe that for each such set there is a recursive sequence \(g\) of natural numbers such that each natural number appears in \(g\) at most finitely many times and for any \(n \in \omega\), \(n \in A\) if the number of occurrences of \(n\) in \(g\) is odd.

\(f\) is going to be \(g\) modified in such a way that we put some fixed points between elements of \(g\), pushing these elements to the right, to ensure that \(f\) is a quasi-block function. We will be able to easily distinguish (within \(f\)) old elements of \(g\) from the new filler elements, because only the new elements are going to be fixed points of \(f\).

We construct \(f\) by finite extension, starting from the empty function. Initially, all elements of sequence \(g\) are unused. At any given stage, suppose that \(g(m)\) is the least unused element of sequence \(g\) and that \(n\) is the least argument such that \(f(n)\) is not defined yet. If \(g(m) > n\), then for each \(i = n, \ldots, g(m)\) assign \(f(i) = i\). Regardless of whether you performed the previous instruction, assign value \(g(m)\) to the least \(i\) such that \(f(i)\) has no value set yet. If the least such \(i\) is equal to \(g(m)\), then put an additional fixed point before it to ensure that all fixed points serve as fillers in \(g\). We declare that \(g(m)\) is used and go to the next stage.

This is a quasi-block function because each argument \(n\) is either a fixed point or is a number from sequence \(g\) which has been pushed so far to the right that
f(n) < n. Hence every finite initial segment of ω is closed under f. However, this is not a block function. If it were, then every m such that f(m) = n would need to be in the same block as n. Then we would be able to count how many times n is assumed as the value of f and hence A would be decidable.

The lower bound of f diverges to ∞ because every value can be assumed only finitely often. However, no such bound is computable because otherwise we would be able to determine the last occurrence of every number in g and A would be computable. Observe we can assume that this bound is non-decreasing. We just need to set f(n) = the largest m such that f(i) ≥ m whenever i ≥ n.

If A is a c.e. set, then we utilise the RS module to show that the degree spectrum of f consists of exactly the c.e. degrees. We can assume without loss of generality that g assumes each of its values only once, then so does f if we ignore fixed points.

We take I_q such that behind it there are no quasi-blocks closed under f. The condition R states that there is an element n ≥ I_q such that f(n) ∈ I_q. Observe that such element is determined uniquely. We want to retrieve I_{q+1} ≥ I_q satisfying R. We need to look for n described above and then to fill in all the missing numbers between I_q and n. Since the segment thus obtained is not a block, it needs to satisfy R. We call this segment I_{q+1}.

3.3. Unusual degree spectrum. In this section we answer Wright’s question (Question 6.2 in [16]). The result we prove here is also relevant for Harrison-Trainor’s question (p. 5 in [5]). Recall a representation of a block function f as an infinite sequence α of (the indices of) types (see, Remark 3.6).

Definition 3.18. Let f be a computable block function with infinitely many types. The counting function for f is defined by c_f(n) = #i : α_f(i) = n.

Proposition 3.19. Let f be a computable block function with infinitely many pairwise non-embeddable types, each occurring finitely often. Then deg(c_f) is c.e. and f_A ≥_T c_f implies that deg(f_A) is c.e.

Proof. C^{<}_f := \{(k, n) : k ≤ c_f(n)\} is c.e., C^{≥}_f := \{(k, n) : k ≥ c_f(n)\} is co-c.e., so deg(C^{<}_f ⊕ C^{≥}_f) is c.e. Since C^{<}_f ⊕ C^{≥}_f ≡_T c_f, c_f is of c.e. degree.

Assume that f_A ≥_T c_f. Hence, Succ_A ≤_T f_A. But Succ_A ≥_T f_A always (for a computable f). Hence f_A ≡_T Succ_A, and thus, by Proposition 3.2 f_A is of c.e. degree.

Theorem 3.20. There exists a total computable function whose degree spectrum strictly contains all c.e. degrees and is strictly contained in the ∆_2 degrees.

We construct a computable block function f with infinitely many types and each c_f(n) finite. We want c_f <_T 0' and a computable copy A of (ω, <) with f_A of non-c.e. degree. Combining this with Proposition 3.19 and a result by Cooper, Lempp and Watson from [5] (see Theorem 3.26) finishes the proof.

For each e, e_1, e_2 ∈ ω, we have the following requirements:

I_e : I ≠ Φ^{I}_{e_1}, J_e : J ≠ Φ^{I}_{e_2}, and R_{<_T} : Φ^{I}_{e_1} ≉ W_n \lor Φ^{I}_{e_2} ≉ W_n \lor Φ^{I}_{e_1} ≉ \Gamma f_A,

where Γ f_A is the graph of f_A. The non-c.e. degree requirements are based on [5] p. 195.

At stage s we have finite sets I_s, J_s, structure A_s = (A_s, <_{A_s}) and a function f_{A_s} : A_s → A_s. Eventually, we set A = \bigcup_{s ∈ ω} A_s. We assume some recursive
3.3.1. Construction. Let \( (C_i)_{i \in \omega} \) be a computable sequence of cycles, where \( C_i \) is of length \( 2^i \) (Figure 2). Put \( I_0 = J_0 = A_0 = f_{A_0} = \emptyset \). Requirements have no reserved numbers, no numbers are frozen. Below we describe stage \( s + 1 \), for \( s \in \omega \).

1. If no requirement needs attention at stage \( s + 1 \), we choose the highest priority requirement with no reservation. If this is some \( I_e \) (or \( J_e \)), we reserve it for the least fresh number \( x \). If the highest priority requirement with no reservation is some \( R_{(e_1,e_2,n)} \), we reserve for it the least fresh \( \langle u,v \rangle \), for \( A_s \) (i.e. \( u,v \) do not occur in \( A_s \)), and three consecutive fresh numbers \( t_0, t_1, t_2 \), called tickets. We apply PtR by setting \( B = A_s \), \( C = D = \emptyset \) and \( E \supseteq \{ u,v \} \) such that \( |E| = 2^{t_0} + 2^{t_1} \) with every \( x \in E \) being fresh for \( A_s \). We build a structure \( E = (E, \prec_E, g) \) where \( \prec_E \) is a linear order satisfying the \( E \)-condition, depicted in Figure 3 which is:

- \( C_{t_0} \cup C_{t_1} \supseteq E \),
- \( u \) is the \( \prec_E \)-last element in the block corresponding to \( C_{t_0} \), and
- \( v \) the \( \prec_E \)-first element in the block corresponding to \( C_{t_1} \).

We set \( A_{s+1} = A_s + E \). We have \( \langle u,v \rangle \notin \Gamma_{f_{A_{s+1}}} \). We enumerate ticket \( t_0 \) into \( I \).

2. If a requirement needs attention, pick the highest one. We say it receives attention. If this is \( I_e \), some \( x \) is reserved for \( I_e \) at stage \( s \) and \( I_s(x) = \Phi^{J_s}_{e,s}(x) \).

Put \( x \) into \( I \), freeze the computation \( \Phi^{J_s}_{e,s}(x) \) and cancel all freezings and reservations for lower priority requirements. Deal with with \( J_e \) accordingly.
with reservation

Below we describe reactions to first and second attention received by some $\langle u, v \rangle$ by fresh numbers, thick part represents pushed numbers.

Figure 4. $A_s$ when $R_{\langle e_1, e_2, n \rangle}$ receives attention for the first time with $\langle u, v \rangle$ and tickets $t_0, t_1, t_2$, assuming that the reservation has been made at stage $r$.

Figure 5. The result of reaction to first attention for $R_{\langle e_1, e_2, n \rangle}$ with reservation $\langle u, v \rangle$ and tickets $t_0, t_1, t_2$. Gray part is occupied by fresh numbers, thick part represents pushed numbers.

Figure 6. $A_s$ when $R_{\langle e_1, e_2, n \rangle}$ receives attention for the second time with $\langle u, v \rangle$ and tickets $t_0, t_1, t_2$, assuming that the reservation has been made at stage $r$.

Suppose the highest priority requirement needing attention is some $R_{\langle e_1, e_2, n \rangle}$. Some $\langle u, v \rangle$ is reserved for $R_{\langle e_1, e_2, n \rangle}$ at stage $s$ with some tickets $t_0, t_1, t_2$.

Below we describe reactions to first and second attention received by $R_{\langle e_1, e_2, n \rangle}$ with reservation $\langle u, v \rangle, t_0, t_1, t_2$.

(i) Suppose the reservation for $R_{\langle e_1, e_2, n \rangle}$ has been made at stage $r$. After $r$ and before $s + 1$ the structure $A$ might have been extended by some $T$ (thick line in Figure 4). The idea is that we push to the right all numbers occupying the highlighted positions in Figure 4 and obtain the structure as in Figure 5.

More formally, divide $A_s$ into $A_s = B + C + D$, where $D \cong A_{r-1}$, $C \cong C_{t_0} + C_{t_1}$, and $D \cong T$, and apply PtR. Take $|C \cup D|$ numbers, fresh for $A_s$, and make $F$ out of them. Build a structure $\mathcal{F} = (F, <_F; g)$, where $<_F$ is a linear order, satisfying the $F$-condition $\mathcal{F} \cong C + D$.

We rebuild $C$ to get $C' = (C, <_C; h)$ where $C'$ satisfies the $C$-condition $C' \cong C_{t_1} + C_{t_0}$. We set $A_{s+1} = B + F + C' + D$ (Figure 5). Observe that pushed numbers from $C + D$ assume in $A_{s+1}$ the same order structure as in $A_s$ but the behavior of $f_{A_{s+1}}$ mimics that on $C_{t_1} + C_{t_0}$. This makes $\Gamma_{f_{A_{s+1}}} (\langle u, v \rangle) = 1$ and thus $R_{\langle e_1, e_2, n \rangle}$ is satisfied at stage $s + 1$. We enumerate $t_1$ into $I$ and invalidate all reservations and freezings for lower priority requirements.

(ii) Suppose $R_{\langle e_1, e_2, n \rangle}$ has made the reservation at stage $r$ and received the first attention at stage $p + 1$. By the time we got to stage $s + 1$, the structure $A$ might have been extended by some $U$ (Figure 6). The idea is that we push all numbers occupying the highlighted positions in Figure 6 and obtain the structure as in Figure 7.
Lemma 3.21. \( A \) is computable.

Proof. It is clear that the domain of \( A \) is \( \omega \). By the construction, once two numbers enter \( A \), their order according to \(<_A \) is never changed. Hence, \( A_s \subset A_{s+1} \), for every \( s \). Thus we can set \( A = \bigcup_{s \in \omega} A_s \). Clearly, \(<_A \) is computable: to decide whether \( x <_A y \) holds it suffices to carry out the construction until some stage \( s_0 \) at which \( x, y \in A_{s_0} \). We know that \( x <_A y \iff x <_{A_{s_0}} y \).

Lemma 3.22. Every requirement is eventually satisfied. Hence, \( I, J \) are intermediate and \( f_A \) is of non-c.e. degree.

Proof. This follows from finite-injury. It remains to observe that each requirement can receive attention only finitely many times with the same numbers reserved for it. This is clear for \( I, J \) (see, e.g. [15] Chap. VII.2). We show that no \( \mathcal{R}_{e_1,e_2,n} \) needs attention more than twice with the same \( \langle u, v \rangle \) and tickets \( t_0, t_1, t_2 \) reserved for it (cf. [5] p. 196). Suppose the reservation was made at stage \( r \), the first attention was at stage \( s+1 \) and the second at stage \( t+1 \). Since \( \langle u, v \rangle, t_0, t_1, t_2 \) are reserved for \( \mathcal{R}_{e_1,e_2,n} \) at stage \( t \geq s+1 \), no requirement with lower priority than \( \mathcal{R}_{e_1,e_2,n} \) has received attention at any stage \( u, t \geq u \geq s+1 \). Actions performed at stage \( t+1 \) lead to \( A_{t+1} \upharpoonright A_s = A_s \upharpoonright A_s \). Therefore, \( \Phi_{e_1}^{\Gamma_{f_{A_{t+1} \upharpoonright A_s}}} [z] = \Phi_{e_1}^{\Gamma_{f_{A_s} \upharpoonright A_s}} [z] = W_{n,s} [z] \). At stage \( s+1 \) we had \( \Phi_{e_2}^{W_{n,s} [\langle u, v \rangle]} = \Gamma_{f_{A_s} \upharpoonright A_s} \langle u, v \rangle \neq \Gamma_{f_{A_s} \upharpoonright A_s} \langle u, v \rangle \). Since at stage \( t+1 \) we had \( \Phi_{e_2}^{W_{n,t} [\langle u, v \rangle]} = \Gamma_{f_{A_t} \upharpoonright A_t} \langle u, v \rangle \) we must have \( W_{n,t} [z] \neq W_{n,s} [z] \). Hence, for some \( x \), \( \Phi_{e_1}^{\Gamma_{f_{A_{t+1} \upharpoonright A_s}} (x) = W_{n,s} (x) \neq W_{n,t} (x) \). Now, observe that \( A_{t+1} \upharpoonright A_s \) does not change at any later stage at which \( \langle u, v \rangle \) is reserved for \( \mathcal{R}_{e_1,e_2,n} \). Hence, for all such stages \( w \geq t+1 \), \( \Phi_{e_1}^{\Gamma_{f_{A_{w \upharpoonright A_s}}} (w) \neq W_{n,s} (x) \) and \( \mathcal{R}_{e_1,e_2,n} \) does not need attention at stage \( w+1 \).

Lemma 3.23. For every \( n \in \omega \), \( c_f(n) \) is never increased due to numbers \( > n+2 \) entering \( I \).
Proof. Suppose the contrary. Then there exists $n$ such that $c_f(n)$ is increased because of some $k > n + 2$ entering $I$. Let $s + 1$ be the stage at which this happens. Since $c_f(n)$ is increased at stage $s + 1$, $A_n$ is present in $A_{s+1}$. Since $c_f(n)$ is increased due to $k$ entering $I$, $k$ must be associated at stage $s + 1$ with some $R_i$. Hence, $k$ is one of the tickets $t_0, t_1, t_2$ paired with $R_i$ at this point. There are three cases. (k = t_0) This is when $R_i$ is initialized and receives tickets $t_0, t_1, t_2$ (see Figure 3). For $c_f(n)$ to increase, we must have $n = t_0$ or $n = t_1$. $n = t_0$ is not possible because then we would have $k = t_0 = n$ which contradicts $k > n + 2$. $n = t_1$ is also not possible because we would have $k = t_0 = t_1 - 1 = n - 1$ which contradicts $k > n + 2$.

\[ k = t_1 \] This is when $R_i$ receives first attention with tickets $t_0, t_1, t_2$ (see Figure 4). $C_n$ must occur somewhere at the highlighted positions in Figure 4 because this fragment of the structure is copied leading to an increase of $c_f$. Hence, $n = t_0$ or $n = t_1$, or $C_n$ occurs in $\mathcal{T}$. $n \neq t_0$ because otherwise $n = t_0$, $k = t_1 = t_0 + 1 = n + 1$ which contradicts $k > n + 2$. $n$ cannot be $t_1$ because otherwise $n = t_1 = k$ which contradicts $k > n + 2$. Hence, $C_n$ occurs in $\mathcal{T}$. However, this is also not possible for the following reason. We know that $k = t_1$ enters $I$ so this is due to $R_i$ acting when receiving the first attention with tickets $t_0, t_1, t_2$. This means that no higher than $R_i$ requirement $R_j$ (i.e., with $j < i$) has received attention after $R_i$ got associated with tickets $t_0, t_1, t_2$ (up to the current stage)—otherwise $R_i$’s tickets would have been reassigned to numbers different than $t_0, t_1, t_2$. This means that $C_n$ entered the construction after $R_i$ was assigned to $t_0, t_1, t_2$. Therefore, by the construction (i.e. the way we choose and assign tickets to requirements (re)entering the construction), $n$ is a ticket for some lower priority requirement $R_r$ ($l > i$). But when $n$ enters the construction as a ticket of such $R_r$, $n$ is chosen as a fresh number so, in particular, $n > t_1 = k$ which contradicts $k > n + 2$.

\[ k = t_2 \] This is when $R_j$ receives attention for the second time with tickets $t_0, t_1, t_2$ (see Figure 5). $C_n$ occurs somewhere at the highlighted positions in Figure 5 i.e. $n = t_0$ or $n = t_1$, or $C_n$ occurs in $\mathcal{T} + \mathcal{U}$. $n \neq t_0$ because otherwise $n = t_0, k = t_2 = t_0 + 2 = n + 2$ which contradicts $k > n + 2$. $n \neq t_1$ because otherwise $n = t_1, k = t_2 = t_1 + 1 = n + 1$ which contradicts $k > n + 2$. Therefore, $n$ occurs in $\mathcal{T} + \mathcal{U}$. The rest of the argument is similar to the analogical place of the case previous case ($k = t_1$).

Lemma 3.24. $c_f \leq_T I$.

Proof. To compute $c_f(n)$, find $s$ such that $I_i[n + 2] = I[n + 2]$. By Lemma 3.23 and the fact that $c_f(n)$ is increased only due to numbers entering $I$, $c_f(n)$ is not increased at stages $s > s$ (no additional copies of $C_n$ are added to $f_A$). Return the number of copies of $C_n$ in $f_A$.

Lemma 3.25. $f_A \leq_T c_f$.

Proof. To compute in $c_f$ the value $f_A(x)$, wait for the earliest $s$ such that $x \in A_s$. At stage $s - 1$, either some requirement needed attention or some requirement made a reservation.
Assume that some requirement needed attention at stage $s - 1$. Hence, $x$ is one of the fresh numbers that have been added to $\mathcal{A}$ in response to the requirement that received attention at stage $s$. These numbers are represented by the gray fragments in Figures 5, 7. It is crucial to observe that blocks created in this manner always preserve their structure throughout the construction. It means that when fresh numbers are put in such blocks, they can be pushed to the right (with no in-between insertions) but once they are pushed, they land on positions on which we recreate the behavior of $f$ from their original positions. Therefore, $f_A(x) = f_{A_0}(x)$.

Now, assume that some some requirement $\mathcal{R}_i$ made a reservation with $\langle u, v \rangle$ and tickets $t_0, t_1, t_2$ at stage $s - 1$. Hence, $x$ is one of the fresh numbers that have been added to $\mathcal{A}$ at stage $s$ as elements of the cycles $C_{t_0}$ and $C_{t_1}$. These numbers are represented by the gray fragment in Figure 5. Note that, later in the construction, $f$ can assume a different behavior on them in response to first or second attention received by $\mathcal{R}_i$ with reservation $\langle u, v \rangle$, $t_0, t_1, t_2$. However, when we take action in such a situation, $c_f(t_0)$ and $c_f(t_1)$ is increased. Since, by the construction, the behavior of $f$ on $x$ cannot change due to other reason than $\mathcal{R}_i$ receiving attention with reservation $\langle u, v \rangle$, $t_0, t_1, t_2$ (or $x$ can be pushed to the right in other circumstances but then the behavior of $f$ on $x$ is faithfully reproduced). Therefore, to compute $f_A(x)$ it suffices to ask the oracle for $c_f(t_0)$ (or $c_f(t_1)$) and carry out the construction up to a stage $s_0$ such that $\mathcal{A}_{s_0}$ contains $c_f(t_0)$ occurrences of cycle $C_{t_0}$. Then we are sure that $f_A(x) = f_{A_{s_0}}(x)$.

By Lemmas 3.23, 3.24, and 3.25, $0 < T f_A \leq c_f \leq T I < T 0'$. The spectrum of $f$ is not trivial by Proposition 3.1. By Theorem 3.1 DgSp(f) contains all c.e. degrees. Since $f_A$ is of non-c.e. degree, DgSp(f) $\neq$ the c.e. degrees. To show that DgSp(f) $\neq$ the $\Delta_2$ degrees, we need

**Theorem 3.26** (Cooper, Lempp and Watson, [3]). Given c.e. sets $U < T V$ there is a proper d.c.e. set $C$ of properly d.c.e. degree such that $U < T C < T V$.

Assume, for a contradiction, that DgSp(f) consists of the $\Delta_2$ degrees. By Theorem 3.26 $DgSp(f) \cap \{\deg(A) : c_f \leq T A \leq T 0'\}$ contains a properly d.c.e. degree. However, by Proposition 3.19 $DgSp(f) \cap \{\deg(A) : c_f \leq T A \leq T 0'\}$ contains only c.e. degrees. This is a contradiction, so the degree spectrum of $f$ is different then the $\Delta_2$ degrees. This completes the proof.

4. Conclusions and open questions

We have investigated the problem of intrinsic complexity of computable relations on $(\omega, <)$, as measured by their degree spectra, in the restricted setting of graphs of unary total computable functions. It has been known that possible candidates for intrinsic complexities of such functions include three sets consisting of precisely: the computable degree, all c.e. degrees, and all $\Delta_2$ degrees. Imposing certain structural contraints on such functions has led us to the notions of block functions (Definition 3.5) and a broader class of quasi-block functions (Definition 3.12). Non-quasi-block functions have intrinsic complexity equal to the c.e. degrees (Theorem 3.15) which redirects all focus to quasi-block functions. We have obtained several results on this class, most prominently the one on block-functions with finitely many types (Theorem 3.10) showing that their intrinsic complexity is either trivial of equal to the $\Delta_2$ degrees. However, the most surprising result is that on an unusual degree spectrum (Theorem 3.20) which proves the existence of a block function having
intrinsic complexity different from the already known three candidates. To the best of our knowledge, this theorem answers Question 6.2 from [10] formulated by Wright who asked whether there are relations on $(\omega, <)$ with other degree spectra (than the three known candidates). Harrison-Trainor obtained a related result though for a different relation. However, for his relation it is not known whether its spectrum is intermediate (see Section 1 for details, as well as [8]).

A few questions arise immediately. Although we have been able to obtain some results on computable block functions with infinitely many types, the spectrum problem for such functions remains largely unsolved. Even for the function constructed in Theorem 3.20 the exact contents of its spectrum are unknown. We finish the paper with an open question: are there infinitely many spectra of unary total computable functions on $(\omega, <)$?

References

[1] Chris J Ash and Julia Knight. *Computable structures and the hyperarithmetical hierarchy.* Elsevier, 2000.
[2] Jennifer Chubb, Andrey Frolov, and Valentina Harizanov. Degree spectra of the successor relation of computable linear orderings. *Archive for Mathematical Logic*, 48(1):7–13, 2009.
[3] S. Barry Cooper, Steffen Lempp, and Philip Watson. Weak density and cupping in the d-r.e. degrees. *Israel Journal of Mathematics*, 67(2):137–152, 1989.
[4] Rod Downey, Bakhadyr Khoussainov, Joseph S Miller, and Liang Yu. Degree spectra of unary relations on $(\omega, <)$. In *Logic, Methodology and Philosophy of Science: Proceedings of the Thirteenth International Congress*, pages 35–55. College Publications, 2009. Publisher: Citeseer.
[5] R.L. Epstein. *Degrees of Unsolvability: Structure and Theory.* Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1979.
[6] Ekaterina B. Fokina, Valentina Harizanov, and Alexander Melnikov. Computable model theory. In R. Downey, editor, *Turing’s legacy: Developments from Turing’s ideas in logic*, volume 42 of *Lecture Notes in Logic*, pages 124–194. Cambridge University Press, Cambridge, 2014.
[7] V.S. Harizanov. *Degree spectrum of a recursive relation on a recursive structure.* PhD thesis, University of Wisconsin-Madison, 1987.
[8] Matthew Harrison-Trainor. Degree spectra of relations on a cone. *Memoirs of the American Mathematical Society*, 253(1208):1–120, 2018.
[9] Denis R. Hirschfeldt. Degree spectra of relations on computable structures. *Bulletin of Symbolic Logic*, 6(2):197–212, 2000.
[10] Carolyn Alexis Knoll. Degree spectra of unary relations on $\omega$ and $\zeta$. Master’s thesis, University of Waterloo, 2009.
[11] Antonio Montalbán. *Computable structure theory: Within the arithmetic.* Cambridge University Press, 2021.
[12] Michael Moses. Relations Intrinsically Recursive in Linear Orders. *Mathematical Logic Quarterly*, 32(25-30):467–472, 1986.
[13] Linda Jean Richter. Degrees of structures. *Journal of Symbolic Logic*, 46(4):723–731, 1981. Publisher: Cambridge University Press.
[14] Stewart Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23(1):14–20, January 1982.
[15] Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Springer-Verlag New York, Inc., New York, NY, USA, 1987.
[16] Matthew Wright. Degrees of relations on ordinals. *Computability*, 7(4):349–365, 2018. Publisher: IOS Press.
