Archive for Mathematical Logic (2023) 62:257–288
https://doi.org/10.1007/s00153-022-00836-4

**Mathematical Logic**

# Generalization of Shapiro's theorem to higher arities and noninjective notations

**Dariusz Kalociński[1] · Michał Wrocławski[2]**

Received: 12 October 2020 / Accepted: 27 May 2022 / Published online: 14 September 2022



## Abstract

In the framework of Stewart Shapiro, computations are performed directly on strings of symbols (numerals) whose abstract numerical interpretation is determined by a notation. Shapiro showed that a total unary function (unary relation) on natural numbers is computable in every injective notation if and only if it is almost constant or almost identity function (finite or co-finite set). We obtain a syntactic generalization of this theorem, in terms of quantifier-free definability, for functions and relations relatively intrinsically computable on certain types of equivalence structures. We also characterize the class of relations and partial functions of arbitrary finite arities which are computable in every notation (be it injective or not). We consider the same question for notations in which certain equivalence relations are assumed to be computable. Finally, we discuss connections with a theorem by Ash, Knight, Manasse and Slaman which allow us to deduce some (but not all) of our results, based on quantifier elimination.


**Keywords** Intrinsic computability · Equivalence relations · Notations for natural numbers · Definability · Learnability

**Mathematics Subject Classification** 03C57

D. Kalociński is funded by the National Science Centre Poland grant no. 2018/31/B/HS1/04018.

✉ Dariusz Kalociński
dariusz.kalocinski@gmail.com

Michał Wrocławski
m.wroclawski@uw.edu.pl

1 Institute of Computer Science, Polish Academy of Sciences, ul. Jana Kazimierza 5, 01-248 Warsaw, Poland

2 Department of Philosophy, University of Warsaw, ul. Krakowskie Przedmieście 3, 00-927 Warsaw, Poland

## 1 Introduction and overview

Philosophical analyses of the concept of computation often involve, more or less explicitly, the syntactical layer of numerals and the semantical layer of abstract natural numbers (see, e.g., [6, 7, 11, 20, 21]). This distinction is justified by the intuition, already present in some of the founding works of computability theory [16, 26], that algorithms are performed directly on strings of symbols, while the notion of computing on numbers is established indirectly by assuming an appropriate mapping from strings to numbers. Stewart Shapiro formalized this idea by introducing the concept of notation—an injective function from a recursive set of numerals onto natural numbers [23]. Each notation determines the class of functions and relations computable in it. Shapiro showed that, in general, computability of a number-theoretic function or relation is not an invariant property across all notations. Classical computability theory may be viewed as a theory of partial functions and relations computable in an acceptable notation, where *acceptable* means *with computable successor*.

While searching for the characterization of the class of acceptable notations, Shapiro proved the following.

**Theorem 1** (Shapiro [23]) *The only unary total functions computable in every injective notation are almost constant and almost identity functions. The only subsets of natural numbers whose characteristic functions are computable in every injective notation are finite and co-finite sets.*[1]

Shapiro concluded that partial recursive functions and relations are not the same as partial functions and relations computable in every injective notation. This was sufficient as an intermediate step in finding criteria for acceptability of notations which was Shapiro's primary goal.

It might be argued that Theorem 1 has some depth on its own, regardless of notations' acceptability. It indicates that effectiveness has a representationally independent core—the class of functions and relations computable in every injective notation. This idea is familiar in computable structure theory (for a general introduction see, e.g. [3, 17]). Given a computable structure $\mathcal{A}$ (i.e. a relational structure whose domain and all basic relations are uniformly computable) and a computable relation $R$, we say that $R$ is intrinsically computable on $\mathcal{A}$ if the image of $R$ is computable in every computable isomorphic copy of $\mathcal{A}$ (see, e.g., [2]). It is easy to observe that injective notations correspond directly to isomorphisms between structures, as known in computable structure theory, and thus both perspectives lead to the same problems, though manifested differently. For example, functions computable in every injective notation are the same as functions intrinsically computable on plain natural numbers (a structure consisting of natural numbers with no additional structure).

In this paper, we consider two groups of results inspired by Shapiro's Theorem 1 and the notion of intrinsic computability. One concerns the problem of relative intrinsic computability on certain types of equivalence structures. Equivalence relation may seem a simple structural addition but it leads to rich theory (see, e.g., [4, 5, 8, 9, 12, 13, 19]). We say that $R$ is relatively intrinsically computable on a computable structure

[1] On a side note, Theorem 1 can be deduced from some results on automorphic triviality in computable structure theory [15].

$\mathcal{A}$ if in every copy $\mathcal{B}$ of $\mathcal{A}$ the image of $R$ is computable relative to $\mathcal{B}$. This problem is tackled in Sect. 3 and some related observations are given in Sect. 4. An important result in this area is that of Ash et al. (Theorem 17) [1]. Some but not all of our results can be deduced from it. This is discussed in the final section.

The second group of results concerns arbitrary notations, i.e. notations that may be noninjective. Under a noninjective notation numbers are allowed to have more than one name. This minor change is not irrelevant (see, e.g., [27–29]). We consider the following problem: which functions are computable in every notation? In Sect. 5, using a long inductive argument, we demonstrate that the class in question consists of the empty function, constant functions and projections. A similar problem arises for equivalence structures: which functions are computable in every notation in which a given equivalence relation is computable? It seems that this notion has no clear equivalent in computable structure theory, though similarities with the notion of intrinsic computability are evident. We also describe functions computable in every notation if arities of input and output are not fixed.

We wrap up our work in Sect. 6 which concerns related work and open questions.

## 2 Background

Familiarity with basic concepts from model theory and recursive function theory is assumed. We review notions that are necessary to follow the proofs. For more information, we refer the reader to textbooks [10, 22, 25].

The set of natural numbers is denoted by $\omega$. $\Sigma$ stands for a finite alphabet, $\Sigma^*$ for the set of all words (finite sequences) over $\Sigma$. Letters $\alpha, \beta, \ldots$, possibly with subscripts, refer to words. A characteristic function of a relation $R$ is denoted by $\chi_R$. We sometimes confuse $R$ with its characteristic function $\chi_R$ which makes writing $R(x_1, \ldots, x_k) = 1$ or $R(x_1, \ldots, x_k) = 0$ meaningful. Finite lists $v_1, v_2, \ldots, v_k$ of elements of a given set are abbreviated by $\vec{v}$.

**Definition 1** ([14, 18]) A function $f : \omega^k \to \omega$ is learnable (in $B$) if there exists a uniformly computable family of computable (in $B$) functions $\{f_t\}_{t\in\omega}$ such that for every $\vec{n} = n_1, n_2, \ldots, n_k$: $f(\vec{n}) = \lim_{t\to\infty} f_t(\vec{n})$. A relation $R \subseteq \omega^k$ is learnable (in $B$) if its characteristic function $\chi_R$ is learnable in $B$.

**Theorem 2** (Limit Lemma [24]) *$f : \omega^k \to \omega$ is learnable in $B$ if and only if $f \leq B'$. Similarly, a relation $A \subseteq \omega^k$ is learnable in $B$ if and only if $A \leq B'$.*

Consider a first-order language with variables $V = \{x_1, x_2, \ldots\}$, individual constants $C = \{c_1, c_2, \ldots\}$ and no functional or relational symbols (except the logical predicate $=$). Sometimes we also consider one relational binary predicate $E$. The set of formulae is defined in a standard way. If we write $\varphi(z_1, z_2, \ldots, z_k)$, all the free variables of $\varphi$ occur among $z_1, z_2, \ldots, z_k$. A model for our language is a pair $\mathcal{M} = (M, \{c_i^M : i \in \omega\})$, where $M$ is a non-empty set and $c_i^M \in M$ is the distinguished element named by $c_i$, for $i \in \omega$. If the language contains $E$, then the model is of the form $\mathcal{M} = (M, E^M, \{c_i^M : i \in \omega\})$, where $E^M$ is the interpretation of $E$. Any function $a : V \to M$ is called an assignment. The satisfaction relation $\models$ between a model $\mathcal{M}$, formula $\varphi$ and an assignment $a$, is defined in a standard way. We write $\mathcal{M} \models \varphi[a]$ to say that $\varphi$ is satisfied in $\mathcal{M}$ under the assignment $a$. Given a formula $\varphi(z_1, \ldots, z_k)$, it is customary to write $\mathcal{M} \models \varphi[a_1, \ldots, a_k]$ where $a_i$ is understood as the element assigned to $z_i$.

If $E$ is an equivalence relation, the equivalence class of an element $e$ is denoted by $[e]_E$, or $[e]$ if $E$ is clear from context. By an equivalence structure we mean $(\omega, E)$ where $E$ is an equivalence relation. The character of such structure is the set $A$ consisting of all such $(k, n) \in \omega^2$ that $E$ has at least $n$ equivalence classes of size $k$. We will consider mainly two types of equivalence relations. The first type are equivalence relations of finite character. Such equivalence relations have finitely many finite equivalence classes. The second type are equivalence relations of unbounded character. We say that the character of $E$ is bounded if there is some finite $k$ such that all finite classes have size at most $k$. Otherwise, we say that the character of $E$ is unbounded. $E$ of unbounded character has arbitrarily large finite classes.

### 2.1 Notations for natural numbers

The following notion is a slight generalization of that of Shapiro. Note that Shapiro used the term *notation* to mean *1-1 notation*.

**Definition 2** (Shapiro [23]) Let $\Sigma$ be a finite alphabet. $(S, \sigma)$ is a notation (for $\omega$) if $S \subseteq \Sigma^*$ is computable, $\sigma : S \to \omega$ is onto. If $\sigma$ is one-one, then $(S, \sigma)$ is called injective or 1-1.

Elements of $S$ are referred to as numerals. The standard notation and standard numerals are understood as the usual decimal notation and its numerals, respectively. When we refer to numerals rather than numbers, we put bars over them: $\overline{n}$ is the standard decimal numeral for the number $n$. However, in a non-standard notation it may represent a different number, or it may not be a valid numeral at all.

**Definition 3** ([29]) Let $(S, \sigma)$ be a notation and let $f : \omega^n \to \omega$ be a partial function. Let $f^\sigma$ be the class of all partial functions $F : S^n \to S$ such that for any $\alpha_1, \ldots, \alpha_n, \beta \in S$ the following condition is satisfied:

$$(\alpha_1, \ldots, \alpha_n) \in dom(F) \iff (\sigma(\alpha_1), \ldots, \sigma(\alpha_n)) \in dom(f),$$
$$F(\alpha_1, \ldots, \alpha_n) = \beta \implies f(\sigma(\alpha_1), \ldots, \sigma(\alpha_n)) = \sigma(\beta).$$

If a notation is noninjective, then there might be multiple functions in $f^\sigma$. For a 1-1 notation, $f^\sigma$ is a singleton and we identify $f^\sigma$ with its sole element.

If some function from $f^\sigma$ is computable, we say that $f^\sigma$ is computable or that $f$ is computable in $(S, \sigma)$.

There is going to be a certain ambiguity when we talk about computing $f^\sigma$. Unless explicitly stated otherwise, it shall be synonymous with computing any function from the class $f^\sigma$. However, sometimes, when a concrete function from this class has already been specified, it can refer to computing this specific function.

**Definition 4** Let $(S, \sigma)$ be a notation and let $R \subseteq \omega^n$. Let $R^\sigma \subseteq S^n$ be defined in the following way:

$$(\alpha_1, \ldots, \alpha_n) \in R^\sigma \iff (\sigma(\alpha_1), \ldots, \sigma(\alpha_n)) \in R,$$

for all $\alpha_1, \ldots, \alpha_n \in S$. We say that $R$ is computable (or c.e.) in $(S, \sigma)$ if $R^\sigma$ is computable (or c.e.).

The above definitions might be summarized as follows. Computability of a number-theoretic function (or relation) in a notation is understood as the existence of a program which acts on numerals and outputs numerals (or truth values) such that the underlying referents agree with the function (relation) being computed.

**Definition 5** Let $(S, \sigma)$ be an injective notation. A function $f : \omega^k \to \omega$ is learnable in $(S, \sigma)$ if $f^\sigma$ is learnable. Similarly, a relation $R \subseteq \omega^k$ is learnable in $(S, \sigma)$ if $R^\sigma$ is learnable.

The notion of learnability in arbitrary notations is not considered in this paper. This concept can be defined in a few non-equivalent ways. In fact, our notion of computable enumerability for arbitrary notations (Definition 4) is just one possible.

## 3 Some generalizations for injective notations

In this section we consider the problem of relative intrinsic computability of functions on certain types of computable equivalence structures. We obtain a solution for total functions and relations (Theorems 4, 5 and Corollary 1).

It might be tempting to conjecture that Theorem 1 holds for functions and relations of higher arities. However, this is not the case, as demonstrated by Proposition 3.

**Definition 6** We say that a function $f : \omega^k \to \omega$ is almost constant if there exists $y \in \omega$ such that $f(x_1, \ldots, x_k) = y$ holds for all but finitely many tuples $(x_1, \ldots, x_k)$. Similarly, a function $f : \omega^k \to \omega$ is said to be almost projection if there exists $i$ such that $1 \leq i \leq k$ and $f(x_1, \ldots, x_k) = x_i$ holds for all but finitely many tuples $(x_1, \ldots, x_k)$.

**Proposition 3** *There exists a total function which is computable in every injective notation but is neither almost constant nor almost projection.*

***Proof*** Take the function $f : \omega^2 \to \omega$ defined by $f(x_1, x_2) = 0$ if $x_1 = 0 \vee x_2 = 0$ and $f(x_1, x_2) = 1$ if $x_1 \neq 0 \wedge x_2 \neq 0$. $\square$

Generalizing Shapiro's theorem to arbitrary partial functions and relations requires a slightly different perspective. Consider the first-order language $\mathcal{L} = (E)$ with a binary relational symbol $E$ (and the logical predicate $=$), with $x_1, x_2, \ldots$ as variables and $\underline{0}, \underline{1}, \ldots$ as constants. Definition of first-order formulae is standard. We say that a formula $\phi(x_1, \ldots, x_k, y)$ defines a function $f : \omega^k \to \omega$ in an $\mathcal{L}$-model $\mathcal{A} = (\omega, E^A)$ if for all $n_1, \ldots, n_k, m \in \omega$: $\mathcal{A} \models \phi(x_1, \ldots, x_k, y)[n_1, \ldots, n_k, m] \iff f(n_1, \ldots, n_k) = m$. We assume that a constant $\underline{n}$ is interpreted as $n$. A model for $\mathcal{L}$ will be written as $(\omega, E)$, innocuously confusing $E$ with its denotation.

Also, the following a bit technical notion will be quite handy.

**Definition 7** Let $(\omega, E)$ be an equivalence structure and let $X \subseteq \omega^n$. We say that $f : X \to \omega$ is an $E$-projection if there exists $e \in \omega$ such that for every $\vec{x} \in X$, $f(\vec{x}) > e$, $f(\vec{x}) E e$, $f(\vec{x}) \notin \vec{x}$ and $|[e]_E \cap (e, \infty)| = |\{x_i : x_i E e \wedge x_i > e\}| + 1$.

Below we define a family of equivalence relations (indexed by subsets of the universe) over tuples of an equivalence structure. This notion allow us to encapsulate certain cumbersome details in Lemmas 1, 2 and 4 which will be used in the proof of Theorems 4 and 5 .

**Definition 8** Let $\mathcal{X} = (X, E)$ be an equivalence structure and $F \subseteq X$. We say that $\vec{a}, \vec{b} \in X^k$ are of the same $F$-type (in symbols: $\vec{a} \sim_F \vec{b}$) if, for every $i, j : \vec{a}, \vec{b}$ and $x \in F$ satisfy (i-iv).

$$a_i \in F \iff b_i \in F, \tag{i}$$

$$a_i \in F \implies a_i = b_i, \tag{ii}$$

$$a_i = a_j \iff b_i = b_j, \tag{iii}$$

$$a_i E a_j \iff b_i E b_j, \tag{iv}$$

$$a_i E x \iff b_i E x. \tag{v}$$

(For $X = \omega$ and $C = \{0, 1, \ldots, c\}$ we shall write small subscripted $_c$ instead of big subscripted $_C$.) If $\mathcal{Y} = (Y, F)$ is another equivalence structure, $h : \mathcal{X} \cong \mathcal{Y}$, $g \subseteq h$, $\vec{a} \in X^k, \vec{b} \in Y^k$, we use the notation $\vec{a} \sim_g^h \vec{b}$ to mean that $h(\vec{a}) = (h(a_1), \ldots, h(a_k)) \in Y^k$ is of the same $img(g)$-type as $\vec{b}$ or, equivalently, that $h^{-1}(\vec{b}) \in X^k$ is of the same $dom(g)$-type as $\vec{a}$.

Intuitively, $\vec{a}, \vec{b} \in X^k$ are of the same $C$-type if: (i) the positions at which elements from $C$ occur are the same, (ii) at those positions, both $\vec{a}$ and $\vec{b}$ have precisely the same values, (iii-iv) the same equalities and equivalences hold within $\vec{a}, \vec{b}$, position-wise, and, finally, (v) the positions at which elements equivalent to something from $C$ occur, contain equivalent elements.

**Lemma 1** *Let $(X, E)$ be an equivalence structure and let $C \subseteq X$. $\sim_C$ is an equivalence relation on $X^k$. Moreover, if $C$ is finite, then $X^k / \sim_C$ is finite.*

***Proof*** Consider possible arrangements of elements of $C$ in $k$-tuples, possible arrangements of pairs of indices from the set $\{1, 2, \ldots, k\}$ for which equality or equivalence holds, as well as possible arrangements of pairs $(c, i)$ with $c \in C$ and $i \in \{1, 2, \ldots, k\}$ for which we could have $c$ equivalent to the number at position $i$. □

**Lemma 2** *Let $\mathcal{X} = (X, E)$, $\mathcal{Y} = (Y, F)$ be equivalence structures, $h : \mathcal{X} \cong \mathcal{Y}$, $g \subseteq h$. Let $\vec{a}, \vec{b} \in X^k$, $\vec{c} \in Y^k$. If $\vec{a} \sim_{dom(g)} \vec{b}$ and $\vec{c} \sim_g^h \vec{a}$ then $\vec{c} \sim_g^h \vec{b}$.*

***Proof*** Assume $\vec{a} \sim_{dom(g)} \vec{b}$ and $\vec{c} \sim_g^h \vec{a}$. By definition of $\sim_g^h$, $h^{-1}(\vec{c}) \sim_{dom(g)} \vec{a}$. By transitivity of $\sim_{dom(g)}$, $h^{-1}(\vec{c}) \sim_{dom(g)} \vec{b}$. Hence, $\vec{c} \sim_g^h \vec{b}$. □

**Lemma 3** *Let $\phi(\vec{x}, \vec{d})$ be a quantifier-free formula in the language $\mathcal{L} = \{E\}$. If $E$ is of finite or unbounded character then there exists $c$ such that, for every $\vec{a} \sim_c \vec{b}$: $(\omega, E) \models \phi[\vec{a}] \iff (\omega, E) \models \phi[\vec{b}]$.*

***Proof*** Let $\phi(\vec{x}, \vec{d})$ be a quantifier-free formula. For equivalence structure $(\omega, E)$ with unbounded character choose $c$ as the maximum over (the values of) all constants occurring in $\phi$ and over the maxima of finite $[x]_E$ such that $x \leq$ some constant in $\phi$. For $(\omega, E)$ with finite character, we choose $c$ as the maximum over (the values of) all constants occurring in $\phi$ and over the maxima of all finite equivalence classes. The rest of the paragraph applies to both types of equivalence relations, with $c$ chosen appropriately.

Let $\vec{a} \sim_c \vec{b}$. We show that $(\omega, E) \models \phi[\vec{a}] \iff (\omega, E) \models \phi[\vec{b}]$. It suffices to prove this for atomic formulae $\psi$ in $\phi$. The cases $\psi := (\underline{d} = \underline{d}')$ and $\psi := (\underline{d} E \underline{d}')$ are obvious. Note that, for any $d$ occurring in $\phi$, $d \leq c$ which, by Definition 8(i-ii), implies that $a_i = d \iff b_i = d$ and, by Definition 8(v), $a_i E d \iff b_i E d$. This is sufficient for cases $\psi := (x_i = \underline{d})$ and $\psi := (x_i E \underline{d})$. Cases $\psi := (x_i = x_j)$ and $\psi := (x_i E x_j)$ are evident by Definition 8(iii-iv).

□

**Lemma 4** *Let $(\omega, E)$ be an equivalence structure and $f : \omega^n \to \omega$.*

*(a) If $E$ is of finite character then $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters iff for some $c$ every $f \restriction [\vec{x}]_{\sim_c}$ is constant or a projection.*
*(b) If $E$ is of unbounded character then $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters iff for some $c$ every $f \restriction [\vec{x}]_{\sim_c}$ is constant, projection or an $E$-projection.*

***Proof*** Let $f$ be definable in $(\omega, E)$ by a quantifier-free formula $\phi(\vec{x}, y, \vec{d})$. Choose $c$ as in Lemma 3, according to the type of equivalence relation. Fix $\vec{n}$ and let $f(\vec{n}) = m$.

We consider case (a) and show that if $m$ does not occur in $\vec{n}$ then $m \leq c$. The proof is by contradiction: *our assumption is that $m$ does not occur in $\vec{n}$ but $m > c$.* Observe that $[m]_E$ is not finite because otherwise we would have $m \leq c$ by the choice of $c$ (see Lemma 3). Therefore $[m]_E$ is infinite. But then there exists $m' > \max\{c, m, n_1, \ldots, n_k\}$ such that $m' E m$. We have $(\vec{n}, m) \sim_c (\vec{n}, m')$ and thus, by Lemma 3, $(\omega, E) \models \phi[\vec{n}, m] \iff (\omega, E) \models \phi[\vec{n}, m']$ so $f(\vec{n}') = m \neq m' = f(\vec{n}')$ which is impossible.

We continue the left-to-right implication for the case (a).

First, suppose $m$ occurs in $\vec{n}$ and $m \leq c$. We show that $f \restriction [\vec{n}]_{\sim_c}$ are constant. Let $\vec{n}' \sim_c \vec{n}$. Clearly, $m$ must occur in $\vec{n}'$ in exactly the same places as in $\vec{n}$. Therefore, $(\vec{n}', m) \sim_c (\vec{n}, m)$. Since $(\omega, E) \models \phi[\vec{n}, m]$, by Lemma 3 we also have $(\omega, E) \models \phi[\vec{n}', m]$, and thus $f(\vec{n}') = m$ so $f \restriction [\vec{n}]_{\sim_c}$ is constant.

Second, suppose that $m$ occurs in $\vec{n}$ and $m > c$. We show that $f \restriction [\vec{n}]_{\sim_c}$ are projections. Let $\vec{n}' \sim_c \vec{n}$. Choose $i \in \{1, 2, \ldots, k\}$ such that $n_i = m$. Observe that we have $(\vec{n}', n_i') \sim_c (\vec{n}, n_i)$. But $(\omega, E) \models \phi[\vec{n}, n_i]$ holds, so, by Lemma 3, $(\omega, E) \models \phi[\vec{n}', n_i']$ holds as well. Therefore, $f(\vec{n}') = n_i'$ which proves that $f \restriction [\vec{n}]_{\sim_c}$ is a projection.

Finally, we consider the case when $m$ does not occur in $\vec{n}$ and $m \leq c$.

We want to show that $f \restriction [\vec{n}]_{\sim_c}$ is constant. Let $\vec{n}' \sim_c \vec{n}$. Observe that $(\vec{n}, m) \sim_c (\vec{n}', m)$ by Definition 8(v) applied to $\vec{n}, \vec{n}'$. Therefore, by Lemma 3, $f \restriction [\vec{n}]_{\sim_c}$ is constant.

We proceed to (b). Suppose $E$ is an equivalence relation of unbounded character. We want to show that $f \restriction [\vec{n}]_{\sim_c}$ is constant, projection or an $E$-projection.

We begin with the case $m \notin \vec{n}$ and $m > c$.

First, consider the sub-case when $m$ is $E$-equivalent to some $n_i$ but $n_i$ is not $E$-equivalent with any element $\leq c$. Clearly, every element of $[m]_E$ is $> c$. Let $n_{i_1}, n_{i_2}, \ldots, n_{i_k}$ be all elements of $\vec{n}$ that are $E$-equivalent with $m$. We find $u > c$ with all elements of $[u]_E$ being $> c$ and such that $|[u]_E| \geq k + 2$. We can pick $\vec{u} = u_1, u_2, \ldots, u_k \in [u]_E$ so that $\vec{n} \sim_c \vec{n}'$, where $\vec{n}' = \vec{n}[u_1/n_{i_1}, \ldots, u_k/n_{i_k}]$. Let $v, v \in [u]_E$ such that $v \neq v'$ and $v \notin \vec{u}$, $v' \notin \vec{u}$. Observe that $(\vec{n}, m) \sim_c (\vec{n}', v) \sim_c (\vec{n}', v')$ which leads to $f(\vec{n}') = v \neq v = f(\vec{n}')$ which is a contradiction.

Second, consider the sub-case when $m$ is not $E$-equivalent with any $n_i$ nor with any element $\leq c$. Find $v$ such that all elements of $[v]_E$ are $> c$ and $v$ is not $E$-equivalent to $m$ nor to any $n_i$. Observe that $(\vec{n}, m) \sim_c (\vec{n}, v)$. Hence, by Lemma 3, $f(\vec{n}) = m \neq v = f(\vec{n})$ which is impossible.

Third, we consider the sub-case when $m$ is $E$-equivalent to some element $\leq c$ but is not $E$-equivalent to any $n_i$. We observe that there is no $m' > c$ with $m \neq m'$ and $mEm'$. Otherwise, we would have $(\vec{n}, m) \sim_c (\vec{n}, m')$ which, by Lemma 3, breaks functionality of $f$. Let $\vec{n} \sim_c \vec{n}'$. Note that $m$ is not $E$-equivalent to any $n_i'$ because otherwise we would have, for some $n_i', n_i' Em$ but since $m$ is equivalent to some element $\leq c$ and $\vec{n} \sim_c \vec{n}'$, would would also have, by Definition 8(v), $n_i Em$ which contradicts our assumption. Therefore, it is clear that $(\vec{n}, m) \sim_c (\vec{n}', m)$. By Lemma 3, $f(\vec{n}') = m$, so $f \restriction [\vec{n}]_{\sim_c}$ is constant.

The last sub-case is as follows: $m$ is $E$-equivalent to some element $e \leq c$ and is $E$-equivalent to some $n_i$ (assume $e$ is the largest such number $\leq c$). Let $n_{i_1}, \ldots, n_{i_k}$ be all elements of $\vec{n}$ that are $E$-equivalent to $m$. Clearly, we must have $|\{u : uEm \wedge u > c\}| = k + 1$ for otherwise there would be $m' > c$ different from $m, n_{i_1}, \ldots, n_{i_k}$ and $E$-equivalent to $m$ which would lead to $(\vec{n}, m) \sim_c (\vec{n}, m')$ thus breaking functionality of $f$. Now, given any $\vec{n}' \sim_c \vec{n}$, each $n_{i_j}'$, for $j = 1, \ldots, k$, must land in $[m]_E$. If such $n_{i_j}'$ is $\leq c$ then clearly $n_{i_j} = n_{i_j}'$. If such $n_{i_j}'$ is $> c$ then $n_{i_j}' \in \{u : uEm \wedge u > c\}$. Given any such arrangement of $n_{i_j}'$s there always remains one vacant number, denote it by $m'$, different from any $n_{i_j}'$ with $m' \in \{u : uEm \wedge u > c\}$. We observe that $(\vec{n}, m) \sim_c (\vec{n}', m')$. Hence, by Lemma 3, $f(\vec{n}') = m'$. We also have $m' > c, m'Ee, m \notin \vec{n}$ and

$|[e]_E \cap (e, \infty)| = |\{n_i : n_i E e \wedge n_i > e\}| + 1$ which means that $f \upharpoonright [\vec{n}]_{\sim_c}$ is an $E$-projection.

We continue with (b) by considering the case $m \in \vec{n}$ and $m \leq c$. It is easy to observe that for any $\vec{n}' \sim_c \vec{n}$ we have $(\vec{n}, m) \sim_c (\vec{n}', m)$. Therefore, by Lemma 3, $f \upharpoonright [\vec{n}]_{\sim_c}$ is constant.

The next case is $m \in \vec{n}$ and $m > c$. Let $\vec{n}' \sim_c \vec{n}$. Observe that $(\vec{n}, n_i) \sim_c (\vec{n}', n_i')$. Therefore, by Lemma 3, $f(\vec{n}') = n_i'$ so $f \upharpoonright [\vec{n}]_{\sim_c}$ is a projection.

The last case is as follows: $m \notin \vec{n}$ and $m \leq c$. Let $\vec{n}' \sim_c \vec{n}$. It is easy to observe that $(\vec{n}, m) \sim_c (\vec{n}', m)$. Therefore, by Lemma 3, $f(\vec{n}') = m$ so $f \upharpoonright [\vec{n}]_{\sim_c}$ is constant.

To prove ($\Leftarrow$) in (a), assume that there exists $c \in \omega$ such that for every equivalence class $N$ of $\sim_c$, $f \upharpoonright N$ is constant or a projection. Choose an appropriate $c$. By Lemma 1, let $N_1, N_2, \ldots, N_p$ be all equivalence classes of $\sim_c$. Observe that for each $N_i$ there exists a quantifier-free formula with parameters $\phi_i(\vec{x})$ defining $N_i$ in $(\omega, E)$. Now, if $f \upharpoonright N_i$ is constant and $f(\vec{n}) = d$ for all $\vec{n} \in N_i$, let $\psi_i := (\phi_i \Rightarrow y = \underline{d})$. If $f \upharpoonright N_i$ is a projection, i.e. for some $j \in \{1, 2, \ldots, k\}$, $f(\vec{n}) = n_j$, for all $\vec{n} \in N_i$, let $\psi_i := (\phi_i \Rightarrow y = x_j)$. Finally, let $\phi(\vec{x}, y) := \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_p$. Formula $\phi$ is quantifier-free and defines $f$ in $(\omega, E)$.

Finally, we consider the case (b). We choose $c$ such that each $f \upharpoonright [\vec{n}]_{\sim_c}$ is constant, projection or an $E$-projection. Again, let $N_1, \ldots, N_p$ be all equivalence classes of $\sim_c$ and let $\phi_1, \ldots, \phi_p$ be their defining quantifier-free formulae. If $f \upharpoonright N_i$ is constant or a projection, we proceed as in the paragraph above and obtain a suitable $\psi_i$. Suppose $f \upharpoonright N_i$ is neither constant nor projection but an $E$-projection. Let $\vec{n} \in N_i$ and $m = f(\vec{n})$. By the definition of $E$-projection, we choose $e \leq c$ such that $f(\vec{n}) E e \in [e]_E$. We also know that $f(\vec{n}) \notin \vec{n}$ and that $|[e]_E \cap (c, \infty)| = |\{n_i : n_i E e \wedge n_i > c\}| + 1$. Let $d_0, \ldots, d_k$ be all numbers $> c$ and $E$-equivalent to $e$. Hence, $f(\vec{n})$ must be one of $d_0, \ldots, d_k$. We construct formulae $\xi_j$, for $j = 0, \ldots, k$:

$$\xi_j(\vec{x}, y) = \Big( \bigwedge_{1 \leq t \leq k} x_{i_t} \neq d_j \Big) \implies y = d_j.$$

Now, $\psi_i := (\phi_i \implies \bigwedge_{1 \leq t \leq k} \xi_t)$. Finally, we let $\phi(\vec{x}, y) := \psi_1 \wedge \ldots \wedge \psi_p$. $\phi$ is quantifier-free and defines $f$ in $(\omega, E)$. □

**Theorem 4** *Let $f$ be a total function of arbitrary arity. Let $(\omega, E)$ be a computable equivalence structure with no infinite classes and such that there exist arbitrarily large cardinalities, each assumed by infinitely many classes. Then the following are equivalent:*

*(1) $f$ is relatively intrinsically computable on $(\omega, E)$,*
*(2) $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters.*

Proving (2) $\Rightarrow$ (1) is rather easy. A quantifier-free formula $\phi(\vec{x}, y)$ that defines $f$ in $(\omega, E)$ gives rise to a simple program which, if provided with $E^\sigma$ as oracle, where $\sigma$ is any 1-1 notation, computes $f^\sigma$: on input $\vec{\alpha}$, search for the unique $\beta$ such that $\phi(\vec{\alpha}, \beta)$ holds (in this $\phi$ parameters are replaced by their names according to $\sigma$).

To prove (1) $\Rightarrow$ (2), suppose $f$ is not definable in $(\omega, E)$ by any quantifier-free formula with parameters. We will construct a notation $(T, \tau)$ such that $f$ is not computable in $(T, \tau)$ relative to $(\omega, E)$. Let $T \subseteq \Sigma^*$ be an infinite computable set with 1-1 recursive enumeration $\beta_0, \beta_1, \ldots$.

**Construction.**

At stage $n+1$ we have a finite injection $\tau_n : [0, l_n] \to T$ which we extend to a finite injection $\tau_{n+1} : [0, l_{n+1}] \to T$, $l_n \leq l_{n+1}$. $\tau_{n+1}$ is chosen so that the following requirement is satisfied:

$\mathcal{R}_n$ : for each 1-1 notation $(T, \sigma)$ such that $\sigma \succ \tau_{n+1}$, $\Phi_n(E^\sigma) \not\simeq f^\sigma$ .

We say that $x$ is non-fresh if $x < |\tau_n|$. Otherwise, $x$ is called fresh. A fresh element that is $E$-equivalent to some element $< |\tau_n|$ is referred to as an orbital one. Otherwise, it is said to be free. A tuple $\vec{x} \in \omega^n$ is said to be fresh if some number occurring in $\vec{x}$ is fresh. Similarly, we say that a numeral $\alpha$ is fresh if $\alpha$ does not occur in $\tau_n$. A tuple $\vec{\alpha} = \alpha_1, \ldots, \alpha_n$ is said to be fresh if some numeral occurring in $\vec{\alpha}$ is fresh. Given an injection $\sigma$ mapping a number $x$ to a numeral $\alpha$, we sometimes write $x^\sigma$ to denote $\alpha$, and $\alpha_\sigma$ to denote $x$.

Extensions of $\tau_n$s are chosen so that each element of $T$ occurs in some $\tau_n$. The final notation is defined by $\tau = \bigcup_{n \in \omega} \tau_n$ (or, rather, $\tau^{-1}$, to remain consistent with Definition 2).

Occasionally, we may write $xE\vec{x}$ to mean that $x$ is equivalent to some element in $\vec{x}$. $\neg xE\vec{x}$ means that $x$ is not $E$-equivalent to any element from $\vec{x}$. Given a notation $\sigma$, we may sometimes write $\alpha E x$ or $\alpha E \beta$, etc. Strictly speaking, we should write $\alpha E x^\sigma$ or $\alpha E^\sigma \beta$, respectively. But this should be clear, given the underlying isomorphism $\sigma$.

In the construction we use a certain condition for which we introduce the following abbreviation:

$$\Gamma(\vec{\alpha}, \vec{x}, \tau_n) := \forall \alpha_j \in \vec{\alpha}((\forall \gamma \in \tau_n \neg \alpha_j E \gamma) \implies |[\alpha_j]_E| \leq |[x_j]_E|). \tag{$\Gamma$}$$

Below we describe the method of swap-and-transfer which is used throughout the construction. After that we proceed to the construction itself.

$\boxed{\text{Swap-and-transfer}}$ We are given a notation $\sigma \succ \tau_n$, $\vec{\alpha}$ and $\vec{x} \sim^\sigma_{\tau_n} \vec{\alpha}$ with $\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow= \beta$. First, we take a sufficiently large $\rho$, $\tau_n \preceq \rho \prec \sigma$, so that $\Phi_n^{E^\rho}(\vec{\alpha}) \downarrow= \beta$ with $\rho$ containing all $\alpha_i$s, $x_i$s, $\beta$ and, additionally, including each class $[\alpha_i]$, $[x_i]$, $[\beta]$, if it happens to be finite. We want to transform $\rho$ into $\widetilde{\rho}$ such that $\Phi_n^{E^{\widetilde{\rho}}}(\vec{\alpha}) \downarrow= \beta$ with each $\alpha_i$ sitting on $x_i$ in $\widetilde{\rho}$. Non-fresh elements $\alpha_j$ remain untouched. We perform the algorithms described below in the order listed. Along the way, we keep changing $\rho$ (we store the changes in the variable $\rho'$, initially $\rho' = \rho$) until the final $\widetilde{\rho}$ is reached which is $\rho'$ after performing all modifications. Below, we say that $\alpha_j$ is bad if the position of $\alpha_j$ in $\rho'$ is different from $x_j$. Otherwise $\alpha_j$ is good.

**Swapping.** As long as there is some bad orbital element $\alpha_j \in \vec{\alpha}$, or some bad free element $\alpha_j \in \vec{\alpha}$ with $\alpha_j E x_j$: pick such a bad $\alpha_j$ and swap the places of $\alpha_j$ and $\rho'(x_j)$.

We note that, before the swap, when dealing with a bad orbital $\alpha_j$, the position of $\alpha_j$ is $E$-equivalent to $x_j$ with $x_j$ being also a fresh orbital element. We also see that $E^{\rho'}$ is the same for $\rho'$ before and after the swap and that $\alpha_j$ is good after the swap.

After swapping, all bad $\alpha_j$s in the current $\rho'$ have the following property: $[\alpha_j] \neq [x_j]$.

Transferring moves bad $\alpha_j$s and transfers numerals from $[\alpha_j] \upharpoonright \rho'$ to good positions in $[x_j]$. We should be careful, because the numerals taken out from the positions in $[x_j] \upharpoonright |\rho'|$ may want to find their good positions as well (transferring is a kind of iterative process). We highlight that $\rho'$ might become incomplete before transfer is completed. Incompleteness means that some intermediate positions may be empty. Eventually, however, once the transfer is complete, the resulting $\widetilde{\rho}$ does not have any empty intermediate positions.

**Transferring.** First, we construct a transfer graph containing information about all necessary transfers. Nodes of the graph are $[\alpha_j]_{E^\sigma} \upharpoonright \rho$ for all bad $\alpha_j$ and corresponding $[x_j]_E \upharpoonright |\rho|$. Note that for an infinite equivalence class $[\alpha_j]_{E^\sigma}$ or $[x_j]_E$, the corresponding node contains only finitely many representatives of the class that happen to be in $\rho$. For convenience, we refer to nodes of the graph as if they were full classes. However, one should bear in mind that, strictly speaking, it is not true. For each node node $[\alpha_j]$ we add a directed edge $[\alpha_j] \to [x_j]$. It means that the elements of $[\alpha_j] \upharpoonright \rho'$ should be placed at positions $[x_j]$. We can safely assume that $|[\alpha_j]| \leq |[x_j]|$ because swap-and-transfer will be only run when such a condition is satisfied. For nodes $[\alpha_p]$, $[x_q]$ such that $[\alpha_p] = [x_q]^\sigma$ (notice that $p \neq q$) we add an undirected edge $[\alpha_p] - [x_q]$. It means that the numerals from $[\alpha_p] \upharpoonright \rho'$ which currently reside at positions from $[x_q]$ should be transferred to position from some class other than $[x_q]$ but, also, that numerals other than $[\alpha_p]$ should be transferred to positions from $[x_q]$. This ends the description of the transfer graph.

We say that a node $[\alpha_j]$ is an origin if it is not connected with any $[x_l]$ by an undirected edge. Practically, it means that no bad element should be transferred to positions at which currently the elements of $[\alpha_j]$ reside and, therefore, that the transfer can start from $[\alpha_j]$.

Figure 1 shows three simple examples of such graphs with an overall idea of how the transfer should work for them.

Transferring algorithm uses the following three procedures. Sometimes, we write $[A]_E$ to mean the closure of $A$ with respect to $E$.

$Subst(\rho', A, X)$ If there are numerals at positions $X$ in $\rho'$, cut them and store them in $A'$ (otherwise $A'$ will be empty). If the elements of $A$ are in $\rho'$, cut them from $\rho'$, along with the elements equivalent to those from $A$ in $\rho'$ (we call these additional elements companions). Paste the elements of $A$ (with companions) on positions from $X$ so that each $\alpha_j \in A$ lands on position $x_j$. Now, we have two cases. The first case is that there are not enough positions in $X$ to accommodate all members of $A$ (with companions). In this situation, we find enough positions outside $\rho'$ equivalent to those from $X$ to accommodate the rest of $A$ (with companions), thus extending $\rho'$ and filling intermediate empty positions with fresh numerals, if necessary. The second case is that there are enough positions in $X$ to accommodate all members of $A$ (with companions). In this situation, we accommodate them and if after that there remain empty positions in $X$, we fill them with fresh numerals. Output $A'$.

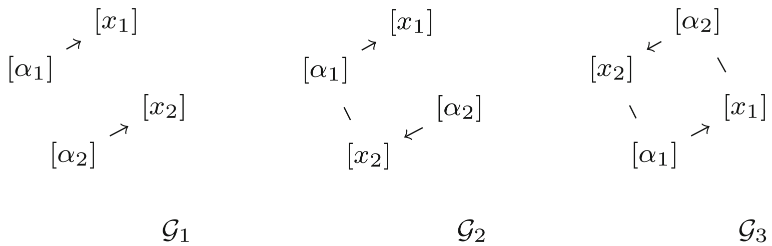


**Fig. 1** $\mathcal{G}_1$ has two origins: $\alpha_1$ and $\alpha_2$. Elements of $[\alpha_1]$ should be cut and pasted to positions from $[x_1]$, empty positions should be filled with fresh numerals while the numerals taken out from the positions in $[x_1]$ should be placed on some fresh equivalence class. The same applies to $\alpha_2$ and $x_2$. $\mathcal{G}_2$ has one origin: $\alpha_2$. Elements of $[\alpha_2]$ should be cut and pasted to positions from $[x_2]$, numerals taken out from $[x_2]$ (i.e. elements of $[\alpha_1]$) should be placed at positions from $[x_1]$, while the numerals taken out from $[x_1]$ should be moved to some fresh equivalence class. Finally, empty positions (after cutting $[\alpha_2]$) should be filled with fresh numerals. $\mathcal{G}_3$ has no origin. We pick an arbitrary node $[\alpha_i]$, say $[\alpha_1]$. We perform the same actions as above, i.e. elements of $[\alpha_1]$ are moved to positions from $[x_1]$, numerals taken out from $[x_1]$ (i.e. elements of $[\alpha_2]$) are moved to the positions from $[x_2]$. Here, no empty intermediate positions will be left

$Transfer(\rho', A)$ Let $X = [x] \upharpoonright |\rho'|$ be such that $[A] \to [x]$ is in the transfer graph. Let $A' := Subst(\rho', A, X)$. If $[A']$ contains a node of the transfer graph (i.e., some node $\subseteq [A']$)[2] then run $Transfer(\rho', A')$. Otherwise, if $A' \neq \emptyset$, run $Finalize(\rho', A')$.

$Finalize(\rho', A)$ Find a sufficiently large new equivalence class (i.e. not intersected by the current $\rho'$) and place the elements from $A$ on it. Fill empty intermediate positions with fresh numerals.

Now, the overall transferring for the whole graph works as follows. We set $\rho' = \rho$. If the graph has no origins, we pick an arbitrary node $A$ and we run $Transfer(\rho', A)$. Otherwise, for every origin node $A$, we run $Transfer(\rho', A)$ in succession. This way, we transform $\rho$ into $\widetilde{\rho}$.

We see that $E^{\widetilde{\rho}} \upharpoonright \rho$ is the same as $E^{\rho}$ and that each $\alpha_j$ is good in $\widetilde{\rho}$.

⋆⋆⋆

*Stage* 0. Set $\tau_0 = \emptyset$.

*Stage* $n+1$. In the questions below, Q1-Q5, $\sigma$ ranges over notations, so it is an infinite object.

1. Check whether

$$\exists \sigma \succ \tau_n \exists \vec{\alpha} \exists \vec{x} \sim^{\sigma}_{\tau_n} \vec{\alpha}[\Phi_n^{E^{\sigma}}(\vec{\alpha}) \downarrow =: \beta, \beta \in \tau_n, \beta_\sigma \neq f(\vec{x}), \Gamma(\vec{\alpha}, \vec{x}, \tau_n)]. \quad \text{(Q1)}$$

If not, go to Q2. Otherwise, we choose $\sigma, \vec{\alpha}, \vec{x}$ as above and we perform swap-and-transfer. This gets us $\widetilde{\rho} \succ \tau_n$ such that $\Phi_n^{E^{\widetilde{\rho}}}(\vec{\alpha}) \downarrow =: \beta$ with each $\alpha_i$ sitting on $x_i$ but the position of $\beta$ in $\sigma$ and $\widetilde{\rho}$ is the same. Hence, $\beta_{\widetilde{\rho}} = \beta_\sigma \neq f(\vec{x}) = f(\vec{\alpha}_{\widetilde{\rho}})$, so $\mathcal{R}_n$ is satisfied with $\tau_{n+1} = \widetilde{\rho}$.

[2] Observe that transferring may add to $\rho'$ some elements which are equivalent to $\alpha_j$ in $\rho'$ but which were not present in $\rho$ and thus are not included in the node $[\alpha_j]_{E^{\sigma}} \upharpoonright \rho$. To transfer such $\alpha_j$ we obviously cut all elements from $\rho'$ which are equivalent to $\alpha_j$ and proceed accordingly.

2. Check whether

$$\exists \sigma \succ \tau_n \exists \, \text{fresh} \, \vec{\alpha} \exists \, \text{free} \, \beta \notin \vec{\alpha} \; [\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow = \beta]. \tag{Q2}$$

If not, go to Q3. Choose $\sigma, \vec{\alpha}, \beta$ as above. Let $\rho$ be such that $\tau_n \prec \rho \prec \sigma$, $\Phi_n^{E^\rho}(\vec{\alpha}) = \beta$ and $\rho$ includes $\beta$, every $\alpha_j$ and every $[\alpha_j]_{E^\sigma}$ (if finite). Our goal is to modify $\rho$ and obtain $\widetilde{\rho} \succ \tau_n$ which preserves the computation $\Phi_n^{E^{\widetilde{\rho}}}(\vec{\alpha}) = \beta$ and $f(\vec{x}) \neq \beta_{\widetilde{\rho}}$. This way, $\mathcal{R}_n$ will be satisfied. Clearly, $[\beta]_{E^\rho}$ consists of free elements (notice that $[\beta]_{E^\rho}$ might contain some elements of $\vec{\alpha}$). We find large enough new equivalence class of cardinality $\geq |[\beta]_{E^\rho}| + 1$ and we put all elements of $[\beta]_{E^\rho}$ on the positions from the new class with one fresh extra numeral to cover the additional element. Empty positions are handled as usual. This way, we obtain $\rho' \succ \tau_n$ satisfying $E^\rho = E^{\rho'} \restriction \rho$ which preserves the computation. Now, we compute $f(\vec{\alpha}_{\rho'})$. If this is $\beta_{\rho'}$, then we swap $\beta$ with the extra numeral and denote the result by $\widetilde{\rho}$. Notice that, since $\beta \notin \vec{\alpha}$, moving $\beta$ does not affect the position of $\vec{\alpha}$. Clearly, $E^{\widetilde{\rho}}$ preserves the computation and we have $\beta_{\widetilde{\rho}} \neq f(\vec{\alpha}_{\widetilde{\rho}})$, thus satisfying $\mathcal{R}_n$. Hence, we can set $\tau_{n+1} := \widetilde{\rho}$. If $f(\vec{\alpha}_{\rho'}) \neq \beta_{\rho'}$, we can immediately set $\tau_{n+1} := \rho'$.
3. Check whether

$$\exists \sigma \succ \tau_n \exists \, \text{fresh} \, \vec{\alpha} \exists \vec{x} \sim_{\tau_n}^{\sigma} \vec{\alpha} \exists i [\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow =: \alpha_i, \, f(\vec{x}) \neq x_i, \, \Gamma(\vec{\alpha}, \vec{x}, \tau_n)]. \tag{Q3}$$

If not, go to Q4. Otherwise, choose $\sigma, \vec{\alpha}, \vec{x}$ and $i$ as above and we perform swap-and-transfer to get $\widetilde{\rho}$. We have $\Phi_n^{E^{\widetilde{\rho}}}(\vec{\alpha}) = \beta$ with each $\alpha_j$ sitting on $x_j$. Therefore, $f(\vec{\alpha}_{\widetilde{\rho}}) \neq \alpha_{i\,\widetilde{\rho}}$.
4. Check whether

$$\exists \sigma \succ \tau_n \exists \, \text{fresh} \, \vec{\alpha} \exists \, \text{orbital} \, \beta \notin \vec{\alpha} [\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow =: \beta, \\ |[\beta] - \tau_n| > |\{\text{fresh} \, \alpha_i : \alpha_i E \beta\}| + 1]. \tag{Q4}$$

If not, go to Q5. Choose $\sigma, \vec{\alpha}, \beta$ as above. Let $\rho$ be such that $\tau_n \prec \rho \prec \sigma$, $\Phi_n^{E^\rho}(\vec{\alpha}) = \beta$ and $\rho$ includes $\beta$ with every $[\alpha_j]_{E^\sigma}$. Observe that there is another orbital numeral $\gamma \neq \beta$ (different from any $\alpha_i E \beta$, if there is any such $\alpha_i$) such that $\gamma E \beta$. This follows from $|[\beta] - \tau_n| > |\{\text{fresh} \, \alpha_i : \alpha_i E \beta\}| + 1$. We check whether $f(\vec{\alpha}_\rho) \neq \beta_\rho$. If so, we set $\tau_{n+1} = \widetilde{\rho}$. Otherwise, we set $\tau_{n+1}$ as $\rho$ with $\gamma, \beta$ swapped (clearly, this does not affect the computation). $\mathcal{R}_n$ is satisfied.
5. Check whether

$$\exists \sigma \succ \tau_n \exists \, \text{fresh} \, \vec{\alpha} \exists \vec{x} \sim_{\tau_n}^{\sigma} \vec{\alpha} \exists \, \text{orbital} \, \beta \notin \vec{\alpha} [\Phi_n^{E^\sigma}(\vec{\alpha}) = \beta, \\ \neg f(\vec{x}) E \beta_\sigma \vee \exists i (f(\vec{x}) = x_i \wedge x_i E \beta_\sigma), \Gamma(\vec{\alpha}, \vec{x}, \tau_n)]. \tag{Q5}$$

Choose $\sigma, \vec{\alpha}, \vec{x}, \beta$ as above. We apply swap-and-transfer which yields $\widetilde{\rho}$. Clearly, $\beta_{\widetilde{\rho}}$ lands on the same equivalence class as $\beta_\sigma$. If $\neg f(\vec{\alpha}_{\widetilde{\rho}}) E \beta_\sigma$, then, in particular, $f(\vec{\alpha}_{\widetilde{\rho}}) \neq \beta_{\widetilde{\rho}}$ and we set $\tau_{n+1} = \widetilde{\rho}$. If $\exists i (f(\vec{x}) = x_i \wedge x_i E \beta_\sigma)$ then we let $\alpha_{i_1}, \ldots, \alpha_{i_l}$ be all numerals from $\vec{\alpha}$ occupying the same $E$-equivalence class as $\beta$.

After swap-and-transfer, each $\alpha_{i_j}$, $j = 1, \dots, l$, sits on $x_{i_j}$ in $\widetilde{\rho}$. But $\beta$ is different from each $\alpha_{i_j}$, for $j = 1, \dots, l$, and, therefore, $\beta_{\widetilde{\rho}} \neq f(\vec{x})$, so we can set $\tau_{n+1} = \widetilde{\rho}$.

**Verification.**

The following lemmas imply that for every $n$, for every 1-1 notation $\sigma \succ \tau_{n+1}$, $\Phi_n(E^\sigma) \not\simeq f^\sigma$. Since each $\tau_{n+1}$ is a prefix of $\tau$, we have $\Phi_n^{E^\tau} \not\simeq f^\tau$ for every $n$, as needed.

**Lemma 5** *If at stage $n+1$ some question Q1-Q5 is answered affirmatively, then for every 1-1 notation $\sigma \succ \tau_{n+1}$, if $\Phi_n^{E^\sigma}$ is total then $\Phi_n^{E^\sigma} \neq f^\sigma$.*

***Proof*** This should be obvious by the construction. The first question with positive answer at stage $n+1$ leads to $\tau_{n+1}$ such that, for some $\vec{\alpha}$, $\Phi_n^{E^{\tau_{n+1}}}(\vec{\alpha}) \downarrow \neq f^{\tau_{n+1}}(\vec{\alpha})$. □

**Lemma 6** *If questions* Q1-Q5 *are all answered negatively at stage* $n+1$, *then for every 1-1 notation $\sigma \succ \tau_{n+1}$, $\Phi_n^{E^\sigma}$ is not total.*

***Proof*** Fix $n$. At stage $n+1$ we already have $\tau_n$. Fix $\sigma \in T^\omega$ such that $\sigma \succ \tau_n$. Suppose that questions Q1-Q5 are all answered negatively at stage $n+1$. Towards a contradiction, suppose that $\Phi_n^{E^\sigma}$ is total. We will show that $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters. In general, our aim is to obtain suitable quantifier-free definitions of each $f \upharpoonright [\vec{n}]_{\sim_c}$.

Fix $\vec{n}$. Let $c = |\tau_n| - 1$. Clearly, if $\vec{n}$ is not fresh, $[\vec{n}]_{\sim_c}$ is a singleton and thus $f \upharpoonright [\vec{n}]_{\sim_c}$ is constant. In the reminder, $\vec{n}$ is fresh.

Let $l$ be the number of all free classes having representatives in $\vec{n}$ and let $n_{i_1}, n_{i_2}, \dots, n_{i_l}$ be such representatives from $\vec{n}$, one for each free class (observe that these representatives must be pair-wise non-$E$-equivalent). We may assume $i_1 < i_2 < \cdots < i_l$. For each $i_j$, $j = 1, 2, \dots, l$, we take $k_{i_j} = |\{p : n_p E n_{i_j}\}|$. Below we show how to select numbers $k'_{i_j}$ for $j = 1, \dots, l$. Numbers $k'_{i_j}$ will play an important role later.

Suppose we want to select $k'_{i_j}$. Observe that there is a cardinality $\geq k_{i_j}$ that is realized infinitely often by free equivalence classes. We choose $k'_{i_j}$ to be the least such cardinality. If some cardinalities $k$, $k_{i_j} \leq k < k'_{i_j}$, are realized by free equivalence classes then, by the choice of $k'_{i_j}$, they are realized only finitely often. All free classes that realize such cardinalities are referred to as $i_j$-exceptions. Let $C_{i_j}$ be the set of all $i_j$-exceptions.

Now, we define $B^j_{[\vec{n}]_{\sim_c}}$, for $j = 1, \dots, l$, as the set of all $\vec{m} \sim_c \vec{n}$ such that each free $m_p$ satisfying $m_{i_j} E m_p$ belongs to some $i_j$-exception. We see that the set $B_{[\vec{n}]_{\sim_c}} = \bigcap_{j=1}^{l} B^j_{[\vec{n}]_{\sim_c}}$ is finite.

In the remainder of the proof we examine what happens with $f \upharpoonright ([\vec{n}]_{\sim_c} - B_{[\vec{n}]_{\sim_c}})$. Eventually we will see that this restriction is definable in $(\omega, E)$ by a quantifier-free formula with parameters.

Consider an arbitrary $J \subseteq \{1, \dots, l\}$. For such $J$ we define a family $\mathcal{B}^{\vec{n}}_J$ of all injections $g : J \to \bigcup_{j \in J} C_{i_j}$ satisfying $g(j) \in C_{i_j}$. Now, for each $J \subseteq \{1, \dots, l\}$

with nonempty $\mathcal{B}^{\vec{n}}_J$, for every $g \in \mathcal{B}^{\vec{n}}_J$, we select $\vec{\alpha}^g$ in the following way. If free $n_p$ is not $E$-equivalent with any $n_{i_j}$ for $j \in J$, then we take $\alpha^h_p$ from a free $E$-equivalence class of cardinality $k'_{i_j}$. We also guarantee that for any other free $n_q$ non-$E$-equivalent to any $n_{i_j}$ for $j \in J$ but satisfying $n_p E n_q$, $\alpha^g_q$ is taken from the same $E$-class as $n_p$. Now, if free $n_p$ is $E$-equivalent to $n_{i_j}$ for some $j \in J$, then we take $\alpha^g_p$ sitting on the $E$-class $g(j)$. We see that such $\vec{\alpha}^g$ can be selected in the above way and satisfy $\vec{n} \sim^{\sigma}_{\tau_n} \vec{\alpha}^g$. One can easily observe that there are only finitely many such functions $g$ and thus only finitely many corresponding $\vec{\alpha}^g$s. We also define $\vec{\alpha}^{\emptyset}$ which we associate with $J = \emptyset$: we simply take free $\alpha^{\emptyset}_p$ from an $E$-class of cardinality $k'_{i_j}$, if $n_p E n_{i_j}$. Again, such $\vec{\alpha}^{\emptyset}$ can be selected as advised and satisfy $\vec{n} \sim^{\sigma}_{\tau_n} \vec{\alpha}^{\emptyset}$. We note that when selecting non-free $\alpha^h_p$ or $\alpha^{\emptyset}_p$, it must be $\tau_n(n_p)$ if it is non-fresh, and it must sit on the same class as $n_p$, if it is orbital.

Observe that given any $\vec{m} \in [\vec{n}]_{\sim_c} - B_{[\vec{n}]_{\sim_c}}$ we have two possibilities. First is that $\vec{m} \in H_{\emptyset}$, where $H_{\emptyset} := \{\vec{m} : \vec{m} \sim_c \vec{n} \wedge \forall^l_{j=1} m_{i_j} \notin \bigcup C_{i_j}\}$ and in that case each $\vec{m} \in H_{\emptyset}$ satisfies $\vec{m} \sim^{\sigma}_{\tau_n} \vec{\alpha}_{\emptyset}$. The second possibility is that there exists $J \subseteq \{1, \ldots, l\}$, non-empty $\mathcal{B}^{\vec{n}}_J$ and $g \in \mathcal{B}^{\vec{n}}_J$ such that $\vec{m} \sim^{\sigma}_{\tau_n} \vec{\alpha}_g$. We always choose maximal such $J$ in the sense that if we see that $m_{i_j}$ belongs to one of $i_j$-exceptions, $j$ is added to $J$. Moreover, we can see that the set $H_{\emptyset}$ and sets $H_g = \{\vec{m} : \vec{m} \sim^{\sigma}_{\tau_n} \vec{\alpha}^g \wedge m_{i_j} \in g(j),\ \text{for } j \in dom(g)\}$ are definable by a quantifier-free formulae with parameters.

Notice that $[\vec{n}]_{\sim_c}$ without any free elements is finite (by the definition of $\sim_c$ and the fact that all classes of $E$ are finite). In that case, $f \restriction [\vec{n}]_{\sim_c}$ is trivially definable by a quantifier-free formula.

Let $[\vec{n}]_{\sim_c}$ be such that $\vec{n}$ has some free elements. Let $\vec{m} \in [\vec{n}]_{\sim_c} - B_{[\vec{n}]_{\sim_c}}$ and choose $\vec{\alpha}^g$ accordingly. Recall that $\Phi^{E^\sigma}_n$ is total so let $\beta = \Phi^{E^\sigma}_n(\vec{\alpha}^g)$. By the negative answer to Q2, $\beta$ is not free or $\beta \in \vec{\alpha}^g$.

Suppose $\beta \in \vec{\alpha}^g$. Choose $i$ such that $\Phi^{E^\sigma}_n(\vec{\alpha}^g) = \alpha^g_i$. Observe that by the negative answer to Q3 the following holds: $\forall \vec{x} \sim^{\sigma}_{\tau_n} \vec{\alpha}^g [\Phi^{E^\sigma}_n(\vec{\alpha}^g) \downarrow =: \alpha^g_i \wedge f(\vec{x}) \neq x_i \implies \neg\Gamma(\vec{\alpha}^g, \vec{x}, \tau_n)]$. We are interested in $\vec{x}$ such that $\vec{x} \in H_g$.

Let us unpack $\neg\Gamma(\vec{\alpha}^g, \vec{x}, \tau_n)$:

$$\exists \alpha^g_j \in \vec{\alpha}^g((\forall \gamma \in \tau_n \neg \alpha^g_j E^\sigma \gamma) \wedge |[\alpha^g_j]_{E^\sigma}| > |[x_j]_E|). \tag{$\neg\Gamma$}$$

Can it be the case that $f(\vec{x}) \neq x_i$? Suppose, towards a contradiction, that $f(\vec{x}) \neq x_i$. But then ($\neg\Gamma$) holds. We show that it cannot be the case. For let $x_r$ by any free element of $\vec{x}$. If $x_r E x_{i_j}$ for some $j \in dom(g)$, then $|[\alpha^g_r]| = |[x_r]|$ because, by the definition of $H_g$, $\alpha^g_r$ sits on the class $g(j)$ and this is precisely the class on which $x_r$ sits. So we cannot have $|[\alpha^g_r]| > |[x_r]|$ for such $x_r$. The remaining case is when $x_r$ is not $E$-equivalent to any $x_{i_j}$ with $j \in dom(g)$. Since $J$ is maximal (see one of the paragraphs above), $x_r$ does not come from any $i_j$-exception satisfying $x_r E x_{i_j}$. Therefore, $|[x_r]| \geq k'_{i_j}$. But, by the construction of $\vec{\alpha}^g$, $\alpha^g_r$ sits on an $E$-equivalence class of cardinality $k'_{i_j}$. Hence, again, we cannot have $|[\alpha^g_r]| > |[x_r]|$. We have arrived at a contradiction. Therefore, $f \restriction H_g$ is a projection.

The remaining case is that $\beta \notin \vec{\alpha}^g$. By the negative answer to Q2, $\beta$ is not free. If $\beta \in \tau_n$, then by the negative answer to Q1, every $\vec{x} \sim^{\sigma}_{\tau_n} \vec{\alpha}^g$ satisfies $f(\vec{x}) = \beta_\sigma$ (otherwise, we would have $\neg\Gamma$ which is impossible by similar argument as above). Hence, $f \upharpoonright H_g$ is constant.

Finally, we consider the case when $\beta \notin \vec{\alpha}^g$, $\beta$ is not free and $\beta \notin \tau_n$. Hence, $\beta$ is an orbital. By the negative answer to Q5, $\forall \vec{x} \sim^{\sigma}_{\tau_n} \vec{\alpha}^g[\Phi_n^{E^\sigma}(\vec{\alpha}^g) = \beta \wedge (\neg f(\vec{x})E\beta_\sigma \vee \exists i(f(\vec{x}) = x_i \wedge x_i E\beta_\sigma)) \Longrightarrow \neg\Gamma(\vec{\alpha}^g, \vec{x}, \tau_n)]$. We cannot have $\neg f(\vec{x})E\beta_\sigma \vee \exists i(f(\vec{x}) = x_i \wedge x_i E\beta_\sigma$, because then $\neg\Gamma(\vec{\alpha}^g, \vec{x}, \tau_n)$ and we obtain a contradiction as before. Therefore, $\neg(\neg f(\vec{x})E\beta_\sigma \vee \exists i(f(\vec{x}) = x_i \wedge x_i E\beta_\sigma)$ which is equivalent to $f(\vec{x})E\beta_\sigma \wedge \forall i(x_i E\beta_\sigma \Longrightarrow f(\vec{x}) \neq x_i)$. Now, we use the negative answer to Q4. It follows that $|[\beta] - \tau_n| \leq |\{\text{fresh } \alpha_i^g : \alpha_i^g E\beta\}| + 1$ which means that the number of orbitals $E$-equivalent to $\beta$ is precisely equal to the number of such orbitals in $\vec{\alpha}^g$ plus one. Note that $f(\vec{x})$ is uniquely determined. It follows that $f \upharpoonright H_g$ is an $E$-projection, because for some $e \leq c$, $f(\vec{x}) \in [e]_E$, $f(\vec{x}) \notin \vec{x}$ and $|[e]_E \cap (c, \infty)| = |\{x_i : x_i Ee \wedge x_i > c\}| + 1$.

Now, we shall put everything together to show that $f$ is definable $(\omega, E)$ by a quantifier-free formula. First, recall that by Lemma 1 there are only finitely many classes of the form $[\vec{n}]_{\sim_c}$ and let $[\vec{n}^{(1)}]_{\sim_c}, [\vec{n}^{(2)}]_{\sim_c}, \ldots, [\vec{n}^{(k)}]_{\sim_c}$ be all of them. Each class $[\vec{n}^{(i)}]_{\sim_c}$ can be defined by some quantifier-free formula $\alpha_i$. We look at the behavior of $f \upharpoonright [\vec{n}^{(i)}]_{\sim_c}$.

Let $[1, k] = O \cup F$, where $O \cap F = \emptyset$ and $O$ consists of precisely all $i \in [1, k]$ such that $\vec{n}^{(i)}$ has no free elements.

If $i \in O$, then we have already seen that $f \upharpoonright [\vec{n}^{(i)}]_{\sim_c}$ is definable by a quantifier free formula, say $\psi_i$.

Now, we consider $i \in F$. Obviously, we have $([\vec{n}^{(i)}]_{\sim_c} - B_{[\vec{n}^{(i)}]_{\sim_c}}) \cup B_{[\vec{n}^{(i)}]_{\sim_c}}$. Recall that $B_{[\vec{n}^{(i)}]_{\sim_c}}$ is finite, hence definable by a quantifier-free formula, say $\beta_i$. Also, $f \upharpoonright B_{[\vec{n}^{(i)}]_{\sim_c}}$ is definable by a quantifier-free formula, say $\beta_i'$. Now, we look at $f \upharpoonright ([\vec{n}^{(i)}]_{\sim_c} - B_{[\vec{n}^{(i)}]_{\sim_c}})$. Notice that the set $[\vec{n}^{(i)}]_{\sim_c} - B_{[\vec{n}^{(i)}]_{\sim_c}}$ is a finite disjoint union of $H_\emptyset$ and sets $H_g$ for $g \in \bigcup_{J \in 2^{[1,l]}} \mathcal{B}_J^{\vec{n}^i}$ which is finite. Note that $H_\emptyset$ and sets $H_g$ are definable by quantifier-free formulae (this is obvious by looking at how these sets are defined), say $\theta_\emptyset, \theta_g$, for $g \in \bigcup_{J \in 2^{[1,l]}} \mathcal{B}_J^{\vec{n}^i}$. We have shown that each $f \upharpoonright H_g$, for $g \in \{\emptyset\} \cup \bigcup_{J \in 2^{[1,l]}} \mathcal{B}_J^{\vec{n}^i}$, is constant, projection or an $E$-projection. Therefore, by Lemma 4, each such $f \upharpoonright H_g$ is definable by a quantifier-free formula, say $\psi_h$. Now, the overall formula defining $f$ as as follows:

$$\bigwedge_{i \in O}(\alpha_i \Rightarrow \psi_i) \wedge \bigwedge_{i \in F}\{(\alpha_i \wedge \beta_i \Rightarrow \beta_i') \wedge [(\alpha_i \wedge \neg\beta_i) \Rightarrow \bigwedge_{g \in \bigcup_{J \subseteq [1,l]} \mathcal{B}_J^{\vec{n}^i}} (\theta_g \Rightarrow \psi_g)]\}$$

□

This completes the verification.

The following result can be deduced from Theorem 17 (discussed in conclusions). A relatively easy application of the techniques developed for the previous theorem achieves it as well.

**Theorem 5** *Let $f$ be a total function of arbitrary arity. Let $(\omega, E)$ be a computable equivalence structure with finite character. Then the following are equivalent:*

*(1) $f$ is relatively intrinsically computable on $(\omega, E)$,*
*(2) $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters.*

***Proof*** We apply the same construction as in Theorem 4 with two exceptions. First, at the initial stage we start with $\tau_0$ sufficiently large that every $n \geq |\tau_0|$ belongs to an infinite $E$-equivalence class. Second, instead of $\Gamma$ we use a stronger condition:

$$\Gamma'(\vec{\alpha}, \vec{x}, \tau_n) := \forall \alpha_j \in \vec{\alpha}(\alpha_j \notin \tau_n \implies |[\alpha_j]_E| \leq |[x_j]_E|). \tag{$\Gamma'$}$$

Its negation, used in verification, is as follows:

$$\exists \alpha_j \in \vec{\alpha}(\alpha_j \notin \tau_n \wedge |[\alpha_j]_{E^\sigma}| > |[x_j]_E|). \tag{$\neg\Gamma'$}$$

Lemma 5 remains unchanged. The proof of Lemma 6 is simpler. The first two paragraphs of its proof remain the same and we start from there.

We want to show that $f \restriction [\vec{n}]_{\sim_c}$, where $\vec{n}$ is fresh, is definable by a quantifier-free formula with parameters.

We take $\vec{\alpha}$ such that $\vec{n} \sim^\sigma_{\tau_n} \vec{\alpha}$.

Suppose that $\Phi_n^{E^\sigma}(\vec{\alpha}) = \alpha_i$. We use the negative answer to Q3: $\forall \vec{x} \sim^\sigma_{\tau_n} \vec{\alpha}[\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow =: \alpha_i \wedge f(\vec{x}) \neq x_i \implies \neg\Gamma'(\vec{\alpha}, \vec{x}, \tau_n)]$. We claim that $f(\vec{x}) = x_i$ for every such $\vec{x}$. For suppose it is not the case. Then we have $\neg\Gamma'$. But this is not possible, because each $\alpha_p \notin \tau_n$, as well as corresponding $x_p$, sits on an infinite $E$-equivalence class. Therefore, we cannot have $|[\alpha_j]_{E^\sigma}| > |[x_j]_E|$. Hence, $f \restriction [\vec{n}]_{\sim_c}$ is a projection.

Suppose $\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow = \beta \notin \vec{\alpha}$. By the negative answer to Q2, $\beta$ is not free.

We show that $\beta$ must occur in $\tau_n$. Assume otherwise. Hence, $\beta$ is an orbital. By the negative answer to Q4: $\forall$ orbital $\beta \notin \vec{\alpha}[\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow =: \beta \implies |[\beta] - \tau_n| \leq |\{\text{fresh } \alpha_i : \alpha_i E \beta\}| + 1]$. Therefore, $|[\beta] - \tau_n| \leq |\{\text{fresh } \alpha_i : \alpha_i E \beta\}| + 1]$ which is impossible because $|\{\text{fresh } \alpha_i : \alpha_i E \beta\}| + 1] < \infty$ while $\beta$ sits on an infinite equivalence class.

We work with $\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow = \beta \notin \vec{\alpha}$ such that $\beta \in \tau_n$. We use the negative answer to Q1: $\forall \vec{x} \sim^\sigma_{\tau_n} \vec{\alpha}[\Phi_n^{E^\sigma}(\vec{\alpha}) \downarrow =: \beta, \beta \in \tau_n, \beta_\sigma \neq f(\vec{x}) \implies \neg\Gamma'(\vec{\alpha}, \vec{x}, \tau_n)]$. However, $\neg\Gamma'$ cannot hold for the same reason as before. Therefore, $\beta_\sigma = f(\vec{x})$. Hence, $f \restriction [\vec{n}]_{\sim_c}$ is constant.

Application of Lemma 4 finishes the proof. □

**Corollary 1** *A total function is intrinsically computable on $\omega$ iff it is definable in $\omega$ by a quantifier-free formula with parameters.*

***Proof*** Take $(\omega, E)$ where $E$ is a trivial computable equivalence relation $E = \omega^2$. For such $E$, the term 'relatively' can be safely omitted in Theorem 5 because $E$ (and $=$) is always computable regardless of isomorphism. Hence, by Theorem 5, $f$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters. Clearly, any atomic formula including the symbol $E$ can be easily eliminated, and we are left with a formula in the empty language. □

## 4 Learnability and other types of functions

Consider the following question: does replacing the notion of computability in every notation by learnability in every notation gives us more functions? In other words, does the class of relatively intrinsically learnable functions extend the class of relatively intrinsically computable ones? A natural notion of relative intrinsic learnability could be as follows: $R$ is relatively intrinsically learnable on a computable structure $\mathcal{A}$ if in all copies $\mathcal{B}$ of $\mathcal{A}$, the image of $R$ is learnable in $\mathcal{B}$, i.e. $\Delta^0_2(\mathcal{B})$.

The answer to the above question depends on the underlying structure. Here, we will focus on computable equivalence structrue $(\omega, E)$ where $E$ has finite character. By the results of Ash et al. [1] (see, also, Theorem 10.1 in [3]), if $R$ is relatively intrinsically learnable on $(\omega, E)$ then it is definable in $(\omega, E)$ by a computable $\Sigma_2$ formula (see, Chapter 7 in [3]). The theory of this structure (with countably many constants naming each element from the universe) has quantifier elimination. Hence, a computable $\Sigma_2$ formula which defines $R$ in $(\omega, E)$ can be transformed (using quantifier elimination and contraction of countable disjuctions/conjuctions of quantifier-free formulae to finitary quantifier-free formulae) to a quantifier-free formula. Hence, $R$ is definable in $(\omega, E)$ by a quantifier-free formula with parameters. Therefore, by Theorem 5, on such $(\omega, E)$, every relatively intrinsically learnable relation is also relatively intrinsically computable.

Previous section partially characterizes relative intrinsic computability of total functions (and, by an obvious extension, for relations)[3] over computable equivalence structures. In this section we also consider related questions about partial functions, vector-valued functions and functions of non-fixed arity.

**Proposition 1** *Let $(\omega, E)$ be a computable equivalence structure and let the character of $E$ be finite. The class of partial functions relatively intrinsically computable on $(\omega, E)$ is the class of partial functions definable $(\omega, E)$ by quantifier-free formula with parameters.*

***Proof*** The proof of ($\Leftarrow$) is easy—consider the program based on a quantifier-free formula that defines $f$.

To prove the left-to-right implication, let $f$ be a partial function relatively intrinsically computable on $(\omega, E)$. Observe that in every copy $(\omega, E')$ of $(\omega, E)$, the image of $dom(f)$ is c.e. in $E'$ and hence learnable in $E'$. Therefore, $dom(f)$ is relatively intrinsically learnable on $(\omega, E)$. By the paragraph preceding Proposition 1, $dom(f)$ can be defined in $(\omega, E)$ by some quantifier-free $\phi(\vec{x})$. Choose $\alpha_0 \in S$. Observe that, in every copy $(\omega, E')$ of $(\omega, E)$, following function $g$ is computable in $E'$:

$$g(\vec{n}) = \begin{cases} f(\vec{n}) & \text{if } (\omega, E) \models \phi(\vec{x})[\vec{n}], \\ \sigma(\alpha_0) & \text{otherwise.} \end{cases}$$

[3] We note that in injective notations, computing a relation is equivalent to computing its characteristic function. However, this property does not hold anymore in noninjective notations (see Theorem 2.9 in [29]).

$g$ is total. By Theorem 5, choose a quantifier-free formula $\psi(\vec{x}, y)$ which defines $g$ in $(\omega, E)$. The following quantifier-free formula defines $f$ in $(\omega, E)$:

$$[\phi(\vec{x}) \Rightarrow \psi(\vec{x}, y)] \wedge [\neg\phi(\vec{x}) \Rightarrow y \neq y]. \tag{1}$$

□

**Corollary 2** *A partial function is intrinsically computable on $\omega$ iff it is definable in $\omega$ by a quantifier-free formula with parameters.*

We proceed to consider vector-valued functions and functions with non-fixed arity. For any partial function $f : \omega^n \to \omega^m$ we use $f_j$ (for $1 \leq j \leq n$) to denote the projection of the value of $f$ on $j$-th coordinate.

**Theorem 6** *A partial function $f : \omega^n \to \omega^m$ is computable in every injective notation iff $f$ is definable in $\omega$ by a quantifier-free formula with parameters.*

***Proof*** It is an easy consequence of Corollary 2 and the fact that, in injective notations, computability of a relation is equivalent to computability of its characteristic function (consider $k = n + m$).

□

Below we show yet another generalisation of Shapiro's theorem. We consider functions $f : \omega^* \to \omega^*$. These are functions whose both arguments and values are finite sequences of natural numbers, of non-fixed arity. Observe that all the functions considered above are special cases of this notion.

To deal with such functions, we need a first-order logic over the same alphabet as described above but with a certain modification. We need to allow some infinite formulae. A formula $\bigvee_{i\in\omega} \varphi_i$, where each $\varphi_i$ is a finite (quantifier-free) formula, is an infinite (quantifier-free) alternative. If for each finite formula $\varphi$ it can be determined whether there is such $i$ that $\varphi = \varphi_i$, then the infinite formula is recursive.

**Theorem 7** *A partial function $f : \omega^* \to \omega^*$ is computable in every injective notation iff it is qf-definable with parameters by a recursive infinite alternative.*

***Proof*** ($\Leftarrow$) For any $a_1, \ldots, a_n$ search for such formula $\varphi_i$ in the infinite alternative and such $b \in \omega$ that $\mathcal{N} \models \varphi(a_1, \ldots, a_n, b)$. When you encounter such $b$, return $f(a_1, \ldots, a_n) = b$.

($\Rightarrow$) Denote $f^{i,j}$ to be $f$ restricted to all the arguments of arity $i$ such that the value of the function is of arity $j$. Observe that if $f$ is computable in every notation, then so is every $f^{i,j}$. By earlier theorem each $f^{i,j}$ is then qf-definable by a certain formula $\varphi_{i,j}$. Then $f$ is qf-definable by $\bigvee_{i,j\in\omega} \varphi_i$.

□

## 5 Generalizations for (not necessarily injective) notations

Many results contained in this part of the article were earlier published in the PhD thesis [29].

**Theorem 8** *The only unary functions computable in every notation are constant and identity functions.*

***Proof*** It is clear that all constant and identity functions are computable in every notation. We need to prove that only these functions are. Suppose $f$ is neither constant nor identity. Thanks to the Theorem 1, we only need to consider the following two cases:

1. $f$ is almost constant but not constant,
2. $f$ is almost identity but not identity.

**Case 1** We construct a notation $(S, \sigma)$ in which $f$ is not computable. Let $S$ be the standard set of numerals. Let $k$ be the value of $f$ for nearly all arguments and $a_0, \ldots, a_m$ be all arguments for which $f$ takes values other than $k$.

Let $A \subseteq \omega$ be a set not computable in the standard notation. We construct $\sigma$ as follows:

$$\sigma(\overline{0}) = k.$$

Let $b_0, b_1, b_2, \ldots$ be an enumeration of all numbers from $A \setminus \{0\}$ and $c_0, c_1, c_2, \ldots$ — an enumeration of all numbers from $(\omega \setminus A) \setminus \{0\}$.

For $i \in \omega$, to each of the numerals $\overline{b_i}$, function $\sigma$ assigns one of numbers $a_0, \ldots, a_m$ and to each of the numerals $\overline{c_i}$ it assigns one of numbers from the set $\omega \setminus \{a_0, \ldots, a_m\}$. It is done in such a way that function $\sigma$ is surjective and for every positive natural number $t$: $\sigma(\overline{t}) \neq k$.

This is possible unless $k$ is the only argument for which $f$ assumes a value different from $k$. We deal with this case later.

Suppose that $f$ is computable in $(S, \sigma)$. We show that, contrary to our assumption, $A$ is computable in the standard notation.

The algorithm provided below only works for $n \neq k$ but that does not need to bother us since the answer for $k$ can be given explicitly.

Let $n \in \omega \setminus \{k\}$. We want to know whether $n \in A$. We calculate $f^\sigma(\overline{n})$. The following are equivalent:

1. $f^\sigma(\overline{n}) \neq \overline{0}$,
2. $f(\sigma(\overline{n})) \neq k$,
3. $\sigma(\overline{n}) \in \{a_0, a_1, \ldots, a_m\}$,
4. $\overline{n} \in \{\overline{b_i} : i \in \omega\}$,
5. $n \in \{b_i : i \in \omega\}$,
6. $n \in A$.

Analogously, we can prove that $n \notin A \Leftrightarrow f^\sigma(\overline{n}) = \overline{0}$ and we have obtained a contradiction.

To complete the proof of the first case, we need to consider $f$ of the following form:

$$f(n) = \begin{cases} k & \text{if } n \neq k, \\ l & \text{if } n = k, \end{cases}$$

where $k \neq l$. This is because in such a case it is not possible to construct $\sigma$ in the way described above.

Let $A \subseteq \omega$ be a set of natural numbers not computable in the standard notation. Let $a_0, a_1, a_2, \ldots$ be an enumeration of all numbers from $A \setminus \{0\}$ and $b_0, b_1, b_2, \ldots$—an enumeration of all numbers from $(\omega \setminus A) \setminus \{0\}$.

We construct a notation $(S, \sigma)$ in which $f$ is not computable. Let $S$ be the standard set of numerals. We construct $\sigma$ as follows:

$$\sigma(\overline{0}) = l,$$
$$\sigma(\overline{a_i}) = k, \text{ for all } i \in \omega,$$

and to all the numerals $\overline{b_0}, \overline{b_1}, \overline{b_2}, \ldots$ we assign all the numbers other than $k$ and $l$.

Suppose that $f$ is computable in $(S, \sigma)$. We show that, contrary to our assumption, $A$ is computable in the standard notation.

The algorithm provided below works only for $n \neq 0$ but that does not need to bother us because we can give the answer for 0 explicitly.

Let $n \in \omega \setminus \{0\}$. We want to know whether $n \in A$. We calculate $f^\sigma(\overline{n})$. The following are equivalent:

1. $f^\sigma(\overline{n}) = \overline{0}$,
2. $f(\sigma(\overline{n})) = l$,
3. $\sigma(\overline{n}) = k$,
4. $\overline{n} \in \{\overline{a_i} : i \in \omega\}$,
5. $n \in \{a_i : i \in \omega\}$,
6. $n \in A$.

Analogously, we can prove that $n \notin A \Leftrightarrow f^\sigma(\overline{n}) \neq \overline{0}$ and we have obtained a contradiction.

**Case 2** Assume that $f$ is almost identity but not identity. Let $A \subseteq \omega$ be a set of natural numbers not computable in the standard notation. Let $a_0, a_1, a_2, \ldots$ be an enumeration of all numbers from $A$ and $b_0, b_1, b_2, \ldots$—an enumeration of all numbers from $\omega \setminus A$.

We construct a notation $(S, \sigma)$ in which $f$ is not computable. $S$ is the standard set of numerals. Let $c_0, c_1, \ldots, c_m$ be natural numbers such that $f(c_i) \neq c_i$ for $i = 0, \ldots, m$ and let them be the only natural numbers with such a property.

Now let us construct $\sigma$. To each of the numerals $\overline{a_i}$, $\sigma$ assigns one of the numbers $c_i$ and to each of the numerals $\overline{b_i}$, $\sigma$ assigns one of the other numbers. This is done in such a way that each natural number is assigned to at least one numeral.

Suppose for the sake of contradiction that $f$ is computable in $(S, \sigma)$. We provide an algorithm for $A$. Let $n \in \omega$. We want to know whether $n \in A$. We calculate $f^\sigma(\overline{n})$. For any $n$, the following conditions are equivalent:

1. $f^\sigma(\overline{n}) \neq \overline{n}$,
2. $\sigma(\overline{n}) \in \{c_0, c_1, \ldots, c_m\}$,
3. $\overline{n} \in \{\overline{a_i} : i \in \omega\}$,
4. $n \in \{a_i : i \in \omega\}$,
5. $n \in A$.

Analogously, for every natural number $n$ the following holds:

$$f^{\sigma}(\overline{n}) = \overline{n} \Leftrightarrow n \notin A.$$

Therefore we have obtained a contradiction.

It follows that the only functions computable in every notation are constant and identity functions.

□

**Definition 9** A function $f : \omega^k \to \omega$ is a projection if there exists $i \in \{1, \ldots, k\}$ such that:

$$\forall x_1 \ldots \forall x_k \; f(x_1, \ldots, x_k) = x_i.$$

**Theorem 9** *The only total functions $f : \omega^k \to \omega$ computable in every notation are constant functions and projections.*

***Proof*** The implication ($\Leftarrow$) is obvious.

We prove the implication ($\Rightarrow$) by induction over $k$. The previous theorem constitutes the base case for this induction. Now suppose that the only functions of $k$ arguments computable in every notation are constant functions and projections. Let $f : \omega^{k+1} \to \omega$ be a function computable in every notation. We want to show that it is either constant or a projection. In this proof we utilise Lemmas 7, 8, 9 and 10 , included below.

For every $1 \leq i \leq k+1$ and every $j \in \omega$, let us define a function:

$$f_{i,j} : \omega^k \to \omega$$

such that for all $x_1 \ldots, x_{i-1}, x_{i+1}, \ldots, x_{k+1}$:

$$f_{i,j}(x_1 \ldots, x_{i-1}, x_{i+1}, \ldots, x_{k+1}) = f(x_1 \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_{k+1}),$$

i.e. a function obtained by substituting the value $j$ for the variable $x_i$ in $f$.

All functions $f_{i,j}$ are computable in every notation because they are obtained by substituting a value for a variable in $f$, and $f$ is computable in every notations. Therefore, by inductive assumption, each of them is either constant or a projection.

We want to show that $f$ is either a constant function or a projection. We have two cases to consider.

**Case 1** Suppose that among all functions $f_{i,j}$ there is at least one projection $f_{i_0,j_0} = x_l$. Then by Lemmas 7 and 8 , every function $f_{i,j}$ is also a projection on the same coordinate $x_l$, unless $i = l$. Hence, for any $i \neq l$ and any $x_1, \ldots, x_{k+1}$:

$$f(x_1, \ldots, x_{k+1}) = f_{i,x_i}(x_1 \ldots, x_{i-1}, x_{i+1}, \ldots, x_{k+1}) = x_l.$$

Therefore, $f$ is a projection on $x_l$.

**Case 2** Suppose that all functions $f_{i,j}$ are constant. Then by Lemmas 9 and 10 all these functions are identical and always equal to the same value $c$. Hence $f$ is also constant and equal to $c$. □

Note that functions $f_{i,j}$ mentioned in subsequent lemmas are those defined in the proof of Theorem 9.

**Lemma 7** *If $f_{i_1,j_1}$ is a projection on $x_l$ and $i_2 \neq l$ , then $f_{i_2,j_2}$ is also a projection.*

***Proof*** Suppose to the contrary that the function $f_{i_2,j_2}$ is not a projection, i.e. it is constant. Assume that for all arguments:

$$f_{i_1,j_1}(x_1,\dots,x_{k+1}) = x_l$$

and

$$f_{i_2,j_2}(x_1,\dots,x_{k+1}) = c.$$

Let us consider the following cases:

**Case 1** Suppose that $i_1 \neq i_2$. We know that for every sequence of arguments:

$$f_{i_1,j_1}(x_1,\dots,x_{i_1-1},x_{i_1+1},\dots,x_{i_2-1},x_{i_2},x_{i_2+1},\dots,x_{k+1}) = x_l.$$

In particular, for $x_{i_2} = j_2$:

$$f_{i_1,j_1}(x_1,\dots,x_{i_1-1},x_{i_1+1},\dots,x_{i_2-1},j_2,x_{i_2+1},\dots,x_{k+1}) = x_l.$$

But this is also the value of $f_{i_2,j_2}$, with $j_1$ substituted for $x_{i_1}$. This is however a contradiction since this is always equal to $x_l$, and we assumed that $f_{i_2,j_2}$ is constant. Therefore, $f_{i_2,j_2}$ must be a projection.

**Case 2** Suppose that $i_1 = i_2$. If $j_1 = j_2$, then it is trivial. Hence suppose that $j_1 \neq j_2$.

Let $A \subseteq \omega$ be uncomputable. We construct a notation $(S,\sigma)$. Let $S$ be the standard set of decimal numerals and let

$$\sigma(\overline{2n}) = \begin{cases} c & \text{if } n = 0, \\ j_1 & \text{if } n > 0 \wedge n \in A, \\ j_2 & \text{if } n > 0 \wedge n \notin A. \end{cases}$$

Assign the remaining numbers to numerals of the form $\overline{2n+1}$ in any injective way.

To obtain a contradiction, we want to construct an algorithm which decides whether $n \in A$. The answer for $n = 0$ is given explicitly as a special case. Assume $n > 0$. Since $f$ is computable in every notation, we compute the value of $f^\sigma$, where we substitute the numeral $\overline{2n}$ for $x_{i_1}$ (which is the same variable as $x_{i_2}$), the numeral $\overline{1}$ for $x_l$, and for other variables we substitute any numerals.

Due to the construction of $\sigma$ and because $n > 0$, the numeral substituted for $x_{i_1}$ represents either $j_1$ or $j_2$. If it represents $j_1$, then $f$ is a projection on $x_l$ and it has to return a numeral which represents the same number as the numeral $\overline{1}$; hence it has to return the numeral $\overline{1}$, since no other numeral represents the same number. If, on the other hand, the numeral substituted for $x_{i_1}$ represents $j_2$, then $f$ is a constant function always equal to $c$ and in this case the algorithm returns the numeral $\overline{0}$.

Therefore, if the algorithm returns $\overline{1}$, $n \in A$. If it returns $\overline{0}$, $n \notin A$. □

**Lemma 8** *If $f_{i_1,j_1}$ and $f_{i_2,j_2}$ are both projections, then they are projections on the same coordinate.*

***Proof*** Suppose that $f_{i_1,j_1} = x_{l_1}$ and $f_{i_2,j_2} = x_{l_2}$ are different projections, i.e. $l_1 \neq l_2$. Let us consider the following cases:

**Case 1** Suppose that $i_1 \neq i_2$. We know that for every sequence of arguments:

$$f_{i_1,j_1}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2}, x_{i_2+1}, \ldots, x_{k+1}) = x_{l_1}.$$

In particular, for $x_{i_2} = j_2$:

$$f_{i_1,j_1}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, j_2, x_{i_2+1}, \ldots, x_{k+1}) = x_{l_1}.$$

But this is also equal to $f_{i_2,j_2}$, with $j_1$ substituted for $x_{i_1}$, hence it is always equal to $x_{l_2}$. This is a contradiction since we can substitute different values for $x_{l_1}$ and $x_{l_1}$.

**Case 2** Suppose that $i_1 = i_2$. If $j_1 = j_2$, then it is trivial. Hence suppose that $j_1 \neq j_2$. Let $A \subseteq \omega$ be a set uncomputable in the standard notation. We construct a notation $(S, \sigma)$. Let $S$ be the standard set of numerals and let:

$$\sigma(\overline{2n}) = \begin{cases} j_1 & \text{if } n \in A, \\ j_2 & \text{if } n \notin A. \end{cases}$$

Assign the rest of numbers to the remaining numerals in any injective way.

To obtain a contradiction, we construct an algorithm which decides whether $n \in A$. Since $f$ is computable in every notation, we are going to compute $f^\sigma$, with $\overline{2n}$ substituted for $x_{i_1}$ (which is equal to $x_{i_2}$), $\overline{1}$ — for $x_{l_1}$ and $\overline{3}$ — for $x_{l_2}$. If the output numeral is $\overline{1}$, then the algorithm has computed projection on coordinate $x_{l_1}$. Then $\sigma(\overline{2n}) = j_1$ and $n \in A$. Analogously, if the output numeral is $\overline{3}$, then $n \notin A$. Hence $A$ is computable in the standard notation and we have obtained a contradiction.

□

**Lemma 9** *If $i_1 \neq i_2$ and functions $f_{i_1,j_1} = c_1$ and $f_{i_2,j_2} = c_2$ are constant, then $c_1 = c_2$.*

***Proof*** Let these functions be constant and assume values, respectively, $c_1$ and $c_2$. Without loss of generality assume that $i_1 < i_2$. We show that $c_1 = c_2$. Then for any $x_1, \ldots, x_{k+1}$:

$$\begin{aligned} c_1 &= f_{i_1,j_1}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, j_2, x_{i_2+1}, \ldots, x_{k+1}) \\ &= f(x_1, \ldots, x_{i_1-1}, j_1, x_{i_1+1}, \ldots, x_{i_2-1}, j_2, x_{i_2+1}, \ldots, x_{k+1}) \\ &= f_{i_2,j_2}(x_1, \ldots, x_{i_1-1}, j_1, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2+1}, \ldots, x_{k+1}) \\ &= c_2. \end{aligned}$$

□

**Lemma 10** *If $f_{i_0,j_0} = c$ is constant, then:*

1. *$f_{i_0,j} = c$, for every $j$, if all the functions $f_{i,j}$ are constant,*
2. *$f_{i_0,j} = j$, for every $j$, if at least one function $f_{i,j}$ is a projection.*

***Proof*** First note that if $f_{i_0,j_0} = c$ is constant, then $f_{i_0,j}$ must be constant for every $j$. Otherwise $f_{i_0,j}$ would be a projection, for some $j$, and then, by Lemma 7, $f_{i_0,j_0}$ would also be a projection. That would be a contradiction because $f_{i_0,j_0}$ is constant.

Suppose that all the functions $f_{i,j}$ are constant. Consider the function $f_{i_1,j_1}$ such that $i_1 \neq i_0$. By Lemma 9, it is also equal to $c$. Then, for any $j$ we can again apply Lemma 9 to $f_{i_1,j_1}$ and $f_{i_0,j}$ and we conclude that $f_{i_0,j} = c$, for every $j$.

Now suppose that the function $f_{i_1,j_1}$ is a projection on $x_l$. Then, by Lemmas 7 and 8 , all functions $f_{i,j}$ are projections on $x_l$ unless $i = l$. Since all functions $f_{i_0,j}$ are constant, it follows that $l = i_0$. Then for all $i \neq i_0$ and all $j$, functions $f_{i,j}$ are projections on $x_{i_0}$, and for all $j$, functions $f_{i_0,j}$ are constant and equal to $j$.

□

**Definition 10** Let the notation $(S, \sigma)$ of $\omega$ be defined as follows:

The alphabet $\Sigma$ consists of standard digits $\overline{0}, \ldots, \overline{9}$, brackets (, ) and a comma. The set of numerals $S$ consists of all inscriptions of the form $(\overline{a}, \overline{b})$, where $\overline{a}, \overline{b}$ are standard numerals.

Let $B \subseteq \omega$ be such that neither $B$ nor $\omega \setminus B$ is c.e. in the standard notation. We define $\sigma$ as follows:

$$\sigma((\overline{a}, \overline{b})) = \begin{cases} a & \text{if } b \notin B, \\ a+1 & \text{if } b \in B. \end{cases}$$

**Lemma 11** $(S, \sigma)$ *defined as above is a correct notation for $\omega$.*

***Proof*** The only condition that might not be obvious is that for every natural number $n$ there is a numeral $(\overline{a}, \overline{b}) \in S$ representing $n$. Let $n \in \omega$. Since $B$ is not c.e., it follows that $B \neq \omega$. Let $b \in \omega \setminus B$. Then $\sigma((\overline{n}, \overline{b})) = n$.

□

**Lemma 12** *Let $A \subseteq \omega$. Then $A$ is c.e. in $(S, \sigma)$ if and only if $A = \emptyset$ or $A = \omega$.*

***Proof*** The implication ($\Leftarrow$) is obvious. To prove ($\Rightarrow$), we show that if $A$ is neither $\emptyset$ nor $\omega$, then it is not c.e. in $(S, \sigma)$. Then there is such $n$ that either $n$ or $n+1$ is in $A$ but not both of them.

Suppose to the contrary that $A$ is c.e. in $(S, \sigma)$. If $n \in A$ and $n+1 \notin A$, then we enumerate all the elements of $A$ and whenever we reach an element $(\overline{n}, \overline{a})$, we know that it represents number $n$ and hence $a \notin B$. Hence $\omega \setminus B$ is c.e. and this is a contradiction.

If $n+1 \in A$ and $n+1 \notin A$, then we enumerate the elements of $A$ as above and whenever we reach $(\overline{n}, \overline{a})$, we know that it represents number $n+1$ and hence $a \in B$. This means that $B$ is c.e. and we obtain a contradiction.

□

**Lemma 13** *Let $R \subseteq \omega^k$. Then $R$ is c.e. in a notation $(S, \sigma)$ if and only if $R = \emptyset$ or $R = \omega^k$.*

***Proof*** The implication ($\Leftarrow$) is obvious. To prove ($\Rightarrow$), suppose that $R \neq \emptyset$, $R \neq \omega^k$ and that $R$ is c.e. in $(S, \sigma)$.

For any $(n_1, \dots, n_k)$, $(n'_1, \dots, n'_k) \in \omega^k$, we call them neighbouring elements if they differ only on one coordinate, and on this coordinate they differ only by 1, i.e. if there is $1 \leq i \leq k$ such that $n_i = n'_i + 1$ or $n'_i = n_i + 1$ and for all $1 \leq j \leq k$, if $j \neq i$, then $n_j = n'_j$.

If $R \neq \emptyset$ and $R \neq \omega^k$, then there must obviously exist $(n_1, \dots, n_k)$ and $(n'_1, \dots, n'_k)$—two neighbouring elements of $\omega^k$ such that $(n_1, \dots, n_k) \in R$ and $(n'_1, \dots, n'_k) \notin R$. Without loss of generality we can assume that $n_1 = n'_1 + 1$, and that $n_j = n'_j$, for $1 < j \leq k$. Let us fix $n_2, \dots, n_k$.

Let $C = \{a \in \omega : (a, n_2, \dots, n_k) \in R\}$. Since $C$ is neither $\emptyset$, nor $\omega$, it follows from Lemma 12 that $C$ is not c.e. in $(S, \sigma)$. Then $R$ is not c.e. in $(S, \sigma)$ either. Thus we have obtained a contradiction. Therefore the only relations c.e. in $(S, \sigma)$ are $\emptyset$ and $\omega^k$. □

**Theorem 10** *The only relations on natural numbers whose characteristic functions are c.e. in every notation are $\emptyset$ and $\omega^k$, for $k \in \omega$.*

**Corollary 3** *The only relations on natural numbers whose characteristic functions are computable in every notation are $\emptyset$ and $\omega^k$, for $k \in \omega$.*

**Lemma 14** *If a nonempty partial function is computable in every notation, then it is total.*

***Proof*** Suppose that such a function is not total. Then its domain is neither $\emptyset$ nor $\omega$ and hence is not c.e. in $(S, \sigma)$ as described above. But if a function is computable in a given notation, then its domain is c.e. in it. Hence this function is not computable in $(S, \sigma)$. □

**Theorem 11** *The only nonempty partial functions computable in every notation are constant functions and projections.*

**Definition 11** $R \subseteq \omega^k$ is qf-definable in terms of relations $S_1, \dots S_m$ (possibly infinitely many) if $R$ is definable by a quantifier-free formula in the first order logic (without $=$) in which $S_1, \dots, S_m$ are the only non-logical symbols.

**Theorem 12** *If $E \subseteq \omega^2$ is an equivalence relation and $R \subseteq \omega$, then $R$ is computable in every notation in which $E$ is iff $R$ is qf-definable in terms of $E$.*

***Proof*** The implication ($\Leftarrow$) is straightfoward. We wish to prove ($\Rightarrow$). Suppose that $R$ is not qf-definable in terms of $E$ (by a finite formula).

First consider the case when $E$ is not qf-definable by an infinite formula either. Enumerate all equivalence classes of $E$ as $P_0, P_1, \dots$ (possibly finitely many). Observe that $R$ is not a Boolean combination (finite or even infinite) of equivalence classes of $E$. We wish to construct such $(S, \sigma)$ that $E$ is computable in $(S, \sigma)$ but $R$ is not.

We can take $S$ to be the standard set of decimal numerals. We divide $S = \bigcup_{i \in I} S_i$, where $I$ is the set of indices of sets $P_i$ and all $S_i$ are all infinite, pairwise disjoint and uniformly computable. Each $S_i$ is going to denote numbers from the equivalence class $P_i$.

Since $R$ is not a Boolean combination of all $P_i$, then there is such $j$ and such $a, b \in P_j$, that $a \in R$ and $b \notin R$. We can assume without loss of generality that there are also some other elements in $P_j$. Then divide $S_j = S_j^1 \cup S_j^2 \cup S_j^3$ (all these sets being infinite and pairwise disjoint, $S_j^1$ and $S_j^2$ noncomputable and $S_j^3$ computable). Construct $\sigma$ in such a way that numerals from $S_j^1$ all denote $a$, numerals from $S_j^2$ denote $b$ and numerals from $S_j^3$ denote $P_j \setminus \{a, b\}$.

Observe that $E$ is computable in $(S, \sigma)$ but $R$ is not.

Now consider the case when $R$ is an infinite (but not finite) Boolean combination of equivalence classes. We are going to use a modified version of the above argument. Divide $S$ into infinitely many infinite, pairwise disjoint, uniformly computable sets $S_i$. Each of these sets is going to denote the equivalence class $P_i$. However, at this point we still have not established the precise enumeration of classes - we wish to do it later. We wish to construct $(S, \sigma)$ in which $E$ is computable but $R$ is not.

Observe that $E$ is computable in $(S, \sigma)$. Now we wish to ensure that $R$ is not. Since $R$ is an infinite but not a finite combination of classes $P_i$, there are infinitely many classes contained in $R$ (call them $P_1^1, P_2^1, \ldots$) and infinitely many not contained (call them $P_1^2, P_2^2, \ldots$). To ensure that $R$ is not computable in $(S, \sigma)$, take a noncomputable set of indices $W \subseteq \omega$. Classes $P_i^1$ are going to be denoted by sets of numerals $S_i$ where $i \in W$, the others by those where $i \notin W$. Hence $E$ is computable in $(S, \sigma)$ but $R$ is not.

$\square$

**Theorem 13** *If $E \subseteq \omega^2$ is an equivalence relation, $R \subseteq \omega^n$ and $R$ is computable in every notation in which $E$ is, then whenever $(x_1, \ldots, x_n) \in R$ and $(y_1, \ldots, y_n) \notin R$, it follows that there is such $i$ that $\neg x_i E y_i$.*

***Proof*** Suppose that $(x_1, \ldots, x_n) \in R$ and $(y_1, \ldots, y_n) \notin R$ but for all $i \leq n$, $x_i E y_i$. We call equivalence classes of $E$ as $P_0, P_1, \ldots$ (possibly finitely many). We wish to construct $(S, \sigma)$ in which $E$ is computable but $R$ is not.

Observe that if there are such $(x_1, \ldots, x_n) \in R$ and $(y_1, \ldots, y_n) \notin R$ that $x_i E y_i$ for each $i$, then there are such $(x_i', \ldots, x_n') \in R$ and $(y_1', \ldots, y_n') \notin R$ that $x_i' E y_i'$ for each $i$ and there is a unique $j$ such that $x_j' \neq y_j'$.

Suppose to the contrary this is not the case, hence whenever $(x_1, \ldots x_n) \in R$, $(y_1, \ldots, y_n) \notin R$ and $x_i E y_i$ for all $i$, then they differ on multiple positions. Suppose that $k$ is the least number of positions on which each such pairs of tuples differ. We can assume without loss of generality that $x_i \neq y_i$ for $i \leq k$ and $x_i = y_i$ otherwise.

Then consider the tuple $(x_1, \ldots x_{k-1}, y_k, x_{k+1}, \ldots x_n)$. Since it differs from $(x_1, \ldots, x_n)$ on less than $k$ positions (and all the necessary elements are equivalent), it belongs to $R$. However, it also differs from $(y_1, \ldots, y_n)$ on less than $k$ positions, hence it does not belong to $R$. This is a contradiction.

Now consider $(x_1, \ldots x_n)$ and $(y_1, \ldots, y_n)$ fixed at the beginning of the proof. We can assume that they differ on a unique position $j$. Also, assume without loss of generality that $j = n$. Suppose that $x_1, \ldots x_{n-1}$ are denoted by numerals $\alpha_1, \ldots, \alpha_{n-1}$ and that both $x_n$ and $y_n$ belong to $P_k$. We use decimal numerals and we divide the set of numerals $S = \bigcup_{i \in I} S_i$ like in the previous proof. Each $S_i$ is going to denote numerals from $P_i$.

We divide $S_k = S_k^1 \cup S_k^2 \cup S_k^3$ similarly to the previous proof. The numerals from $S_k^1$ are going to denote $x_n$, the numerals from $S_k^2$ are going to denote $y_n$ and from $S_k^3$ - the remaining numbers from $P_k$. Recall that $S_k^1$ and $S_k^2$ are noncomputable. If $R$ was computable in $(S, \sigma)$, then for any $\beta \in S_k$, by asking whether $R^\sigma(\alpha_1, \ldots, \alpha_{n-1}, \beta)$, we would be able to determine $\sigma(\beta)$ belongs to which $S_k^j$ (assuming that $\sigma(\beta) \notin S_k^3$ but this condition is decidable).

Hence $E$ is computable in $(S, \sigma)$ but $R$ is not.

□

**Theorem 14** *If $E \subseteq \omega^2$ is an equivalence relation with finitely many equivalence classes and $R \subseteq \omega^n$, then $R$ is computable in every notation in which $E$ is iff $R$ is qf-definable in terms of $E$.*

***Proof*** The implication ($\Leftarrow$) is straightforward. Now we want to show ($\Rightarrow$). Suppose that $R$ is not qf-definable in terms of $E$. Then $R$ is not qf-definable in terms of equivalence classes of $E$. Then there are such $(x_1, \ldots, x_n) \in R$ and $(y_1, \ldots, y_n) \notin R$ that $x_i E y_i$ for all $i \leq n$. Then it follows from the previous theorem that there is $(S, \sigma)$ in which $E$ is computable but $R$ is not.

□

**Definition 12** For any partial function $f : \omega^* \to \omega^*$, its type is a partial function $T^f : \omega^2 \to \omega^*$ defined as follows:

1. $T^f(i, j) = 0$ iff $f_j^i$ is constant,
2. $T^f(i, j) = k$ iff $f_j^i$ is a projection on the $k$-th coordinate of the argument of $f$,
3. $T^f(i, j)$ is undefined otherwise,

where $f_j^i$ is the projection on the $j$-th coordinate of the value of $f^i$, $f^i$ is $f$ restricted to arguments from $\omega^i$. We also define the following sets: $Cons^f = \{a \in \omega \mid \exists_{i,j}\ (f_j^i \text{ is constant and equal to } a)\}$, $C_a^f = \{(i, j) \mid f_j^i = a\}$ for all $a \in Cons^f$ and $C^f = \bigcup_{a \in Cons^f} C_a^f$.

In a certain part of the proof of the theorem below we are going to use another notion of computability.

**Definition 13** A sequence $f : \omega \to \omega$ is computable in $(S, \sigma)$ if there is an algorithm which produces (arbitrarily long initial segments) of a sequence of numerals $(\alpha_n)_{n \in \omega}$ such that for each $n$, $\sigma(\alpha_n) = f(n)$.

When we wish to utilise the above notion of computability we are going to refer to $f$ as sequence rather than a function.

**Theorem 15** *A partial function $f : \omega^* \to \omega^*$ is computable in every notation iff the following conditions are satisfied:*

1. *for each $i$ the arity of the value of $f^i$ is fixed and the function assigning such arity to each $i$ is computable,*
2. *each $f^i_j$ is either constant, a projection or empty,*
3. *$Cons^f$ is finite and for every $a \in Cons^f$, $C^f_a$ is c.e.,*
4. *$T^f$ is a partial computable function.*

***Proof*** We want to show ($\Leftarrow$). Fix any function $f$ and notation $(S, \sigma)$. Consider $(\alpha_1, \ldots, \alpha_i)$ on the input. Condition 1 allows us to calculate arity of the output. Call this arity $m$. Now we want to calculate values of every $f^i_j$ for $1 \leq j \leq m$.

Fix $j$. From condition 2 we know that $f^i_j$ is either constant, a projection or empty. From condition 4, we calculate $T^f(i, j)$.

If $T^f(i, j) = 0$, then $f^i_j$ is constant and we are able to determine the value of that constant from Condition 3. If $T^f(i, j) = k > 0$, then $f^i_j$ is a projection on the $k$-th coordinate of the argument. If the algorithm computing $T^f(i, j)$ does not halt, then $f^i_j$ is empty. In either of these cases, $f^i_j$ is computable in every notation as a consequence of Theorem 11.

We want to show ($\Rightarrow$). Suppose that $f$ is computable in every notation.

We want to show condition 1. Consider a function $f_a : \omega^i \to \{j, j', k\}$, where $j$ and $j'$ are some arities of $f^i$, $j \neq j'$ and $k$ is any number different from both of them. $f_a$ assigns the value $j$ or $j'$ to any $\alpha \in \omega^i$ whenever $f(\alpha) \in \omega^j$ or $f(\alpha) \in \omega^{j'}$, otherwise it assigns value $k$. Observe that if the range of a function is finite, then from the point of view of computability it is irrelevant which notation is used for the output. Hence we can conclude that if $f$ is computable in any $(S, \sigma)$, then so is $f_a$.

Since $f_a$ is not empty, it is either a projection or constant (as a consequence of Theorem 11. However, the former is impossible since the domain of $f_a$ on every coordinate contains numbers which are not in the range of this functions. Hence $f_a$ is constant. But this is also impossible because both $j$ and $j'$ belong to its range and $j \neq j'$. This is a contradiction.

To show that the function described in this condition is computable, consider an alogrithm which calculates $f$ on any input of length $i$ supplied on the input. If it halts with an output $j$, return $j$.

Condition 2 is straightforward from Theorem 11.

We wish to prove condition 3. Observe that if $f$ is computable in every notation, then $C^f$ and each $C^f_a$ are computable. We want to show that $Cons^f$ is finite. Suppose to the contrary that it is infinite. Consider a set $A \subseteq \omega^2$ such that for each $a \in Cons^f$ there is a unique pair $(i, j) \in A$ such that $f^i_j$ is constant and equal to $a$. Since $f$ is computable in every notation, there is a c.e. set $A$ as described above. For brevity we can enumerate elements of $A$ with natural numbers and think of them as of natural numbers rather than pairs. Thus we obtain a natural bijective sequence $g : \omega \to Cons^f$. Since $f$ is computable in every notation, so is the sequence $g$. However, this would mean that every permutation of the set $Cons^f$ is computable but that is impossible because there are uncountably many such permutations.

We wish to prove condition 4. For any $i \in \omega$, calculate $f(1, \ldots, i)$ and $f(i + 1, \ldots, 2i)$ (in the standard notation). We want to determine $T^f(i, j)$. Consider the $j$-th coordinate of each of both obtained values. If they are both the same, then $f^i_j$ is constant, hence $T^f(i, j) = 0$. If they are not, then the only possibility is that in the former case we obtained the value $k$ for some $k \leq i$ and in the latter we obtained $k + i$, hence $T^f(i, j) = k$. Observe that no other options are possible since we have already determined in condition 2 what kind of function each $f^i_j$ is and we know that it is not undefined since $i \in \omega$.

□

**Theorem 16** *A partial function $f : \omega^n \to \omega^m$ is computable in every notation iff for every $1 \leq i \leq n$, $1 \leq j \leq m$, $f^i_j$ is constant, a projection or empty.*

***Proof*** This is a direct consequence of Condition 2 from the previous theorem.

□

## 6 Discussion and concluding remarks

Shapiro's result (see, Theorem 1) does not fully characterize (relative) intrinsic computability over plain natural numbers (i.e., natural numbers with no additional structure assumed)—it does so only for unary total functions. We have extended this result by covering partial functions and relations of arbitrary finite arities and even when arities of input and output are not fixed. By doing so, we contributed to the programme of syntactic characterization of computational notions.

A separate path of investigation was to consider the class of functions computable in notations (injective or not) in which it is assumed that a certain equivalence relation is computable. The question we have asked is the following: let $(S, \sigma)$ be a notation in which an equivalence relation $E$ is computable. What other relations are guaranteed to be computable in $(S, \sigma)$? Separately, we consider the problem of relative intrinsic computability on a given computable equivalence structure $(\omega, E)$: we ask what relations are computable relative to $E$ in every injective notation.

Results of this sort have been obtained in computable structure theory. For example, the following theorem characterizes relative intrinsic computable enumerability of a relation (which, by definition, is equivalent to the second condition below) on an arbitrary computable structure.

**Theorem 17** *(Ash, Knight, Manasse and Slaman [1]) For a computable structure $\mathcal{A}$ with a further relation $R$, the following are equivalent:*

1. *$R$ is definable in $\mathcal{A}$ by a computable $\Sigma_1$ formula,*
2. *in all copies $\mathcal{B}$ of $\mathcal{A}$, the image of $R$ is $\Sigma^0_1(\mathcal{B})$.*

Recall Theorem 5 which characterizes total functions relatively instrinsically computable on $(\omega, E)$, where $E$ is a computable equivalence relation of finite character, as those which are quantifier-free definable in $(\omega, E)$ (with parameters). This theorem can be obtained from the above result based on the observation that the theory of an equivalence structure of finite character has quantifier elimination.[4] However, the

[4] Take a computable $\Sigma_1$ formula that defines $R$ in $(\omega, E)$, replace each finitary $\Sigma_1$ formula in it by an equivalent quantifier-free formula and then observe that an infinite disjuction of quantifier-free formulae is equivalent to a single quantifier-free formula (noting that, in total, only finitely many parameters are allowed).

theory of a structure considered in Theorem 4 does not admit such elimination. Therefore, Theorem 4 cannot be directly deduced from the above result based on simple elimination-like argument.

There are a few immediate questions that we have not considered. Can a result similar to Theorem 4 be obtained for computable $E$ for which there is some finite $k$ such that $E$ has infinitely many classes of size $k$, there exists $l$ such that for every $p$, if $E$ has infinitely many classes of size $p$, then $p \leq l$, and either of the following conditions hold:

1. $E$ has unbounded character,
2. $E$ has an infinite class,
3. $E$ has infinitely many classes of size $k'$, for some $k' \neq k$.

To the best of our knowledge, these structures do not admit quantifier elimination and thus cannot be deduced this way from Theorem 17.

Another question concerning equivalence relations is related to noninjective notations. We wish to prove the following hypothesis: for any equivalence relation $E$ and any $R \subseteq \omega^n$, $R$ is computable in every notation in which $E$ is iff $R$ is qf-definable in terms of $E$. The case that we still have not proved is for some $E$ with inifinitely many equivalence classes and for $n > 1$.

**Acknowledgements** We would like to thank Nikolay Bazhenov and Alex Kruckman for helpful discussions. We are also grateful to the reviewer for comments.



## References

1. Ash, C., Knight, J., Manasse, M., Slaman, T.: Generic copies of countable structures. Annals of Pure and Applied Logic **42**(3), 195–205 (1989). Publisher: North-Holland
2. Ash, C., Nerode, A.: Intrinsically recursive relations. In: Crossley, J. (ed.) Aspects of Effective Algebra, pp. 26–41. Upside Down A Book Company, Monash University, Australia (1981)
3. Ash, C.J., Knight, J.: Computable structures and the hyperarithmetical hierarchy. Elsevier (2000)
4. Bazhenov, N., Fokina, E., Rossegger, D., San Mauro, L.: Degrees of bi-embeddable categoricity of equivalence structures. Archive for Mathematical Logic **58**(5), 543–563 (2019). https://doi.org/10.1007/s00153-018-0650-3
5. Bazhenov, N., Marchuk, M.: A note on decidable categoricity and index sets. Siberian Elect. Math. Reports **17**, 1013–1026 (2020). https://doi.org/10.33048/semi.2020.17.076
6. Benacerraf, P.: What Numbers Could Not Be. Philosophical Review **74**(1), 47–73 (1965)
7. Benacerraf, P.: Recantation or Any Old $\omega$-sequence Would Do After All. Philosophia Mathematica **4**(2), 184–189 (1996)
8. Blinov, K.V.: Primitively Recursive Categoricity for Unars and Equivalence Structures. Siberian Math. J **62**(6), 994–1009 (2021). https://doi.org/10.1134/S0037446621060033

9. Calvert, W., Cenzer, D., Harizanov, V., Morozov, A.: Effective categoricity of equivalence structures. Annals of Pure and Applied Logic **141**(1), 61–78 (2006). https://doi.org/10.1016/j.apal.2005.10.002. https://www.sciencedirect.com/science/article/pii/S0168007205001557
10. Chang, C.C., Keisler, H.J.: Model theory, Studies in Logic and the Foundations of Mathematics, vol. 73. North-Holland Press, Amsterdam, New York (1973)
11. Copeland, B.J., Proudfoot, D.: Deviant Encodings and Turing's Analysis of Computability. Studies in History and Philosophy of Science Part A **41**(3), 247–252 (2010)
12. Downey, R., Melnikov, A.G., Ng, K.M.: On $\Delta^0_2$-categoricity of equivalence relations. Annals of Pure and Applied Logic **166**(9), 851–880 (2015). https://doi.org/10.1016/j.apal.2015.04.003. https://www.sciencedirect.com/science/article/pii/S0168007215000354
13. Downey, R.G., Melnikov, A.G., Ng, K.M.: A Friedberg enumeration of equivalence structures. J. Math. Logic **17**(02), 1750008 (2017). https://doi.org/10.1142/S0219061317500088
14. Gold, E.M.: Limiting Recursion. J. Symbolic Logic **30**(1), 28–48 (1965)
15. Knight, J.F.: Degrees Coded in Jumps of Orderings. The Journal of Symbolic Logic **51**(4), 1034–1042 (1986). http://www.jstor.org/stable/2273915. Publisher: [Association for Symbolic Logic, Cambridge University Press]
16. Markov, A.A.: The theory of algorithms. Trudy Matematicheskogo Instituta imeni V.A. Steklova **38**, 176–189 (1951)
17. Montalbán, A.: Computable structure theory: Within the arithmetic. Cambridge University Press (2021)
18. Putnam, H.: Trial and Error Predicates and the Solution to a Problem of Mostowski. J. Symbolic Logic **30**(1), 49–57 (1965)
19. Quinn, S.B.: Scott sentences for equivalence structures. Archive for Mathematical Logic **59**(3), 453–460 (2020). https://doi.org/10.1007/s00153-019-00701-x
20. Quinon, P.: A Taxonomy of Deviant Encodings. In: D.N. Florin Manea Russell G. Miller (ed.) 14th Conference on Computability in Europe, CiE 2018, Lecture Notes in Computer Science, vol. 10936 LNCS, p 338–348. Springer Verlag, Kiel (2018)
21. Rescorla, M.: Copeland and Proudfoot on Computability. Studies in History and Philosophy of Science Part A **43**(1), 199–202 (2012)
22. Rogers, H., Jr.: Theory of Recursive Functions and Effective Computability. McGraw-Hill series in higher mathematics. McGraw-Hill Book Company, New York, NY, USA (1967)
23. Shapiro, S.: Acceptable Notation. Notre Dame Journal of Formal Logic **23**(1), 14–20 (1982)
24. Shoenfield, J.R.: On degrees of unsolvability. Annal. math. **69**, 644–653 (1959)
25. Soare, R.I.: Recursively Enumerable Sets and Degrees. Springer-Verlag, New York Inc, New York, NY, USA (1987)
26. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. J. of Math **58**(345–363), 5 (1936)
27. Wrocławski, M.: Representing Numbers. Filozofia Nauki **26**(4), 57–73 (2018)
28. Wrocławski, M.: Representations of Natural Numbers and Computability of Various Functions. In: F. Manea, B. Martin, D. Paulusma, G. Primiero (eds.) 15th Conference on Computability in Europe, CiE 2019, Lecture Notes in Computer Science. Springer Verlag (2019)
29. Wrocławski, M.: Representations of Numbers and their Computational Properties. Ph.D. thesis, Institute of Philosophy, University of Warsaw, Poland (2019)