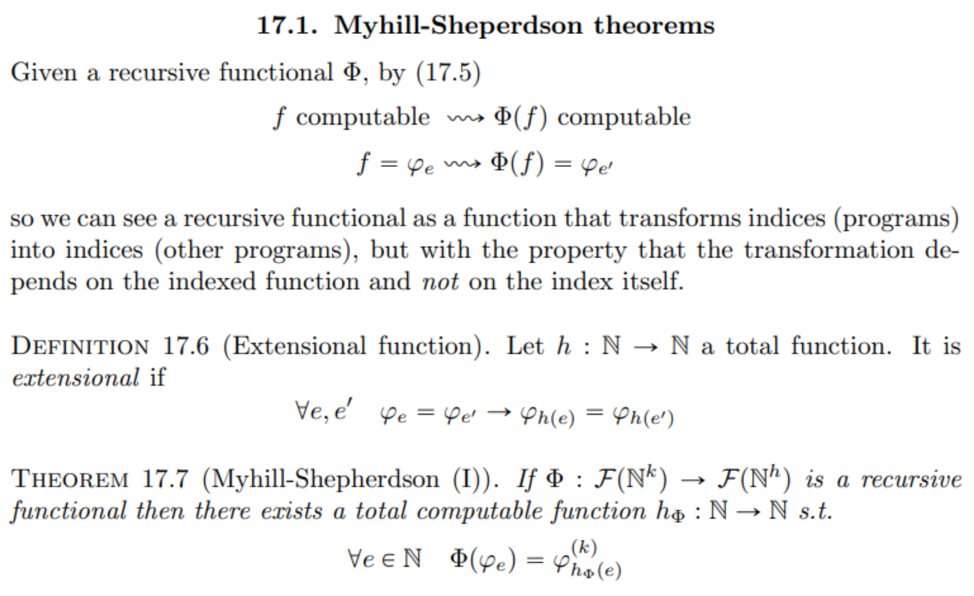
Lessons touched by this meeting according to schedule:

- 20. 16/12/2024
  - Recursive functionals (recursive operators, in the book)
  - Myhill-Shepherdson Theorem
  - First recursion theorem [§10.1, §10.2, §10.3, without proofs]
- 21. 17/12/2024
  - Second recursion theorem [§11.1, §11.2]

- functional: A functional (also called "operator") is a total function $\Phi: F(\mathbb{N}^k) \to F(\mathbb{N}^h)$ (considering F is the set of all the functions and the others are the arguments and the indices of the k arguments)
  - A functional has to be effective, given both input and output can be infinite
  - They calculate in finite time using only a finite part of the input function (according to Cutland book definition)
  - In general, a *functional type*, in which a function takes in input a function of same type and gives as output another function of same type

- To do this, we introduce the concept of recursive functionals
  - A functional $\Phi: F(\mathbb{N}^k) \to F\mathbb{N}N^h)$ is recursive if there is a total computable function $\phi: \mathbb{N}^{h+1} \to \mathbb{N}\ s.t. \forall f \in F(\mathbb{N}^k)$
    $\forall \vec{x} \in \mathbb{N}^h$
    $\Phi(f)(\vec{x}) = y \quad iff\ there\ exists\ \theta \subseteq f\ s.t. \phi(\tilde{\theta}, \vec{x}) = y$
  - In simpler terms, recursive functionals essentially produce outputs of the same type as a finite part of the input function, acting as both input and output themselves.

- fixed point: A function is a fixed point/fixpoint of a functional Φ (a function which is not changed from the transformation and is an element mapped to itself by the function) , i.e. $f: \mathbb{N} \to \mathbb{N}\ s.t. \Phi(f) = f$.
  - Looking here, a fixed point x in a set X s.t. $x \in X$ is a fixed point with a map to itself such that $f(x) = x$.



17.1. Myhill-Sheperdson theorems

Given a recursive functional Φ, by (17.5)

$$f \text{ computable } \rightsquigarrow \Phi(f) \text{ computable}$$
$$f = \varphi_e \rightsquigarrow \Phi(f) = \varphi_{e'}$$

so we can see a recursive functional as a function that transforms indices (programs) into indices (other programs), but with the property that the transformation depends on the indexed function and *not* on the index itself.

DEFINITION 17.6 (Extensional function). Let $h : \mathbb{N} \to \mathbb{N}$ a total function. It is *extensional* if

$$\forall e, e' \quad \varphi_e = \varphi_{e'} \to \varphi_{h(e)} = \varphi_{h(e')}$$

THEOREM 17.7 (Myhill-Shepherdson (I)). *If $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$ is a recursive functional then there exists a total computable function $h_\Phi : \mathbb{N} \to \mathbb{N}$ s.t.*

$$\forall e \in \mathbb{N} \quad \Phi(\varphi_e) = \varphi^{(k)}_{h_\Phi(e)}$$

This is defined as extensional function: $\forall e, e' \in \mathbb{N}\ s.t.\ \phi_e = \phi_{e'}\ then\ \phi_{h_\Phi(e)} = \phi_{h(\Phi)(e')}$

(from programs which compute the same function, if you apply the functional transformation, the two programs will compute the same function)

It should be noted that the book defines precisely the functional as continuous and monotone; I add this because the professor notions given up until this last one precisely state this.

Myhill-Shepherdson Theorem:

Definition

(1) Let $\Phi: F(\mathbb{N}^k) \rightarrow F(\mathbb{N}^i)$ be a recursive function. Then, there exists a total computable function $h_\Phi: \mathbb{N} \rightarrow \mathbb{N}\ s.t. \forall e \in \mathbb{N}, \Phi\left(\phi_e^{(k)}\right) = \phi_{h_{(\Phi)(e)}}^{(i)}$ and h_Φ is extensional.

(For this first part - intuitively, the behaviour of the recursive functional on computable functions is captured by a total extensional function on the indices)

(2) Let $h: \mathbb{N} \rightarrow \mathbb{N}$ be a total computable function and h extensional.

Then, there is a unique recursive functional $\Phi: F(\mathbb{N}^k) \rightarrow F(\mathbb{N}^i)$ s.t. for all $e \in \mathbb{N}$ (possible programs)

$$\Phi\left(\phi_e^{(k)}\right) = \phi_{h(e)}^{(i)}$$

(For this second part - Computable extensional functions uniquely identify computable functions through program transformations. In contrast, recursive functionals extend this identification to non-computable functions, highlighting that all functions, even non-computable ones, can be approximated precisely by computable functions, like finite subfunctions).

In Wikipedia, I see this can generally extended as "Myhill isomorphism theorem", which provides a characterization for two numberings (assignments of natural numbers to sets of similar objects) to induce the same notion of computability on a set. Basically, there exists a total computable bijection, which maps elements reducible to each other in both directions, given the functions are extensional.

The Myhill-Shepherdson Theorem, stemming from the Rice-Shapiro Theorem, defines the computable type 2 functionals. These functionals operate on computable partial functions, yielding numbers as results in cases of termination. Notably, they adhere to a specific effectiveness criterion and exhibit continuity as functionals. This can be also found here.

THEOREM 17.9 (First recursion theorem (Kleene)). *Let $\Phi : \mathcal{F}(\mathbb{N}^k) \rightarrow \mathcal{F}(\mathbb{N}^h)$ be a* ***recursive functional****. Then Φ has a* ***least fixed point*** *f_Φ which is* ***computable****, i.e.*

(1) $\Phi(f_\Phi) = f_\Phi$

(2) $\forall g \in \mathcal{F}(\mathbb{N}^k) \quad \Phi(g) = g \Rightarrow f_\Phi \subseteq g$

(3) f_Φ *is computable*

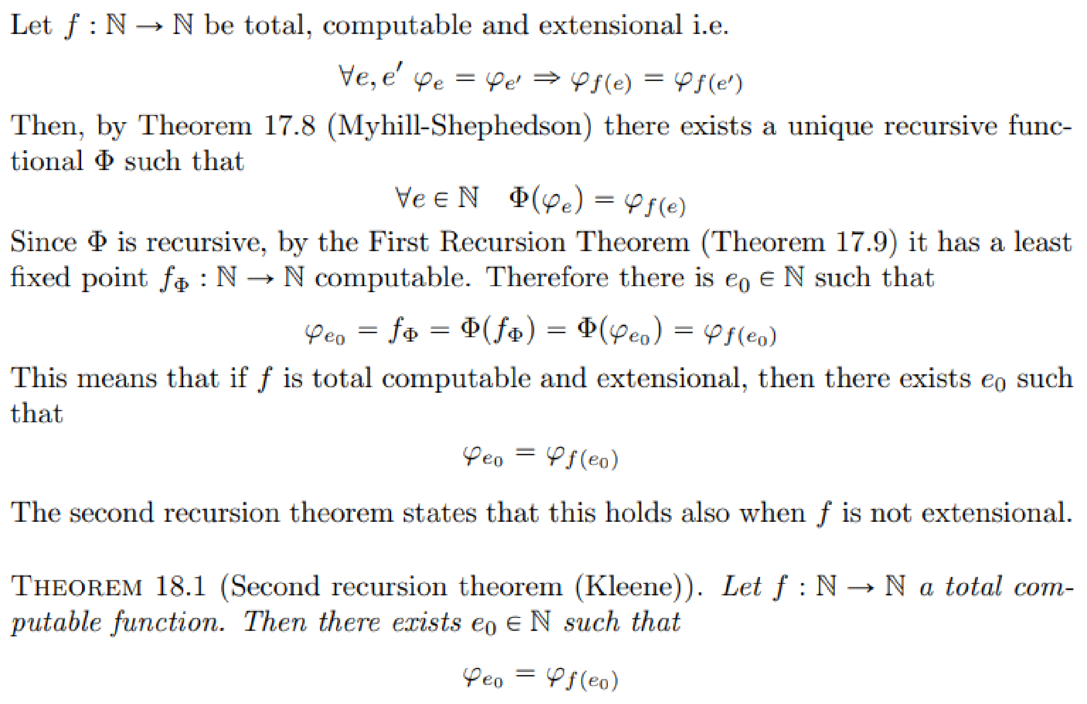
and we can see that $f_\Phi = \bigcup_n \Phi^n(\varnothing)$.

The theorem above implies the closure of the set of computable functions with respect to extremely general forms of recursion.

The First Recursion Theorem is an important result in computability theory that is used in several key ways in the exercises provided:

1. Proving the existence of fixed points for recursive functionals. The theorem states that every recursive functional has a least fixed point that is computable. This is applied in examples like showing that the Ackermann functional has the Ackermann function as its least fixed point.
2. Classifying sets as recursively enumerable or not. The theorem can be used to show that certain sets, like the set A={x | ϕ_x(y)=x^2 for infinitely many y}, are likely recursively enumerable but not recursive. The proof sketch leverages the First Recursion Theorem.
3. Proving undecidability results. As a corollary, the theorem allows proving Rice's Theorem, which states that any non-trivial property of computable functions is undecidable. This in turn is used to prove the undecidability of the Halting Problem.
4. Showing the non-extensionality of certain sets. The theorem is applied to prove that the Halting Set K={x | ϕ_x(x)↓} is not saturated (i.e. extensional). The diagonalization argument in the proof relies on the recursion theorem.

The Second Recursion Theorem, also known as Kleene's Fixed Point Theorem, states that for any total computable function f : N → N, there exists a fixed point e ∈ N such that ϕ_e = ϕ_f(e). In other words, there is a program e that, when executed, behaves exactly like the program obtained by applying f to e.



Let $f : \mathbb{N} \to \mathbb{N}$ be total, computable and extensional i.e.

$$\forall e, e' \; \varphi_e = \varphi_{e'} \Rightarrow \varphi_{f(e)} = \varphi_{f(e')}$$

Then, by Theorem 17.8 (Myhill-Shephedson) there exists a unique recursive functional Φ such that

$$\forall e \in \mathbb{N} \quad \Phi(\varphi_e) = \varphi_{f(e)}$$

Since Φ is recursive, by the First Recursion Theorem (Theorem 17.9) it has a least fixed point $f_\Phi : \mathbb{N} \to \mathbb{N}$ computable. Therefore there is $e_0 \in \mathbb{N}$ such that

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

This means that if f is total computable and extensional, then there exists e_0 such that

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

The second recursion theorem states that this holds also when f is not extensional.

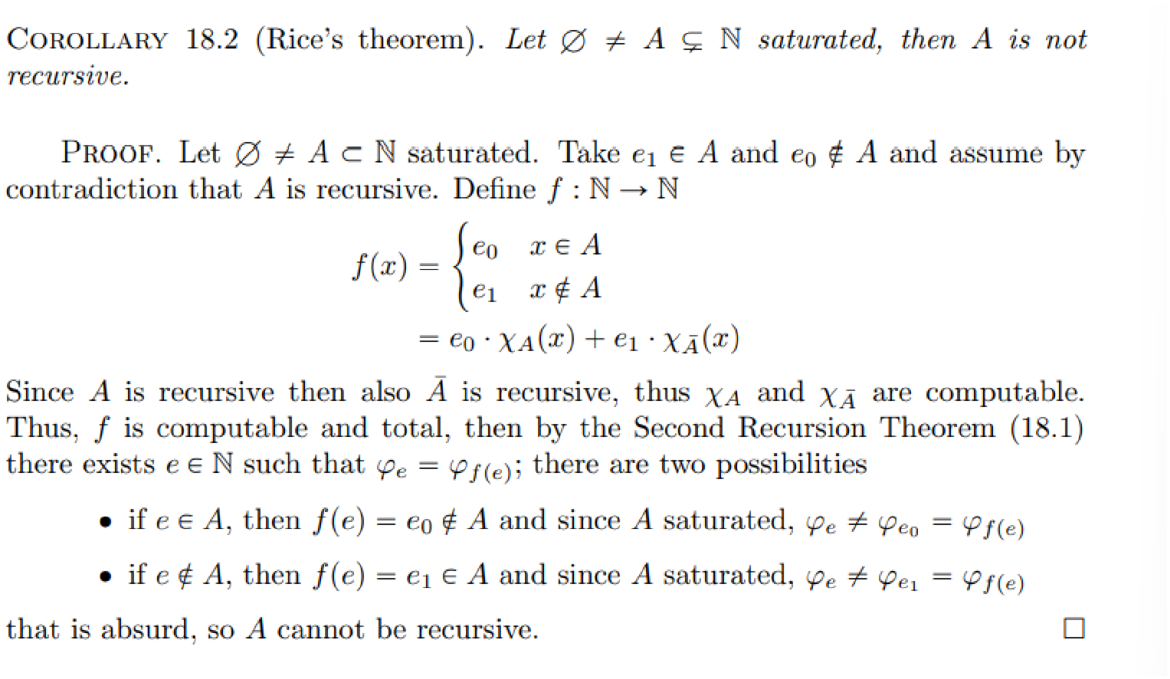
THEOREM 18.1 (Second recursion theorem (Kleene)). *Let $f : \mathbb{N} \to \mathbb{N}$ a total computable function. Then there exists $e_0 \in \mathbb{N}$ such that*

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

This theorem is used in the course to:

1. Show the non-extensionality of certain sets. For example, the Halting Set K = {x | ϕ_x(x)↓} can be proven to be non-extensional using the Second Recursion Theorem. The proof involves constructing a program that behaves differently when given its own index as input.
2. Demonstrate the existence of self-referential programs. The Second Recursion Theorem allows us to create programs that can access their own source code or index during execution. This is a powerful technique for constructing counterexamples and proving the limitations of computable functions.

Let's use these proofs as exercises:



Corollary 18.2 (Rice's theorem). *Let $\varnothing \neq A \subsetneq \mathbb{N}$ saturated, then A is not recursive.*

Proof. Let $\varnothing \neq A \subset \mathbb{N}$ saturated. Take $e_1 \in A$ and $e_0 \notin A$ and assume by contradiction that A is recursive. Define $f : \mathbb{N} \to \mathbb{N}$

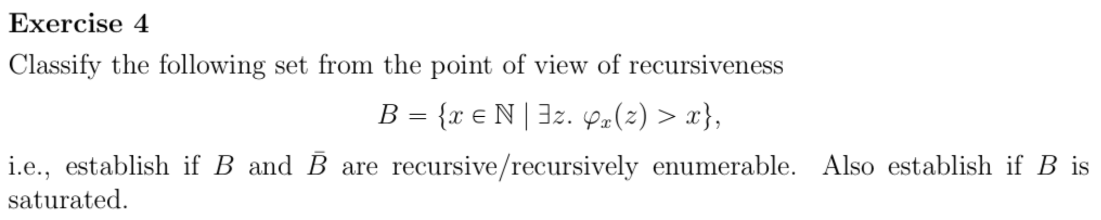
$$f(x) = \begin{cases} e_0 & x \in A \\ e_1 & x \notin A \end{cases}$$
$$= e_0 \cdot \chi_A(x) + e_1 \cdot \chi_{\bar{A}}(x)$$

Since A is recursive then also $\bar{A}$ is recursive, thus χ_A and $\chi_{\bar{A}}$ are computable. Thus, f is computable and total, then by the Second Recursion Theorem (18.1) there exists $e \in \mathbb{N}$ such that $\varphi_e = \varphi_{f(e)}$; there are two possibilities

- if $e \in A$, then $f(e) = e_0 \notin A$ and since A saturated, $\varphi_e \neq \varphi_{e_0} = \varphi_{f(e)}$
- if $e \notin A$, then $f(e) = e_1 \in A$ and since A saturated, $\varphi_e \neq \varphi_{e_1} = \varphi_{f(e)}$

that is absurd, so A cannot be recursive. □

Consider some real examples (exercises):

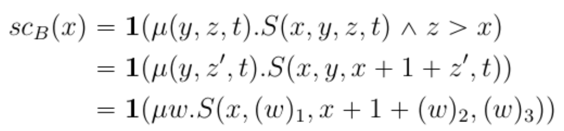


Exercise 4

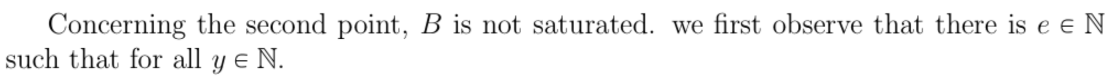
Classify the following set from the point of view of recursiveness

$$B = \{x \in \mathbb{N} \mid \exists z.\ \varphi_x(z) > x\},$$

i.e., establish if B and $\bar{B}$ are recursive/recursively enumerable. Also establish if B is saturated.



$$\begin{aligned} sc_B(x) &= \mathbf{1}(\mu(y,z,t).S(x,y,z,t) \land z > x) \\ &= \mathbf{1}(\mu(y,z',t).S(x,y,x+1+z',t)) \\ &= \mathbf{1}(\mu w.S(x,(w)_1,x+1+(w)_2,(w)_3)) \end{aligned}$$



Concerning the second point, B is not saturated. we first observe that there is $e \in \mathbb{N}$ such that for all $y \in \mathbb{N}$.

$$\varphi_e(y) = e + 1$$

To this aim define $g : \mathbb{N}^2 \to \mathbb{N}$ as

$$g(x, y) = x + 1$$

This is clearly computable and thus, by smn theorem, there exists $s : \mathbb{N} \to \mathbb{N}$ such that for all $x, y \in \mathbb{N}$ we have

$$\varphi_{s(x)}(y) = g(x, y) = x + 1$$

By the second recursion theorem there is $e \in \mathbb{N}$ such $\varphi_{e)}(y) = \varphi_{s(e)}(y)$ and thus for all $y \in \mathbb{N}$

$$\varphi_e(y) = \varphi_{s(e)}(y) = e + 1$$

Thus $e \in B$.

Now, there are infinitely many indexes for the function φ_e, hence there is $e' \in \mathbb{N}$, $e' > e$ such that $\varphi_e = \varphi_{e'}$ and thus for all $y \in \mathbb{N}$ $\varphi_{e'}(y) = \varphi_e(y) = e + 1 \leqslant e'$. Hence $e' \notin B$.

Summing up, $e \in B$, $e' \notin B$ and $\varphi_e = \varphi_{e'}$. Hence B is not saturated.

Let's jump to some more exercises:

Exercise 9.12. State the second recursion theorem. Use it for proving that the set $C = \{x \in \mathbb{N} \mid x \in E_x\}$ not saturated.

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

For answering the question, define

$$g(x, y) = x$$

which is a computable function and thus, by smn theorem, there is a total computable function $s : \mathbb{N} \to \mathbb{N}$ such that for each $x, y \in \mathbb{N}$

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

By the II recursion theorem there exists an index e such that $\varphi_{s(e)} = \varphi_e$ and then

$$\varphi_e(y) = e$$

Therefore $E_e = \{e\}$ and therefore $e \in C$.

Given any $e' \neq$ *and* such $\varphi_{e'} = \varphi_e$ one has that $e \notin E_{e'} = E_e$ and therefore $e \notin C$. Therefore C is not is saturated. □

Exercise 9.14. State the second recursion theorem. Prove that, given a function $f : \mathbb{N} \to \mathbb{N}$ total computable injective, the set $C_f = \{x : f(x) \in W_x\}$ is not saturated.

Solution: Define

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

By the smn theorem, we obtain a function $s : \mathbb{N} \to \mathbb{N}$ total computable, such that $g(x, y) = \varphi_{s(x)}(y)$ and by the second recursion theorem there exists $e \in \mathbb{N}$ such that $\varphi_e = \varphi_{s(e)}$. Therefore:

$$\varphi_e(y) = \varphi_{s(e)}(y) = g(e, y) = \begin{cases} f(e) & \text{if } x = f(e) \\ \uparrow & \text{otherwise} \end{cases}$$

Thus $e \in C_f$. Now, if we take a different index e such that $\varphi_e = \varphi_{e'}$ we will have that, by injectivity of f, it holds $f(e') \neq f(e)$ and thus $f(e') \notin W_{e'} = W_e = \{f(e)\}$. Hence $e' \notin C_f$. □

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

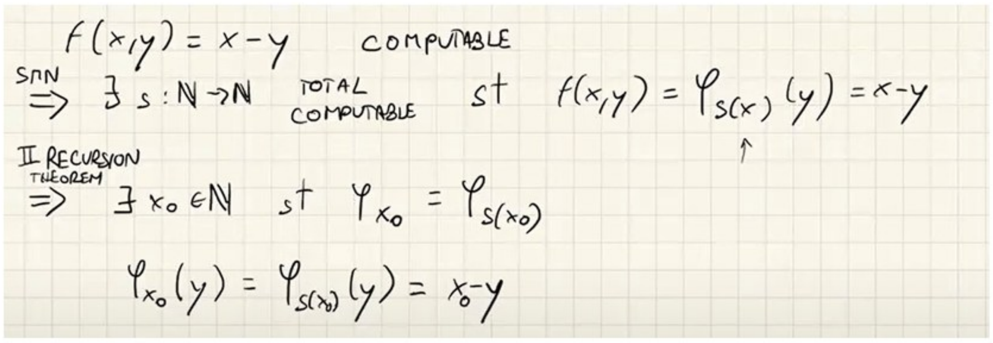
The second recursion theorem states that for each total computable function $h: \mathbb{N} \to \mathbb{N}$ there exists $e \in \mathbb{N}$ such that $\phi_e = \phi_{h(e)}$.

Define a function of two arguments as follows:

$$g(x, y) = \begin{cases} x - y, & x \in W_x \\ \uparrow, & otherwise \end{cases}$$

By the smn-theorem, there exists $n \in \mathbb{N}\ s.t.\ g(x, y) = \phi_{h(x)}(y)$ and by the second recursion theorem, there exists an index $e \in \mathbb{N}\ s.t.\ \phi_e = \phi_{h(e)}$ and $\phi_e(y) = \phi_{s(x)}(y) = g(e, y) = e - y\ \forall y \in \mathbb{N}$.

As solved by an old tutor:



$f(x,y) = x - y$ COMPUTABLE

SMN $\Rightarrow \exists\, s : \mathbb{N} \to \mathbb{N}$ TOTAL COMPUTABLE st $f(x,y) = \varphi_{s(x)}(y) = x - y$ $\uparrow$

II RECURSION THEOREM $\Rightarrow \exists\, x_0 \in \mathbb{N}$ st $\varphi_{x_0} = \varphi_{s(x_0)}$

$\varphi_{x_0}(y) = \varphi_{s(x_0)}(y) = x_0 - y$

<u>Exercise</u> (2019-01-24)

State the Second Recursion Theorem and use it to show there exists $x \in \mathbb{N}$ such that $|W_x| = x$

Define:

```
g(x,y) = {
    y       if y < x
    ↑       otherwise
}
```

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

```
φs(x)(y) = g(x,y) for all x,y ∈ N
```

By the Second Recursion Theorem, there exists e ∈ N such that: `φe = φs(e)`

Therefore:

```
φe(y) = φs(e)(y) = g(e,y) = {
    y       if y < e
    ↑       otherwise
}
```

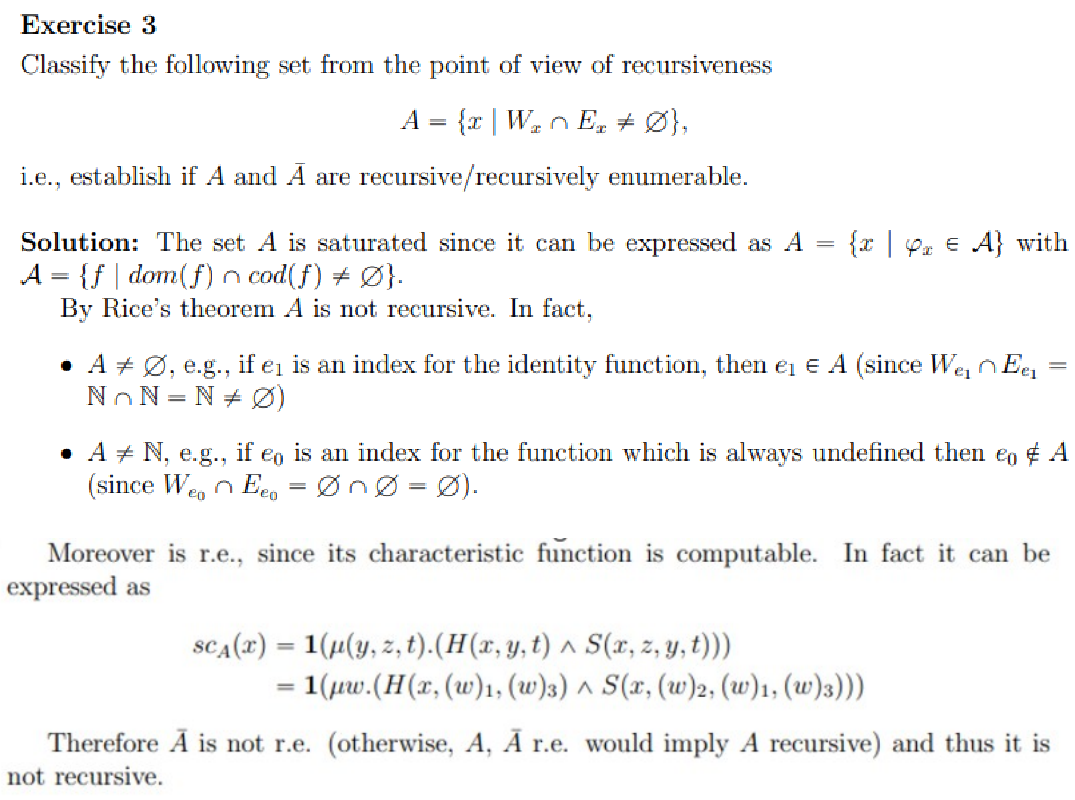
This means:

```
We = dom(φe) = {y ∈ N | y < e}
```

Therefore:

```
|We| = |{y ∈ N | y < e}| = e
```

Let’s jump to some recursiveness exercises:



Exercise 3
Classify the following set from the point of view of recursiveness

$$A = \{x \mid W_x \cap E_x \neq \varnothing\},$$

i.e., establish if A and $\bar{A}$ are recursive/recursively enumerable.

Solution: The set A is saturated since it can be expressed as $A = \{x \mid \varphi_x \in \mathcal{A}\}$ with $\mathcal{A} = \{f \mid dom(f) \cap cod(f) \neq \varnothing\}$.
By Rice’s theorem A is not recursive. In fact,

- $A \neq \varnothing$, e.g., if e_1 is an index for the identity function, then $e_1 \in A$ (since $W_{e_1} \cap E_{e_1} = \mathbb{N} \cap \mathbb{N} = \mathbb{N} \neq \varnothing$)
- $A \neq \mathbb{N}$, e.g., if e_0 is an index for the function which is always undefined then $e_0 \notin A$ (since $W_{e_0} \cap E_{e_0} = \varnothing \cap \varnothing = \varnothing$).

Moreover is r.e., since its characteristic function is computable. In fact it can be expressed as

$$\begin{aligned} sc_A(x) &= \mathbf{1}(\mu(y,z,t).(H(x,y,t) \wedge S(x,z,y,t))) \\ &= \mathbf{1}(\mu w.(H(x,(w)_1,(w)_3) \wedge S(x,(w)_2,(w)_1,(w)_3))) \end{aligned}$$

Therefore $\bar{A}$ is not r.e. (otherwise, A, $\bar{A}$ r.e. would imply A recursive) and thus it is not recursive.

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.

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.

For 8.1:

A is not r.e.: Consider the identity function id $\notin$ A since dom(id) = N is infinite.

However, we can find a finite subfunction $\theta \subseteq$ id defined as:

θ(x) = {

  0 if x = 0

  1 if x = 1

  ↑ otherwise

}

Clearly θ ∈ A since |dom(θ)| = 2. By Rice-Shapiro theorem, A is not r.e.

For 8.2:

For A = {x ∈ N : x ∈ Wx ∩ Ex}, we first prove A is not recursive by showing K ≤m A.

Let's define:

```
g(x,y) = {
    y    if x ∈ K
    ↑    otherwise
}
```

g is computable since g(x,y) = y · sc_K(x).

By the s-m-n theorem, there exists s: N → N total computable such that ϕs(x)(y) = g(x,y).

We claim s is a reduction function K ≤m A. Indeed:

- If x ∈ K then ϕs(x)(y) = y for all y, so s(x) ∈ Ws(x) and s(x) ∈ Es(x), thus s(x) ∈ A
- If x ∉ K then ϕs(x)(y) ↑ for all y, so s(x) ∉ Ws(x), thus s(x) ∉ A

Additionally, A is r.e. since:

```
scA(x) = 1(μw.H(x,x,(w)1) ∧ S(x,(w)1,x,(w)2))
```

is computable.

Therefore A is r.e. but not recursive, which implies $\bar{A}$ is not r.e.

Exercise 2

Define the class of primitive recursive functions. Using only the definition show that the function $a : \mathbb{N}^2 \to \mathbb{N}$ defined below is primitive recursive

$$a(x,y) = \begin{cases} 1 & \text{if } x > 0 \text{ and } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

Solution: The class of primitive recursive functions is the least class of functions $\mathcal{PR} \subseteq \bigcup_k (\mathbb{N}^k \to \mathbb{N})$ containing the base functions (zero, successor, projections) and and closed under composition and primitive recursion.

In order to show that f is primitive recursive observe that it can be defined as

$$\begin{cases} a(x,0) & = 0 \\ a(x,y+1) & = sg(x) \end{cases}$$

where $sg : \mathbb{N} \to \mathbb{N}$, is the sign function, which can be defined by primitive recursion as

$$\begin{cases} sg(0) & = 0 \\ sg(y+1) & = 1 \end{cases}$$

Exercise 2

Define the class of primitive recursive functions. Using only the definition show that the function $f : \mathbb{N} \to \mathbb{N}$ defined below is primitive recursive

$$f(x,y) = \begin{cases} 1 & \text{if } x \leqslant y \\ 0 & \text{otherwise} \end{cases}$$

Solution: The class of primitive recursive functions is the least class of functions $\mathcal{PR} \subseteq \bigcup_k (\mathbb{N}^k \to \mathbb{N})$ containing the base functions (zero, successor, projections) and and closed under composition and primitive recursion.

In order to show that f is primitive recursive, first observe that the following functions are so:

- $y \dot{-} 1$
  It can be defined as
  $$\begin{cases} 0 \dot{-} 1 & = 0 \\ (y+1) \dot{-} 1 & = y \end{cases}$$
- $x \dot{-} y$
  It can be defined as
  $$\begin{cases} x \dot{-} 0 & = x \\ x \dot{-} (y+1) & = (x \dot{-} y) \dot{-} 1 \end{cases}$$
- $\overline{sg}(y) = 1$ if $y = 0$ and 0 otherwise
  It can be defined as
  $$\begin{cases} \overline{sg}(0) & = 1 = succ(0) \\ \overline{sg}(y+1) & = 0 \end{cases}$$

Finally we conclude by observing that

$$f(x,y) = \overline{sg}(x \dot{-} y)$$

Exercise 1

a. Provide the definition of reduction between sets, i.e., given $A, B \subseteq \mathbb{N}$ define $A \leqslant_m B$.

b. Show that if B is r.e. and $A \leqslant B$ then A is r.e.

c. Is it possible to have $A, B \subseteq \mathbb{N}$ with A infinite and B finite such that $A \leqslant_m B$. Give an example or prove that it is not possible.

Solution:

1. Given $A, B \subseteq \mathbb{N}$ we write $A \leqslant_m B$ if there is a reduction function $f : \mathbb{N} \to \mathbb{N}$, total and computable and such that for all $x \in \mathbb{N}$, $x \in A$ iff $f(x) \in B$.

2. Given $A, B \subseteq \mathbb{N}$ with $A \leqslant_m B$, let $f : \mathbb{N} \to \mathbb{N}$ be the reduction function. Assume that B is r.e., i.e., its semi-characteristic function $sc_B : \mathbb{N} \to \mathbb{N}$ defined by

$$sc_B(x) = \begin{cases} 1 & \text{if } x \in B \\ \uparrow & \text{otherwise} \end{cases}$$

   is computable.

   Then we simply observe that the semi-characteristic function of A can be defined as $sc_A(x) = sc_B(f(x))$. Since it is the composition of computable functions, it is computable. This shows that A is r.e.

3. There can be $A, B \subseteq \mathbb{N}$ with A infinite and B finite such that $A \leqslant_m B$. Just take $A = \mathbb{N}$ and $B = \{0\}$. It is immediate to see that the constant 0 is a reduction function for $A \leqslant_m B$.