Representations of Natural Numbers and Computability of Various Functions

Michał Wrocławski(✉)

Institute of Philosophy, University of Warsaw, Warsaw, Poland
michalwro@wp.pl

Abstract. We discuss various ways of representing natural numbers in computations. We are primarily concerned with their computational properties, i.e. which functions each of these representations allows us to compute. We show that basic functions, such as successor, addition, multiplication and exponentiation are largely computationally independent from each other, which means that in most cases computability of one of them in a certain representation does not imply that others will be computable in it as well.

We also examine what difference can be made if we restrict our attention only to those representations in which it is decidable whether two numerals represent the same number. It turns out that the impact of such restriction is huge and that it allows us to rule out representations with certain unusual properties.

Keywords: Representations of numbers · Computable functions · Characteristic functions

1 Introduction

Various authors have been considering the view that the notion of computability applies in the first place to functions on numerals, rather than on numbers themselves. Such position has been suggested by Shapiro in [4] and further discussed, among others, by Rescorla in [3], Copeland and Proudfoot in [1] and Quinon in [2]. I have also considered related issues in [5].

All algorithms are performed on strings of symbols which denote numbers (or other objects)—but a certain number can be represented by different strings. E.g. the number 6 is represented as VI when we use Roman numerals, but by 110 if we want to use binary numerals. Computation of a function such as addition is different in each of these cases.

According to Church's thesis, computable functions are exactly recursive functions. However, if we allow non-standard ways of encoding numbers, this does not have to be true. A set of numerals (satisfying a few additional conditions specified in the next section) together with a function assigning a natural number to each numeral, shall be called a representation.

© Springer Nature Switzerland AG 2019
F. Manea et al. (Eds.): CiE 2019, LNCS 11558, pp. 298–309, 2019.
https://doi.org/10.1007/978-3-030-22996-2_26

In this paper we are going to examine computability of the most important functions on natural numbers: successor, addition, multiplication and exponentiation. While they are all recursive and hence their computability is normally taken for granted, we want to show that this is not always the case (i.e. not in every representation). Furthermore, as it turns out, these functions are computationally largely independent from each other—i.e. the assumption of computability of one of them in most cases does not guarantee computability of the others.

We shall also provide a suggestion of an additional constraint on representations which will allow us to rule out representations with particularly irregular properties. Namely, if for a certain representation there exists an algorithm which for any two numerals determines whether they represent the same number or not, then such representations exhibit properties much more similar to representations usually employed.

2 Defining the Concept of Representation

In this section we are going to define some basic notions regarding representations.

Definition 1. *Let Σ be a finite alphabet. We shall call (S, σ) a representation of $\mathbb{N}$, where $S \subseteq \Sigma^*$ is an infinite computable set and $\sigma : S \to \mathbb{N}$ is a surjection.*

Definition 2. *Let (S, σ) be a representation of $\mathbb{N}$. We shall say that this representation is unambiguous iff for every $n \in \mathbb{N}$ there exists exactly one numeral $\alpha \in S$ such that $\sigma(\alpha) = n$. Otherwise we shall call the representation ambiguous.*

The basic example of a representation is the unary representation defined as follows:

Let $\Sigma = \{\overline{1}\}$. S is the set of all finite sequences comprised of $\overline{1}$ and the empty word ε, and the function σ is defined in the following way:

$$\sigma(\varepsilon) = 0,$$

$$\text{if } \sigma(\alpha) = n, \text{then } \sigma(\alpha \frown \overline{1}) = n + 1.$$

Another representation, which we shall refer to throughout this paper as the standard representation, is the decimal representation, defined as follows:

Let $\Sigma = \{\overline{0}, \overline{1}, ..., \overline{9}\}$. S is the set of all standard decimal numerals (i.e. the set consisting of the numeral $\overline{0}$ and of all finite sequences of digit from Σ which do not begin with $\overline{0}$), and the function σ is defined in the following way:

$$\sigma(\overline{a_n}...\overline{a_0}) = \sum_{i=0}^{n} a_i \cdot 10^i,$$

Both these representations are unambiguous.

In unambiguous representations, the concept of computability is simple. A function is computable if there exists an algorithm which for every numeral (representing a certain number) supplied on the input, returns the numeral representing the value of the function on the output. The issue gets more complicated when it comes to ambiguous representations. This is how we define computability in general case:

Definition 3. *Let (S, σ) be a representation of $\mathbb{N}$. Then for any function, $f : \mathbb{N}^n \to \mathbb{N}$, by $f^\sigma : S^n \to S$ we shall denote a function such that for any $\alpha_1, ..., \alpha_n, \beta \in S$ the following condition is satisfied:*

$$f^\sigma(\alpha_1, ..., \alpha_n) = \beta \Rightarrow f(\sigma(\alpha_1), ..., \sigma(\alpha_n)) = \sigma(\beta).$$

If there exists a computable function f^σ satisfying the above condition, than we shall say that f is computable in (S, σ).

Note that in case of ambiguous representations, many such functions f^σ can exist. It is possible that some of them are computable, and some are not. We adopt a convention that "to compute the function f in (S, σ)" and "to compute f^σ" are both going to mean "to compute any function f^σ which satisfies the above condition".

We will also want to be able to compute Boolean functions, i.e. functions whose values are $TRUE$ and $FALSE$.

Definition 4. *Let $R \subseteq \mathbb{N}^n$. The characteristic function of the relation R is the function χ_R such that for any $a_1, ..., a_n \in \mathbb{N}$ the following holds:*

$$\chi_R(a_1, ..., a_n) = TRUE \Leftrightarrow R(a_1, ...a_n).$$

$$\chi_R(a_1, ..., a_n) = FALSE \Leftrightarrow \neg R(a_1, ...a_n).$$

In this paper we are going to be particularly concerned with the characteristic function of identity:

$$\chi_=(a_1, a_2) = TRUE \Leftrightarrow a_1 = a_2,$$

$$\chi_=(a_1, a_2) = FALSE \Leftrightarrow a_1 \neq a_2.$$

The computability of characteristic functions is defined in a similar way as in the case of numerical functions.

Definition 5. *Let (S, σ) be a representation of $\mathbb{N}$. Then for any relation $R \subseteq \mathbb{N}^n$ we shall define $R^\sigma \subseteq S^n$ in the following way:*

$$(\alpha_1, ..., \alpha_n) \in R^\sigma \Leftrightarrow (\sigma(\alpha_1), ..., \sigma(\alpha_n)) \in R,$$

for all $\alpha_1, ..., \alpha_n \in S$. We shall say that χ_R is computable (or simply that R is computable) in (S, σ) if and only if R^σ is computable.

Note that $TRUE$ and $FALSE$ are neither numerals, nor numbers, but they are entirely different symbols.

3 Computability of Successor, Addition, Multiplication and Exponentiation in Representations of Natural Numbers

In this section we are going to show what are the relations between computability of some basic functions. In particular, we want to emphasise the role of computability of characteristic function of identity $\chi_{=}$.

The proofs of Theorems 6 (in a modified form) and 7 come from my paper [5]. The former theorem is a generalised version of Shapiro's result included in his paper [4]. Shapiro considered only unambiguous representations (in his terminology—notations), which is a very common approach among authors dealing with this subject. I have generalised his result to include all types of representations.

Theorem 6. *Let (S, σ) be a representation of $\mathbb{N}$ in which successor and $\chi_{=}$ are computable. Then all functions computable in the standard representation, including addition, multiplication and exponentiation, are also computable in (S, σ).*

Proof. Let (S, σ) be a representation of $\mathbb{N}$ in which the successor function (denoted as $Succ$) and $\chi_{=}$ are computable. In this representation there is a numeral representing number 0. Let us denote such a numeral as α, i.e. let $\alpha \in S$ be such that $\sigma(\alpha) = 0$. Note that for the purpose of this proof we only need to know that such α exists, not which numeral it is. This is because it is our aim here only to prove the existence of an algorithm, not to state which exactly algorithm it is.

We shall first show how to translate numerals from (S, σ) to the standard representation.

Let $\overline{n}$ be a numeral representing n in the standard representation, for every natural number n. The purpose of this convention is to clearly distinguish between standard numerals and numbers which they denote.

Let λ be a numeral of (S, σ). For every natural number n, let us denote $\lambda_n = Succ^{\sigma}(Succ^{\sigma}(...(\alpha)...))$, where the successor is iterated n times in λ_n. We compare one by one each λ_n with λ until we find such n that $\chi^{\sigma}_{=}(\lambda, \lambda_n) = TRUE$. Then $\sigma(\lambda) = n$, so the numeral $\overline{n}$ represents the same number in the standard representation as the numeral λ in (S, σ).

Let $\overline{n}$ be a numeral of the standard representation. To find its counterpart in (S, σ), we calculate λ_n defined as above.

Now suppose that f is computable in the standard representation. We want to compute this function in (S, σ) on some given input. In order to do so, we translate this input to the standard representation, perform an algorithm in the standard representation and then translate the output back to (S, σ).

Theorem 7. *There exists a representation (S, σ) of $\mathbb{N}$ in which the successor function is computable, but addition, multiplication and exponentiation are not computable.*

Proof. We construct (S, σ) as follows:

The alphabet consists of symbols: $\overline{0}$, $\overline{1}$, a.

The set of numerals S consists of all finite non-empty sequences of symbols from the alphabet which contain at most one occurrence of a.

Let $A \subseteq \mathbb{N}$ be uncomputable in the standard representation.

We construct σ in the following way:

$$\sigma(\overline{0}) = 0,$$
$$\sigma(\overline{1}) = 1,$$
$$\sigma(a) = 0 \Leftrightarrow 1 \notin A,$$
$$\sigma(a) = 1 \Leftrightarrow 1 \in A.$$

Also, for any $\alpha \in S$:

$$\sigma(\alpha \frown \overline{0}) = \sigma(\alpha),$$
$$\sigma(\alpha \frown \overline{1}) = \sigma(\alpha) + 1,$$
$$\sigma(\alpha \frown a) = \sigma(\alpha) \Leftrightarrow lh(\alpha) + 1 \notin A,$$
$$\sigma(\alpha \frown a) = \sigma(\alpha) + 1 \Leftrightarrow lh(\alpha) + 1 \in A,$$

where $\frown$ is a concatenation and $lh(\alpha)$ is the length of the sequence α.

This is a correct representation because every natural number n is represented by at least one numeral, namely $\overline{1}...\overline{1}$ consisting of n digits $\overline{1}$, with the exception of number 0, which is represented by the numeral $\overline{0}$.

For any $\alpha \in S$, let $\#_{\overline{1}}(\alpha)$ denote the number of occurrences of symbol $\overline{1}$ in the numeral α.

The successor function in (S, σ) can be computed as follows:

$$Succ^{\sigma}(\alpha) = \alpha \frown \overline{1}.$$

We shall show that addition is not computable in this representation. Suppose to the contrary that it is.

For any natural number $n \geq 1$ let us denote:

$$\lambda_n = \overline{0}...\overline{0}a,$$

where λ_n consists of $n - 1$ digits $\overline{0}$ followed by one occurrence of a.

We want to find out whether $n \in A$. We compute $\lambda_n + \lambda_n$ in (S, σ). We know that $\sigma(\lambda_n)$ is equal to 0 or 1. Thus $\sigma(\lambda_n +^{\sigma} \lambda_n)$ is equal to 0 or 2.

If $n \in A$, then $\sigma(\lambda_n) = 1$ and $\sigma(\lambda_n +^{\sigma} \lambda_n) = 2$. Then $\#_{\overline{1}}(\lambda_n +^{\sigma} \lambda_n) \geq 1$. If, however, $n \notin A$, then $\sigma(\lambda_n) = \sigma(\lambda_n +^{\sigma} \lambda_n) = 0$ and then $\#_{\overline{1}}(\lambda_n +^{\sigma} \lambda_n) = 0$.

It is easy to find out which of these cases occurs and thus—whether $n \in A$. It follows that A is computable in the standard representation, which contradicts our assumption. Therefore, addition is not computable in (S, σ).

Similarly we show that multiplication and exponentiation are not computable in (S, σ). Let us denote:

$$\delta_n = \overline{1}...\overline{1}a,$$

where λ_n consists of $n-1$ digits $\overline{1}$ followed by one occurrence of a. Then we compute respectively $\delta_n \cdot \delta_n$ or ${\delta_n}^{\overline{11}}$ in (S,σ) (note that they both return the same result, we shall only provide a proof for the case with multiplication).

Suppose that multiplication is computable in (S,σ). We shall prove that A is also computable. Let $n \in \mathbb{N}$. We want to find out whether $n \in A$. Without loss of generality we can assume that $n \geq 2$.[1] Let $\alpha \in S$ be the result of multiplication $\delta_n \cdot \delta_n$ in (S,σ). We know that $\sigma(\delta_n)$ is equal to either $n-1$ or n. Therefore:

1. If $\sigma(\delta_n) = n-1$, then $\sigma(\delta_n \cdot^{\sigma} \delta_n) = (n-1)^2 = n^2 - 2n + 1$. Therefore $\#_{\overline{1}}(\alpha) = n^2 - 2n$ or $\#_{\overline{1}}(\alpha) = n^2 - 2n + 1$.
2. If $\sigma(\delta_n) = n$, then $\sigma(\delta_n \cdot^{\sigma} \delta_n) = n^2$. Therefore $\#_{\overline{1}}(\alpha) = n^2 - 1$ or $\#_{\overline{1}}(\alpha) = n^2$.

Note that for $n \geq 2$ we can find out which of these cases occurs. If the first case occurs, then $n \notin A$, otherwise $n \in A$. Thus we have obtained contradiction with the assumption that A is not computable. Therefore multiplication (and similarly exponentiation) is not computable in (S,σ).

Theorem 8. *Let (S,σ) be a representation of $\mathbb{N}$ in which addition is computable. Then the successor function is also computable in this representation.*

Proof. Let (S,σ) be a representation of $\mathbb{N}$ in which addition is computable. In (S,σ) there must be a numeral representing number 1. Let us denote this numeral as β. Then we can calculate the successor function in (N,σ) as follows:

$$Succ(\alpha) = \alpha +^{\sigma} \beta.$$

Theorem 9. *There exists a representation (S,σ) of $\mathbb{N}$ in which addition (and thus also successor) is computable, but multiplication and exponentiation are not computable.*

Proof. For any natural number n, let $\overline{n}$ denote the numeral which represents n in the standard representation of $\mathbb{N}$.

We construct the following representation (S,σ):

The alphabet Σ consists of digits $\overline{0}$, ... , $\overline{9}$, of symbols (,) and the comma.

We construct the set S of numerals as follows:

For any standard numerals $\overline{a_0}$, ... , $\overline{a_n}$, the sequence $(\overline{a_0}, ..., \overline{a_n})$ is a numeral of the representation (S,σ) if $a_0 \geq \sum_{i=1}^{n} a_i$.

Let $A \subseteq \mathbb{N}$ be uncomputable in the standard representation such that $0 \in A$.

For any $(\overline{a_0}, ..., \overline{a_n}) \in S$ the function σ is defined as follows:

$$\sigma((\overline{a_0}, ..., \overline{a_n})) = \sum_{i=0}^{n} (a_i \cdot \chi_A(i)),$$

where for any natural number i: $\chi_A(i) = 1$ if $i \in A$, and $\chi_A(i) = 0$ if $i \notin A$.

[1] The algorithm which is supposed to find out whether $n \in A$ will have answers for $n \in \{0,1\}$ explicitly given as special cases.

This representation is well-defined because every natural number is represented by at least one numeral, in particular n is represented by $(\overline{n})$.

For any numerals $(\overline{a_0}, ..., \overline{a_m})$ and $(\overline{b_0}, ..., \overline{b_n})$ (without loss of generality we assume that $m \leq n$), we define addition in (S, σ) in the following way:

$$(\overline{a_0}, ..., \overline{a_m}) +^{\sigma} (\overline{b_0}, ..., \overline{b_n}) = (\overline{a_0 + b_0}, ..., \overline{a_m + b_m}, \overline{b_{m+1}}, ...\overline{b_n}),$$

where $+_S$ is interpreted as addition of numbers represented by respective numerals in the standard representation. It is obviously computable.

We shall prove that multiplication is not computable in this representation. Suppose that it is computable. We shall show that then A is computable in the standard representation which leads to a contradiction.

We want to find out whether $n \in A$.

For any natural number n we define the following numeral:

$$\lambda_n = (\overline{1}, \overline{0}, ..., \overline{0}, \overline{1}),$$

where λ_n has $\overline{1}$ on the zeroth and n-th position and $\overline{0}$ on all the other positions.

We compute the multiplication $\lambda_n \cdot \lambda_n$ in (S, σ). There are two possible cases:

If $n \in A$, then $\sigma(\lambda_n) = 2$ and $\sigma(\lambda_n \cdot \lambda_n) = 4$. From the condition that $a_0 \geq \sum_{i=1}^{n} a_i$ it follows that $a_0 \geq 2$ for every numeral representing number 4 in this representation.

If $n \notin A$, then $\sigma(\lambda_n) = 1$ and $\sigma(\lambda_n \cdot \lambda_n) = 1$, so $a_0 = 1$ in a numeral representing number 1 in this representation.

We determine which of these cases occurs and thus we can find out whether $n \in A$. Therefore A is a computable set in the standard representation, which leads to a contradiction. It follows that multiplication is not computable in (S, σ).

Similarly, by considering the result of the computation ${\lambda_n}^{\lambda_n}$ we can show that exponentiation is not computable in this representation.

We compute ${\lambda_n}^{\lambda_n}$ in (S, σ). There are two possible cases:

If $n \in A$, then $\sigma(\lambda_n) = 2$ and $\sigma({\lambda_n}^{\lambda_n}) = 4$. From the condition that $a_0 \geq \sum_{i=1}^{n} a_i$ it follows that $a_0 \geq 2$ for every numeral representing number 4 in this representation.

If $n \notin A$, then $\sigma(\lambda_n) = 1$ and $\sigma({\lambda_n}^{\lambda_n}) = 1$, so $a_0 = 1$ in a numeral representing number 1 in this representation.

We determine which of the cases occurs and thus we can find out whether $n \in A$. Therefore A is a computable set in the standard representation, which leads to a contradiction. It follows that exponentiation is not computable in (S, σ).

Theorem 10. *There exists a representation (S, σ) of $\mathbb{N}$ in which multiplication and $\chi_=$ are computable, but addition and exponentiation are not computable.*

Proof. Let π be a permutation of $\mathbb{N}$ uncomputable in the standard representation.

We construct the following representation. The alphabet consists of the digits $\overline{0}$, ... , $\overline{9}$, of the symbols (,) and the comma.

The admissible numerals in (S, σ) are all finite sequences of the form $(\overline{a_0}, ..., \overline{a_n})$, where each a_i is a natural number. Additionally, the numeral $\overline{0}$ belongs to S.

We construct σ as follows:

$$\sigma(\overline{0}) = 0,$$

$$\sigma((\overline{a_0}, ..., \overline{a_n})) = p_{\pi(0)}^{a_0} \cdot ... \cdot p_{\pi(n)}^{a_n},$$

where $\overline{a_i}$ is the numeral representing a_i in the standard representation and p_i is the i-th prime number.

It is a correct representation because each natural number is represented by a certain numeral, which results from the fundamental theorem of arithmetic.

For any numerals $(\overline{a_0}, ..., \overline{a_m})$ and $(\overline{b_0}, ..., \overline{b_n})$ (without loss of generality we assume that $k \leq l$), we define multiplication in (S, σ) in the following way:

$$(\overline{a_0}, ..., \overline{a_m}) \cdot^{\sigma} (\overline{b_0}, ..., \overline{b_n}) = (\overline{a_0 + b_0}, ..., \overline{a_m + b_m}, \overline{b_{m+1}}, ...\overline{b_n}),$$

where $+$ is interpreted as addition of numbers in the standard representation.

Additionally, for any $\alpha \in S$, let $\alpha \cdot^{\sigma} \overline{0} = \overline{0} \cdot^{\sigma} \alpha = \overline{0}$.

Hence, multiplication is computable in (S, σ). The function $\chi_{=}$ is also computable, as a consequence of the fundamental theorem of arithmetic. We shall show that addition and exponentiation are not computable in this representation.

Let us assume that addition is computable in this representation. We shall show that then the permutation π must be computable in the standard representation, which leads to a contradiction.

Let n be any natural number. We want to find the value of $\pi^{-1}(n)$. We take any non-zero numeral $\lambda \in S$ and we calculate $\underbrace{\lambda + ... + \lambda}_{p_n \text{ times}}$ in (S, σ). Then we check on which position of λ the number has increased by 1 (note that it can also be a new position on which $\overline{1}$ has appeared). The number of this position is equal to $\pi^{-1}(n)$. Thus we can compute the permutation π^{-1} in the standard representations. However, if π^{-1} is computable, then obviously π is also computable.

Now suppose that exponentiation is computable in this representation. We shall prove that then the permutation π must be computable in the standard representation.

For any natural number n we shall find $\pi(n)$ using the following method:

Let λ_n be a numeral of the form $(\overline{0}, ..., \overline{0}, \overline{1})$, where the digit $\overline{1}$ is proceeded by n occurrences of the digit $\overline{0}$. Then $\sigma(\lambda_n) = p_{\pi(n)}$. We compute the result of $(\overline{1})^{\lambda_n}$ in (S, σ). Obviously:

$$\sigma((\overline{1})^{\lambda_n}) = p_{\pi(0)}^{p_{\pi(n)}}.$$

When we calculate this exponentiation, we will get the numeral $(\overline{p_{\pi(n)}})$ as a result. Thus we find out the value of the $\pi(n)$-th prime number, so we can easily compute $\pi(n)$.

Theorem 11. *Let (S, σ) be a representation of $\mathbb{N}$ in which exponentiation and $\chi_=$ are computable. Then multiplication and addition are also computable in this representation.*

Proof. Let $\alpha, \beta \in S$. We want to calculate $\alpha \cdot^\sigma \beta$ and $\alpha +^\sigma \beta$. Let λ be a numeral representing number 2 in (S, σ) and let $(\zeta_n)_{n\in\mathbb{N}}$ be a recursive enumeration of all numerals from S. For each ζ_n we check if the following equality holds:

$$(\lambda^\alpha)^\beta = \lambda^{\zeta_n}$$

until we find a numeral for which it is true. Such ζ_n shall be the result of calculating $\alpha \cdot^\sigma \beta$ in (S, σ).

To calculate $\alpha +^\sigma \beta$ in (S, σ), for each ζ_n we check whether the following equality holds:

$$\lambda^\alpha \cdot^\sigma \lambda^\beta = \lambda^{\zeta_n}.$$

until we find a numeral for which it is true. Such ζ_n shall be the result of calculating $\alpha +^\sigma \beta$ in (S, σ).

We conclude that addition and multiplication are computable in (S, σ).

Theorem 12. *There exists a representation (S, σ) of $\mathbb{N}$ in which exponentiation is computable, but successor, addition and multiplication are not computable.*

Proof. We construct such a representation as follows:

The alphabet consists of digits $\overline{0}$, ... , $\overline{9}$, symbols $\overline{\pi}$, E, (,) and the comma.

The set of numerals S is the smallest set satisfying the following conditions:

Every numeral of the standard representation belongs to S.

If $\alpha, \beta \in S \setminus \{\overline{0}, \overline{1}\}$, then $E(\alpha, \beta) \in S$.

If $\alpha \in S$ and α represents a prime number in the standard representation, then $\overline{\pi}(\alpha) \in S$.

We construct the function σ in the following way:

Let π be a permutation of prime numbers (i.e. a bijection from prime numbers onto prime numbers) uncomputable in the standard representation such that $\pi(2) = 2$.

For any standard numeral $\overline{n}$, let $\sigma(\overline{n}) = n$

For any $\alpha, \beta \in S \setminus \{\overline{0}, \overline{1}\}$, let $\sigma(E(\alpha, \beta)) = \sigma(\alpha)^{\sigma(\beta)}$.

For any prime number p, let $\sigma(\overline{\pi}(\overline{p})) = \pi(p)$.

This representation is well-defined because every natural number is represented by a certain numeral, in particular by the same numeral as in the standard representation.

We define exponentiation in (S, σ) as follows:

$\alpha^\beta = E(\alpha, \beta)$, for $\alpha, \beta \in S \setminus \{\overline{0}, \overline{1}\}$,

$\alpha^{\overline{0}} = \overline{1}$, $\alpha^{\overline{1}} = \alpha$, $\overline{1}^\alpha = \overline{1}$, for any $\alpha \in S$,

$\overline{0}^\alpha = \overline{0}$, for any $\alpha \in S \setminus \{\overline{0}, \overline{1}\}$.

Exponentiation is computable in this representation.

We shall prove that successor is not computable in (S, σ). Suppose to the contrary that it is computable. We shall show that π is then computable in the standard representation, which leads to a contradiction.

Let $T = (\alpha_{ij})_{i,j\in\mathbb{N}}$ be defined as follows:

$$\alpha_{ij} = Succ^{\sigma}(Succ^{\sigma}(...(\overline{\pi}(\overline{p_i}))...)),$$

where the successor is iterated j times, and p_i is the i-th prime number.

Since we assumed that the successor function is computable, it follows that T is a computable family of numerals indexed by pairs of natural numbers.

Note that each prime number p is represented by exactly two numerals in (S, σ), namely $\overline{p}$ and $\overline{\pi}(\overline{q})$, for a certain prime number q. Let us consider the following cases:

Case 1. Suppose that there exists a prime number p and that there exist natural numbers i, j such that $\alpha_{ij} = \overline{p}$ (by this we understand the equality of numerals, not just the equality of numbers represented by them) and for every prime number $p\prime > p$ and for any natural numbers $i\prime, j\prime$ the following holds: $\alpha_{i\prime j\prime} \neq \overline{p\prime}$. Then we consider the infinite sequence of results of the following computations (which is a row of T, possibly with the exception of a certain initial segment):

$$\overline{p}, Succ^{\sigma}(\overline{p}), Succ^{\sigma}(Succ^{\sigma}(\overline{p})), ...$$

It is a sequence of numerals representing consecutive natural numbers, starting with $\sigma(\overline{p})$. For any natural numbers i, j, if α_{ij} is a numeral representing a certain prime number p, then $\alpha_{ij} = \overline{p}$ or $\alpha_{ij} = \overline{\pi}(\overline{q})$, for a certain prime number q. According to our assumption, there are only finitely many prime numbers represented in the first of these two ways. In each row of T nearly all prime numbers are represented by numerals of the second type. Since T is computable, by calculating consecutive numerals from any row of T and choosing only those of them which represent prime numbers, we obtain an infinite sequence representing consecutive prime numbers:

$$\overline{\pi}(\overline{p_{i_0}}), \overline{\pi}(\overline{p_{i_1}}), \overline{\pi}(\overline{p_{i_2}}), ...$$

Then we compute π in the following way: nearly all of its elements can be obtained from the above sequence, the rest of them (which are finite in number) can be enumerated as special cases.

Case 2. Suppose that there exists a natural number i such that for any natural number j and any prime number p: $\alpha_{i,j} \neq \overline{p}$. Then from the i-th row of T, like in the previous case, we calculate nearly all values of π. Since there are only finitely many values outside of this row, it follows that π is computable.

Case 3. Suppose that there is no natural number which satisfies either of the conditions from cases 1 and 2. Therefore, for every natural number i there exist a natural number j and a prime number p such that $\alpha_{i,j} = \overline{p}$. Let us take any natural number i. We shall show how to compute $\pi(p_i)$, where p_i is the i-th prime number. Let j, p be such that $\alpha_{i,j} = \overline{p}$, where p is a prime number. Also:

$$\alpha_{i,j} = Succ^{\sigma}(Succ^{\sigma}(...(\overline{\pi}(\overline{p_i}))...)),$$

where the successor function is iterated j times.

Therefore $\pi(p_i) + j = p$. It follows that $\pi(p_i) = p - j$. We have obtained a contradiction with the assumption that π is not computable in the standard representation. Therefore the successor function is not computable in (S, σ).

From this and Theorem 8 it follows that addition is not computable in (S, σ) either.

We shall show that multiplication is not computable in (S, σ). Assume to the contrary that it is. We shall show that π is then computable in the standard representation and thus we shall obtain a contradiction. Let $p > 2$ be a prime number. We shall show how to compute $\pi(p)$.

Let us calculate $\overline{2} \cdot^{\sigma} \overline{\pi}(\overline{p})$. From the definition of (S, σ) it follows that the result of this calculation cannot be of the form $E(\alpha, \beta)$ for any numerals α, β. It cannot be of the form $\overline{\pi}(\overline{q})$, for any prime number q because the result of this multiplication is not a prime number. Therefore it must be a certain standard numeral $\overline{n}$. However, all such numerals are interpreted in (S, σ) just like in the standard representation. Therefore $\pi(p) = \frac{n}{2}$.

It follows that π is computable in the standard representation, which contradicts our assumption. Therefore, multiplication is not computable in (S, σ).

4 Conclusions

In this paper we have considered computability of the most important functions on natural numbers. We believe that we have managed to establish in what ways their computability depends on each other.

Based on these results it seems that except for some trivial cases, we can usually construct a representation in which one function is computable and the other is not. The example of such a trivial case was that computability of addition implies computability of successor—which is not surprising because successor function can be obtained by addition by substituting a constant for one of the arguments.

It is our purpose to find general rules governing such dependencies. Suppose that for a function f, we define computational closure of f as the set of all functions computable in every representation in which f is computable. Certainly f will be closed under such operations as substitution, composition of functions, etc. But is it possible to give a complete description of what functions belong to such closure? This is a question we are currently investigating.

Another important conclusion is that the computational landscape dramatically changes as soon as Boolean functions are included. The assumption of computability of $\chi_=$ ensures that we are already able to say a lot more about properties of various representations. The question arises whether there are other equivalence relations of similar importance.

References

1. Copeland, B.J., Proudfoot, D.: Deviant encodings and Turing's analysis of computability. Stud. Hist. Philos. Sci. Part A **41**(3), 247–252 (2010)
2. Quinon, P.: A taxonomy of deviant encodings. In: Manea, F., Miller, R.G., Nowotka, D. (eds.) CiE 2018. LNCS, vol. 10936, pp. 338–348. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94418-0_34
3. Rescorla, M.: Church's thesis and the conceptual analysis of computability. Notre Dame J. Formal Logic **48**(2), 253–280 (2007)
4. Shapiro, S.: Acceptable notation. Notre Dame J. Formal Logic **23**(1), 14–20 (1982)
5. Wrocławski, M.: Representing Numbers. Filozofia Nauki **26**(4), 57–73 (2018)