1. URM machines

Exercise 1.7

Exercise 1.7(p). Define the operation of primitive recursion and prove that the set $\mathcal{C}$ of URM-computable functions is closed with respect to this operation.

PROOF. Let $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^{k+2} \to \mathbb{N}$ be computable functions. We want to prove that $h : \mathbb{N}^{k+1} \to \mathbb{N}$ defined through primitive recursion

$$\begin{cases} h(\vec{x}, 0) = f(\vec{x}) \\ h(\vec{x}, y+1) = g(\vec{x}, y, h(\vec{x}, y)) \end{cases}$$

is computable.

Let F, G programs in standard form for f, g. We want a program H for h. We proceed as suggested by the definition.

We start from

x_1	...	x_k	y	0	...

we save the parameters and we start to compute $h(\vec{x}, 0)$ using F.

If $y = 0$ we are done, otherwise we save $h(\vec{x}, 0)$ and compute $h(\vec{x}, 1) = g(\vec{x}, 0, h(\vec{x}, 0))$ with G. We do the same for $h(\vec{x}, i)$ until we arrive at $i = y$.

As usual we need registers not used by F and G, $m = max\{\rho(F), \rho(G), k+2\}$ and we build the program for h as follows:

1	...	$m+1$	...	$m+k$	$m+k+1$	...	$m+k+3$	
...	...	...	$\vec{x}$	...	i	$h(\vec{x}, 2)$	y	0

$$\begin{array}{ll}
 & T(1, m+1) \\
 & \dots \\
 & T(k, m+k) \\
 & T(k+1, m+k+3) \\
 & F[m+1, \dots, m+k \to m+k+2] \qquad \texttt{// compute } h(\vec{x}, 0) \\
LOOP: & J(m+k+1, m+k+3, END) \qquad \texttt{// i=y?} \\
 & G[m+1, \dots, m+k+2 \to m+k+2] \\
 & S(m+k+1) \qquad \texttt{// i = i+1} \\
 & J(1, 1, LOOP) \\
END: & T(m+k+2, 1)
\end{array}$$

□

3. Smn-theorem

Exercise 3.1

Exercise 3.1(p). State the smn theorem and prove it (it is sufficient to provide the informal argument using encode/decode functions).

Theorem (smn-theorem) (according to Kleene)

Given $m, n \geq 1$ there is a total computable function $s_{m,n}: \mathbb{N}^{m+1} \to \mathbb{N}$ such that $\forall \vec{x} \in \mathbb{N}^m, \forall \vec{y} \in \mathbb{N}^n, \forall e \in \mathbb{N}$

$$\phi_e^{(m+n)}(\vec{x}, \vec{y}) = \phi_{s_{m,n}(e,\vec{x})}^{(n)}(\vec{y})$$

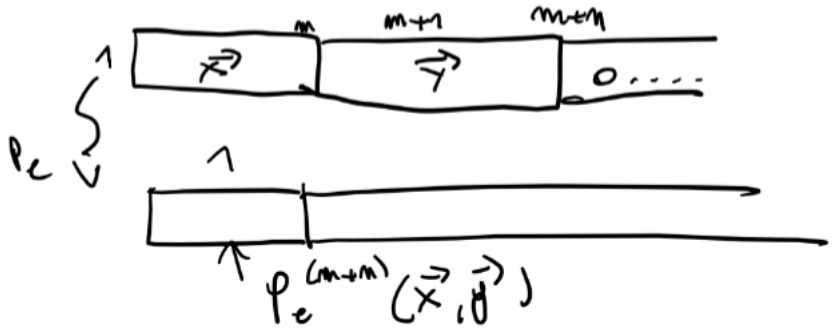
The smn-theorem states that given a function $g(x, y)$ which is computable, there exists a total and computable function s such that $\phi_{s(x)}(y) = g(x, y)$, basically "fixing" the first argument of g – usually, we fix x in favor of y. It's like partially applying an argument to a function. This is generalized over m, n tuples for x, y.

Basically, we have that the program over two indices m, n is the same as the transformed function over x inputs. Fixing an index, we obtain a program with the same features.

Proof

Intuitively, given $e \in \mathbb{N}, \vec{x} \in \mathbb{N}$

- We get the program $P_e = \gamma^{-1}(e)$ in standard form that computes $\phi_e^{(m+n)}$, so starting from the first drawing (this below), in which we compute the $\phi_e^{(m+n)}$ over all inputs $(\vec{x}, \vec{y})$ (so $\phi_e^{(m+n)}(\vec{x}, \vec{y})$)



The formal proof of computability is long and involves the combination of many parametrized functions as auxiliary to construct the smn function (which is primitive recursive for all indices). Consider each for cases:

1) sequential composition of programs

Given two program codes (e_1, e_2),we want the sequential encoding like $\rightarrow \gamma\binom{P_{e_1}}{P_{e_2}}$

Consider the *update* function:

$$upd: \mathbb{N}^2 \rightarrow \mathbb{N}$$

where $upd(e,h)$ = (code of the program obtained from $P_e = \gamma^{-1}(e)$ by updating all jump instructions $J(m,n,t)$ *to* $J(m,n,t+h)$). Basically, this updates a program's jump instructions based on e and h.

We define an auxiliary function working on each single instruction encoded with β:

$$upd^{\sim}: \mathbb{N}^2 \rightarrow \mathbb{N}$$

where $upd^{\sim}(i,h) = \beta$ (code of the instruction obtained from $\beta^{-1}(i)$, updating the target if it is a jump)

Note: $\beta\big(j(m,n,t)\big) = v(m-1, n-1, t-1) * 4 + 3$ (calculates an encoded value based on parameters m, n, t)

Given $i, h \in N, q = qt(4,i), r = rm(4,i)$, we formally define like:

$$upd^{\sim}(i,h) = \begin{cases} i, & if\ rm(4,i) \neq 3 \\ v(v_1(q), v_2(q), v_3(q)+h) * 4 + 3, & if\ rm(4,i) = 3, q = qt(4,i) \end{cases}$$

$$= i * sg(|rm(4,i) - 3|) + (v(v_1(q), v_2(q), v_3(q)+h) * 4 + 3) * \overline{sg}(|rm(4,i) - 3|)$$

Overall, this function updates the encoded instruction i based on whether it's a jump instruction or not, adjusting the jump target when required, calculating each time a new value thanks to the present triplet (which consider the two sign functions in order to "encode the two different characteristic functions, covering all cases in which will be either 1 or 0, deciding if it's useful to jump or not").

Now:

$$upd(e,h) = \tau\big(upd^{\sim}(a(e,1),h)\big) \quad upd^{\sim}(a(e,2),h) \quad upd^{\sim}\big(a\big(e,l(e)\big),h\big)$$

$$= \Big(\prod_{i=1}^{l(e)-1} p_i^{upd^{\sim}(a(e,i),h)}\Big) * p_i^{upd^{\sim}(a(e,l(e)),h)+1} - 2$$

This, in words, represents a formal way to express the update of a program by modifying its jump instructions, considering the effect of the other two instructions when applied recursively.

Basically, we are doing a kind of pattern matching via parametrization and making it computable thanks to composition and primitive recursion. The goal is to make the entire program computable by ensuring that jump targets are correctly adjusted, reflecting the desired program behavior, also doing some "fine-tuning" (the minus 2).

We encode the input sequence of values like the following, where each instruction in adjusted to the current one, the next one and the fine-tuning that we quoted above:

$$\tau(y_1, \dots y_m) = \prod_{i=1}^{n-1} P_i^{y_i} * P_m^{y_n+1} - 2$$

Consider $l(e)$ = length of the encoded sequence. For $1 \leq i \leq l(e)$, we get back the corresponding component, which is $a(e,i) = i^{th}\ component$.

Now we will use the *concatenation of sequences* as a function.

$$c: \mathbb{N}^2 \to \mathbb{N}$$

The concatenation of sequences follows here, considering the mapping of each encoding with

$$c(e_1, e_2) = \tau(a(e_1,1) \ldots . a(e_1, l(e_1))\ \ a(e_2,1) \ldots . a(e_2, l(e_2))$$

The *concatenation of programs* can be defined as:

$$seq: \mathbb{N}^2 \to \mathbb{N}$$

$$seq(e_1, e_2) = \gamma \begin{pmatrix} P_{e_1} \\ P_{e_2} \end{pmatrix} = c(e_1, upd(a, l(e_2)))$$

Essentially, in words, we concatenate the first one with the update over the length of the second one.

Then we use the *transfer function*:

2) $transf: \mathbb{N}^2 \to \mathbb{N}$

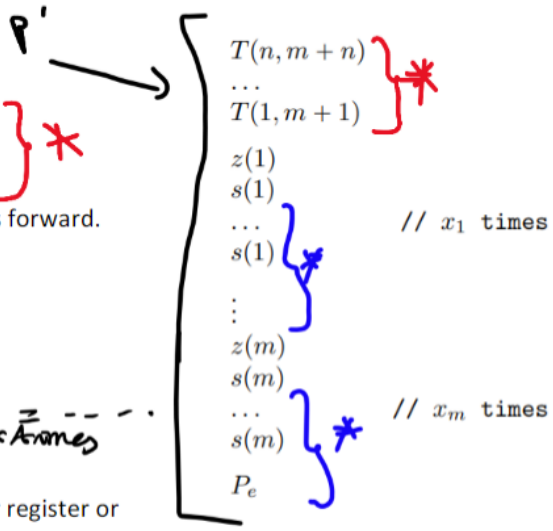
$$transf(m,n) = \gamma \begin{pmatrix} T(n, n+m) \\ T(1, m+1) \end{pmatrix}$$

Technically, this one simply shift registers n positions forward.

Continuing, we will use the *set function*:

3) $set: \mathbb{N}^2 \to \mathbb{N}$

$$set(i,x) = \gamma \begin{pmatrix} Z(i) \\ S(i) \\ S(i) \end{pmatrix}$$



It allows you to set a specific value x into a particular register or location i and possibly perform some operations like incrementing the value.

Finally, this brings us to:

$$s_{m,n}(e, \vec{x}) =$$

$$seq(transf(m,n),$$

$$seq(set(1, x_1),$$

$$\ldots\ldots$$

$$seq(set(m, x_m), e) \ldots)$$

All of this mess can refer to the following *prefix* function because it defines a sequence concatenating multiple sub-sequences "prefixing" other ones, hence obtaining the final function.

This proves that the smn-function is computable and total (actually, we can observe it's also primitive recursive) since it is a composition of primitive recursive functions, themselves effective.

Exercise 3.3

Exercise 3.3. State the smn theorem and use it to prove that there exists a total computable function $s : \mathbb{N}^2 \to \mathbb{N}$ such that $W_{s(x,y)} = \{z : x * z = y\}$

Proof:

The smn theorem states that given computable functions $g : \mathbb{N}^{m+n} \to \mathbb{N}$ and $h : \mathbb{N}^m \to \mathbb{N}$, there exists a total computable function $s : \mathbb{N}^{m+1} \to \mathbb{N}$ such that

$$\phi^{(n)}_{s(e,\vec{x})}(\vec{y}) = \phi^{(m+n)}_e(\vec{x},\vec{y}) = g(h(\vec{x}),\vec{y})$$

for all $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$, and $\vec{y} \in \mathbb{N}^n$.

To prove the theorem, we start by defining the function $f : \mathbb{N}^3 \to \mathbb{N}$ as follows:

$$f(x,y,z) = \begin{cases} 0 & \text{if } x * z = y \\ \uparrow & \text{otherwise} \end{cases}$$

which can be written in terms of the computable functions product, equality and minimization:

$$f(x,y,z) = \mu w.\, |x * z - y|$$

Therefore, f is a computable function.

Now, by the smn theorem, there exists a total computable function $s : \mathbb{N}^2 \to \mathbb{N}$ such that for all $x, y, z \in \mathbb{N}$:

$$\phi_{s(x,y)}(z) = f(x,y,z)$$

We can then conclude that for any $x, y \in \mathbb{N}$:

$$\begin{aligned} W_{s(x,y)} &= \{z \in \mathbb{N} : \phi_{s(x,y)}(z) \downarrow\} \\ &= \{z \in \mathbb{N} : f(x,y,z) = 0\} \\ &= \{z \in \mathbb{N} : x * z = y\} \end{aligned}$$

which proves the theorem. □

6. Functions and Computability

Exercise 6.13

Exercise 6.13. Say if there is a total non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$f(x) \neq \varphi_x(x)$$

only on a single argument $x \in \mathbb{N}$. If the answer is negative provide a proof, if the answer is positive give an example of such a function.

To prove there exists a total non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x) \neq \phi_x(x)$ only on a single argument $x \in \mathbb{N}$, consider the following function:

$$f(x) = \begin{cases} \phi_x(x) + 1 & \text{if } \phi_x(x) \downarrow \\ 0 & \text{if } \phi_x(x) \uparrow \end{cases}$$

This function f is total by definition. Moreover, f differs from every computable function ϕ_x on the argument x:

- If $\phi_x(x) \downarrow$, then $f(x) = \phi_x(x) + 1 \neq \phi_x(x)$
- If $\phi_x(x) \uparrow$, then $f(x) = 0 \neq \phi_x(x)$

Therefore, f is a total non-computable function that differs from each computable function ϕ_x only on the single argument x.

Exercise 6.14

Exercise 6.14. Is there non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$f(x) \neq \varphi_x(x)$$

only on a single $x \in \mathbb{N}$? If the answer is negative provide a proof of non-existence, otherwise give an example of such a function.

To show that there is no non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x) \neq \phi_x(x)$ only on a single $x \in \mathbb{N}$, suppose for contradiction that such an f exists.

Let $x_0 \in \mathbb{N}$ be the unique value where $f(x_0) \neq \phi_{x_0}(x_0)$. Define a new function $g : \mathbb{N} \to \mathbb{N}$ as follows:

$$g(x) = \begin{cases} f(x) & \text{if } x \neq x_0 \\ \phi_{x_0}(x_0) & \text{if } x = x_0 \end{cases}$$

Observe that g is computable, since it differs from the non-computable function f only at the single argument x_0, where it takes on the computable value $\phi_{x_0}(x_0)$.

However, by construction, $g(x_0) = \phi_{x_0}(x_0)$ and for all $x \neq x_0$, $g(x) = f(x) = \phi_x(x)$. Therefore, $g = \phi_{x_0}$, contradicting the assumption that f differs from every computable function on some argument.

Hence, our initial assumption must be false, and there cannot exist a non-computable function that differs from every computable function on only a single argument.

Exercise 6.16

Exercise 6.16. Say if there is a non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that the set $D = \{x \in \mathbb{N} \mid f(x) \neq \phi_x(x)\}$ is finite. Justify your answer.

To show that there cannot exist a non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that the set $D = \{x \in \mathbb{N} | f(x) \neq \phi_x(x)\}$ is finite, we will prove that the existence of such a function leads to a contradiction.

Suppose for the sake of argument that such a function f exists. Then we can define a new function $g : \mathbb{N} \to \mathbb{N}$ as follows:

$$g(x) = \begin{cases} f(x) & \text{if } x \notin D \\ f(x) + 1 & \text{if } x \in D \end{cases}$$

Since D is finite by assumption, g differs from f only on a finite number of inputs. This implies that g is also non-computable, as a finite change to a non-computable function cannot make it computable.

However, by construction, for all $x \in \mathbb{N}$ we have:

- If $x \notin D$, then $g(x) = f(x) \neq \phi_x(x)$, and thus $g \neq \phi_x$
- If $x \in D$, then $g(x) = f(x) + 1 \neq f(x) = \phi_x(x)$, and thus $g \neq \phi_x$

Therefore, g differs from every computable function ϕ_x on input x. By the diagonalization theorem, this implies that g is in fact computable, contradicting our earlier observation that g is non-computable.

We conclude that our initial assumption must be false, i.e., there cannot exist a non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that the set $D = \{x \in \mathbb{N} | f(x) \neq \phi_x(x)\}$ is finite. The function f simply cannot exist under these constraints, as its existence would lead to a logical contradiction.

Exercise 6.20

Exercise 6.20. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(x) = \begin{cases} x + 2 & \text{if } \varphi_x(x) \downarrow \\ \dot{x-1} & \text{otherwise} \end{cases}$$

Is it computable? Justify your answer.

Let's analyze the function $f : \mathbb{N} \to \mathbb{N}$ defined as:

$$f(x) = \begin{cases} x + 2 & \text{if } \varphi_x(x) \downarrow \\ x - 1 & \text{otherwise} \end{cases}$$

To determine if f is computable, we can express it using the computable functions and operations we know.

First, note that the predicate "$\varphi_x(x) \downarrow$" is equivalent to "$x \in K$", where $K = \{x \mid x \in W_x\}$ is the halting set. We know that K is recursively enumerable (r.e.) but not recursive.

Let's denote the characteristic function of K as χ_K, which is defined as:

$$\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{otherwise} \end{cases}$$

Although χ_K is not computable, we can use it to express f:

$$\begin{aligned} f(x) &= (x + 2) \cdot \chi_K(x) + (x - 1) \cdot (1 - \chi_K(x)) \\ &= (x + 2) \cdot \chi_K(x) + (x - 1) - (x - 1) \cdot \chi_K(x) \\ &= x + 2 \cdot \chi_K(x) - 1 + \chi_K(x) \\ &= x + 3 \cdot \chi_K(x) - 1 \end{aligned}$$

If χ_K were computable, then f would also be computable as a composition of computable functions. However, since χ_K is not computable, we cannot conclude that f is computable.

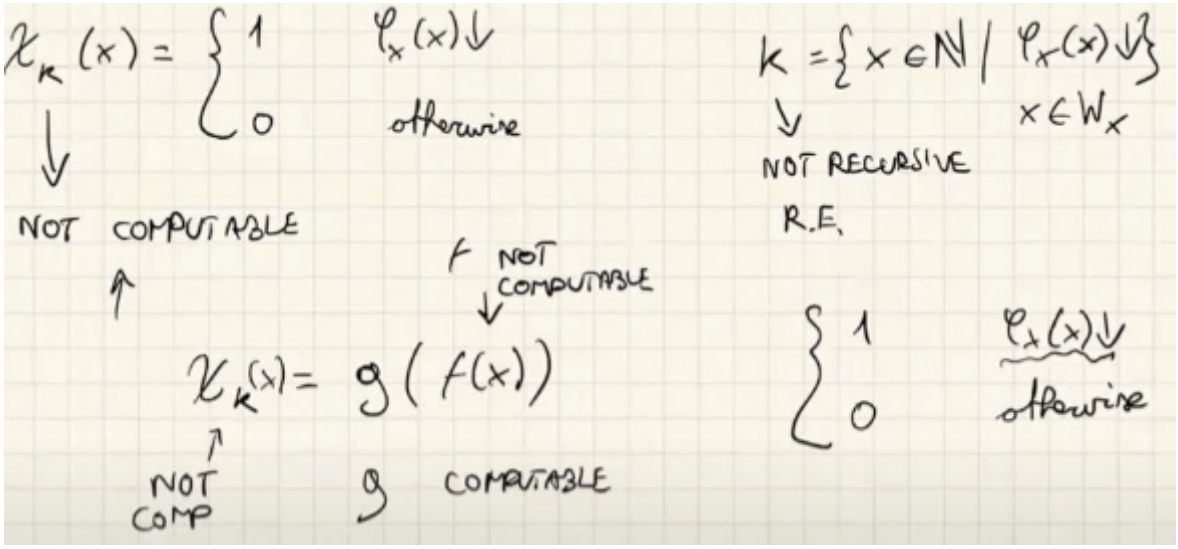
In fact, if f were computable, then we could compute χ_K using:

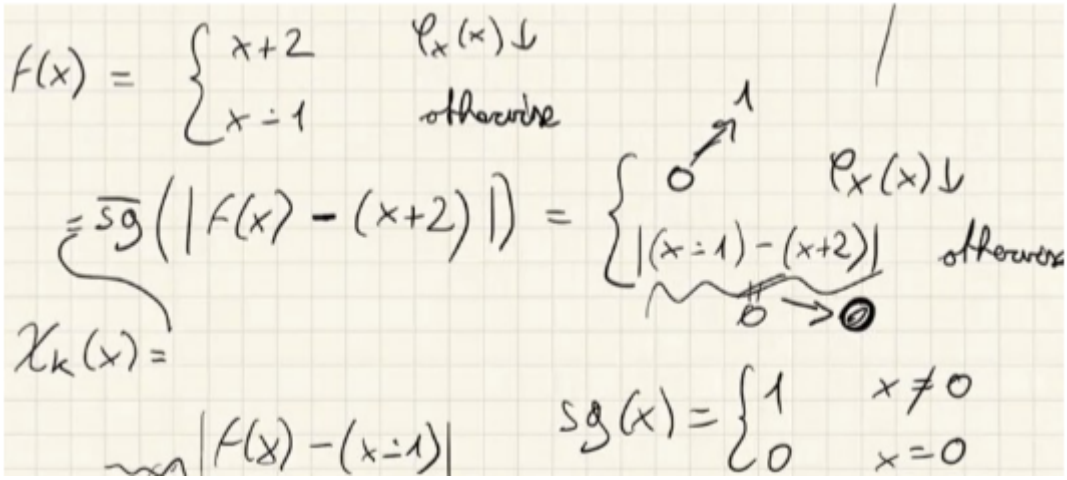
$$\chi_K(x) = \frac{f(x) - x + 1}{3}$$

But this contradicts the fact that χ_K is not computable.

Therefore, the function f is not computable.

Similarly, it was solved by an old tutor the following way:





Exercise 6.21

Exercise 6.21. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(x) = \begin{cases} \varphi_x(x+1) + 1 & \text{if } \varphi_x(x+1) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Is it computable? Justify your answer.

Assume by contradiction that f is computable. Then there exists an index e such that $f = \phi_e$.

Therefore, for any $x \in \mathbb{N}$:

$$\phi_e(x) = \begin{cases} \phi_x(x+1) + 1 & \text{if } \phi_x(x+1) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

In particular, for $x = e$:

$$\phi_e(e) = \begin{cases} \phi_e(e+1)+1 & \text{if } \phi_e(e+1)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Now consider two cases:

1. If $\phi_e(e+1)\downarrow$, then $\phi_e(e) = \phi_e(e+1)+1 > \phi_e(e+1)$
2. If $\phi_e(e+1)\uparrow$, then $\phi_e(e)\uparrow$

In either case, we cannot have $f = \phi_e$.

Conclusion: f is not computable.

Exercise 6.22

Exercise 6.22. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(x) = \begin{cases} \varphi_x(x)+1 & \text{if } \varphi_y(y)\downarrow \text{ for each } y \leqslant x \\ 0 & \text{otherwise} \end{cases}$$

Is it computable? Justify your answer.

Assume by contradiction that f is computable. Then there exists an index e such that $f = \phi_e$.

Let's consider $x = e$. Then:

1. If $\phi_y(y)\downarrow$ for all $y \leq e$, then: $\phi_e(e) = f(e) = \phi_e(e)+1$ This is a contradiction as no natural number equals its successor.
2. If there exists some $y \leq e$ such that $\phi_y(y)\uparrow$, then: $\phi_e(e) = f(e) = 0$ This means $\phi_e(e)\downarrow$, contradicting the fact that $\phi_y(y)\uparrow$ for some $y \leq e$.

Therefore, we cannot have $f = \phi_e$ for any e.

Conclusion: f is not computable.

7. Reduction, Recursiveness and Recursive Enumerability

Exercise 7.11

Exercise 7.11. Let $\pi : \mathbb{N}^2 \to \mathbb{N}$ be the function encoding pairs of natural numbers into the natural numbers. Prove that a function $f : \mathbb{N} \to \mathbb{N}$ is computable if and only if the set $A_f = \{\pi(x, f(x)) \; x \in \mathbb{N}\}$ is recursively enumerable.

($\Rightarrow$) First, let's prove if f is computable then A_f is r.e.

If f is computable, then there exists an index e such that $f = \phi_e$. Define:

$$sc_{A_f}(z) = \mu w.\,(S(e, (\pi_1(z)), \pi_2(z), (w)_1))$$

where π_1, π_2 are the projection functions for the pair encoding.

This function is computable because:

- S is decidable (step counting predicate)
- π_1, π_2 are computable
- The minimal search μ preserves computability

Therefore A_f is r.e.

($\Leftarrow$) Now let's prove if A_f is r.e. then f is computable.

If A_f is r.e., then there exists an index e such that:

$$A_f = W_e = \{\pi(x, f(x)) | x \in \mathbb{N}\}$$

We can define f as:

$$f(x) = \pi_2(\mu z.\, (z \in W_e \wedge \pi_1(z) = x))$$

This function is computable because:

- W_e is r.e.
- π_1, π_2 are computable
- The minimal search μ is over a semi-decidable predicate
- The equality $\pi_1(z) = x$ is decidable

Therefore f is computable.

Conclusion: f is computable if and only if A_f is recursively enumerable.

Exercise 7.12

Exercise 7.12. Prove that a set $A \subseteq \mathbb{N}$ is recursive if and only if $A \leqslant_m \{0\}$.

($\Rightarrow$) Assume A is recursive. Then its characteristic function χ_A is computable. Define the reduction function $f : \mathbb{N} \to \mathbb{N}$ as $f(x) = 1 - \chi_A(x)$. Clearly, f is computable, and $x \in A \Leftrightarrow f(x) = 0 \Leftrightarrow f(x) \in 0$. Thus, $A \leq_m 0$.

($\Leftarrow$) Assume $A \leq_m 0$ via a computable function f. Then $x \in A \Leftrightarrow f(x) = 0$. Define $\chi_A(x) = 1 - \mathrm{sg}(f(x))$, where sg is the signum function. Since f and sg are computable, χ_A is computable, and thus A is recursive.

Exercise 7.13

Exercise 7.13. Let $A \subseteq \mathbb{N}$ be a non-empty set. Prove that A is recursively enumerable if and only if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $dom(f)$ is the set of prime numbers and $img(f) = A$.

($\Rightarrow$) Assume A is recursively enumerable. Then there exists a computable function $g : \mathbb{N} \to \mathbb{N}$

such that $\text{img}(g) = A$. Let p_i denote the i-th prime number. Define $f : \mathbb{N} \to \mathbb{N}$ as $f(p_i) = g(i)$ for all $i \in \mathbb{N}$. Clearly, $\text{dom}(f)$ is the set of prime numbers and $\text{img}(f) = \text{img}(g) = A$.

($\Leftarrow$) Assume there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $\text{dom}(f)$ is the set of prime numbers and $\text{img}(f) = A$. Define $g : \mathbb{N} \to \mathbb{N}$ as $g(i) = f(p_i)$ for all $i \in \mathbb{N}$, where p_i is the i-th prime number. Since the sequence of prime numbers is computable, g is computable, and $\text{img}(g) = \text{img}(f) = A$. Thus, A is recursively enumerable.

Exercise 7.19

Exercise 7.19. Prove that a set $A \subseteq \mathbb{N}$ is recursive if and only if A and $\bar{A}$ are r.e.

($\Rightarrow$) Assume A is recursive. Then its characteristic function χ_A is computable. Define the semi-characteristic functions $\text{sc}_A(x) = \chi_A(x)$ and $\text{sc}_{\overline{A}}(x) = 1 - \chi_A(x)$. Both are computable, so A and $\overline{A}$ are r.e.

($\Leftarrow$) Assume A and $\overline{A}$ are r.e. Then there exist computable semi-characteristic functions sc_A and $\text{sc}_{\overline{A}}$. Define the characteristic function $\chi_A(x) = \mu y.\, [\text{sc}_A(x) = 1 \vee \text{sc}_{\overline{A}}(x) = 1]$. Since $x \in A \cup \overline{A}$ for all $x \in \mathbb{N}$, the minimalization always terminates, and χ_A is computable. Thus, A is recursive.

Exercise 7.20

Exercise 7.20. State and prove Rice's theorem(without using the second recursion theorem).

Rice's theorem states that for any non-trivial property of partial computable functions, the set of indices of functions with that property is undecidable.

Formally, let C be the set of all partial computable unary functions and $A \subseteq C$ such that:

1. $A \neq \emptyset$
2. $A \neq C$
3. A is extensional, i.e., if $f, g \in C$ and $f =^* g$ then $f \in A \Leftrightarrow g \in A$, where $f =^* g$ means $f(x) = g(x)$ whenever $f(x)$ and $g(x)$ are both defined.

Then the index set $I_A = e \in \mathbb{N} \mid \varphi_e \in A$ is not recursive (i.e., not decidable).

PROOF. We show that $K \leqslant_m A$. Let e_0 such that $\phi_{e_0}(x) \uparrow \forall x$. We distinguish two cases depending on whether $e \in A$ or not.

($e_0 \notin A$) Suppose $e_0 \notin A$ and let $e_1 \in A (\neq \varnothing)$. Now define

$$g(x,y) = \begin{cases} \phi_{e_1}(y) & x \in K \\ \phi_{e_0}(y) & x \notin K \end{cases}$$
$$= \begin{cases} \phi_{e_1}(y) & x \in K \\ \uparrow & x \notin K \end{cases}$$
$$= \phi_{e_1}(y) \cdot \mathbf{1}(\Psi_U(x,x))$$

it is computable. By *smn* theorem there is $s\mathbb{N} \to \mathbb{N}$ such that $\phi_{s(x)}(y) = g(x,y)$.

Now observe that s is a reduction function for $K \leqslant_m A$

- $x \in K \Rightarrow \forall y\ \varphi_{s(x)}(y) = \varphi_{e_1}(y) \Rightarrow s(x) \in A$
- $x \notin K \Rightarrow \forall y\ \varphi_{s(x)}(y) = \varphi_{e_0}(y) \uparrow \Rightarrow s(x) \notin A$

Hence $K \leqslant_m A$, K not recursive, thus A.

($e_0 \in A$) If $e_0 \in A$ then $e_0 \notin \bar{A}$. Then $\bar{A} \subseteq \mathbb{N}, \bar{A} \neq \varnothing, \bar{A} \neq \mathbb{N}$, and $\bar{A}$ is saturated since A is. Therefore, by the first part,o $\bar{A}$ is not recursive, and therefore A is not recursive either.

□

Exercise 7.21

Exercise 7.21. Define what it means for a set $A \subseteq \mathbb{N}$ to be saturated and prove that K is not is saturated.

A set $A \subseteq \mathbb{N}$ is saturated if for all $x, y \in \mathbb{N}$, $\varphi_x = \varphi_y$ implies $x \in A \Leftrightarrow y \in A$.

To prove that K is not saturated, we show there exist $e_1, e_2 \in \mathbb{N}$ such that $\varphi_{e_1} = \varphi_{e_2}$ but $e_1 \in K$ and $e_2 \notin K$.

Consider the computable function $g(x,y) = 1$ if $x = y$, and undefined otherwise. By the s-m-n theorem, there exists a computable function $s : \mathbb{N} \to \mathbb{N}$ such that $\varphi_{s(x)}(y) = g(x,y)$ for all $x, y \in \mathbb{N}$.

By the Recursion Theorem, there exists e such that $\varphi_e = \varphi_{s(e)}$. Thus, $\varphi_e(y) = 1$ if $y = e$, and undefined otherwise.

Now, $e \in K$ as $\varphi_e(e) = 1$. Choose any $e' \neq e$ such that $\varphi_{e'} = \varphi_e$. Then $\varphi_{e'}(e') \uparrow$, so $e' \notin K$. Therefore, K is not saturated.

PROOF. First observe that there is n_0 s.t. $\varphi_{n_0} = \{(n_0, n_0)\}$. In fact, define

$$g(n,x) = \begin{cases} 0 & \text{if } x = n \\ \uparrow & \text{otherwise} \end{cases} = \mu z.\ |x - n|$$

Since g is computable there is $s : \mathbb{N} \to \mathbb{N}$ total computable such that $\varphi_{s(n)}(x) = g(x,y)$. By the Second Recursion Theorem there is n_0 such that $\varphi_{n_0} = \varphi_{s(n_0)}$. Therefore

$$\varphi_{n_0}(x) = \varphi_{s(n_0)}(x) = g(n_0, x) = \begin{cases} 0 & \text{if } x = n_0 \\ \uparrow & \text{otherwise} \end{cases}$$

Observe that $n_0 \in K$. Moreover we know that there are infinitely many indices for the same function. Thus let $n \neq n_0$ s.t. $\varphi_n = \varphi_{n_0}$. Then

$$\varphi_n(n) = \varphi_{n_0}(n) \uparrow$$

Hence $n \notin K$ and thus K is not saturated. □

Exercise 7.22

> **Exercise 7.22.** Let $\mathcal{A} \subseteq \mathcal{C}$ be a set of functions computable and let $f \in \mathcal{A}$ such that for any function over $\theta \subseteq f$ is worth $\theta \notin \mathcal{A}$. Prove that $A = \{x \in \mathbb{N} \mid \varphi_x \in \mathcal{A}\}$ is not r.e.

Proof by contradiction. Assume I_A is r.e. Choose $e_0 \in \mathbb{N}$ such that φ_{e_0} is the everywhere undefined function.

Define $g : \mathbb{N}^2 \to \mathbb{N}$ as:

$$g(x, y) = \begin{cases} \varphi_{e_0}(y) & \text{if } x \notin K \\ f(y) & \text{if } x \in K \end{cases}$$

Clearly, g is computable. By the s-m-n theorem, there exists a computable function $s : \mathbb{N} \to \mathbb{N}$ such that $\varphi_{s(x)}(y) = g(x, y)$ for all $x, y \in \mathbb{N}$.

Now, for any $x \in \mathbb{N}$:

- If $x \notin K$, then $\varphi_{s(x)} = \varphi_{e_0} \notin A$, so $s(x) \notin I_A$.
- If $x \in K$, then $\varphi_{s(x)} = f \in A$, so $s(x) \in I_A$.

Therefore, $x \in K \Leftrightarrow s(x) \in I_A$. Since K is not r.e., this contradicts the assumption that I_A is r.e.

8. Characterization of sets

Exercise 8.1

> **Exercise 8.1.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : |W_x| \geqslant 2\}$, i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

We will prove that neither A nor $\bar{A}$ are recursively enumerable (r.e.).

1. First, observe that A is saturated since $A = x|\phi_x \in \mathcal{A}$ where $\mathcal{A} = f \in \mathcal{C} : |dom(f)| \geq 2$.
2. Let's prove A is not r.e. using Rice-Shapiro theorem:
   - Let id be the identity function. Note that $id \notin A$ since $|dom(id)| = \infty \ngeq 2$
   - Define $\theta \subset id$ as:

$$\theta(x) = \begin{cases} x & \text{if } x \leq 1 \\ \uparrow & \text{otherwise} \end{cases}$$

   - Then $\theta \subset id$ and $\theta \in A$ since $|dom(\theta)| = 2$
   - By Rice-Shapiro theorem, A is not r.e.
3. Let's prove $\bar{A}$ is not r.e.:
   - Consider θ as defined above. Clearly $\theta \notin \bar{A}$
   - Let $\emptyset$ be the empty function. Then $\emptyset \subset \theta$ and $\emptyset \in \bar{A}$ since $|dom(\emptyset)| = 0 < 2$
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore, since neither A nor $\bar{A}$ are r.e., A is not recursive.

Exercise 8.2

> **Exercise 8.2.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : x \in W_x \cap E_x\}$, i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

We will prove that A is r.e. but not recursive.

1. First, let's prove A is r.e.:
   - Define the semi-characteristic function:

$$sc_A(x) = \begin{cases} 1 & \text{if } x \in W_x \cap E_x \\ \uparrow & \text{otherwise} \end{cases}$$

   - This is computable since:

$$sc_A(x) = 1(\mu w.\, (H(x, x, w) \wedge S(x, x, x, w)))$$

2. To prove A is not recursive, we show $K \leq_m A$:
   - Define:

$$g(x, y) = \begin{cases} x & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

   - By s-m-n theorem, $\exists s$ total computable such that $\forall x, y : \phi_{s(x)}(y) = g(x, y)$
   - Then s is a reduction function because:
     - If $x \in K$: $\phi_{s(x)}(s(x)) = x$ thus $s(x) \in W_{s(x)} \cap E_{s(x)}$, so $s(x) \in A$
     - If $x \notin K$: $\phi_{s(x)}(s(x)) \uparrow$ thus $s(x) \notin W_{s(x)}$, so $s(x) \notin A$

Therefore A is r.e. but not recursive, and consequently $\bar{A}$ is not r.e.

Exercise 8.5

> **Exercise 8.5.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : \exists y, z \in \mathbb{N}.\ z > 1 \ \wedge \ x = y^z\}$, i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

We'll show that $K \leq_m A$ to prove A is not recursive.

Define a function $g(x, y)$ as:

$$g(x, y) = \begin{cases} 2^2 & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

By s-m-n theorem, $\exists$s total computable such that $\phi_{s(x)}(y) = g(x, y)$. Then s is a reduction function because:

- If $x \in K$ then $s(x) = 4$ which can be written as 2^2, so $s(x) \in A$
- If $x \notin K$ then $\phi_{s(x)}$ is undefined everywhere, so $s(x) \notin A$

Therefore A is not recursive. However, A is r.e. since:

$$sc_A(x) = 1(\mu\langle y, z, w\rangle.\,(z > 1 \wedge x = y^z))$$

Given A is r.e. but not recursive, $\bar{A}$ is also not recursive (otherwise both would be recursive) and also r.e.

Exercise 8.6

> **Exercise 8.6.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : \phi_x(y) = y \text{ for infinitely many } y\}$, i.e., establish if A and $\bar{A}$ are recursive/recursive enumerable.

The set A is saturated since $A = \{x \mid \varphi_x \in \mathcal{A}\}$ where
$\mathcal{A} = f \in \mathcal{C} : f(y) = y \text{ for infinitely many } y.$

By Rice-Shapiro theorem:

1. A is not r.e.:
   - Let id be the identity function. Then $id \in A$
   - Let $\theta \subset id$ be any finite subfunction. Then $\theta \notin A$
   - By Rice-Shapiro theorem, A is not r.e.
2. $\bar{A}$ is not r.e.:
   - Let $f(x) = x + 1$. Then $f \notin A$ (thus $f \in \bar{A}$)
   - Let $\emptyset \subset f$. Then $\emptyset \in A$ (thus $\emptyset \notin \bar{A}$)
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore A is not recursive.

Exercise 8.7

> **Exercise 8.7.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : W_x \subseteq E_x\}$, i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

The set A is saturated since $A = \{x \mid \varphi_x \in \mathcal{A}\}$ where $\mathcal{A} = f \in \mathcal{C} : dom(f) \subseteq cod(f)$.

By Rice-Shapiro theorem:

1. A is not r.e.:
   - Let $f(x) = x + 1$. Then $f \notin A$ since $\mathbb{N} \nsubseteq \mathbb{N} \setminus 0$
   - Let $\emptyset$ be the empty function. Then $\emptyset \subset f$ and $\emptyset \in A$
   - By Rice-Shapiro theorem, A is not r.e.
2. $\bar{A}$ is not r.e.:
   - Let id be the identity function. Then $id \in A$, so $id \notin \bar{A}$
   - Let $\theta \subset id$ where $dom(\theta) \nsubseteq cod(\theta)$. Then $\theta \in \bar{A}$
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore A is not recursive.

This was also solved by prof. Baldan in some older lessons:

(8.7) $A = \{ x \mid W_x \subseteq E_x \}$

* A is saturated

$A = \{ x \mid \varphi_x \in \mathcal{A} \}$ $\quad \mathcal{A} = \{ f \mid dom(f) \subseteq cod(f) \}$

* A is not re.

$\mathbb{1} \notin \mathcal{A}$ $\quad dom(\mathbb{1}) = \mathbb{N} \not\subseteq \{1\} = cod(\mathbb{1})$

$\vartheta = \emptyset \subseteq \mathbb{1}$ $\quad dom(\vartheta) = \emptyset \subseteq \emptyset = cod(\vartheta) \Rightarrow \vartheta \in \mathcal{A}$

$\Rightarrow$ A is not re. (by Rice-Shapiro)

* $\bar{A}$ is not re.

$pred(x) = x \dot{-} 1$ $\quad dom(pred) = \mathbb{N} \subseteq cod(pred) = \mathbb{N}$

$\downarrow pred \in \mathcal{A} \Rightarrow pred \notin \bar{A}$

$\vartheta = \begin{cases} 0 & x \leq 1 \\ \uparrow & \text{otherwise} \end{cases} = \begin{cases} x \dot{-} 1 & x \leq 1 \\ \uparrow & \text{otherwise} \end{cases}$ $\quad \vartheta \in pred$ finite

$dom(\vartheta) = \{0,1\} \not\subseteq \{0\} = cod(pred)$

$\vartheta \notin \mathcal{A} \Rightarrow \vartheta \in \bar{A}$

$\Rightarrow \bar{A}$ is not re.

Hence $A, \bar{A}$ are neither re. $\square$

(we could have considered

$f(x) = \begin{cases} 1 & x=0 \\ 0 & x=1 \\ \uparrow & \text{otherwise} \end{cases}$

$\vartheta(x) = \begin{cases} 1 & x=0 \\ \uparrow & \text{otherwise} \end{cases}$

$f \notin \bar{A}$ $\quad \vartheta \subseteq f$ $\quad \vartheta \in \bar{A}$)

$g(x,y)$ computable

by smn theorem (corollary) $\exists$ total computable $s: \mathbb{N} \to \mathbb{N}$ such that

$\varphi_{s(x)}(y) = g(x,y)$

Exercise 8.8

> **Exercise 8.8.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : |W_x| > |E_x|\}$, i.e. establish whether A and $\bar{A}$ are recursive/recursively enumerable.

The set A is saturated since $A = \{x \mid \phi_x \in \mathcal{A}\}$ where $\mathcal{A} = f \in \mathcal{C} : |dom(f)| > |cod(f)|$.

By Rice-Shapiro theorem:

1. A is not r.e.:
   - Let id be the identity function. Then $id \notin A$
   - Let $\theta \subset id$ with $dom(\theta) > cod(\theta)$. Then $\theta \in A$
   - By Rice-Shapiro theorem, A is not r.e.
2. $\bar{A}$ is not r.e.:
   - Let f with finite domain be in A. Then $f \notin \bar{A}$
   - Let $\emptyset \subset f$. Then $\emptyset \in \bar{A}$
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore A is not recursive.

Exercise 8.12

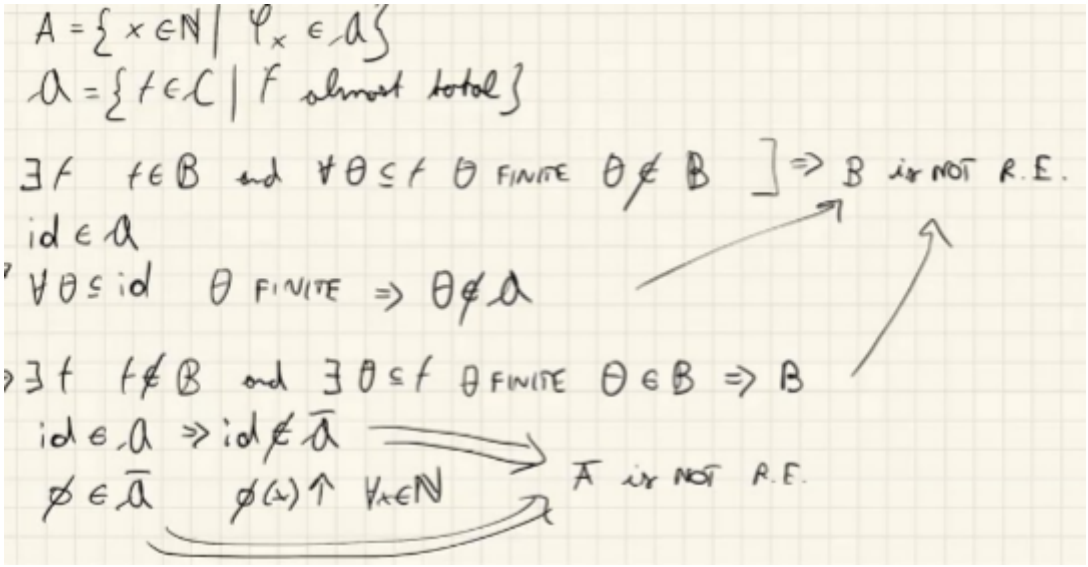
> **Exercise 8.12.** Say that a function $f : \mathbb{N} \to \mathbb{N}$ is *almost total* if it is undefined on a finite set of points. Study the recursiveness of the set $A = \{x \mid \varphi_x \text{ almost total}\}$, i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

The set A is saturated since $A = x|\phi_x \in \mathcal{A}$ where $\mathcal{A} = f \in \mathcal{C} : |\mathbb{N} \setminus dom(f)| < \infty$.

1. Let's prove A is not r.e. using Rice-Shapiro theorem:
   - Let id be the identity function. Then $id \in A$ (since $dom(id) = \mathbb{N}$)
   - Let $\theta \subset id$ be any finite subfunction of id. Then $\theta \notin A$ since $|\mathbb{N} \setminus dom(\theta)| = \infty$
   - By Rice-Shapiro theorem, A is not r.e.
2. Let's prove $\bar{A}$ is not r.e.:
   - Let $f \notin A$ (so $f \in \bar{A}$)
   - Let $\emptyset \subset f$ be the empty function. Then $\emptyset \in A$ (so $\emptyset \notin \bar{A}$)
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore, since neither A nor $\bar{A}$ are r.e., A is not recursive (and also $\bar{A}$ otherwise both would be recursive).

This was also solved by an old tutor:



Exercise 8.13

> **Exercise 8.13.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : W_x \cap E_x = \varnothing\}$, i.e., establish whether A and $\bar{A}$ are recursive/recursively enumerable.

The set A is saturated since $A = x|\phi_x \in \mathcal{A}$ where $\mathcal{A} = f \in \mathcal{C} : dom(f) \cap cod(f) = \emptyset$.

1. Let's prove A is not r.e. using Rice-Shapiro theorem:
   - Consider id. Then $id \notin A$ since $dom(id) \cap cod(id) = \mathbb{N} \neq \emptyset$
   - Let $\emptyset$ be the empty function. Then $\emptyset \subset id$ and $\emptyset \in A$
   - By Rice-Shapiro theorem, A is not r.e.
2. Let's prove $\bar{A}$ is not r.e.:
   - Let f be any function where $dom(f) \cap cod(f) \neq \emptyset$. Then $f \in \bar{A}$
   - Let $\emptyset \subset f$. Then $\emptyset \in A$ (so $\emptyset \notin \bar{A}$)
   - By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore A is not recursive (and $\bar{A}$ also, otherwise both would be recursive).

Exercise 8.22

> **Exercise 8.22.** Study the recursiveness of the set $A = \{x \in \mathbb{N} : \varphi_x(y) = y \text{for infinities} y\}$, that is, say if A and $\bar{A}$ are recursive/recursive enumerable.

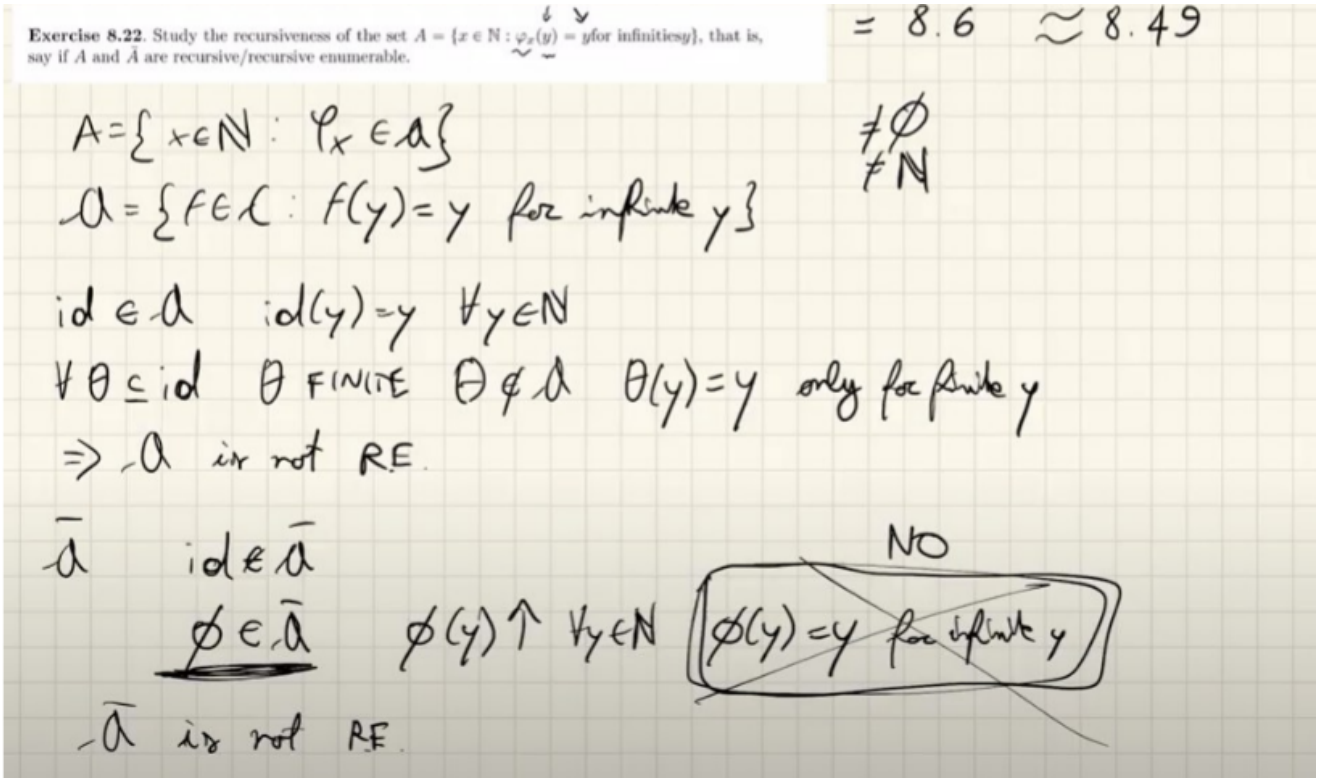
The set A is saturated since $A = x|\phi_x \in \mathcal{A}$ where $\mathcal{A} = f \in \mathcal{C} : |y : f(y) = y| = \infty$.

1. Let's prove A is not r.e. using Rice-Shapiro theorem:
   - Let id be the identity function. Then $id \in A$
   - Any finite subfunction $\theta \subset id$ has only finitely many fixed points, so $\theta \notin A$
   - By Rice-Shapiro theorem, A is not r.e.
2. Let's prove $\bar{A}$ is not r.e.:
   - Let $f(x) = x + 1$. Then $f \in \bar{A}$ since it has no fixed points

- Let $\emptyset \subset f$. Then $\emptyset \in A$ (so $\emptyset \notin \bar{A}$)
- By Rice-Shapiro theorem, $\bar{A}$ is not r.e.

Therefore A is not recursive.

This was also solved by an old tutor:



Exercise 8.35

> **Exercise 8.35.** Study the recursiveness of the set $B = \{x \in \mathbb{N} : x \in E_x\}$, i.e., establish if B and $\bar{B}$ are recursive/recursively enumerable.

Let's prove B is r.e. but not recursive by showing $K \leq_m B$.

1. First, B is r.e. since:

   $$sc_B(x) = 1(\mu\langle w, t\rangle . S(x, w, x, t))$$

   is computable.
2. To prove B is not recursive, we show $K \leq_m B$: Define $g(x, y)$:

   $$g(x, y) = \begin{cases} x & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

   By s-m-n theorem, $\exists s$ total computable such that $\phi_{s(x)}(y) = g(x, y)$. Then:
   - If $x \in K$: $\phi_{s(x)}(s(x)) = x$, so $s(x) \in E_{s(x)}$, thus $s(x) \in B$
   - If $x \notin K$: $\phi_{s(x)}$ undefined everywhere, so $s(x) \notin E_{s(x)}$, thus $s(x) \notin B$

Therefore B is r.e. but not recursive, and consequently $\bar{B}$ is not r.e.

Exercise 8.36

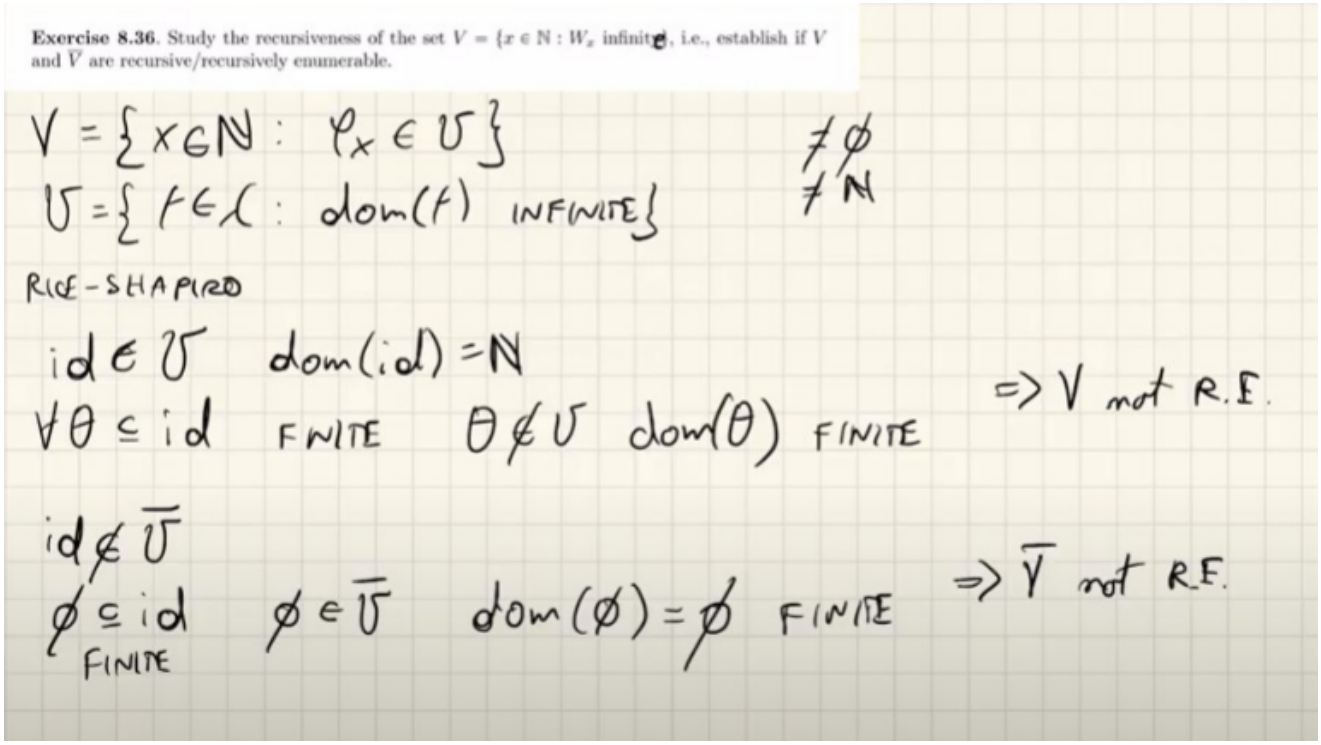
Exercise 8.36. Study the recursiveness of the set $V = \{x \in \mathbb{N} : W_x \text{ infinity}\}$, i.e., establish if V and $\overline{V}$ are recursive/recursively enumerable.

The set V is saturated since $V = x|\phi_x \in \mathcal{V}$ where $\mathcal{V} = f \in \mathcal{C} : |dom(f)| = \infty$.

1. V is not r.e. by Rice-Shapiro theorem:
   - Let id be the identity function. Then $id \in V$
   - Any finite subfunction $\theta \subset id$ has finite domain, so $\theta \notin V$
   - By Rice-Shapiro theorem, V is not r.e.
2. $\bar{V}$ is not r.e.:
   - Consider any $f \in \bar{V}$ (finite domain)
   - Let $\emptyset \subset f$. Then $\emptyset \in \bar{V}$
   - By Rice-Shapiro theorem, $\bar{V}$ is not r.e.

Therefore V is not recursive.

This was also solved by an old tutor:



Exercise 8.37

Exercise 8.37. Study the recursiveness of the set $V = \{x \in \mathbb{N} : \exists y \in W_x.\ \exists k \in \mathbb{N}.\ y = k \cdot x\}$, i.e., establish if V and $\overline{V}$ are recursive/recursive enumerable.

1. V is r.e. since:

$$sc_V(x) = 1(\mu\langle y, k, t\rangle.\ (H(x, y, t) \wedge y = k \cdot x))$$

   is computable.

2. To prove V is not recursive, we show $K \leq_m V$: Define $g(x, y)$:

$$g(x,y) = \begin{cases} 2x & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

By s-m-n theorem, $\exists s$ total computable such that $\phi_{s(x)}(y) = g(x, y)$. Then:

- If $x \in K$: $2x \in W_{s(x)}$ and $2x = 2 \cdot x$, so $s(x) \in V$
- If $x \notin K$: $W_{s(x)} = \emptyset$, so $s(x) \notin V$

Therefore V is r.e. but not recursive, and consequently $\bar{V}$ is not r.e.

Exercise 8.38

> **Exercise 8.38.** Study the recursiveness of the set $V = \{x \in \mathbb{N} : |W_x| > 1\}$, i.e., establish if V and $\overline{V}$ are recursive/recursive enumerable.

The set V is saturated since $V = x|\phi_x \in \mathcal{V}$ where $\mathcal{V} = f \in \mathcal{C} : |dom(f)| > 1$.

1. V is not r.e. by Rice-Shapiro theorem:
   - Consider id. Then $id \in V$
   - Let $\theta \subset id$ be any function with $|dom(\theta)| = 1$. Then $\theta \notin V$
   - By Rice-Shapiro theorem, V is not r.e.
2. $\bar{V}$ is not r.e.:
   - Let f be any function with $|dom(f)| > 1$. Then $f \notin \bar{V}$
   - Let $\emptyset \subset f$. Then $\emptyset \in \bar{V}$
   - By Rice-Shapiro theorem, $\bar{V}$ is not r.e.

Therefore V is not recursive (and also $\bar{V}$ otherwise both would be recursive).

Exercise 8.39

> **Exercise 8.39.** Let P be the set of even numbers and Pr the set of prime numbers. Show that $P \leqslant_m Pr$ and $Pr \leqslant_m P$.

- First, let's show $P \leq_m Pr$:
  - Define $f(x) = 2x + 3$
  - f is total and computable
  - For any $x \in \mathbb{N}$:
    - If $x \in P$ (x is even), then $f(x) = 2x + 3$ is odd and ≥ 5, thus not prime
    - If $x \notin P$ (x is odd), then $f(x) = 2x + 3$ is prime
  - Therefore $x \in P \iff f(x) \notin Pr$, so $P \leq_m \overline{Pr}$
- Now, let's show $Pr \leq_m P$:
  - Define $g(x) = 2x$
  - g is total and computable
  - For any $x \in \mathbb{N}$:

- If $x \in Pr$ (x is prime), then $g(x) = 2x$ is even
- If $x \notin Pr$ (x is not prime), then $g(x) = 2x$ is even

- Therefore $x \in Pr \iff g(x) \in P$

Exercise 8.41

Exercise 8.41. Let $f : \mathbb{N} \to \mathbb{N}$ be a fixed total computable function. Study the recursiveness of the set $B = \{x \in \mathbb{N} \ img(f) \cap E_x \neq \varnothing\}$, i.e., establish if B and $\bar{B}$ are recursive/recursively enumerable. Please note that $img(f) = \{f(x) \mid x \in \mathbb{N}\}$.

1. First, let's show that B is not recursive by reducing K to B ($K \leq_m B$).

Let's construct a reduction function using s-m-n theorem. Define:

$$\phi_e(y) = \begin{cases} f(0) & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

By s-m-n theorem, there exists a total computable function h such that:

- If $x \in K$, then $E_{h(x)} = f(0)$, thus $img(f) \cap E_{h(x)} \neq \emptyset$
- If $x \notin K$, then $E_{h(x)} = \emptyset$, thus $img(f) \cap E_{h(x)} = \emptyset$

Therefore, $x \in K \iff h(x) \in B$, proving that $K \leq_m B$.

2. Now let's show that B is recursively enumerable.

The semi-characteristic function of B can be written as:

$$sc_B(x) = \mu w.\, (H(x, (w)_1, (w)_2) \wedge \exists y.\, f(y) = (w)_1)$$

This is computable because:

- H is decidable (step counting predicate)
- f is computable by hypothesis
- The existential quantification over a decidable predicate yields a semi-decidable predicate

3. Since B is r.e. but not recursive, $\overline{B}$ cannot be r.e. (otherwise B would be recursive).

Conclusion: B is r.e. but not recursive, and $\overline{B}$ is not r.e.

Exercise 8.42

Exercise 8.42. Study the recursiveness of the set $B = \{x \in \mathbb{N} \ E_x \nexists W_x\}$, i.e., establish if B e $\bar{B}$are recursive/recursively enumerable.

1. First, let's prove that B is not recursive by showing $K \leq_m B$.

By s-m-n theorem, define a reduction function h where:

$$\phi_{h(x)}(y) = \begin{cases} 1 & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

Then:

- If $x \in K$, then $E_{h(x)} = 1$ and $W_{h(x)} = \emptyset$, thus $E_{h(x)} \not\subseteq W_{h(x)}$
- If $x \notin K$, then $E_{h(x)} = \emptyset$ and $W_{h(x)} = \emptyset$, thus $E_{h(x)} \subseteq W_{h(x)}$

Therefore $x \in K \iff h(x) \in B$, establishing $K \leq_m B$.

2. B is recursively enumerable since:

$$sc_B(x) = \mu w.\, (S(x, (w)_1, (w)_2, (w)_3) \wedge \neg H(x, (w)_2, (w)_3))$$

This is computable because:

- S is decidable (step counting predicate)
- H is decidable
- Composition of decidable predicates is decidable

3. Since B is r.e. but not recursive, $\overline{B}$ is not r.e.

Conclusion: B is r.e. but not recursive, and $\overline{B}$ is not r.e.

Exercise 8.43

> **Exercise 8.43.** Let $B = \{x \mid \forall m \in \mathbb{N}.\ m \cdot x \in W_x\}$. Study the recursiveness of the B set, that is to say if B and $\bar{B}$ are recursive/recursively enumerable.

1. First, let's show that B is not recursively enumerable by reducing $\overline{K}$ to B ($\overline{K} \leq_m B$).

By s-m-n theorem, define:

$$\phi_{g(x)}(y) = \begin{cases} 0 & \text{if } x \notin K \\ \uparrow & \text{otherwise} \end{cases}$$

Then for any x:

- If $x \in \overline{K}$, then $\forall m \in \mathbb{N}. m \cdot g(x) \in W_{g(x)}$
- If $x \notin \overline{K}$, then $\exists m \in \mathbb{N}. m \cdot g(x) \notin W_{g(x)}$

Therefore $x \in \overline{K} \iff g(x) \in B$

2. Now let's show that $\overline{B}$ is recursively enumerable.

The semi-characteristic function of $\overline{B}$ is:

$$sc_{\overline{B}}(x) = \mu w.\, (\exists m \leq (w)_1.\, \neg H(x, m \cdot x, (w)_2))$$

This is computable because:

- H is decidable
- Bounded existential quantification preserves semi-decidability

3. Since $\overline{B}$ is r.e. but B is not r.e., B cannot be recursive.

Conclusion: B is not r.e., $\overline{B}$ is r.e., and B is not recursive.

Exercise 8.45

> **Exercise 8.45.** Study the recursiveness of the set $B = \{x \in \mathbb{N} \mid \exists y > x.\ y \in E_x\}$, i.e., establish if$B$ and $\bar{B}$ are recursive/recursively enumerable.

1. First, let's show that B is not recursive by reducing K to B ($K \leq_m B$).

By s-m-n theorem, define:

$$\phi_{g(x)}(y) = \begin{cases} x+1 & \text{if } x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

Then:

- If $x \in K$, then $x+1 \in E_{g(x)}$ and $x+1 > g(x)$
- If $x \notin K$, then $E_{g(x)} = \emptyset$, so $\nexists y > g(x).\, y \in E_{g(x)}$

Therefore $x \in K \iff g(x) \in B$

2. B is recursively enumerable since:

$$sc_B(x) = \mu w.\, (S(x, (w)_1, (w)_2, (w)_3) \wedge (w)_1 > x)$$

This is computable because:

- S is decidable
- Greater-than relation is decidable
- Conjunction of decidable predicates is decidable

3. Since B is r.e. but not recursive, $\overline{B}$ cannot be r.e.

Conclusion: B is r.e. but not recursive, and $\overline{B}$ is not r.e.

9. Second recursion theorem

Exercise 9.1

Exercise 9.1. State and prove the second recursion theorem.

THEOREM 18.1 (Second recursion theorem (Kleene)). *Let $f : \mathbb{N} \to \mathbb{N}$ a total computable function. Then there exists $e_0 \in \mathbb{N}$ such that*

$$\varphi_{e_0} = \varphi_{f(e_0)}$$

PROOF. Let $f : \mathbb{N} \to \mathbb{N}$ a total computable. Take

$$\begin{aligned} g(x, y) &= \varphi_{f(\varphi_x(x))}(y) \\ &= \Psi_U(f(\varphi_x(x)), y) \\ &= \Psi_U(f(\Psi_U(x, x)), y) \end{aligned}$$

it is computable. By the *smn* theorem there exists $s : \mathbb{N} \to \mathbb{N}$ total computable such that

$$\varphi_{s(x)}(y) = g(x, y) = \varphi_{f(\varphi_x(x))}(y) \quad \forall x, y$$

Since s is computable there exists $m \in \mathbb{N}$ such that

$$S = \varphi_m$$

hence

$$\varphi_{\varphi_m(x)}(y) = \varphi_{f(\varphi_x(x))}(y) \quad \forall x, y$$

For $x = m$

$$\varphi_{\varphi_m(m)}(y) = \varphi_{f(\varphi_m(m))}(y) \quad \forall y$$

We set $e_0 = \varphi_m(m)\downarrow$ and we replace in the previous equation

$$\varphi_{e_0}(y) = \varphi_{f(e_0)}(y) \quad \forall y$$

i.e.

$$\varphi_{e_0} = \varphi_{h(e_0)}$$

□

This theorem can therefore be interpreted in the following manner *given any effective procedure to transform programs, there is always at a program such that when modified by the procedure it does exactly what it did before* or *it is impossible to write a program that change the extensional behaviour of of all programs.*

The proof of the theorem can appear mysterious, but after a closer inspection, it clearly appears to be a simple diagonalization. Nevertheless, the result of this theorem is extremely deep; in this way, many theorems we've seen up until now are just corollaries.

Exercise 9.2

Exercise 9.2. State the second recursion theorem and use it to prove that K is not is recursive.

Solution: The second recursion theorem states that for each total computable function $h : \mathbb{N} \to \mathbb{N}$ there exists $n \in \mathbb{N}$ such that $\varphi_n = \varphi_{h(n)}$.

PROOF. Let $k = \{x \mid x \in W_x\}$ recursive for the sake of the argument. and let e_0, e_1 be indexes s.t. $\varphi_{e_0} = \varnothing$ and $\varphi_{e_1} = \lambda x \,.\, x$.

Define $f : \mathbb{N} \to \mathbb{N}$

$$\begin{aligned} f(x) &= \begin{cases} e_0 & x \in K \\ e_0 & x \notin K \end{cases} \\ &= e_0 \cdot \chi_K(x) + e_1 \cdot \chi_{\bar{K}}(x) \end{aligned}$$

If K were recursive, then χ_K and $\chi_{\bar{K}}$ would be computable, thus f would be both computable and total, then by (18.1), there would be $e \in \mathbb{N}$ such that $\varphi_e = \varphi_{f(e)}$, but

- if $e \in K$, then $f(e) = e_0$, so $\varphi_e(e)\downarrow \neq \varphi_{f(e)}(e) = \varphi_{e_0}(e)\uparrow$
- if $e \in \bar{K}$, then $f(e) = e_1$, so $\varphi_e(e)\uparrow \neq \varphi_{f(e)}(e) = \varphi_{e_1}(e) = e\downarrow$

Exercise 9.3

Exercise 9.3. State the Second Recursion Theorem and use it for proving that there exists $x \in \mathbb{N}$ such that $\varphi_x(y) = y^x$, for each $y \in \mathbb{N}$.

The Second Recursion Theorem states that for every total computable function $h : \mathbb{N} \to \mathbb{N}$, there exists $e \in \mathbb{N}$ such that $\varphi_{h(e)} = \varphi_e$.

We want to prove that for each $y \in \mathbb{N}$, there exists $x \in \mathbb{N}$ such that $\varphi_x(y) = y^x$.

Define $h(e) = s(e, y)$, where s is a total computable function given by the s-m-n theorem such that $\varphi_{s(e,y)}(z) = [\varphi_e(y)]^z$.

By the Second Recursion Theorem, there exists $x \in \mathbb{N}$ such that $\varphi_x = \varphi_{h(x)} = \varphi_{s(x,y)}$. Thus, $\varphi_x(y) = [\varphi_x(y)]^y = y^x$.

Exercise 9.4

Exercise 9.4. State the Second Recursion Theorem and use it for proving that there exists $n \in \mathbb{N}$ such that $W_n = E_n = \{x \cdot n : x \in \mathbb{N}\}$.

The Second Recursion Theorem states that for every total computable function $h : \mathbb{N} \to \mathbb{N}$, there exists $e \in \mathbb{N}$ such that $\varphi_{h(e)} = \varphi_e$.
We want to prove that there exists $n \in \mathbb{N}$ such that $W_n = E_n = x \cdot n : x \in \mathbb{N}$.

Define $h(e) = s(e)$, where s is a total computable function given by the s-m-n theorem such that

$$\varphi_{s(e)}(x) = \begin{cases} e \cdot x & \text{if } \varphi_e(x) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

By the Second Recursion Theorem, there exists $n \in \mathbb{N}$ such that $\varphi_n = \varphi_{h(n)} = \varphi_{s(n)}$. Thus,

$$\varphi_n(x) = \begin{cases} n \cdot x & \text{if } \varphi_n(x) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

This implies $W_n = E_n = x \cdot n : x \in \mathbb{N}$.

Exercise 9.6

Exercise 9.6. State the Second Recursion Theorem and use it for proving that there exists $x \in \mathbb{N}$ such that $\varphi_x(y) = x - y$.

The Second Recursion Theorem states that for every total computable function $h : \mathbb{N} \to \mathbb{N}$, there exists $e \in \mathbb{N}$ such that $\varphi_{h(e)} = \varphi_e$.
We want to prove that there exists $x \in \mathbb{N}$ such that $\varphi_x(y) = x - y$.

Define $h(e) = s(e)$, where s is a total computable function given by the s-m-n theorem such that $\varphi_{s(e)}(y) = e - y$.

By the Second Recursion Theorem, there exists $x \in \mathbb{N}$ such that $\varphi_x = \varphi_{h(x)} = \varphi_{s(x)}$. Thus, $\varphi_x(y) = x - y$.