Degree Spectra, and Relative Acceptability of Notations

Nikolay Bazhenov
Sobolev Institute of Mathematics, Novosibirsk, Russia

Dariusz Kalociński
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract

We investigate the interplay between the degree spectrum of a computable relation R on the computable structure $(\omega, <)$, i.e., natural numbers with the standard order, and the computability of the successor, relativized to that relation, in all computable copies of the structure – the property we dub successor's recoverability. In computable structure theory, this property is used to show that of all possible Turing degrees that could belong to the spectrum of R (namely, of all Δ_2 degrees), in fact only the computably enumerable degrees are contained in the spectrum. Interestingly, successor's recoverability (in the unrelativized form) appears also in philosophy of computing where it is used to distinguish between acceptable and deviant encodings (notations) of natural numbers. Since Shapiro's notations are rarely seen through the lens of computable structure theory, we first lay the elementary conceptual groundwork to understand notations in terms of computable structures and show how results pertinent to the former can fundamentally inform our understanding of the latter. Secondly, we prove a new result which shows that for a large class of computable relations (satisfying a certain effectiveness condition), having all c.e. degrees as a spectrum implies that the successor is recoverable from the relation. The recoverability of successor may be otherwise seen as relativized acceptability of every notation for the structure. We end with remarks about connections of our result to relative computable categoricity and to a similar direction pursued by Matthew Harrison-Trainor in [18], and with an open question.

2012 ACM Subject Classification Theory of computation → Computability

Keywords and phrases Computable structure theory, Degree spectra, ω-type order, C.e. degrees, Δ_2 degrees, Acceptable notation, Successor, Learnability

Digital Object Identifier 10.4230/LIPIcs.CSL.2023.11

Related Version *Previous Version*: `https://arxiv.org/abs/2205.00791`

Funding *Nikolay Bazhenov*: The work of Bazhenov is supported by the Mathematical Center in Akademgorodok under the agreement No. 075-15-2022-281 with the Ministry of Science and Higher Education of the Russian Federation.
Dariusz Kalociński: The work of Kalociński is supported by the National Science Centre Poland under the agreement No. 2018/31/B/HS1/04018.

Acknowledgements We would like to thank Matthew Harrison-Trainor and Michał Wrocławski for helpful discussions and anonymous reviewers for their comments.

1 Introduction

Shapiro made a point that computations are performed on syntactic objects, such as strings of symbols, rather than on numbers themselves [34]. Some models of computation are directly based on this premise [38, 25] but some are not [36, 6]. When showing equivalence between these models, one uses some form of notation for natural numbers, e.g., the unary notation. However, as Shapiro observed, not every notation is appropriate for showing the desired equivalence. Therefore, he asked when a notation is appropriate in the above sense, or, in his words, when it should be deemed acceptable. He showed that a notation is acceptable if and only if the successor function is computable in it.

Shapiro's notations became influential in philosophy of computing (see, e.g., [30, 8, 29, 40, 35]). Meanwhile, we have seen rapid development of a separate, though similar in spirit, research program known as computable structure theory which explores the relationship between computability and algebraic structures [1, 26]. Interestingly, Shapiro's framework can be fully rephrased in terms of computable structures under the following slogan: a notation for a given computable structure corresponds to a computable (isomorphic) copy of the structure. To our best knowledge, this simple fact, and its consequences, has been consistently overlooked (though, see [20]). Bringing to light the connection between Shapiro's notations and computable structure theory is the first, methodological, contribution of the paper, contained in Section 2.

We start Section 2 by providing an appropriate translation between Shapiro's notations and computable structures. Our exposition, although restricted to the structure $(\omega, <)$, i.e., natural numbers with the standard order, generalizes to other computable structures. As a proof of concept, we recast some of the notions, such as notation's acceptability, and results of Shapiro (later, in Section 2.1), using tools developed in computable structure theory. After that, we recall an important notion of degree spectrum. The notion of *degree spectrum* is a classical one [31, 32, 17]. The degree spectrum of a computable relation R on $(\omega, <)$ is the set of all Turing degrees of the images of R in all computable isomorphic copies of $(\omega, <)$. We demonstrate the applicability of this notion to notations by rejecting a certain claim, stemming from a philosophical paper by Benacerraf, according to which, in every notation for $(\omega, <)$, the successor is computable [3]. This claim is equivalent to saying that the degree spectrum of the successor on $(\omega, <)$ is trivial, i.e., consists of only the computable degree. The claim is rejected based on a known result according to which this degree spectrum is actually equal to the set of all (and only) computably enumerable degrees [5]. We then set the ground for Section 3, containing the main technical result. Section 2 could appeal to logically inclined philosophers, interested in connections with the advanced framework of computable structure theory, but also to computable structure theorists who could find an independent source of motivation for their own research (otherwise, they could safely skim through Section 2).

The second and main contribution of the paper is a new result, contained in Section 3. The result explores the logical relationship between the following two statements about a computable relation R: (I) R's degree spectrum on $(\omega, <)$ is equal to the set of all (and only) computably enumerable (c.e.) degrees, and (II) (the image $Succ_{\mathcal{A}}$ of) the successor function $Succ$ is computable relative to (the image $R_{\mathcal{A}}$ of) R, in every computable copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ of $(\omega, <)$. If R satisfies (II), we say that the successor is recoverable from R on $(\omega, <)$.

The *recoverability of the successor* appears naturally in the investigation of degree spectra on $(\omega, <)$. Every degree spectrum of a computable relation on $(\omega, <)$ is contained in the set of all Δ_2 degrees (recall that a Δ_2 degree is characterized by containing a subset of ω that can be recursively approximated; such sets are also called algorithmically learnable or limiting recursive [28, 15]). It is known that the recoverability of the successor from R on $(\omega, <)$ fixes R's spectrum to that of all c.e. degrees (see, e.g., [18, 2]) which form a proper subset of all Δ_2 degrees [7]. This means that the implication (II) $\Rightarrow$ (I) holds in general. On the other hand, recoverability of the successor plays an important role in isolating the class of acceptable notations [34]: plain acceptability of a computable copy $\mathcal{A}$ of $(\omega, <)$ means that $Succ_{\mathcal{A}}$ is computable. After relativization, we obtain a more general notion: a computable copy $\mathcal{A}$

of $(\omega, <)$ is acceptable relative to R if $Succ_{\mathcal{A}}$ is computable in $R_{\mathcal{A}}$. Now, recoverability of the successor from R on $(\omega, <)$ means that every computable copy $\mathcal{A}$ of $(\omega, <)$ is acceptable relative to R.

The question tackled in Section 3 deals with the implication (I) $\Rightarrow$ (II). For any known example of a computable relation R, the implication holds. We ask whether it holds in general, namely whether for any computable relation R, if R's degree spectrum on $(\omega, <)$ is equal to all c.e. degrees, then the successor is recoverable from R on $(\omega, <)$. Following [2], we attack this problem for a restricted but nevertheless inclusive class of computable relations, namely the graphs of unary total computable functions. We expand on techniques developed in [2], in particular by reusing the concept of block functions. Our result – which answers the aforementioned question in the affirmative – is proved for computable block functions that satisfy certain intuitive effectiveness condition (Theorem 21).

In Section 4 we remark about related work in computable structure theory. In particular, we comment on key differences between our theorem and a theorem by Matthew Harrison-Trainor, where the technique of recovering the successor is used [18]. We also outline a connection between successor's recoverability and the concept of relative computable categoricity.

The last section concludes the paper. We ask whether the effectiveness condition that we use in proving Theorem 21 can be dropped.

2 Notations and Computable Structures

Originally, Shapiro considered notations for natural numbers (henceforth, ω) with no additional structure. A *notation* for ω is any bijection $\sigma\colon S \to \omega$, where S is a fixed countably infinite set of strings over a finite alphabet.[1] The idea is that $\alpha \in S$ is a numeral denoting $\sigma(\alpha)$ and we think of computations as being performed only on numerals. Computability of an n-ary relation R *on natural numbers* in σ is equated, by definition, with the computability of the σ-preimage of R, $\{(\sigma^{-1}(a_1), \sigma^{-1}(a_2), \dots, \sigma^{-1}(a_n)) : (a_1, a_2, \dots, a_n) \in R\}$, which is a relation *on numerals*.

▶ **Example 1.** Let T be the set of all nonempty words over the alphabet $\{a\}$ and let $\tau(a^n) = n$, where a^n is the word consisting of n consecutive occurrences of a. Clearly, τ is a notation for ω. The τ-preimage of $<$ is equal to $\{(x, y) \in T^2 : \tau(x) < \tau(y)\}$. This set is computable because comparing word lengths is computable, and for every $x, y \in T$, $\tau(x) < \tau(y) \Leftrightarrow |x| < |y|$. Therefore, the relation $<$ (on natural numbers) is computable in τ.

▶ **Example 2.** Let T be as in Example 1 and let $X \subset \omega$ be non-computable. Consider the sequence $a, aa, aaa, \dots$ and obtain a new one in the following way: for each $k \geq 0$, swap the positions of a^{2k+1} and a^{2k+2} if $k \in X$. For example, if $0, 2 \in X$ and $1 \notin X$, the new sequence starts with $aa, a, aaa, aaaa, aaaaaa, aaaaa, \dots$. Let ρ be the inverse of the sequence obtained in this way. ρ is a notation for ω. $<$ is not computable in ρ because if it were computable then X would be, as $k \in X$ iff (a^{2k+2}, a^{2k+1}) belongs to the ρ-preimage of $<$.

Without loss of generality, we may assume that a notation for ω is any bijection from ω to ω. This follows from a simple but important fact that for every infinite computable set S of words over a finite alphabet there exists a computable enumeration of S without

[1] Shapiro's notation should not be confused with Kleene's notation for ordinals. Note, however, that an acceptable notation (to be defined) can indeed be viewed as a notation for the ordinal ω.

repetitions, i.e., a total computable injection $s\colon \omega \to S$ such that $s(\omega) = S$. In other words, s gives us a computable nonrepetitive sequence $s_0, s_1, s_2, \ldots$ which consists of all (and only) elements of S. s may be seen as an encoding of S by ω. We take such identification for granted.

Shapiro considered only notations for plain ω, i.e., for the set of natural numbers without any additional structure (in Section 2.1, we reproduce some of his results in the setting of computable structure theory). However, one can easily extend his notion to cover additional structure. We shall focus on the additional structure in the form of the simplest ordering possible – the standard order on natural numbers, which we denote by $<$. Hence, the structure under investigation is $(\omega, <)$. By a notation for $(\omega, <)$ we shall mean any notation σ for ω in which $<$ is computable.

Now, let us turn our attention to computable structures. A structure $(A, R_1, \ldots, R_n)$ is said to be *computable* if its universe A and each relation R_i is computable. Without loss of generality, we can assume that $A = \omega$, the reason being similar to the one already discussed for S. Clearly, $(\omega, <)$ is a computable structure.

To pinpoint the connection between notations and computable structures, consider the following definition.

▶ **Definition 3.** *$(\omega, \prec)$ is a computable copy of $(\omega, <)$ if $\prec$ is a computable ordering on ω and structures $(\omega, <)$ and $(\omega, \prec)$ are isomorphic.*

Now, we can make the following observations. Let $\sigma\colon \omega \to \omega$ be a notation for $(\omega, <)$ and let $\prec$ be the σ-preimage of $<$, i.e., $\prec := \{(\sigma^{-1}(x), \sigma^{-1}(y)) : x < y\}$. By the definition of notation, $\prec$ is computable. Moreover, by the definition of $\prec$, the structures $(\omega, <)$ and $(\omega, \prec)$ are isomorphic. Therefore, by Definition 3, $(\omega, \prec)$ is a computable copy of $(\omega, <)$.

The next observation goes in the other direction. Let $(\omega, \prec)$ be a computable copy of $(\omega, <)$. Let $h\colon (\omega, <) \cong (\omega, \prec)$ be an isomorphism between the two structures and let $\sigma = h^{-1}$. Obviously, σ is a notation for ω. Also, the σ-preimage of $<$ is equal to $\prec$ and, by Definition 3, $\prec$ is computable. Therefore, σ is a notation for $(\omega, <)$.

Based on the above two observations, we can posit that a notation for $(\omega, <)$ is any isomorphism that maps a computable structure $(\omega, \prec)$ to $(\omega, <)$. Therefore, instead of notations for $(\omega, <)$ we may equivalently speak about computable copies of $(\omega, <)$. This generalizes straightforwardly to arbitrary computable structures.

Let us recall one of the concepts introduced by Shapiro, namely acceptability of notation, and see what does it mean in terms of computable structures. A notation for ω is said to be *acceptable* if the successor function (henceforth, *Succ*) is computable in it (this implies that all recursive functions are); otherwise the notation is called deviant. Acceptability of notation for plain ω can be extended, in an obvious way, to notations for ω with additional structure, in particular to the structure $(\omega, <)$.

What does the desideratum of notation's acceptability mean from the perspective of computable structures? Before we answer this, consider the following convention which is commonplace when referring to isomorphic copies of a given structure.

▶ **Definition 4.** *Let R be a relation on $(\omega, <)$, i.e., $R \subseteq \omega^k$, for some $k \in \omega$, and let $\mathcal{A}$ be a computable copy of $(\omega, <)$. If φ is an isomorphism from $(\omega, <)$ to $\mathcal{A}$, we write $R_{\mathcal{A}}$ for the image of R under φ.*

Now, in terms of computable structures, the desideratum of acceptability of a computable copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ – which uniquely identifies a notation for $(\omega, <)$ – says that $Succ_{\mathcal{A}}$ should be computable.

Shapiro showed, among others, that not every notation for ω is acceptable. The following question arises immediately: is $Succ_{\mathcal{A}}$ computable, if $\mathcal{A}$ is any computable copy $\mathcal{A}$ of $(\omega, <)$? In other words, is every computable copy of $(\omega, <)$ acceptable? In an influential philosophical paper, Benacerraf hinted in passing at the affirmative (cf. p. 276 in [3] and, also, [4]). Essentially, his claim was that, in any given notation, the computability of the successor is equivalent to the computability of the ordering (we take the liberty to identify Benacerraf's intuitive concept of notation with the formal one).

As we will see, Benacerraf's claim is false in view of Proposition 6 below. The proposition uses an important notion of degree spectrum, originating from Richter [31, 32] and Harizanov [17] (see, also, [19, 14]). The notion of degree spectrum – here defined for $(\omega, <)$ but, in general, applicable to any computable structure – is based on the concept of Turing degrees. As a brief reminder, Turing degrees are equivalence classes of the relation $\equiv_T$ on subsets of ω, defined by $A \equiv_T B \Leftrightarrow (A \leq_T B \wedge B \leq_T A)$, where $X \leq_T Y$ means that the characteristic function of X can be computed by a program which can ask questions of the form "$n \in Y$?" for different n's and use the answers to make decisions while the program is running. For more information, see, e.g., [9, 33, 37].

▶ **Definition 5** (degree spectrum of a relation)**.** *The degree spectrum of a computable relation R on $(\omega, <)$, in symbols $DgSp_{(\omega,<)}(R)$, is the set of Turing degrees of $R_{\mathcal{A}}$ over all computable copies $\mathcal{A}$ of $(\omega, <)$.*

We write $DgSp(R)$ as we do not consider degree spectra on structures other than $(\omega, <)$.

As we can see, the notion of degree spectrum encompasses all possible complexities of a relation (formalized as Turing degrees) across all notations for the underlying structure.

Before formulating Proposition 6, let us remind that a Turing degree is computably enumerable (c.e.) if it contains a computably enumerable set, i.e., a set which, if nonempty, can be enumerated by an algorithm. Since there exist noncomputable c.e. sets (e.g., the halting problem), there are c.e. degrees which are different than $\mathbf{0}$ (the degree of computable sets).

▶ **Proposition 6** (see, e.g., Example 1.3 in [5])**.** *The degree spectrum of the successor on $(\omega, <)$ consist of all (and only) c.e. Turing degrees.*

Clearly, Benacerraf's claim must be false, for otherwise every computable copy of $(\omega, <)$ would be acceptable (i.e., the successor would be computable in it) and we would have $DgSp(Succ) = \{\mathbf{0}\}$ which contradicts Proposition 6.

The degree spectrum of the successor on $(\omega, <)$, i.e., the set consisting of all c.e. degrees, is just one example of degree spectrum on $(\omega, <)$, but other examples exist and it is still not known what are all possibilities. One kind of degree spectrum is the trivial one, i.e., $\{\mathbf{0}\}$. Relations having this spectrum are called *intrinsically computable* and were characterized by Moses [27]. There are also other kinds of spectra, including the set of all Δ_2 degrees [10, 39, 18] and other spectra, discovered quite recently [2].

Anyway, the degree spectrum of the successor is, in a sense, special. Essentially, $DgSp(Succ) \subseteq DgSp(R)$ for every computable R which is not intrinsically computable (see, Theorem 4.7 in [39]). To show that $DgSp(R) = DgSp(Succ)$, one typically uses a technique known as *recovering the successor* [2]. The idea is to prove that $R_{\mathcal{A}} \geq_T Succ_{\mathcal{A}}$ holds for every computable copy $\mathcal{A}$ of $(\omega, <)$; this is sufficient because $Succ_{\mathcal{A}} \geq_T R_{\mathcal{A}}$ always, i.e., for every computable copy $\mathcal{A}$ of $(\omega, <)$.[2] In a sense, such a proof shows how the successor (inside $\mathcal{A}$) can be recovered from the relation R (again, inside $\mathcal{A}$). If this can be done, we say that the successor is recoverable from R on $(\omega, <)$.

In the second part of the paper, we are interested whether one can go in the other direction. Specifically, we ask whether $DgSp(R) = DgSp(Succ)$ implies that the successor is recoverable from R on $(\omega, <)$. In other words, given the following properties of a computable relation R:

$$\text{the degree spectrum of } R \text{ on } (\omega, <) \text{ is equal to all c.e. degrees, and} \tag{I}$$

$$\text{for every computable copy } \mathcal{A} \text{ of } (\omega, <),\ R_{\mathcal{A}} \geq_T Succ_{\mathcal{A}}, \tag{II}$$

we ask whether (I) implies (II) (the other direction follows from one of the paragraphs above). Note that (II) can be seen as a relativized variant of acceptability of notation. We may posit that a computable copy (notation) $\mathcal{A} = (\omega, <_{\mathcal{A}})$ is acceptable relative to R if $R_{\mathcal{A}}$ computes $Succ_{\mathcal{A}}$. Thus, the property in question encompasses computable relations such that every computable copy $(\omega, <_{\mathcal{A}})$ is acceptable relative to them.

Although we are not able to prove this implication in full generality, we show that it holds for a wide subclass of all total computable functions (seen as computable binary relations). For certain types of functions, the implication already follows from earlier results which will be discussed at the beginning of Section 3. Our extension concerns the class of block functions, isolated in this context by Bazhenov et al. [2], and satisfying an additional effectiveness condition.

2.1 Shapiro's Theorems Re-proved

In this subsection, some of the results on Shapiro's notations, here Theorems 11 and 12, are re-proved using tools from computable structure theory. This provides concrete evidence of the intrinsic connection between the two frameworks. The proofs are based on early results from computable structure theory, some of which even predate Shapiro's paper.

For Theorem 11, we need a classical notion of the degree spectrum of a structure. It is different from the notion of the degree spectrum of a relation (Definition 5) in that it focuses on the complexity of the copies of the structure itself. Let L be a computable language, i.e., L consists of a computable set of constants, function and relation symbols, and the map assigning arity to each symbol is also computable.

▶ **Definition 7** (degree spectrum of a structure). *For a countably infinite L-structure $\mathcal{A}$, its* degree spectrum *is the set of all Turing degrees* $\mathbf{d}$ *such that there is an L-structure $\mathcal{B}$ with the following properties:*

(i) *$\mathcal{B}$ is isomorphic to $\mathcal{A}$,*

(ii) *the domain of $\mathcal{B}$ equals ω,*

(iii) *the Turing degree of $D(\mathcal{B})$ – the atomic diagram of $\mathcal{B}$ – is equal to* $\mathbf{d}$ *(here a formula ϕ from the atomic diagram is identified with its Gödel number $\ulcorner \phi \urcorner$).*

▶ **Definition 8.** *A structure $\mathcal{A}$ is called* automorphically trivial *if there is a finite set $X \subseteq$* $\mathrm{dom}(\mathcal{A})$ *such that every permutation of* $\mathrm{dom}(\mathcal{A})$ *that fixes X is an automorphism of $\mathcal{A}$.*

[2] To compute in $Succ_{\mathcal{A}}$ whether $(a_1, \dots a_n) \in R_{\mathcal{A}}$, use $Succ_{\mathcal{A}}$ and one parameter, e.g. the $<_{\mathcal{A}}$-minimal element, to identify the positions $i_1, \dots, i_n$ of $a_1, \dots, a_n$ in the ordering $<_{\mathcal{A}}$, and return $R(i_1, \dots, i_n)$.

▶ **Example 9.** Let E be an equivalence relation on ω with finitely many finite equivalence classes and just one infinite equivalence class. The structure (ω, E) is automorphically trivial, because every permutation of ω that fixes $X = \{x \in \omega : \text{the } E\text{-equivalence class of } x \text{ is finite}\}$ is an automorphism of (ω, E). Moreover, the degree spectrum of (ω, E) is $\{\mathbf{0}\}$. For let $\mathcal{B} = (\omega, E_{\mathcal{B}})$ be isomorphic to (ω, E) via h. $\ulcorner a = b \urcorner \in D(\mathcal{B})$ iff $a = b$. $\ulcorner E(a, b) \urcorner \in D(\mathcal{B})$ iff $a, b \notin h^{-1}(X)$ or $a, b \in h^{-1}(X) \wedge E(h(a), h(b))$. The latter is computable by the finiteness of $h^{-1}(X)$. Hence, $D(\mathcal{B})$ is computable, given that the underlying Gödel numbering of formulas is effective.

▶ **Theorem 10** (Theorem 4 of [21]). *If a countably infinite structure $\mathcal{A}$ is not automorphically trivial, then its degree spectrum is closed upwards in Turing degrees.*

▶ **Theorem 11** (T1 and C1 in [34]). *A function $f \colon \omega \to \omega$ is computable in every notation for ω if and only if either f is almost constant (i.e., there is c such that $f(x) = c$ holds for all but finitely many x), or f is almost identity (i.e., $f(x) = x$ for all but finitely many x). A relation $R \subseteq \omega$ is computable in every notation for ω if and only if either R is finite, or R is cofinite.*

Proof. We only consider the nontrivial direction ($\Rightarrow$). Suppose that f is computable in every notation for ω. Fix some standard notation σ_0 – e.g., the one induced by decimal numerals. Our function f must be computable in σ_0. Consider the language $L = \{F\}$, where F is a unary function symbol. We define an L-structure $\mathcal{S}_f$ as follows: the domain of $\mathcal{S}_f$ equals ω, and the function symbol F is interpreted as our function f. Since $f(x)$ is computable in σ_0, it is easy to show that the structure $\mathcal{S}_f$ is computable.

Now, if $f(x)$ is neither almost constant nor almost identity, then one can show that the corresponding structure $\mathcal{S}_f$ is not automorphically trivial. Thus, by Theorem 10, one can obtain an isomorphic copy $\mathcal{A}$ of $\mathcal{S}_f$ such that the atomic diagram $D(\mathcal{A})$ of the structure $\mathcal{A}$ is Turing equivalent to, say, the halting problem. Now we define a notation $\sigma \colon S \to \omega$:

(i) S equals (the decimal representation of) ω, and

(ii) σ is an arbitrary isomorphism from $\mathcal{A}$ onto $\mathcal{S}_f$.

Since $D(\mathcal{A})$ is not computable, one can show that the function $f(x)$ is not computable in σ. ◀

Finally, let us consider the following result by Shapiro, in which notation's acceptability (a) gets an intuitive equivalent (b).

▶ **Theorem 12** ([34]). *For any notation σ for ω, the following are equivalent:*

(a) *For any function f, f is computable in σ if and only if f is computable in the standard notation (e.g., the stroke notation).*

(b) *The successor function is computable in σ.*

To establish the above result, we need the notion of computable categoricity which goes back to the papers by Mal'tsev [23, 24].

▶ **Definition 13.** *A computable L-structure $\mathcal{A}$ is* computably categorical *if for every computable L-structure $\mathcal{B}$, which is isomorphic to $\mathcal{A}$, there is a computable isomorphism g from $\mathcal{B}$ onto $\mathcal{A}$ (i.e., g is an isomorphism which is also a computable function).*

No matter which computable copy of a computably categorical structure we take, it will have the same computability-theoretic properties. A simple example of a computably categorical structure will be considered in the proof of Theorem 12.

▶ **Example 14.** $(\omega, <)$ is not computably categorical. By Proposition 6, there exists a computable structure $(\omega, <_{\mathcal{A}})$ isomorphic to $(\omega, <)$ such that $Succ_{\mathcal{A}}$ is noncomputable. If $(\omega, <)$ were computably categorical, we would have a computable isomorphism g from $(\omega, <_{\mathcal{A}})$ to $(\omega, <)$. But then $Succ_{\mathcal{A}}$ would be computable because $Succ_{\mathcal{A}} = g^{-1} \circ Succ \circ g$.

Recall that a structure $\mathcal{A}$ is finitely generated if there exists a finite set $G \subset dom(\mathcal{A})$ such that the set generated by G in $\mathcal{A}$ – i.e., the least set including G together with all the distinguished elements of $\mathcal{A}$, and closed under the operations of $\mathcal{A}$ – is equal to $dom(\mathcal{A})$.

▶ **Theorem 15** (Theorem 4.1.2 in [23]). *Every finitely generated, computable algebraic structure is computably categorical.*

Proof of Theorem 12. We sketch the proof for the direction (b)⇒(a). Since the computable structure $(\omega, Succ)$ is one-generated, $(\omega, Succ)$ is computably categorical by Theorem 15. Now consider a notation $\sigma \colon S \to \omega$ such that the successor function is computable in σ. Let $Succ^{\sigma}$ be the image of the successor under σ^{-1}. We define a computable L-structure $\mathcal{T}_{S,\sigma}$ as follows.

- The domain of $\mathcal{T}_{S,\sigma}$ equals ω.
- Fix a computable bijection ψ from ω onto S. The unary function symbol F is interpreted as follows: for $k \in \omega$,

$$F(k) = \psi^{-1}(Succ^{\sigma}(\psi(k))).$$

It is not hard to show that the structure $\mathcal{T}_{S,\sigma}$ is a computable isomorphic copy of $(\omega, Succ)$. Then the computable categoricity of $(\omega, Succ)$ allows to choose (the unique) computable isomorphism g from $\mathcal{T}_{S,\sigma}$ onto $(\omega, Succ)$.

After that, one can use the computability of g to easily show that the notation σ satisfies the conditions from the item (a). ◀

3 Spectrum of C.E. Degrees and Recovering the Successor

Following Bazhenov et al. [2], we restrict our attention to computable binary relations R of general interest – graphs of unary total computable functions. The results of Moses [27] imply that the degree spectrum of such an f is trivial if and only if f is almost constant or almost identity (see, also, the associated version of [2]). Clearly, the equivalence (I) ⇔ (II) holds for such functions as they do not satisfy neither (I) nor (II). Bazhenov et al. [2] isolated the class of quasi-block functions (which include almost constant and almost identity functions) and showed that all computable functions f outside this class satisfy (I) and (II) (Theorem 18 in [2]). We recall that f is a quasi-block function if there are arbitrarily long initial segments of $(\omega, <)$ which are closed under f. Further information about quasi-block functions, pertinent also to the problem at hand, can be found in [2].

Among quasi-block functions there is a narrower class of block functions which we define below. Given a linear ordering $\mathcal{L} = (L, \prec)$, by an interval in $\mathcal{L}$ we mean any set $I \subseteq L$ such that for every $x, y, z \in L$, if $x, z \in I$ and $x \prec y \prec z \in I$, then $y \in I$. For $a, b \in L$, $[a; b] = \{x \in L : a \preceq x \preceq b\}$, where $\preceq$ is the non-strict ordering corresponding to $\prec$.

▶ **Definition 16.** *Let $f \colon \omega \to \omega$ be a total function. An interval I in $(\omega, <)$ is* f-closed *if for all $x \in I$, $f(x) \in I$ and $f^{-1}(\{x\}) \subseteq I$. For a finite non-empty interval $I \subset \omega$, the structure $(I, <, f \restriction I)$ is an* f-block *if it has the following properties:*

(i) *I is an f-closed interval and it cannot be written as a disjoint union of at least two f-closed intervals;*

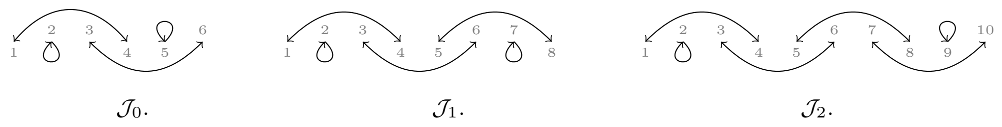

Figure 1 Structures $\mathcal{J}_n = ([1; 6 + 2n], <, f)$, for $n = 0, 1, 2$, where f is the involution such that $f(k) = k$ iff $k = 2$ or $k = 6 + 2n - 1$, and $f(k) = k + 3$ for odd numbers $\leq 6 + 2n - 3$.

(ii) *$\{x \in \omega : x < \min(I)\}$ is f-closed.*
The function f is a block function *if for every $a \in \omega$, there is an f-block containing a. If $(I, <, f \upharpoonright I)$ is an f-block, we refer to its isomorphism type as an* f-type *(or a* type*).*

A block function on $(\omega, <)$ can be visualized as a sequence of blocks, arranged one after another according to $<$. We develop this intution in the examples below.

▶ **Example 17** (borrowed from [2])**.** Consider finite structures $\mathcal{J}_n$ from Figure 1. Let g be the involution such that $(\omega, <, g) \cong \mathcal{J}_0 + \mathcal{J}_1 + \mathcal{J}_2 + \ldots$ Clearly, g is a block function. Condition (ii) in Definition 16 ensures that every element is contained in a unique g-block. Without (ii), each loop, i.e., a substructure of the form $(\{x\}, < \upharpoonright \{x\}, g \upharpoonright \{x\})$ with $g(x) = x$, would be a g-block itself.

▶ **Example 18.** Let f be a function such that each of its blocks is isomorphic to a structure $\mathcal{C}_k = ([0; k], <, f_k)$, for some $k \in \omega$, where $<$ orders $[0; k]$ in the standard way, $f_k(x) = x + 1$, for $0 \leq x < k$, and $f_k(k) = 0$. It is natural to view $\mathcal{C}_k$ as a *cycle*. This illustrative example is worth remembering because we will use it in the proof of the main result, Theorem 21.

In this paper, we consider functions f that satisfy the effectiveness condition $(\star)$ defined below. The problem whether $(\star)$ can be dropped without affecting Theorem 21 is left open (see, Section 5).

For a set X, by $\operatorname{card}(X)$ we denote the cardinality of X.

▶ **Definition 19.** *Let $f : \omega \to \omega$. We define $cp_f : \omega \to \omega \cup \{\infty\}$ as follows:*

$$cp_f(x) = \operatorname{card}(f^{-1}(\{x\})), \text{ for all } x \in \omega.$$

Observe that for the involution g from Example 17 we have $cp_g = 1$. The same holds for functions considered in Example 18.

The effectiveness condition is the following statement about f:

$$cp_f \text{ is computable.} \qquad (\star)$$

Clearly, functions from Examples 17 and 18 satisfy $(\star)$. More generally, any injective computable block function satisfies it because such a function must be composed of finite cycles and thus the preimage of each single element has cardinality 1. Another less trivial example could be a computable block function having only finitely many pairwise nonembeddable types. In general, however, the behavior of cardinalities of the preimages can be quite complicated. For example, it is not hard to produce an example of a computable block function f such that cp_f is noncomputable (see, also, Theorem 27).

Let f be a block function and let $(\omega, <, f) = \mathcal{F}_1 + \mathcal{F}_2 + \ldots$, where each $\mathcal{F}_n$ is an f-block. By an initial segment of $(\omega, <, f)$ we mean any structure $\Sigma_{i=1}^{k} \mathcal{F}_k$, for $k \in \omega$. We say that $(\mathcal{B}_j)_{j \in \omega}$ is a sequence of growing initial segments of $(\omega, \prec, f)$ if each $\mathcal{B}_j$ is an initial segment of $(\omega, \prec, f)$ and for all j we have $dom(\mathcal{B}_j) \subset dom(\mathcal{B}_{j+1})$. Given structures $\mathcal{A}$ and $\mathcal{B}$ over the same relational signature (treat each function as a relation), we say that $\mathcal{A}$ embeds precisely once in $\mathcal{B}$ if there is exactly one embedding from $\mathcal{A}$ to $\mathcal{B}$. Recall that $h\colon dom(\mathcal{A}) \to dom(\mathcal{B})$ is an embedding from $\mathcal{A}$ to $\mathcal{B}$ if h is an injection and for every relational symbol R from the signature, for every tuple $\overline{a} \in dom(\mathcal{A})$ of length equal to the arity of R, $\mathcal{A} \models R(\overline{a})$ if and only if $\mathcal{B} \models R(h(\overline{a}))$.

▶ **Lemma 20.** *Let f be a unary total computable block function. If there exists an infinite computable sequence $(\mathcal{B}_j)_{j\in\omega}$ of growing initial segments of $(\omega, <, f)$ such that each $\mathcal{B}_j$ embeds in $(\omega, <, f)$ precisely once, then the successor is recoverable from f on $(\omega, <)$.*

Proof. To compute $Succ_{\mathcal{A}}(x)$ for a given computable copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ of $(\omega, <)$, look for bigger and bigger substructures $\mathcal{A}_j$ of $(\omega, <_{\mathcal{A}}, f_{\mathcal{A}})$ that look precisely as $\mathcal{B}_j$. We do this by enumerating ω one by one, arranging enumerated elements according to $<_{\mathcal{A}}$ and asking $f_{\mathcal{A}}$ for the values of the enumerated elements. Once $\mathcal{A}_j$ isomorphic to $\mathcal{B}_j$ is discovered, we are sure that $\mathcal{A}_j$ is an initial segment of $(\omega, <_{\mathcal{A}}, f_{\mathcal{A}})$. We continue in this manner and wait for x and at least one $y >_{\mathcal{A}} x$ to enter some $\mathcal{A}_j$. At this point we know the position of x in $<_{\mathcal{A}}$ and $Succ_{\mathcal{A}}(x)$ can be read out from $\mathcal{A}_j$. ◀

We prove (I)⇒(II) by contrapositive. Notice that we can additionally assume that we work with computable block functions which are not almost identities because, as discussed earlier, such functions have the trivial degree spectrum.

▶ **Theorem 21.** *Let f be a computable block function such that it is not almost identity and it satisfies the effectiveness condition $(\star)$. Assume that the successor is not recoverable from f on $(\omega, <)$. Then the degree spectrum of f on $(\omega, <)$ contains a non-c.e. degree.*

▶ **Corollary 22.** *Suppose f is a computable block function satisfying the effectiveness condition $(\star)$. Then* (I) *implies* (II)*, i.e., if the degree spectrum of f on $(\omega, <)$ is equal to all c.e. degrees, then the successor is recoverable from f on $(\omega, <)$.*

Before we start the proof, let us remind a few conventions regarding Turing functionals. A Turing functional, also called a Turing operator or a computable operator, is a program that is allowed to query an oracle. We can enumerate them in an effective way $\Theta_0, \Theta_1, \ldots$. We write $\Theta_e^X(n)$ for the output of the eth Turing functional on input n when it uses $X \subseteq \omega$ as oracle. Θ_e corresponds to a fixed program that can be used with different oracles. We assume in s steps it is possible to read at most the first s entries of the oracle. For a finite string $\sigma \in 2^{<\omega}$, $\Theta_e^\sigma(n)$ means the same as $\Theta_{e,|\sigma|}^\sigma(n)$. Sometimes we explicitly write the number of steps t after the code of the program, $\Theta_{e,t}^\sigma$.

Proof of Theorem 21. Fix an effective enumeration $(\Phi_i, \Psi_i, W_i)_{i\in\omega}$ containing all triples such that Φ_i and Ψ_i are Turing functionals, and W_i is a c.e. set. For brevity, the graph of the function $f_{\mathcal{A}}$ (i.e., the isomorphic image of our function f inside $\mathcal{A}$) is denoted by Γ_A. We build a computable isomorphic copy $\mathcal{A} = (\omega, <_A)$ of the ordering $(\omega, <)$. Along the construction, we satisfy the following requirements:

If $W_i = \Phi_i^{\Gamma_A}$ and $\Gamma_A = \Psi_i^{W_i}$, then there is a computable sequence $(\mathcal{B}_j)_{j\in\omega}$ of growing initial segments of $(\omega, <, f)$ such that each $\mathcal{B}_j$ embeds in $(\omega, <, f)$ precisely once. $(\mathcal{P}_i)$

Satisfying each $\mathcal{P}_i$ is sufficient by Lemma 20. Indeed, since the successor is not recoverable from f, Lemma 20 guarantees that the degree Γ_A is not c.e. Our strategy for satisfying $\mathcal{P}_i$ incorporates the classical construction of a properly d.c.e. Turing degree by Cooper [7] adapted to the setting of block functions [2]. Recall that a Turing degree is properly d.c.e. if it contains a subset X of natural numbers such that X is equal to the difference of two c.e. sets and X is not Turing equivalent to any c.e. set. The eventual success of our strategy is secured by a series of threats which attempt to build $(\mathcal{B}_m)$. This will be a degenerate infinite injury construction, described using the framework of trees of strategies (see, e.g., [22]). Degeneracy here means that infinitary outcomes (to be defined) never turn up as the true outcomes of the construction.

For the sake of simplicity, first we give a construction for the case when each block is isomorphic to a cycle, as in Example 18. The modifications needed for the general case of the theorem will be discussed at the end of the proof.

Strategy for $\mathcal{P}_i$. The strategy attempts to build a computable sequence of initial segments $(\mathcal{B}_m)_{m\in\omega}$. Suppose that our strategy starts working at a stage s_0. Then the strategy proceeds as follows.

(1) Set $m = 0$.

(2) Choose two adjacent blocks $\tilde{\mathcal{C}}_m < \tilde{\mathcal{D}}_m$ in $(\omega, <, f)$ such that we have not copied them into the structure $\mathcal{A}_{s_m}$ yet (where s_m is the current stage), and $\tilde{\mathcal{D}}_m$ contains at least two elements. Such a choice is possible, since the function f is not almost identity (hence, it has infinitely many cycles of size ≥ 2).
We extend $\mathcal{A}_{s_m}$ to a finite structure $\mathcal{A}^1_m$ by copying all missing elements up to the end of the block $\tilde{\mathcal{D}}_m$ (these elements are appended to the end of $\mathcal{A}_{s_m}$). Then the structure $\mathcal{A}^1_m$ can be decomposed as follows:

$$\mathcal{A}^1_m = \mathcal{A}^{1,init}_m + \mathcal{C}_m + \mathcal{D}_m,$$

where $\mathcal{C}_m \cong \tilde{\mathcal{C}}_m$ and $\mathcal{D}_m \cong \tilde{\mathcal{D}}_m$.
Let x_m be the rightmost element of $\mathcal{C}_m$, and let y_m be the leftmost element of $\mathcal{D}_m$. It is clear that at the moment, $\langle x_m, y_m\rangle \notin \Gamma_A$. As usual, we restrict "$\langle x_m, y_m\rangle \notin \Gamma_A$" by forbidding lower priority actions to add new elements between the elements of $\mathcal{A}^1_m$.

(3) Wait for a stage $s' > s_m$ witnessing the following computations: for some $t' \leq s'$, we have (at the stage s')

$$W_i \upharpoonright t' = \Phi_i^{\Gamma_A} \upharpoonright t', \text{ and } 0 = \Gamma_A(\langle x_m, y_m\rangle) = \Psi_i^{W_i\upharpoonright t'}(\langle x_m, y_m\rangle).$$

Note that the current structure $\mathcal{A}_{s'}$ can be decomposed as follows:

$$\mathcal{A}_{s'} = \mathcal{A}^1_m + \mathcal{A}^{2,fin}_m = \mathcal{A}^{1,init}_m + \mathcal{C}_m + \mathcal{D}_m + \mathcal{A}^{2,fin}_m,$$

where $\mathcal{A}^{2,fin}_m$ contains the elements added after the end of Step (2).

(4) We define $\mathcal{B}_m := \mathcal{A}_{s'}$. We extend $\mathcal{A}_{s'}$ by adding *precisely one* fresh element between the rightmost element of $\mathcal{A}^{1,init}_m$ and the leftmost element of $\mathcal{C}_m$.
Inside the resulting structure $\mathcal{A}^3_m$, the number x_m becomes the leftmost element of a copy of the cycle $\mathcal{D}_m$. This implies that at the moment, we have $\langle x_m, y_m\rangle \in \Gamma_A$. Similarly to Step (2), we restrict "$\langle x_m, y_m\rangle \in \Gamma_A$".

(5) We wait for a stage $s'' > s'$ witnessing the following condition: for some $t'' \leq s''$, we have $t' \leq t''$ and

$$W_i \upharpoonright t'' = \Phi_i^{\Gamma_A} \upharpoonright t'', \text{ and } 1 = \Gamma_A(\langle x_m, y_m\rangle) = \Psi_i^{W_i\upharpoonright t''}(\langle x_m, y_m\rangle).$$

When the stage s'' is found, we go back to Step (2) with $m + 1$ (in place of m), while *simultaneously waiting* at Step (6) with m.

(6) We wait for a stage $s''' > s''$ witnessing the following condition: one can embed the elements of $\mathcal{A}_{s'''}$ into $(\omega, <, f) \upharpoonright (2 \cdot \mathrm{card}(\mathrm{dom}(\mathcal{A}_{s'''})) + 1)$ in the following special "semi-isomorphic" way. There exists a 1-1 function $\xi\colon \mathrm{dom}(\mathcal{A}_{s'''}) \to [0, 2 \cdot \mathrm{card}(\mathrm{dom}(\mathcal{A}_{s'''}))]$ such that:

- ξ respects the ordering, i.e., for *all* elements $x, y \in \mathcal{A}_{s'''}$, $x <_{\mathcal{A}_{s'''}} y \Leftrightarrow \xi(x) < \xi(y)$;
- if x (originally) was an element of some block I inside $\mathcal{A}_{s'} = \mathcal{B}_m$, then the whole block I should go into some copy of I inside $(\omega, <, f)$; in other words, the restriction of ξ to the domain of $\mathcal{A}_{s'}$ is an isomorphic embedding from $(\mathrm{dom}(\mathcal{A}_{s'}), <_{\mathcal{A}_{s'}}, f_{\mathcal{A}_{s'}})$ into $(\omega, <, f)$.

Note the following: if such a "semi-isomorphic" embedding ξ exists, then one may assume that this ξ satisfies an additional condition:

If I is a block inside $\mathcal{A}_m^{1,init}$, then I is mapped to its counterpart inside $(\omega, <, f)$ (∗∗)
(i.e., if $x \in I$ and x is the i-th element from the left inside $\mathcal{A}_{s'''}$, then $\xi(x)$ equals i).

Indeed, since the structure $\mathcal{A}_m^3$ from Step (4) is obtained by adding only one element just before $\mathcal{C}_m$, the embedding ξ must move the contents of $\mathcal{C}_m$ to the right of the $(\omega, <, f)$-counterpart of $\mathcal{A}_m^{1,init}$. This allows us not to move $\mathcal{A}_m^{1,init}$, and just map it to its $(\omega, <, f)$-counterpart.

(7) Extend $\mathcal{A}_{s'''}$ to a finite structure by using the "semi-isomorphic" embedding ξ described above. More formally, we take the least unused numbers $y \notin \mathrm{dom}(\mathcal{A}_{s'''})$ to extend $\mathcal{A}_{s'''}$ to a finite structure $\mathcal{A}_m^4$ such that there exist a number N and an isomorphism $h\colon \mathcal{A}_m^4 \cong (\omega, <, f) \upharpoonright N$ with the property $\xi \subseteq h$. Stop the strategy, including the actions for all $m' \neq m$.

Outcomes of $\mathcal{P}_i$.
w_m : Waiting at Step (3) forever for this m. Then either $W_i \neq \Phi_i^{\Gamma_A}$ or $\Gamma_A \neq \Psi_i^{W_i}$.
w'_m : Waiting at Step (5) forever for this m. Then, again, $W_i \neq \Phi_i^{\Gamma_A}$ or $\Gamma_A \neq \Psi_i^{W_i}$
s : "Stop", i.e., some m reached Step (7). Then we have:

- $0 = \Gamma_A(\langle x_m, y_m\rangle) = \Gamma_{A,s'''+1}(\langle x_m, y_m\rangle) = \Psi_{i,s'}^{W_{i,s'}\upharpoonright t'}(\langle x_m, y_m\rangle)$.
- Let u be the use for the computation $\Psi_{i,s'}^{W_{i,s'}\upharpoonright t'}(\langle x_m, y_m\rangle) = 0$ at Step (3). Notice that $u \leq t'$. Since at Step (5) we see $\Psi_{i,s''}^{W_{i,s''}\upharpoonright t''}(\langle x_m, y_m\rangle) = 1$, this implies that there exists an element $a \leq u$ such that $a \in W_{i,s''} \setminus W_{i,s'}$.
- Since $a \notin W_{i,s'}$, at Step (3) we have $0 = \Phi_{i,s'}^{\Gamma_{A,s'}}(a)$. Let v be the use for the computation $\Phi_{i,s'}^{\Gamma_{A,s'}}(a) = 0$.
- The embedding ξ from Step (6) guarantees that $\Gamma_A \upharpoonright v = \Gamma_{A,s'} \upharpoonright v$. Hence, $W_i(a) = W_{i,s''}(a) = 1 \neq 0 = \Phi_i^{\Gamma_A}(a)$.

Therefore, the requirement $\mathcal{P}_i$ is satisfied.
∞ : Eventually waiting at Step (6) for each $m \in \omega$. Then for each m, every $\mathcal{A}_{s'''}$ lacks an appropriate "semi-isomorphic" embedding ξ. This means that each $\mathcal{B}_m$ can be isomorphically embedded only once into $(\omega, <, f)$. As discussed above, this contradicts Lemma 20.

The current outcome of a strategy is equal to:

- s, if the strategy is already stopped.
- Otherwise, let m be the current (maximal) value of our strategy parameter m. If for this m, we wait at Step (3), then the outcome is w_m. If we wait at Step (5), then the outcome is w'_m.

Construction. We use the following ordering of the finitary outcomes: $s < \dots < w'_2 < w_2 < w'_1 < w_1 < w'_0 < w_0$. The tree of strategies includes only the finitary outcomes. More formally, we set $\Lambda = \{s\} \cup \{w_m, w'_m : m \in \omega\}$, and the tree T is equal to $\Lambda^{<\omega}$. The i-th level of the tree contains strategies $\alpha \in T$ devoted to the requirement $\mathcal{P}_i$.

If σ and τ are two finite strings from T, then by $\sigma\widehat{\ }\tau$ we denote the concatenation of σ and τ. As usual, we say that σ is *to the left* of τ if there exist some $\rho \in T$ and $o_1, o_2 \in \Lambda$ such that $o_1 < o_2$, $\sigma \supseteq \rho\widehat{\ }o_1$, and $\tau \supseteq \rho\widehat{\ }o_2$.

As usual, at a stage s of the construction we visit the strategies $\alpha_0, \alpha_1, \dots, \alpha_s$, where $\alpha_0 = \emptyset$, and for each $i < s$ we have $\alpha_{i+1} = \alpha_i\widehat{\ }o$, where o is the current outcome of the strategy α_i. By g_s we denote the current finite path, i.e., the sequence $(\alpha_0, \alpha_1, \dots, \alpha_s)$.

Verification. It is clear that the constructed $\mathcal{A}$ is a computable linear order on ω: indeed, if $x < y$ inside $\mathcal{A}_s$ for some $s \in \omega$, then $x <_{\mathcal{A}_t} y$ for all $t \geq s$.

Let g be the true path of the construction, i.e., the limit $\lim_s g_s$. More formally, for $k \in \omega$ and $\alpha \in T$, here we have

$$g(k) = \alpha \iff \exists t (\forall s \geq t)(g_s(k) = \alpha).$$

Note that in general, g could be a finite sequence.

▶ **Lemma 23.** *The path g is infinite, and every requirement $\mathcal{P}_i$ is satisfied.*

Proof. Suppose that a strategy α belongs to the true path g, and its associated requirement is $\mathcal{P}_i$. Consider the following three cases.

Case 1. There is a number m such that starting from some stage s, the outcome of α is always the same $o \in \{w_m, w'_m\}$. Then it is clear that $\alpha\widehat{\ }o$ belongs to g. In addition, for the number m, the strategy is forever stuck either at Step (3) or at Step (5). This means that $W_i \neq \Phi_i^{\Gamma_A}$ or $\Gamma_A \neq \Psi_i^{W_i}$.

Case 2. At some step α has outcome s. Then $\alpha\widehat{\ }s$ lies on the path g. In addition, $W_i \neq \Phi_i^{\Gamma_A}$ as discussed in the description of the outcome s.

Case 3. Otherwise, for each $m \in \omega$, α eventually goes through the outcomes w_m and w'_m. Since we never reach Step (7), this means that each finite structure $\mathcal{B}_m$ can be isomorphically embedded into $(\omega, <, f)$ only once: indeed, if some $\mathcal{B}_m$ could be embedded twice, then we could eventually use the second such embedding to recover the "semi-isomorphic" map ξ and to reach Step (7) for this m. Then, as discussed in the beginning of the proof of the theorem, Lemma 20 guarantees that Case 3 is impossible.

We conclude that the path g is infinite, and each $\mathcal{P}_i$ is satisfied. ◀

In order to finish the proof of Theorem 21, now it is sufficient to show that the order $\mathcal{A} = (\omega, <_A)$ is isomorphic to $(\omega, <)$.

Recall that at Step (6), we always choose a map ξ satisfying Condition $(**)$. Consider a $\mathcal{P}_i$-strategy α. Condition $(**)$ implies that for every $m \in \omega$ and every $x \in \mathcal{A}_m^{1,init}$, α never adds new elements which are $<_A$-below x.

Suppose that an element x is added to $\mathcal{A}$ by some strategy σ.

Consider an arbitrary strategy α. If α is to the right of σ, then α never works after the starting stage of σ. If $\alpha \supset \sigma$ or α is to the left of σ, then x always belongs to $\mathcal{A}_m^{1,init}$ of this particular α. Hence, α never adds elements $<_A$-below x.

We deduce that new elements which are $<_A$-below x could be added only by $\alpha \subseteq \sigma$. The proof of Lemma 23 implies that each such α adds only finitely many elements to $\mathcal{A}$. Therefore, for an arbitrary element x, $\mathcal{A}$ contains only finitely many elements $<_A$-less than x. This implies that $(\omega, <_A)$ is isomorphic to $(\omega, <)$. This concludes the proof for the case when each f-block is a cycle.

The general case. Now we discuss the general case of the theorem. The proof essentially follows the outline provided above, but we have to address two important details of how to implement the strategy for $\mathcal{P}_i$.

The first detail concerns Step (2), where one needs to choose two adjacent blocks $\tilde{\mathcal{C}}_m$ and $\tilde{\mathcal{D}}_m$. The question is how to *algorithmically* choose them?

Here the computability of the function $cp_f(x)$ is important – this fact guarantees that for a given $x \in \omega$, one can effectively recover the f-block I containing x. This effective recovery allows us to computably find the needed blocks $\tilde{\mathcal{C}}_m$ and $\tilde{\mathcal{D}}_m$.

The recovery of the block I can be arranged as follows. Without loss of generality, we may assume that x is the leftmost element of I. Since we know the value $cp_f(x)$, we can find all elements from the preimage $f^{-1}(\{x\})$. By using the function cp_f several times, we eventually find the *finite* set

$$P_f(x) = \bigcup_{j\in\omega} (f^{-1})^{(j)}(\{x\}).$$

If $P_f(x)$ already forms an f-block, then we stop the algorithm. Otherwise, there is (the least) $y_0 \not\in P_f(x)$ such that $x < y_0 < \max P_f(x)$. We find the finite set $P_f(y_0)$. If the set $P_f(x) \cup P_f(y_0)$ forms an f-block, then we stop. Otherwise, again, we find the least $y_1 \not\in P_f(x) \cup P_f(y_0)$ such that $y_0 < y_1 < \max(P_f(x) \cup P_f(y_0))$. We consider $P_f(x) \cup P_f(y_0) \cup P_f(y_1)$, etc. Since every f-block is finite, eventually we will find the desired f-block I.

The second detail concerns Step (2) and its interaction with Step (4). Since the block function f is not almost identity, f satisfies at least one of the following two conditions:

(a) There are infinitely many elements u such that u is the leftmost element of its block and $f(u) > u$.

(b) There are infinitely many pairs (u, v) such that u and v belong to the same block, u is the leftmost element of the block, $u < v$, and $f(v) = u$.

Assume that f satisfies (b) (the case of (a) could be treated in a similar way). Then in Step (2) of the strategy, we can always choose $\tilde{\mathcal{C}}_m, \tilde{\mathcal{D}}_m$ with the following property: the block $\tilde{\mathcal{D}}_m$ contains a pair $(\tilde{u}, \tilde{v})$ satisfying the condition described in item (b).

After building $\mathcal{A}^1_m$ (as described in Step (2)), we will choose x_m and y_m as follows. The element y_m is the rightmost element of $\mathcal{C}_m$, and the element x_m is the copy (inside $\mathcal{D}_m$) of the element $\tilde{v} \in \tilde{\mathcal{D}}_m$.

The intention behind this particular choice of x_m, y_m is the following. At the end of Step (2), we have $\langle x_m, y_m\rangle \not\in \Gamma_A$. *In addition*, if we add *precisely one* element at Step (4), then we will immediately obtain that the condition $\langle x_m, y_m\rangle \in \Gamma_A$ becomes satisfied (since $\tilde{u} = f(\tilde{v})$ is the leftmost element of $\tilde{\mathcal{D}}_m$).

Taking the discussed details into account, one can arrange the proof of the general case in a straightforward manner. Theorem 21 is proved. ◀

4 Further Observations

In this section, we discuss some additional observations that could be interesting for a reader familiar with computable structure theory. First, we observe a connection with the paper [18] (more specifically, its Proposition 4.13), where a technique of recovering the successor is used. The main takeaway of the first observation is that our Theorem 21 cannot be deduced from this result.

Let $n \geq 1$, and let $k = \lceil n/2 \rceil$. A set $A \subseteq \omega^m$ is *n-c.e.* if there exist c.e. sets $U_1, V_1, U_2, V_2, \ldots, U_k, V_k$ such that $A = (U_1 \setminus V_1) \cup (U_2 \setminus V_2) \cup \cdots \cup (U_k \setminus V_k)$ and if n is odd, then $V_k = \emptyset$. This notion was introduced by Putnam in [28] where he proved that a set A is n-c.e. if and only if there exists a recursive approximation of A (i.e., a recursive family of computable sets (A_s) satisfying $\lim A_s = A$) such that $A_0 = \emptyset$ and $card(\{t : A_t(x) \neq A_{t+1}(x)\}) \leq n$, for all $x \in \omega$. An m-ary relation R on $(\omega, <)$ is *intrinsically n-c.e.* if for every computable copy $\mathcal{B}$ of $(\omega, <)$, the set $R_{\mathcal{B}}$ is n-c.e.

The notion of an intrinsically n-c.e. relation could be extended to the transfinite levels of the Ershov hierarchy [11, 12, 13]: for a computable non-zero ordinal α, one can introduce the definition of an *intrinsically α-c.e.* relation. In addition, one can talk about being intrinsically α-c.e. *on a cone* (see Chapter 1 in [18]). Nevertheless, here we can omit the formal technical details: Harrison-Trainor (Proposition 4.12 in [18]) proved that if a relation R on $(\omega, <)$ is intrinsically α-c.e. on a cone, then R is intrinsically n-c.e. for some n. In the same paper, Harrison-Trainor (Proposition 4.13 in [18]) proved that every intrinsically n-c.e. relation R (on the structure $(\omega, <)$) satisfies (II). If every function f satisfying the premise of our Theorem 21 were intrinsically n-c.e. on $(\omega, <)$, then our result would follow from the aforementioned Proposition 4.13 in [18]. However, according to the following proposition, this is not the case.

▶ **Proposition 24.** *There exists a function f which is not intrinsically n-c.e., for any n, with the following properties: f is a computable block function, f is not almost identity, and f satisfies the effectiveness condition $(\star)$.*[3]

Proof. Let f be any computable block function such that for every $x \in \omega$, the f-block containing x is a cycle (the notion of cycle is defined in Example 18). It suffices to show that for each $n > 0$, there exists a computable copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ of $(\omega, <)$ such that the graph of $f_{\mathcal{A}}$ is not n-c.e.

Fix a standard effective listing $(X_i)_{i \in \omega}$ of all n-c.e. sets. We fix $n > 0$ and construct a desired computable copy $\mathcal{A}$ by finite injury priority argument. At each stage s, the current approximation $(A_s, <_{A_s}; f_{A_s})$ of $(\omega, <_{\mathcal{A}}; f_{\mathcal{A}})$ is isomorphic to the initial segment of $(\omega, <; f)$. Here is the requirement that we need to satisfy, for each natural number i:

$$\text{The graph of } f_{\mathcal{A}} \text{ is not equal to } X_i. \qquad (\mathcal{R}_i)$$

Strategy in isolation. When $\mathcal{R}_i$ enters the construction, say at stage s, we use fresh numbers to append to the current $\mathcal{A}_s$ a copy of the cycle that is immediately to the right of $\mathcal{A}_s$ in the standard copy. We thus obtain $\mathcal{A}_{s+1}$. Let u, v be the new cycle's endpoints (in $\mathcal{A}_{s+1}$) such that $f_{\mathcal{A}_{s+1}}(u) = v$. From now on, we will follow the computable approximation $X_{i,t}(\langle u, v \rangle)$ of our n-c.e. set. When we see that $X_{i,t}(\langle u, v \rangle) = 1$, then (if needed) we push things to the right (in an appropriate way) so that, after pushing, we have $f_{\mathcal{A}_{t+1}}(u) \neq v$. When we see that $X_{i,t}(\langle u, v \rangle) = 0$, then we push to the right to obtain $f_{\mathcal{A}_{t+1}}(u) = v$.

Observe that we can always do the "pushing". If f has only finitely many types of blocks, then we can arrange the construction in a way that each $\mathcal{R}_i$ is attached to a cycle that appears infinitely often. To obtain $f_{\mathcal{A}}(u) \neq v$, it suffices to insert one fresh number just before the cycle containing u, v. To obtain $f_{\mathcal{A}}(u) = v$ again, it suffices to search for the next occurrence of the cycle on which $\mathcal{R}_i$ was initialized and insert, right before v, sufficiently many fresh numbers so that v lands again on the first position of such a cycle (and u on the last).

[3] Notice that these properties correspond to the premise of Theorem 21.

If f has infinitely many types, then f has cycles of arbitrary length. In that case, obtaining $f_{\mathcal{A}}(u) \neq v$ is as above. To obtain $f_{\mathcal{A}}(u) = v$ it suffices to find a long enough cycle (appearing to the right of the current position of u), push to the right (by inserting fresh numbers just before v) and put the right amount of fresh numbers just after v so that u and v become endpoints again. ◀

▶ Remark 25. Observe that there are functions that satisfy the premise of Theorem 21 and have all c.e. degrees as a spectrum. For example, this will be true for any function of the following form: f is as in the proof of Proposition 24, and there exists an infinite c.e. set $L \subset \omega$ such that if $l \in L$, then there is precisely one f-block of cardinality l.

Our second observation looks at our property (II) from the point of view of computable categoricity. A computable structure $\mathcal{M}$ is *relatively computably categorical* if for any countable structure $\mathcal{N}$ such that $\mathrm{dom}(\mathcal{N}) = \omega$, there is an isomorphism $f \colon \mathcal{M} \cong \mathcal{N}$ such that f is computable in the atomic diagram $D(\mathcal{N})$.

Goncharov [16] proved that a computable structure $\mathcal{M}$ is relatively computably categorical if and only if there exists a c.e. family Θ of (finitary) existential formulas (with a fixed tuple of parameters $\bar{c}$ from $\mathcal{M}$) with the following properties:

- every tuple $\bar{a}$ from $\mathcal{M}$ satisfies some formula $\theta \in \Theta$;
- if $\bar{a}$ and $\bar{b}$ are tuples from $\mathcal{M}$ satisfying the same formula $\theta \in \Theta$, then $(\mathcal{M}, \bar{a}) \cong (\mathcal{M}, \bar{b})$.

Such a family Θ is usually called a *formally c.e. Scott family* for the structure $\mathcal{M}$.

▶ **Proposition 26.** *Let R be a computable relation on $(\omega, <)$. If the structure $(\omega, <, R)$ is relatively computably categorical, then the relation R satisfies* (II)*, i.e., for every computable copy $\mathcal{A}$ of $(\omega, <)$, we have $R_{\mathcal{A}} \geq_T Succ_{\mathcal{A}}$.*

Proof. Using a formally c.e. Scott family Θ for the structure $(\omega, <, R)$, we can obtain a finite tuple $\bar{c}$ and a computable sequence $(\psi_i(x, \bar{c}))_{i\in\omega}$ of existential formulas (in the language $\{<, R\}$) such that for each $k \in \omega$, the number k is the unique element x such that $(\omega, <, R) \models \psi_k(x, \bar{c})$.

Now let $\mathcal{A}$ be an arbitrary computable copy of $(\omega, <)$. Let $\bar{c}_{\mathcal{A}}$ be the isomorphic image (inside $\mathcal{A}$) of the tuple $\bar{c}$. Then we have:

- $y = Succ_{\mathcal{A}}(x)$ if and only if
$$(\mathcal{A}, R_{\mathcal{A}}) \models \bigvee_{i\in\omega} [\psi_i(x, \bar{c}_{\mathcal{A}}) \,\&\, \psi_{i+1}(y, \bar{c}_{\mathcal{A}})];$$
- $y \neq Succ_{\mathcal{A}}(x)$ if and only if
$$(\mathcal{A}, R_{\mathcal{A}}) \models (y \leq_{\mathcal{A}} x) \vee \bigvee_{i\in\omega,\, k\geq i+2} [\psi_i(x, \bar{c}_{\mathcal{A}}) \,\&\, \psi_k(y, \bar{c}_{\mathcal{A}})].$$

This implies that the graph of $Succ_{\mathcal{A}}$ is both c.e. and co-c.e. with respect to $R_{\mathcal{A}}$. Thus, we have $R_{\mathcal{A}} \geq_T Succ_{\mathcal{A}}$. ◀

5 Conclusions

The philosophical framework of Shapiro [34] is based on the observation that, strictly speaking, computations are performed on syntactic objects, such as strings of symbols, rather than on natural numbers themselves. This leads to the formal notion of a *notation*. In this paper, we re-cast this framework within the setting of computable structure theory. It turns out that there is a straightforward translation between the two approaches: a notation for a given computable structure corresponds to a computable (isomorphic) copy of the structure. Surprisingly, this fact has been consistently overlooked in the literature (though, see [20]). The first contribution of this paper amounts to bringing this connection to the light, and showing, on the example of the structure $(\omega, <)$, how the two perspectives can inform each other.

In the context of the structure $(\omega, <)$, notations and computable structures find their common point in the property of computing the successor. The successor function was used by Shapiro to isolate the class of acceptable notations – those notations have particularly nice computational properties as functions computable in them are exactly the same as standard computable functions. The successor function has also useful applications in the investigation of degree spectra of computable relations on $(\omega, <)$; a particularly handy property is the recoverability of the successor from R on $(\omega, <)$ (which means that, in all computable copies of $(\omega, <)$, the image of the successor is computable relative to the image of R). This naturally leads us to the notion of relatively acceptable notation, i.e., a notation in which the successor is computable relative to a given additional relation.

The second, and main, contribution of the paper amounts to exploring the relationship between two properties of a computable relation R: (I) the degree spectrum of R on $(\omega, <)$ contains precisely the c.e. degrees, and (II) the successor is recoverable from R on $(\omega, <)$ (equivalently, every notation for $(\omega, <)$ is acceptable relative to R). It is a known fact in computable structure theory that (II) implies (I). Our main result (Theorem 21) concerns the reverse implication with regard to relations R that are graphs of unary total computable functions (actually, a particular subclass of so-called block functions, first considered in [2]). Theorem 21 proves that for a large class K of functions f, the two conditions given above are equivalent. Informally speaking, our proof uses the fact that the functions from the class K have pretty tame combinatorial properties.

As a concluding remark, we note that in general, Theorem 21 does not cover the class of all computable functions – even for the case of block functions. The proof of the following theorem is sketched in Appendix A.

▶ **Theorem 27.** *There exists a computable block function f such that:*

1. *the corresponding function $cp_f(x)$ is not computable;*
2. *each f-block occurs infinitely often in $(\omega, <, f)$;*
3. *the degree spectrum of f on $(\omega, <)$ contains all Δ_2 degrees.*

Nevertheless, we conjecture that the equivalence of (I) and (II) could be established for the class encompassing all unary total computable functions.

References

1 Chris J. Ash and Julia Knight. *Computable structures and the hyperarithmetical hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam, 2000.

2 Nikolay Bazhenov, Dariusz Kalociński, and Michał Wrocławski. Intrinsic complexity of recursive functions on natural numbers with standard order. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2022.8.

3 Paul Benacerraf. What numbers could not be. *The Philosophical Review*, 74(1):47–73, 1965. doi:10.2307/2183530.

4 Paul Benacerraf. Recantation or Any old ω-sequence would do after all. *Philosophia Mathematica*, 4(2):184–189, 1996. doi:10.1093/philmat/4.2.184.

5 Jennifer Chubb, Andrey Frolov, and Valentina S. Harizanov. Degree spectra of the successor relation of computable linear orderings. *Archive for Mathematical Logic*, 48(1):7–13, 2009. doi:10.1007/s00153-008-0110-6.

6 Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936. doi:10.2307/2371045.

7 S. Barry Cooper. *Degrees of unsolvability.* PhD thesis, University of Leicester, 1971.

8 Brian Jack Copeland and Diane Proudfoot. Deviant encodings and Turing's analysis of computability. *Studies in History and Philosophy of Science Part A*, 41(3):247–252, 2010.

9 Nigel Cutland. *Computability: An Introduction to Recursive Function Theory.* Cambridge University Press, Cambridge, MA, 1980.

10 Rod Downey, Bakhadyr Khoussainov, Joseph S. Miller, and Liang Yu. Degree spectra of unary relations on $(\omega, \leq)$. In *Logic, Methodology and Philosophy of Science: Proceedings of the Thirteenth International Congress*, pages 35–55. College Publications, 2009. URL: http://homepages.mcs.vuw.ac.nz/~downey/publications/LOJan24.pdf.

11 Yurii L. Ershov. A hierarchy of sets. I. *Algebra and Logic*, 7(1):25–43, 1968. doi:10.1007/BF02218750.

12 Yurii L. Ershov. On a hierarchy of sets, II. *Algebra and Logic*, 7(4):212–232, 1968. doi:10.1007/BF02218664.

13 Yurii L. Ershov. On a hierarchy of sets. III. *Algebra and Logic*, 9(1):20–31, 1970. doi:10.1007/BF02219847.

14 Ekaterina B. Fokina, Valentina S. 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. doi:10.1017/CBO9781107338579.006.

15 E. Mark Gold. Limiting Recursion. *Journal of Symbolic Logic*, 30(1):28–48, 1965. doi:10.2307/2270580.

16 Sergey S. Goncharov. Autostability and computable families of constructivizations. *Algebra and Logic*, 14(6):392–409, 1975. doi:10.1007/BF01668470.

17 Valentina S. Harizanov. *Degree spectrum of a recursive relation on a recursive structure.* PhD thesis, University of Wisconsin-Madinson, 1987.

18 Matthew Harrison-Trainor. Degree spectra of relations on a cone. *Memoirs of the American Mathematical Society*, 253(1208):1–120, 2018. doi:10.1090/memo/1208.

19 Denis R. Hirschfeldt. Degree spectra of relations on computable structures. *Bulletin of Symbolic Logic*, 6(2):197–212, 2000. doi:10.2307/421207.

20 Dariusz Kalociński and Michał Wrocławski. Generalization of Shapiro's theorem to higher arities and noninjective notations. *Archive for Mathematical Logic*, September 2022. doi:10.1007/s00153-022-00836-4.

21 Julia F. Knight. Degrees coded in jumps of orderings. *Journal of Symbolic Logic*, 51(4):1034–1042, 1986. doi:10.2307/2273915.

22 Steffen Lempp. Priority arguments in computability theory, model theory, and complexity theory, 2012. preprint available on the author's webpage. URL: https://people.math.wisc.edu/~lempp/papers/prio.pdf.

23 Anatolii I. Mal'tsev. Constructive algebras. I. *Russian Mathematical Surveys*, 16(3):77–129, 1961. doi:10.1070/RM1961v016n03ABEH001120.

24 Anatolii I. Mal'tsev. On recursive abelian groups. *Soviet Mathematics. Doklady*, 3:1431–1434, 1962.

25 Andrei A. Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta imeni V.A. Steklova*, 38:176–189, 1951. in Russian.

26 Antonio Montalbán. *Computable structure theory: Within the arithmetic.* Cambridge University Press, 2021.

27 Michael Moses. Relations intrinsically recursive in linear orders. *Mathematical Logic Quarterly*, 32(25-30):467–472, 1986. `doi:10.1002/malq.19860322514`.

28 Hilary Putnam. Trial and Error Predicates and the Solution to a Problem of Mostowski. *Journal of Symbolic Logic*, 30(1):49–57, 1965. `doi:10.2307/2270581`.

29 Paula Quinon. A taxonomy of deviant encodings. In Florin Manea, Russell G. Miller, and Dirk Nowotka, editors, *Sailing Routes in the World of Computation*, volume 10936 of *LNCS*, pages 338–348, Cham, 2018. Springer International Publishing.

30 Michael Rescorla. Church's Thesis and the conceptual analysis of computability. *Notre Dame Journal of Formal Logic*, 48(2):253–280, 2007. `doi:10.1305/ndjfl/1179323267`.

31 Linda J. C. Richter. *Degrees of Unsolvability of Models*. PhD thesis, University of Illinois at Urbana-Champaign, 1977.

32 Linda J. C. Richter. Degrees of structures. *Journal of Symbolic Logic*, 46(4):723–731, 1981. `doi:10.2307/2273222`.

33 Hartley Rogers. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, 1987.

34 Stewart Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23(1):14–20, 1982. `doi:10.1305/ndjfl/1093883561`.

35 Stewart Shapiro, Eric Snyder, and Richard Samuels. Computability, notation, and de re knowledge of numbers. *Philosophies*, 7(1), 2022. `doi:10.3390/philosophies7010020`.

36 John C. Shepherdson and Howard E. Sturgis. Computability of recursive functions. *Journal of the ACM*, 10(2):217–255, 1963. `doi:10.1145/321160.321170`.

37 Robert I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, New York, 1987.

38 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Ser. 2*, 42:230–265, 1936. `doi:10.1112/plms/s2-42.1.230`.

39 Matthew Wright. Degrees of relations on ordinals. *Computability*, 7(4):349–365, 2018. `doi:10.3233/COM-180086`.

40 Michał Wrocławski. *Representations of numbers and their computational properties*. PhD thesis, University of Warsaw, Warsaw, 2019.

A Proof of Theorem 27

▶ **Theorem 27.** *There exists a computable block function f such that:*

1. *the corresponding function $cp_f(x)$ is not computable;*
2. *each f-block occurs infinitely often in $(\omega, <, f)$;*
3. *the degree spectrum of f on $(\omega, <)$ contains all Δ_2 degrees.*

Proof Sketch. As usual, for $i \in \omega$, φ_i denotes the unary partial computable function which has Gödel number i.

We build a computable block function f satisfying the following series of requirements:

$$\text{The function } cp_f \text{ is not equal to } \varphi_i. \qquad (\mathcal{P}_i)$$

$$\text{Each } f\text{-block occurs infinitely often inside } (\omega, <, f). \qquad (\mathcal{R})$$

At a stage s, we define the finite function $f \restriction N_s$ (where $N_s \in \omega$ and $N_s < N_{s+1}$) in such a way that $(\{0, 1, \dots, N_s - 1\}, <, f \restriction N_s)$ consists of blocks.

Beforehand, we put $f(0) = 0$, $f(1) = 2$, and $f(2) = 1$.

The $\mathcal{R}$-strategy is a global one. The $\mathcal{R}$-requirement is satisfied in a simple way, as follows. At the end of each construction stage s, let B_s be the current set of all (isomorphism types of) blocks. Then (before starting the stage $s + 1$) we use fresh numbers x (i.e., the least numbers such that $f(x)$ is not defined yet) to extend f in the following way: for each block I from B_s, we add *precisely two* adjacent copies of I.

Note that in addition to the $\mathcal{R}$-requirement, this procedure will guarantee that in the final structure $(\omega, <, f)$, there will be infinitely many pairs $(x, x+1)$ such that $f(x) = x$ and $f(x+1) = x+1$. This preliminary observation will help us in the verification.

Strategy for $\mathcal{P}_i$. Suppose that the strategy starts working at a stage $s_0 + 1$.

(1) Choose a large fresh size l_i such that the cycle of size l_i cannot be isomorphically embedded into the current structure $(\{0, 1, \ldots, N_{s_0} - 1\}, <, f \restriction N_{s_0})$. (See Example 18 for the definition of a cycle).
Extend f by adding a copy of the cycle of size l_i. Let w_i be the leftmost element of this copy.

(2) Wait for a stage $s' > s_0 + 1$ such that $\varphi_{i,s'}(w_i)\downarrow = 1$.

(3) Extend f by taking the least number x such that $f(x)$ is still undefined, and setting $f(x) := w_i$.
This concludes the description of the strategy. It is clear that it satisfies the requirement $\mathcal{R}_i$. Indeed, assume that the function φ_i is total. If $\varphi_i(w_i) \neq 1$, then the element w_i is a part of a cycle, and $cp_f(w_i) = 1 \neq \varphi_i(w_i)$. If $\varphi_i(w_i) = 1$, then we have $cp_f(w_i) = 2$.

Construction. The construction is arranged as a standard finite injury argument. The requirements are ordered: $\mathcal{P}_0 < \mathcal{P}_1 < \mathcal{P}_2 < \ldots$. When a higher priority strategy $\mathcal{P}_i$ acts (i.e., it extends f in its Step (3)), it initializes all lower priority strategies $\mathcal{P}_j$, $j > i$.

Verification. The non-computability of the function $cp_f(x)$ can be proved by a standard argument for finite injury constructions.

A more hard part is how to show that every Δ_2 degree belongs to the degree spectrum of the constructed f. This can be achieved by arranging an argument similar to the argument of Case (a) of Theorem 14 in [2]. Roughly speaking, given an arbitrary Δ_2 set X, we should encode it via a computable copy $\mathcal{A} = (\omega, <_A)$ as follows. For each $e \in \omega$, the numbers $4e$ and $4e + 2$ will be $<_A$-adjacent, and more importantly:

- if $e \in X$, then $f_{\mathcal{A}}(4e) = 4e + 2$ and $f_{\mathcal{A}}(4e + 2) = 4e$;
- if $e \notin X$, then $f_{\mathcal{A}}(4e) = 4e$ and $f_{\mathcal{A}}(4e + 2) = 4e + 2$.

Our preliminary observation helps us to arrange this encoding. This concludes the proof sketch of Theorem 27. ◀