Computability theory

FAMAF - UNC

SLP

Contents

1 Introduction

These are my notes on computability theory. The only definitions and conventions that ought to be known first-hand are the following.

Let Σ denote an arbitrary alphabet, $\omega := \mathbb{N} \cup \{0\}$, and $n, m \geq 0$ fixed elements in ω. We use $\omega^n \times \Sigma^{*m}$ to denote the set of tuples $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m)$ with $x_i \in \omega, \alpha_i \in \Sigma^*$, and not the cartesian product. Then:

- A Σ-mixed function is a function s.t. $\mathscr{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$ with either $\varphi = \omega$ or $\varphi = \Sigma^*$.
- A Σ-mixed function is Σ-effectively computable if we can find some algorithmic procedure that, given an input in $\omega^n \times \Sigma^{*m}$, outputs the value of $f(\overrightarrow{x}, \overrightarrow{\alpha})$.
- Any set $S \subseteq \omega^n \times \Sigma^{*m}$ is termed a Σ-mixed set.
- If there is an algorithmic procedure that enumerates S, we say S is Σ-effectively enumerable.
- If there is an algorithmic procedure that decides the belonging to S, we say S is Σ-effectively computable.

Here, the notion of "algorithmic procedure" is non-rigorous. The principal subject of this study are three formalizations of this concept: Turing computability, recursive computability (Godel), and imperative computability (von Neumann).

2 Coding infinite tuples

We define $\omega^{\mathbb{N}} := \{(s_1, s_2, \ldots) : s_i \in \omega\}$ and $\omega^{[\mathbb{N}]} \subseteq \omega^{\mathbb{N}} := \{(s_1, s_2, \ldots) : s_i \in \omega \wedge \exists k \in \mathbb{N} : i \geq k \Rightarrow s_i = 0\}$.

Define

$$\begin{aligned} p : \mathbb{N} &\mapsto \omega \\ n &\mapsto \text{ the } n\text{th prime number} \end{aligned}$$

Theorem 1 *For all $x \in \mathbb{N}$ there is a unique infinituple $\overrightarrow{s} \in \omega^{[\mathbb{N}]}$ s.t.*

$$x = \prod_{i=1}^{\infty} p(i)^{s_i}$$

Proof 1 *(Complete induction) The base case is trivial. Assume the statement holds for all $n \leq k$. The fundamental theorem of arithmetic ensures that $k+1 = p_1 \cdot \ldots \cdot p_m$ where p_i is prime. Assume the factorization above is ordered (this is, $p_{j+1} > p_j$ for all $j \in [1, m]$). Then $k + 1 = p_m \cdot q$ with $q = p_1 \cdot \ldots \cdot p_{m-1}$.*

Subproof. We will prove $k + 1 = p_m \cdot q \Rightarrow q \leq k$. Assume the premise holds and the consequence does not. Since $q > k$ we have $q \cdot x > k+1$ for all $x > 1$. Then $q \cdot x > k + 1$ for all x that is prime. Then $q \cdot p_m \neq k + 1$ which is a contradiction. Then, if $k + 1 = q \cdot p_m$, we have $q \leq k$. ■

Since $q \leq k$, via inductive hypothesis, q takes the productorial form of the theorem above. Then $k + 1 = q \cdot p(j)$ where $p(j) = p_m$. Then the theorem holds for all $n \in \mathbb{N}$.

Theorem 2 *If $p, p_1, \ldots, p_m$ are prime ($m \geq 1$) and $p \mid p_1 \ldots p_m$, then $p = p_i$ for some i.*

Proof 2 ***Proof.*** *Assume $p \mid p_1 \ldots p_m$. $\therefore p \leq p_1 \ldots p_m$. The uniqueness of the prime factorization tells us that $p_1 \ldots p_n$ are factors of a unique integer x.*

Assume $p \nmid p_j$ for any $j = 1, \ldots, m$. $\therefore p$ is a prime factor of $p_1 \ldots p_m$ distinct from p_j for all p_j. $\therefore x = p_1 \ldots p_m p \neq x$. $\therefore \perp$

$\therefore p \mid p_i$ for some $i = 1, \ldots, m$. ■

We use $\langle s_1, s_2, \ldots \rangle$ to denote the number $x = \prod_{n=1}^{\infty} p(n)^{s_n}$. We use $(x)_i$ to denote s_i in said tuple and (x) to denote the infinituple itself.

Theorem 3 *The functions*

$$\begin{aligned} \mathbb{N} &\mapsto \omega^{[\mathbb{N}]} & \omega^{[\mathbb{N}]} &\mapsto \mathbb{N} \\ x &\mapsto (x) = ((x)_1, (x)_2, \ldots) & (s_1, s_2, \ldots) &\mapsto \langle s_1, s_2, \ldots \rangle \end{aligned}$$

are bijections each the inverse of the other.

The theorem should be intuitive. The function that maps a number x to the infinituple of its prime exponents is the inverse of the function which takes an infinituple and maps it to the product of all primes with the corresponding exponents.

Theorem 4

$$(x)_i = \max_t \left(p(i)^t \mid x \right)$$

Proof. Let $x \in \mathbb{N}$ be s.t. $x = p_1^{s_1} \ldots p_m^{s_m}$ with p_j prime and $m_j \neq 0$ for all $j = 1, \ldots, m$. Evidently $(x)_i = s_i$.

Let $x \in \mathbb{N}$ and

$$\mathscr{P}_i := \left\{ t \in \omega : p(i)^t \mid x \right\}$$

Since $\mathscr{P}_i \subseteq \mathbb{N}$ is necessary finite it has a maximum m_i. The fundamental theorem of arithmetic directly gives $x = p(1)^{m_1} p(2)^{m_2} \ldots$ Then by definition $m_i = (x)_i$. ■

We define

$$\begin{aligned} Lt : \mathbb{N} &\mapsto \omega \\ x &\mapsto \begin{cases} \max_i \ (x)_i \neq 0 & x \neq 1 \\ 0 & x = 1 \end{cases} \end{aligned}$$

The function returns the index of the maximum prime factor (that is not zero-exponentiated) in the factorization of x. Since, in this factorizations, all prime factors beyond $Lt(x)$ are zero, $Lt(x)$ can be understood as an upper bound of the factorization. This is formalized in the following theorem.

Theorem 5

$$x = \prod_{i=1}^{Lt(x)} p(i)^{(x)_i}$$

Problem 1 *Prove the previous theorem.*

$x = \prod_{i=1}^{\infty} p(i)^{(x)_i} \wedge (x) \in \omega^{[\mathbb{N}]} \therefore \exists k \in \omega : i \geq k \Rightarrow (x)_i = 0 \therefore x = \prod_{i=1}^{k-1} p(i)^{(x)_i} \times \prod_{i=k}^{\infty} p(i)^0 = \prod_{i=1}^{k-1} p(i)^{(x)_i}$. But $k-1 := \max_i ((x)_i \neq 0) := Lt(x)$. ■

Prime numbers closely relate to set enumerability. The reason is that the prime decomposition of a number associates to any unique $x \in \mathbb{N}$ an infinite number of integers $(x)_1, (x)_2, \ldots$. Say we want to generate all possible pairs $(x, y, z) \in \omega^3$ given a unique input $x \in \mathbb{N}$. Then, letting $(x, y, z) = ((x)_1, (x)_2, (x)_3)$, we have

$$
\begin{aligned}
x = 1 &\mapsto (0, 0, 0) \\
x = 2 &\mapsto (1, 0, 0) \\
x = 3 &\mapsto (0, 1, 0) \\
x = 4 &\mapsto (2, 0, 0) \\
x = 5 &\mapsto (0, 0, 1) \\
x = 6 &\mapsto (1, 1, 0) \\
x = 7 &\mapsto (0, 0, 0) \\
x = 8 &\mapsto (3, 0, 0) \\
x = 9 &\mapsto (0, 2, 0) \\
&\vdots
\end{aligned}
$$

It is easy to see that any $(x, y, z) \in \omega^3$ is reached. This generalizes to $\omega^n, n \in \omega$ and to $\omega^n \times \Sigma^{*m}$ via the $*^{\leq}$ function (which we shall study in the following section).

2.1 Orders over Σ

Let Σ an alphabet with n symbols. We want to find a bijection between ω and Σ^* assuming some order $\leq$ over Σ. Let $s^{\leq} : \Sigma^* \mapsto \Sigma^*$ be

$$
\begin{aligned}
s^{\leq}((a_n)^m) &= (a_1)^{m+1} && m \geq 0 \\
s^{\leq}(\alpha a_i (a_n)^m) &= \alpha a_{i+1} (a_1)^m && 1 \leq i < n, m \geq 0
\end{aligned}
$$

This function enumerates the language ordered Σ. For example, consider $\Sigma = \{@, !\}$ with $@ <!$. Then

$$
\begin{aligned}
s^{\leq}(\varepsilon) &= s^{\leq}(!^0) = @ \\
s^{\leq}(@) &= s^{\leq}(\varepsilon @ (!)^0) = \varepsilon ! \varepsilon = \ ! \\
&\vdots
\end{aligned}
$$

Repeated application of this logic outputs the following enumeration:

$$@, !, @@, @!, !@, !!, @@@, @@!, @!@, @!!, !@@, !@!, !!@, !!!, \ldots$$

The reason why $s^{\leq}(\beta)$ enumerates the language is that every β is either of the form $(a_n)^m$ or $\alpha a_i (a_n)^m$. This is, it is either a word with only the last character to a certain exponent, or a word with some subchain before the last character to a certain exponent.

Now we are ready to define a bijection between ω and Σ^*. Let

$$\begin{aligned} *^{\leq} : \omega &\mapsto \Sigma^* \\ x &\mapsto \begin{cases} \varepsilon & x = 0 \\ s^{\leq}\left(*^{\leq}(i)\right) & x = i + 1 \end{cases} \end{aligned}$$

For example, using the same alphabet as before, this function maps

$$\begin{aligned} 0 &\mapsto \varepsilon \\ 1 &\mapsto @ \\ 2 &\mapsto ! \\ 3 &\mapsto @@ \\ 4 &\mapsto @! \\ 5 &\mapsto !@ \\ 6 &\mapsto !! \\ 7 &\mapsto @@@ \\ &\vdots \end{aligned}$$

Now, observe that any $\alpha \in \Sigma^*$ is a concatenation of unique symbols, and that each of this unique symbols is the ith element of Σ^* for some i. We write to express this $\alpha = a_{i_k} \ldots a_{i_0}$ where $i_k, i_{k-1}, \ldots, i_{k_0} \in \{1, \ldots, n\}$. Then we define the inverse of the previous function as follows:

$$\begin{aligned} \#^{\leq} : \Sigma^* &\mapsto \omega \\ \varepsilon &\mapsto 0 \\ a_{i_k} \ldots a_{i_0} &\mapsto i_k n^k + \ldots + i_0 n^0 \end{aligned}$$

For example, consider $\alpha = @!@ = a_1a_2a_1$. Then $\#^{\leq}(\alpha) = 1 \times 2^2 + 2 \times 2^1 + 1 \times 2^0 = 4 + 4 + 1 = 9$. It is easy to verify that $*^{\leq}(9) = @!@$.

Thus, the functions given produce a perfect bijection between numbers and words. Each word can be univocally determined by its numeric position in the language; each number can be univocally determined by a word whose position in the language is that number.

Theorem 6 *Let $n \geq 1$. Then any $x \in \mathbb{N}$ is uniquely written as $x = i_k n^k + i_{k-1} n^{k-1} + \ldots + i_0 n_0$ with $k \geq 0, 1 \leq i_j \leq n$ for all j.*

2.2 Extending the order to words

We can extend $\leq$ from Σ onto Σ^* by letting $\alpha \leq \beta$ if and only if $\#^{\leq}(\alpha) \leq \#^{(\leq)}(\beta)$.

3 Enumerable and computable sets

Let $\mathcal{F} : \mathcal{D}_{\mathcal{F}} \subseteq \omega^k \times \Sigma^{*l} \to \omega^n \times \Sigma^{*m}$. We define $\mathcal{F}_{(i)} = p_i^{n,m} \circ \mathcal{F}$. Then

$$\begin{aligned} \mathcal{F}_{(i)} &: \mathcal{D}_{\mathcal{F}} \subseteq \omega^k \times \Sigma^{*l} \mapsto \omega && 1 \leq i \leq n \\ \mathcal{F}_{(i)} &: \mathcal{D}_{\mathcal{F}} \subseteq \omega^k \times \Sigma^{*l} \mapsto \Sigma^* && n+1 \leq i \leq m \end{aligned}$$

We say a set $S \subseteq \omega^n \times \Sigma^{*m}$ is Σ-effectively enumerable if it is empty or there is a function $\mathcal{F} : \omega \to \omega^n \times \Sigma^{*m}$ s.t. $Im_{\mathcal{F}} = S$ and $\mathcal{F}_{(i)}$ is Σ-computable for all $1 \leq i \leq n+m$.

Theorem 7 *A non-empty set $S \subseteq \omega^n \times \Sigma^{*m}$ is Σ-effectively enumerable if and only if there is an effective procedure $\mathcal{P}$ s.t.*

- *The input space is ω*
- *$\mathcal{P}$ halts for all $x \in \omega$*
- *The output set is S—i.e. whenever $\mathcal{P}$ halts, it outputs an element of S, and for every $(\overrightarrow{x}, \overrightarrow{\alpha}) \in S$ there is some input $x \in \omega$ s.t. $\mathcal{P}(x) \mapsto_{halting} (\overrightarrow{x}, \overrightarrow{\alpha})$.*

Theorem 8 *Let $S \subseteq \omega^n \times \Sigma^{*m}$. The following statements are equivalent.*

(1) S is Σ-effectively enumerable.

(2) S is the domain of a Σ-effectively computable function f.

(3) S is the image of a function $F : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto S$ s.t. each $F_{(i)}$ is Σ-effectively computable.

That (1) $\Leftrightarrow$ (3) is trivial. (1) $\Leftrightarrow$ (2) can be proven as follows.

Proof. ($\Rightarrow$). Assume S is Σ-effectively enumerable. Then there is a procedure $\mathbb{P}$ s.t. for any $x \in \omega$ the procedure outputs a value of S, and all values of S are mapped. Consider the procedure $\mathbb{P}'$ s.t. given an input $(\vec{x}, \vec{\alpha}) \in S$ it *a.* computes $\mathbb{P}$ with input $x = 0, 1, \ldots$ until $\mathbb{P}$ maps to $(\vec{x}, \vec{\alpha})$ and *b.* then returns x. Evidently $\mathbb{P}'$ computes the inverse of the function computed by $\mathbb{P}$. Observe that $\mathbb{P}'$ will only halt if $(\vec{x}, \vec{\alpha}) \in S$. Then S is the domain of a Σ-effectively computable function.

($\Leftarrow$) Assume S is the domain of a Σ-effectively computable function. Let $(\vec{w}, \vec{\beta})$ an arbitrary element of S. Consider a procedure $\mathbb{P}$ which does the following with an input $x \in \omega$.

(1) Produce the enumeration of $\omega \times \omega^n \times \Sigma^{*m}$ given by x. This will produce a variable $(t, \vec{x}, \vec{\alpha})$.
(2) Run t steps of $\mathbb{P}_f$ with input $(\vec{x}, \vec{\alpha})$. If the program terminated, return $(\vec{x}, \vec{\alpha})$. If not, return $(\vec{w}, \vec{\beta})$.

Evidently, $\mathbb{P}$ enumerates S. ■

Since any Σ-effectively computable set is Σ-effectively enumerable, point *(1)* of the previous theorem can be substituted by S is Σ-effectively enumerable—with some loss of generality.

Theorem 9 *Let $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$ with φ either ω or Σ^*. Let $S \subseteq \mathcal{I}_f$. Then if f is Σ-effectively computable and S is Σ-effectively enumerable, $f^{-1}(S) = \{(\vec{x}, \vec{\alpha}) : f(\vec{x}, \vec{\alpha}) \in S\}$ is Σ-effectively enumerable .*

Proof. Assume f is Σ-effectively computable and $S \subseteq \mathcal{I}_f$ is Σ-effectively enumerable. Let $(\vec{w}, \vec{\beta})$ an arbitrary element of $f^{-1}(S)$. Consider the following effective procedure operating over an input $x \in \omega$:

(0) If $x = 0$ return $(\vec{w}, \vec{\beta})$. Else proceed.
(1) Enumerate an element of $s \in S$ using input x.
(2) Use $(\vec{x}, \vec{\alpha}) := ((x)_1, \ldots, (x)_n, *^{\leq}((x)_{n+1}), \ldots, *^{\leq}((x)_m))$ to compute a value of $f(\vec{x}, \vec{\alpha})$.
(3) If $f(\vec{x}, \vec{\alpha}) = s$ return $(\vec{x}, \vec{\alpha})$. Else return $(\vec{w}, \vec{\beta})$.

Evidently, the procedure enumerates $f^{-1}(S)$. ■

The theorem states that the set of inputs which map to a region of a computable function's range is enumerable if that region is enumerable.

Theorem 10 *If $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$ with φ either ω or Σ^*, then if $S \subseteq \mathcal{D}_f$ is Σ-effectively enumerable, $f_{|S}$ is Σ-effectively computable.*

3.1 Prime numbers and enumerable sets

Let $\Sigma \neq \emptyset$ be an alphabet with a total order $\leq$. Let $S \subseteq \omega^n \times \Sigma^{*m}$ a Σ-mixed set of arbitrary dimensions. Notice that for any n-tuple $(x_1, \ldots, x_n)$, with $x_i \in \omega$, we can find a corresponding $\varphi \in \mathbb{N}$ s.t.

$$\varphi = 2^{x_1} 3^{x_2} \ldots p(n)^{x_n}$$

In other words, $(x_1, \ldots, x_n)$ corresponds to the exponents of the n prime factors of a unique natural number. At the same time, the m-tuple $(\alpha_1, \ldots, \alpha_m)$ corresponds to a unique $\psi \in \mathbb{N}$ s.t.

$$\psi = 2^{y_1} 3^{y_2} \ldots p(m)^{y_m}$$

where $\alpha_j = *^{\leq}(y_j)$. In other words, $(\alpha_1, \ldots, \alpha_m)$ corresponds to a unique natural number whose m prime factors have exponents given by the position of each word in the language.

Both of these relations come from the uniqueness of prime factorizations. They provide a way to enumerate Σ-mixed sets. In particular, if S is Σ-total we enumerate it mapping each $x \in \omega$ to $\big((x)_1, \ldots, (x)_n, *^{\leq}((x)_{n+1}), \ldots, *^{\leq}((x)_m)\big)$. If S is not Σ-total, then one can still enumerate it assuming that it is Σ-computable. Indeed, one maps x to the corresponding $(n + m)$-tuple described above if the tuple is in S, and leaves the procedure undefined (or without halt) otherwise. This can be expressed as follows:

Because Σ-total sets are enumerable (as pointed out above), any Σ-mixed set that is Σ-computable is enumerable (via restriction of the Σ-total enumeration).

Theorem 11 *If $S \subseteq \omega^n \times \Sigma^{*m}$ is Σ-effectively computable, then it is Σ-effectively enumerable.*

4 Godel

Definition 1 *A set $S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m$ is rectangular if $S_i \subseteq \omega, L_i \subseteq \Sigma^*$ for all i.*

Lemma 1 *S is rectangular if and only if $(\overrightarrow{x}, \overrightarrow{\alpha}) \in S \wedge (\overrightarrow{y}, \overrightarrow{\beta}) \in S$ implies $(\overrightarrow{x}, \overrightarrow{\beta}) \in S$.*

Example. The set $\{(0, \#\#), (1, \%\%\%)\}$ is not rectangular ($(1, \#\#), (0, \%\%\%)$ are not in S.) Observe how this set cannot be expressed as a product of subsets of ω and Σ. Thus, the concept of *rectangular set* is equivalent to *a set formed via Cartesian product*.

Notation. If $f : \omega_1 \times \ldots \times \omega_n \times \alpha_1 \times \alpha_m \rightarrow \Lambda$ we write $f \sim (n, m, \Lambda)$, and read f is of type n, m to Λ.
Notation. If $f_1, \ldots, f_n$ Σ-mixed functions, then

$$[f_1, \ldots, f_2]\,(\overrightarrow{x}, \overrightarrow{\alpha}) = \left(f_1(\overrightarrow{x}, \overrightarrow{\alpha}), \ldots, f_n(\overrightarrow{x}, \overrightarrow{\alpha})\right)$$

4.1 Primitive recursion

Let R a Σ-mixed function s.t. $R \sim (n, m, \varphi)$, where $\varphi = \omega$ or $\varphi = \Sigma^*$. Primitive recursion consists in recursively defining R with two primitive functions f, which gives the base case, and g, which gives the recursive step. The recursion is done over ω, associating $R(t, \vec{x}, \vec{\alpha})$ with $R(t-1, \vec{x}, \vec{\alpha})$, or over Σ^*, associating $R(\vec{x}, \vec{\alpha}, \alpha a)$ with $R(\vec{x}, \vec{\alpha}, \alpha)$. Since the computation of R may depend on the element which the recursion dismisses (t in the numeric case, a in the alphabetic case), g incorporates this value into its arguments. Thus, we have that:

- If $\varphi = \omega$
  - If the recursion is over ω, $f \sim (n-1, m, \omega), g \sim (n+2, m, \omega)$
  - If the recursion is over Σ^*, $f \sim (n, m-1, \omega), g \sim (n+1, m+1, \omega)$.
- If $\varphi = \Sigma^*$
  - If the recursion is over ω, $f \sim (n-1, m, \Sigma^*), g \sim (n+1, m+1, \Sigma^*)$
  - If the recursion is over Σ^*, $f \sim (n, m-1, \Sigma^*), g \sim (n, m+2, \Sigma^*)$.

We say f, g construct R via primitive recursion and we denote R as $R(f, g)$.

4.2 Numeric to numeric primitive recursion

Let $R \sim (n, m, \#)$. Then functions $f \sim (n-1, m, \#), g \sim (n+1, m, \#)$ recursively define R if and only if

$$\begin{cases} R(0, \overrightarrow{x}, \overrightarrow{\alpha}) & = f(\overrightarrow{x}, \overrightarrow{\alpha}) \\ R(t+1, \overrightarrow{x}, \overrightarrow{\alpha}) & = g\left(R(t, \overrightarrow{x}, \overrightarrow{\alpha}), t, \overrightarrow{x}, \overrightarrow{\alpha}\right) \end{cases}$$

We use the notation $R(f, g)$ to say R is defined by primitive recursion by f and g.

4.3 Numeric to alphabet primitive recursion

Let $R \sim (n, m, \Sigma)$. Then functions $f \sim (n-1, m, \Sigma), g \sim (n, m+1, \Sigma)$ recursively define R if and only if

$$\begin{aligned} R(0, \overrightarrow{x}, \overrightarrow{\alpha}) &= f(\overrightarrow{x}, \overrightarrow{\alpha}) \\ R(t+1, \overrightarrow{x}, \overrightarrow{\alpha}) &= g\left(t, \overrightarrow{x}, \overrightarrow{\alpha}, R(t, \overrightarrow{x}, \overrightarrow{\alpha})\right) \end{aligned}$$

4.4 Alphabet to numeric primitive recursion

If Σ an alphabet, then a Σ-indexed family of functions is a function $\mathcal{G}$ s.t. $D_{\mathcal{G}} = \Sigma$ and for each $a \in D_{\mathcal{G}}$ there is a function $\mathcal{G}(a)$. We write $\mathcal{G}_a$ instead of $\mathcal{G}(a)$.

If $R \sim (n, m, \omega)$ then R can be recursively defined by $f \sim (n, m-1, \omega)$ an indexed family $\mathcal{G}$ s.t. $\mathcal{G}_a \sim (n+1, m, \omega)$ as follows:

$$\begin{cases} R(F, \mathcal{G})(\overrightarrow{x}, \overrightarrow{\alpha}, \varepsilon) = f(\overrightarrow{x}, \overrightarrow{\alpha}) \\ R(f, \mathcal{G})(\overrightarrow{x}, \overrightarrow{a}, \alpha a) = \mathcal{G}_a\left(R(\overrightarrow{x}, \overrightarrow{\alpha}, \alpha), \overrightarrow{x}, \overrightarrow{\alpha}, \alpha\right) \end{cases}$$

4.5 Alphabet to alphabet primitive recursion

If $R \sim (n, m, *)$ then $f \sim (n, m-1, *)$ and $\mathcal{G}$ a Σ-indexed family, with $\mathcal{G}_a \sim (n, m+1, *)$ for all $a \in \Sigma$, define R via primitive recursion if

$$\begin{cases} R(\overrightarrow{x}_n, \overrightarrow{\alpha}_{m-1}, \varepsilon) & = f(\overrightarrow{x}, \overrightarrow{\alpha}) \\ R(\overrightarrow{x_n}, \overrightarrow{\alpha}_{m-1}, \alpha a) & = \mathcal{G}_a\left(\overrightarrow{x}, \overrightarrow{\alpha}, \alpha, R(\overrightarrow{x}, \overrightarrow{\alpha}, \alpha)\right) \end{cases}$$

4.6 The point of primitive recursion

Theorem 12 *If f, g are Σ-computable then $R(f, g)$ is too.*

4.7 The primitive recursive set

Let Σ a language. We define $PR_0^\Sigma = \left\{Suc, Pred, C_0^{0,0}, C_\varepsilon^{0,0}\right\} \cup \{d_a\} \cup \left\{p_j^{n,m}\right\}$. Observe that every $\mathcal{F} \in PR_0^\Sigma$ is Σ-computable. Then we define

$$PR_{k+1} = PR_k^\Sigma \cup \left\{f \circ [f_1 \dots f_r] : f \text{ and } f_i \in PR_k^\Sigma \cup\right\} \cup \left\{R(f, g) : f, g \in PR_k^\Sigma\right\}$$

and $PR^\Sigma = \bigcup_{k \geq 0} PR_k^\Sigma$.
Note. Observe that when we include $R(f, g) : f, g \in PR_k^\Sigma$, we also include the case where $g = \mathcal{G}$ an indexed family of functions.

I provide a list of functions that are in PR^Σ for any Σ.

- Addition, multiplication and factorial
- String concatenation and string length
- All constant functions $C_k^{n,m}$ for any $k, n, m \in \omega$.
- Two-variable exponentiation: $\lambda xy\,[x^y]$.
- Two-variable string exponentiation: $\lambda x\alpha\,[\alpha^x]$.

With $x - y := \max(x - y, 0)$ the list may continue:

- The maximum of two numeric variables
- The predicates $x = y, x \leq y, \alpha = \beta$.
- The predicate x is even.
- The predicate $x = |\alpha|$.
- The predicate $\alpha^x = \beta$.

4.8 Predicates

The $\vee, \wedge$ operators are defined only for predicates of the same type. In other words, $P \circ Q$, where $\circ \in \{\wedge, \vee\}$, is defined only if $P \sim (n, m, \#) \wedge Q \sim (n, m, \#)$. If P, Q are Σ-p.r. then $P \circ Q$ and $\neg P$ also are. Furthermore, P, Q must have the same domains.

4.9 Primitive recursive sets

Definition 2 *A Σ-mixed $S \sim (n, m)$ set is primitive recursive if and only if its characteristic function $\chi_S^{\omega^n \times \Sigma^{m*}}$ is Σ-p.r.*

If S_1, S_2 are Σ-p.r. then their union, intersection and difference are. The proof follows from the fact that

$$\begin{aligned}
\chi_{S_1 \cup S_2} &= \left(\chi_{S_1} \vee \chi_{S_2}\right) \\
\chi_{S_1 \cap S_2} &= \left(\chi_{S_1} \wedge \chi_{S_2}\right) \\
\chi_{S_1 - S_2} &= \lambda xy[x - y] \circ \left[\chi_{S_1}, \chi_{S_2}\right]
\end{aligned}$$

The only property here that may not be immediately intuitive is the last one. But observe that $S_1 - S_2 = \{s \in S_1 : s \notin S_2\}$. Now, let $\chi_{S_1}(\overrightarrow{x}, \overrightarrow{\alpha}) = a, \chi_{S_2}(\overrightarrow{x}, \overrightarrow{\alpha}) = b$. Evidently, if the $n+m$-tuple is in S_1 but not in S_2, $a - b = 1$. If the tuple is in both sets, $a - b = 0$. Etc.

Theorem 13 *A rectangular set $S_1 \times \ldots \times S_n \times L_1 \times \ldots L_m$ is Σ-p.r. if and only if each $S_1, \ldots, S_n, L_1, \ldots, L_m$ is Σ-p.r.*

This theorem is important, insofar as it allows us to evaluate whether a Cartesian product is Σ-p.r. only by looking at its set factors. This theorem should follow from the properties of primitive recursive sets mentioned before.

Theorem 14 *If $f \sim (n, m, \Omega)$ is Σ-p.r (not necessarily Σ-total) and S is a Σ-p.r. set, then $f_{|S}$ is Σ-p.r.*

The previous theorem is useful in proving a function is Σ-p.r. For example, let $P = \lambda x \alpha \beta \gamma \left[x = |\gamma| \wedge \alpha = \gamma^{Pred(|\beta|)}\right]$. We cannot use the fact that both predicate functions are Σ-p.r. to conclude that P is Σ-p.r., because $P_1 = \lambda x \alpha [x = |\alpha|]$ and $P_2 = \lambda x \alpha \beta \gamma [\alpha = \gamma^{Pred(|\beta|)}]$ do not have the same domains. Simply observe that β cannot take the value ε in P_2, but it can take in P_1.

However, observe that $\mathcal{D}_P = \omega \times \Sigma^* \times (\Sigma^* - \varepsilon) \times \Sigma^*$. This set is Σ-p.r. because $\chi_{\mathcal{D}_P}^{1,3} = \neg\lambda [\alpha = \beta] \circ \left[p_3^{1,3}, C_\varepsilon^{1,3}\right]$ is Σ-p.r. Now, we can safely say that $P = P_{1|\mathcal{D}_P} \wedge P_2$, ensuring with the restriction that both predicates have the same domain. Since $\mathcal{D}_P$ is Σ-p.r. so is $P_{1|\mathcal{D}_P}$, form which readily follows that so is P. ■

Theorem 15 *A set S is Σ-p.r. if and only if it is the domain of a Σ-p.r. function.*

Theorem 16 *If S_1, S_2 are Σ-p.r. sets, then $S_1 \cup S_2, S_1 \cap S_2, S_1 - S_2$ are Σ-p.r. sets.*

From the two previous theorems follows that if f_1, f_2 are Σ-p.r. then $f_1 \cap f_2$ are Σ-p.r. The reason is that $\mathcal{D}_{f_1 \cap f_2} = \mathcal{D}_{f_1} \cap \mathcal{D}_{f2}$, which are both Σ-p.r. by virtue of being domains of Σ-p.r. functions. Then their intersection is Σ-p.r. Then they are the domain of a Σ-p.r.—namely $f_1 \cap f_2$.

The same *does not* happen with $f_1 \cup f_2$ because this set may violate the definition of a function; namely, that to each $(\vec{x}, \vec{\alpha})$ corresponds a unique element $((\vec{x}, \vec{\alpha}), f(\vec{x}, \vec{\alpha}))$.

4.10 Case division

If $f_1, \ldots, f_n$ are s.t. $D_{f_j} \cap D_{f_k} = \emptyset$ for $j \neq k$ and $f_j \mapsto \Omega$, then $\mathcal{F} = f_1 \cup \ldots \cup f_n$ is s.t.

$$\mathcal{F} : D_{f_1} \cup \ldots \cup D_{f_n} \to \Omega$$

$$e \to \begin{cases} f_1(e) & e \in D_{f_1} \\ \vdots & \\ f_n(e) & e \in D_{f_n} \end{cases}$$

Under the same constraints, if f_i is Σ-p.r. for all i, then $\mathcal{F}$ is Σ-p.r. This reveals a proving method. Given a function $\mathcal{H}$, we can prove it is Σ-p.r. by proving it is the union of Σ-p.r. functions, under the constraint that the domains of these functions are disjoint.

For example, this can be used to prove that $\lambda\alpha\,[[\alpha]_i]$ is Σ-p.r. Assume a language Σ. Then

$$[\alpha a]_i = \begin{cases} a & i = |\alpha| + 1 \\ [\alpha]_i & \text{otherwise} \end{cases}$$

for any $a \in \Sigma$. The base case is the trivial $[\varepsilon]_i = \varepsilon$. From this follows that $R = [\alpha]_i \sim (1, 1)$ is difined via primitive recursion by $f = C_\varepsilon^{1,0}$ and $\mathcal{G}$ an indexed family where $\mathcal{G}_a$ is of the form above for every a. Evidently f is Σ-p.r.; now we want to prove $\mathcal{G}_a$ is Σ-p.r. for any $a \in \Sigma$.

Observe that the sets $S = \{(i, \alpha, \zeta) : i = |\alpha| + 1\}$ and its complement $\overline{S}$ are disjoint and Σ-p.r. (We skip the proof of this statement.) It follows from the division by cases that

$$\mathcal{G}_a = p_3^{1,2}{}_{|S} \cup C_a^{1,2}{}_{|\overline{S}}$$

is Σ-p.r. Thus, $R = [\alpha]_i$ is Σ-p.r.

Problem 2 *Let $\Sigma = \{@, \$\}$. Let $h : \mathbb{N} \times \Sigma^+ \mapsto \omega$ be x^2 if $x + |\alpha|$ is even, 0 otherwise. Prove that f is Σ-p.r.*
Let $S = \{(n, \alpha) \in \mathbb{N} \times \Sigma^* : n + |\alpha| \equiv 0 \mod 2\}$ and $g := \lambda x \alpha \left[x^2\right]$.
g is Σ-p.r.
$\therefore$ $g_S \cup C_0^{1,1}{}_{\overline{S}} = f$ is Σ-p.r.

Problem 3 *Let h have $\mathcal{D}_h = \{(x, y, \alpha) : x \leq y\}$ and be s.t. $R \mapsto x^2$ if $|\alpha| \leq y$, zero otherwise. Show h is Σ-p.r.*

Let $S := \{(x, y, \alpha) \in \mathcal{D}_h : y \leq |\alpha|\}$. Evidently, $h = f_1 = C_0^{2,3}$ when $|\alpha| > y$ (this is, when the argument is in $\overline{S}$). When the argument is in S, it is $f_2 = \lambda x[x^2] \circ [p_1^{2,1}]$. It is trivial to observe both functions are Σ-p.r. Then $h = f_{1|\overline{S}} \cup f_{2|S}$, where of course $S \cup \overline{S} = \mathcal{D}_h$.

4.11 Summation, product and concatenation

Let $f \sim (n + 1, m, \#)$ with domain $\mathcal{D}_f = \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m$, with $S_i \subseteq \omega, L_i \subseteq \Sigma^*$. Then we define $\sum_{t=x}^{t=y} f(t, \overrightarrow{x}, \overrightarrow{\alpha})$ in the usual way, with the constraint that the sum is 0 if $y > x$. In the same way we deifne $\prod_{t=x}^{t=y} f(t, \overrightarrow{x}, \overrightarrow{\alpha})$ and the concatenation $\subset_{t=x}^{t=y} f(t, \overrightarrow{x}, \overrightarrow{\alpha})$ for the case $I_f \subseteq \Sigma^*$.

The domain of each of these is $\mathcal{D} = \omega \times \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m$, where the first two ω elements are the x, y domains of the sum.

Theorem 17 *If f is Σ-p.r. then the functions are Σ-p.r.*

To understand why, let $G = \lambda t x \overrightarrow{x} \overrightarrow{\alpha} \left[\sum_{i=x}^{i=t} f(i, \overrightarrow{x}, \overrightarrow{\alpha})\right]$. Evidently, $G = \circ \left[p_2^{n+2,m}, p_1^{n+2,m}, p_3^{n+2,m}, \ldots, p_{n+2+m}^{n+2,m}\right]$ and so we only need to prove G is Σ-p.r. Observe that

$$G(0, x, \overrightarrow{x}, \overrightarrow{\alpha}) = \begin{cases} 0 & x > 0 \\ f(0, \overrightarrow{x}, \overrightarrow{\alpha}) & x = 0 \end{cases}$$

$$G(t+1, x, \overrightarrow{x}, \overrightarrow{\alpha}) = \begin{cases} 0 & x > t+1 \\ G(t, x, \overrightarrow{x}, \overrightarrow{\alpha}) + f(t+1, \overrightarrow{x}, \overrightarrow{\alpha}) \end{cases}$$

Thus, if we let each of these functions be called h, g we have that $G = R(h, g)$. Suffices to show h, g are Σ-p.r. This can be proven using division by cases and domain restriction.

Problem 4 *Prove that* $G = \lambda x x_1 \left[\sum_{t=1}^{t=x} Pred(x_1)^t\right]$ *is* Σ*-p.r.*

We know $f = \lambda x t \left[Pred(x)^t\right]$ is Σ-p.r. (trivial to show). Let $\mathcal{G} = \lambda x y x_1 \left[\sum_{t=x}^{t=y} f(x_1, t)\right]$. We know from the last theorem that $\mathcal{G}$ is Σ-p.r. It is evident that $G = \mathcal{G} \circ \left[C_1^{2,0}, p_1^{2,0}, p_2^{2,0}\right]$. Then G is Σ-p.r. ■

Show it to me. Well, $G(x, x_1) = \left(\mathcal{G} \circ \left[C_1^{2,0}, p_1^{2,0}, p_2^{2,0}\right]\right)(x, x_1) = \mathcal{G}(0, x, x_1) = \sum_{t=0}^{t=x} f(x_1, t)$.

Problem 5 *Show that* $G = \lambda x y \alpha \left[\prod_{t=y+1}^{t=|\alpha|} (t + |\alpha|)\right]$ *is* Σ*-p.r.*

It is trivial to show $f = \lambda t \alpha \left[t + |\alpha|\right]$ is Σ-p.r. Let

$$\mathcal{G} = \lambda x y \alpha \left[\prod_{t=x}^{t=y} (t + |\alpha|)\right]$$

which is Σ-p.r. Observe that $G(x, y, \alpha) = \mathcal{G}(y + 1, |\alpha|, \alpha)$. Then

$$G = \mathcal{G} \circ \left[Suc \circ p_2^{2,1}, \lambda \alpha [|\alpha|] \circ p_3^{2,1}, p_3^{2,1}\right]$$

Then G is Σ-p.r. ■

Problem 6 *Prove that*

$$\lambda x y z \alpha \beta \left[\mathop{\subset}_{t=3}^{t=z+5} \alpha^{Pred(z) \cdot t} \beta^{Pred(Pred(|\alpha|))}\right]$$

is Σ*-p.r.*

Let G denote the function in question. First of all, observe that $\mathcal{D}_G = \omega^2 \times \mathbb{N} \times \Sigma^{*2}$—which means G is not Σ-total. Let us divide our proof by parts.
(1) Let $\mathcal{F} = \lambda xy\alpha\beta\left[\alpha^{Pred(x)\cdot y}\beta^{Pred(Pred(|\alpha|))}\right]$, where evidently $\mathcal{F} \sim (2,2,*)$ with $x \in \mathbb{N}$. Observe that

$$\begin{aligned}
\mathcal{F}_1 &:= \lambda xy\alpha\left[\alpha^{Pred(x)y}\right] \\
&= \lambda x\alpha\left[\alpha^x\right] \circ \left[\lambda xy\left[xy\right] \circ \left[Pred \circ p_1^{2,1}, p_2^{2,1}\right], p_3^{2,1}\right] \\
\mathcal{F}_2 &:= \lambda\alpha\beta\left[\alpha^{Pred(Pred(|\alpha|))}\right] \\
&= \lambda x\alpha[\alpha^x] \circ \left[p_1^{0,2}, Pred \circ \left[Pred \circ \left[\lambda\alpha[|\alpha|] \circ p_2^{0,2}\right]\right]\right]
\end{aligned}$$

and evidently

$$\begin{aligned}
\mathcal{F} &= \lambda xy\alpha\beta[\mathcal{F}_1(x,y,\alpha)\mathcal{F}_2(\beta,\alpha)] \\
&= \lambda\alpha\beta[\alpha\beta] \circ \left[\mathcal{F}_1 \circ \left[p_1^{2,2}, p_2^{2,2}, p_3^{2,2}\right], \mathcal{F}_2 \circ \left[p_4^{2,2}, p_3^{2,2}\right]\right]
\end{aligned}$$

This proves $\mathcal{F}$ is Σ-p.r.
(2) It is evident that $G = \lambda xyz\alpha\beta\left[\subset_{t=3}^{t=z+5} \mathcal{F}(z,t,\alpha,\beta)\right]$.
If we let

$$\mathcal{G} := \lambda xyz\alpha\beta\left[\overset{t=y}{\underset{t=x}{\subset}} \mathcal{F}(z,t,\alpha,\beta)\right]$$

it is evident that $G = \mathcal{G} \circ \left[C_3^{3,2}, \lambda z[z+5] \circ p_3^{3,2}, p_3^{3,2}, p_4^{3,2}, p_5^{3,2}\right]$.
Then G is Σ-p.r. ■

4.12 Predicate quantification

If $P : S_0 \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m$ is a predicate and $S \subseteq S_0$, then
$(\forall t \in S)_{t \leq x} P(t, \overrightarrow{x}, \overrightarrow{\alpha})$ is 1 when $P(t, \overrightarrow{x}, \overrightarrow{\alpha}) = 1$ for all $t \in \{u \in S : u \leq x\}$. The domain of the quantified proposition is $\omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m$, where the first argument (accounted by ω) is the upper bound x. We generalize, where $L \subseteq L_{m+1}$, $S \subseteq S_0$:

$$(\forall t \in S)_{t\leq x}P(t,\overrightarrow{x},\overrightarrow{\alpha}) : \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \to \{0,1\}$$
$$(\exists t \in S)_{t\leq x}P(t,\overrightarrow{x},\overrightarrow{\alpha}) : \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \to \{0,1\}$$
$$(\forall \alpha \in L)_{|\alpha|\leq x}P(\overrightarrow{x},\overrightarrow{\alpha},\alpha) : \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \to \{0,1\}$$
$$(\exists \alpha \in L)_{|\alpha|\leq x}P(\overrightarrow{x},\overrightarrow{\alpha},\alpha) : \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \to \{0,1\}$$

It is important to observe that the set over which the quantification is done is a subset of the set from which comes the driving variable t (in the numeric case) or α (in the alphabetic case).

Theorem 18 *(1) If $P : S_0 \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \to \omega$ a predicate Σ-p.r., and $S \subseteq S_0$ is Σ-p.r., then both quantifications over P are Σ-p.r.*
(2) If $P : S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m L_{m+1} \to \omega$ a predicate Σ-p.r., and $L \subseteq L_{m+1}$ is Σ-p.r., then both quantifications over P are Σ-p.r.

The theorem above states that the quantification over a Σ-p.r. set of a Σ-p.r. predicate is itself Σ-p.r. Though unbounded quantification does not preserve these properties, in general a bound exists "naturally" for quantifications, which serves to prove that a bounded quantification is Σ-p.r.. Consider the following example.

Example. The predicate $\lambda xy[x \mid y]$ is Σ-p.r, because $P = x_1x_2[x_2 = tx_1]$ is Σ-p.r. Since P is Σ-p.r., any **bounded** quantification of it over a Σ-p.r. set is itself Σ-p.r. For example,

$$\lambda x x_1 x_2 \left[(\exists t \in \omega)_{t\leq x} x_2 = tx_1\right]$$

is Σ-p.r. Now, observe that if $x_2 = tx_1$ then it is necessary that $t \leq x_2$. But

$$\begin{aligned}&\lambda x_1 x_2 \left[(\exists t \in \omega)_{t\leq x_2} x_2 = tx_1\right]\\ =&\lambda x x_1 x_2 \left[(\exists t \in \omega)_{t\leq x} x_2 = tx_1\right] \circ \left[p_2^{2,0}, p_1^{2,0}, p_2^{2,0}\right]\end{aligned}$$

Then the **bounded** quantification, with x_2 as bound, is Σ-p.r.

Problem 7 *Let* $\Sigma = \{@, !\}$. *Show that* $S = \{(2^x, @^x, !) : x \in \omega \wedge x \text{ impar}\}$ *is* Σ*-p.r.*
For clarity, observe that a few elements of S are

$$(2, @, !), (8, @@@, !), (32, @@@@@, !), \ldots$$

Let $P_1 = \lambda xy\alpha \left[x = 2^{y+1}\right], P_2 = \lambda xy\alpha[\alpha = @^{y+1}]$. It is clear that $\mathscr{D}_{P_1} = \mathscr{D}_{P_2}$. It is trivial to prove that both are Σ-p.r. Then $P_1 \wedge P_2$ is Σ-p.r. Then

$$\chi_S^{1,2} = \lambda xy\alpha\beta\left[(\exists k \in \omega)_{k \leq x}\left(P_1(y, k, \alpha) \wedge P_2(y, y, \alpha)\right) \wedge \beta = !\right]$$

is Σ-p.r.

4.13 Minimization of numeric variable

Let P an arbitrary predicate over a numeric variable. If there is some $t \in \omega$ s.t. $P(t, \overrightarrow{x}, \overrightarrow{\alpha})$ holds, we use $\min_t P(t, \overrightarrow{x}, \overrightarrow{\alpha})$ to denote the minimum t that holds. This is **not defined** if there is no tuple $(\overrightarrow{x}, \overrightarrow{\alpha})$ over which the predicate holds. Furthermore, $\min_t P(t, \overrightarrow{x}, \overrightarrow{\alpha}) = \min_i P(i, \overrightarrow{x}, \overrightarrow{\alpha})$; this is, $\min_t$ does not depend on the variable t.

We define

$$M(P) = \lambda \overrightarrow{x}\,\overrightarrow{\alpha}\left[\min_t P(t, \overrightarrow{x}, \overrightarrow{\alpha})\right]$$

We say $M(P)$ is obtained via minimization of the numeric variable from P.
Example. Let $Q : \omega \times \mathbb{N}$ be s.t. $Q(x, y)$ denotes the quotient of $\frac{x}{y}$. This quotient is by definition the maximum element of $\{t \in \omega : ty \leq x\}$. Let $P = \lambda txy\,[ty \leq x]$. Observe that

$$\mathscr{D}_{M(P)} = \left\{(x, y) \in \omega^2 : (\exists t \in \omega)P(t, x, y) = 1\right\}$$

If $(x, y) \in \omega \times \mathbb{N}$, one can show that $\min_t x < ty = Q(x, y) + 1$. Then $M(P) = Suc \circ Q$.

The U rule. If f is a Σ-mixed function with type $(n, m, \#)$ and we want to find a predicat P s.t. $f = M(P)$, it is sometimes useful to design P so

$$f(\overrightarrow{x}, \overrightarrow{\alpha}) = \text{only } t \in \omega \text{ s.t. } P(t, \overrightarrow{x}, \overrightarrow{\alpha})$$

Problem 8 *Use the **U rule** to find a predicate P s.t. $M(P) = \lambda x[$integer part of $\sqrt{x}]$.*
Let $f(x)$ denote the integer part of $\sqrt{x}$. If $f(x) = y$ then $y^2 \leq x \wedge (y+1)^2 > x$. Then letting $P = \lambda xy \left[x^2 \leq y \wedge (x+1)^2 > y\right]$ ensures that $M(P(x, y)) = f(x)$.

Problem 9 *Find P s.t. $M(P) = \lambda xy\,[x - y]$.*

Since $x - y$ is unique for each pair x, y, $P = \lambda xyz[z = x - y]$. Then $\min_z P(x, y, z) = \lambda xy[x - y]$. For example, $3 - 5 = 0$ and $\min_z P(3, 5, z) = 0$.

Theorem 19 *If P a predicate that is effectively computable and $\mathcal{D}_P$ is effectively computable, then $M(P)$ is effectively computable.*

4.14 Recursive function

Now we define $R_0^\Sigma = PR_0^\Sigma$ and

$$\begin{aligned} R_{k+1}^\Sigma = & R_k^\Sigma \\ & \cup \left\{f \circ [f_1, \ldots, f_n] : f_i \in R_k^\Sigma\right\} \\ & \cup \{R(f, g) : f, g \in R_k^\Sigma\} \\ & \cup \left\{M(P) : \text{P is } \Sigma\text{-total} \wedge P \in R_k^\Sigma\right\} \end{aligned}$$

In other words, recursive functions are all primitive recursive functions plus all predicate minimization functions over Σ-total and recursive predicates.
We define $R^\Sigma = \bigcup_{k \geq 0} R_k^\Sigma$.

Theorem 20 *If $f \in R^\Sigma$ then f is Σ-effectively computable.*

Theorem 21 *Not every Σ-recursive function is Σ-p.r. In other words,*

$$PR^{\Sigma} \subseteq R^{\Sigma} \text{ but } PR^{\Sigma} \neq R^{\Sigma}$$

It is obvious by definition that if f is Σ-p.r. then it is recursive. But if a function is recursive, it could very well be a minimization predicate over a Σ-total function that is not Σ-p.r. itself! In other words,

$$R^{\Sigma} - PR^{\Sigma} = \left\{M(P) : P \text{ is } \Sigma\text{-p.r.} \wedge P \in R^{\Sigma} \wedge M(P) \text{ is not } \Sigma\text{-p.r.}\right\}$$

In fact, the theorems in previous sections ensured that if P is Σ-p.r. and so is $\mathfrak{D}_P$, then $M(P)$ is Σ-effectively computable. Which doesn't entail that it is Σ-p.r.

Theorem 22 *If $P \sim (n+1, m, \#)$ is a Σ-p.r. predicate then (1) $M(P)$ is Σ-recursive. If there is a Σ-p.r.function $f \sim (n, m, \#)$ s.t. $M(P)(\overrightarrow{x}, \overrightarrow{\alpha}) = \min_t P(t, \overrightarrow{x}, \overrightarrow{\alpha}) \leq f(\overrightarrow{x}, \overrightarrow{\alpha})$ for all $(\overrightarrow{x}, \overrightarrow{\alpha}) \in \mathfrak{D}_{M(P)}$, then $M(P)$ is Σ-p.r.*

The theorem above gives the conditions to say whether $M(P)$ is recursive and whether it is Σ-p.r. It is recursive simply if P is Σ-p.r. And it is Σ-p.r. if $M(P)$ is bounded by some function f for all values in the domain of $M(P)$.

Theorem 23 *The quotient function, the remainder function, and the ith prime function are Σ-p.r.*

4.15 Minimization of alphabetic variable

We define $M^{\leq}(P) = \lambda \overrightarrow{x}\, \overrightarrow{\alpha} \left[\min_{\alpha}^{\leq} P(\overrightarrow{x}, \overrightarrow{\alpha}, \alpha)\right]$, where $\leq$ is some order over the language Σ in question.

Theorem 24 *If P is Σ-p.r. predicate over a string, then the same conditions apply for $M(P)$ to be Σ-p.r. as in the theorem for predicates over numbers.*

Problem 10 *Prove that $\lambda\alpha[\sqrt{\alpha}]$ is Σ-p.r.*

Observe that $\lambda\alpha\left[\sqrt{\alpha}\right] = \min_\alpha \lambda\alpha\beta[\beta = \alpha\alpha]$. The predicate, which we call P, is trivially Σ-p.r. This means that $\lambda\alpha[\sqrt{\alpha}] \in R^\Sigma$.
Let $M(P)$ denote the minimization above. Then $M\,(P\,(\alpha, \beta)) \leq \beta$. In other words, $M(P)$ is bounded by $f = \lambda\alpha[\alpha]$.
Then $\lambda\alpha[\sqrt{\alpha}] \in PR^\Sigma$.

4.16 Enumerable sets

We say $S \subseteq \omega^n \times \Sigma^{*2}$ is Σ-recursively enumerable if it is empty or there is a function $\mathcal{F} : \omega \to \omega^n \times \Sigma^{*2}$ s.t.

- $Im_{\mathcal{F}} = S$
- $\mathcal{F}_{(i)}$ is Σ-recursive for every $1 \leq 1 \leq n + m$.

Here, Σ-recursive functions model Σ-computable functions.

4.17 Recursive sets

The Godelian model of a Σ-effectively computable set is simple. A set S is Σ-recursive when χ_S is Σ-recursive.

4.18 Alphabet independence

Theorem 25 *Let Σ, Γ two alphabets. If f is Σ-mixed and Γ-mixed, then f is Σ-recursive iff it is Γ-recursive. The analogue applies to recursive sets and this extends to primitive recursion.*

The theorem above states that recursiveness or primitive-recursiveness is independent of any given alphabet.

5 Neumann

5.1 The $\mathcal{S}^{\Sigma}$ language

We provide von Neumann's model of Σ-effectively computable function. We use $Num = \{0, 1, \ldots, 9\}$ a set of *symbols* (not numbers) and define $S : Num^* \mapsto Num^*$ as

$$\begin{aligned} S(\varepsilon) &= 1 \\ S(\alpha 0) &= \alpha 1 \\ S(\alpha 2) &= \alpha 3 \\ &\vdots \\ S(\alpha 9) &= S(\alpha)0 \end{aligned}$$

It is easy to observe that S is a "counting" or "enumerating" function of the alphabet Num. We define

$$\begin{aligned} \overline{} : \omega &\mapsto Num^* \\ \overline{0} &\mapsto \varepsilon \\ \overline{n+1} &\mapsto S(\overline{n}) \end{aligned}$$

In other words, $\overline{n}$ simply denotes the alphabetic symbol of Num that denotes the number n. The whole syntax of the S^{Σ} language is given by $\Sigma \cup \Sigma_p$, where

$$\Sigma_p = Num \cup \{\leftarrow, +, =, \ .\ , \neq, \curvearrowright, \varepsilon, N, K, P, L, I, F, G, O, T, B, E, S\}$$

It is important to note that these are *symbols* or *strings*, not values. The ε in Σ_p is not the empty letter, but the symbol that denotes it. The $\overline{+}$, $-$ signs are not the operations plus and minus, but the same symbols that denote these operations.

5.2 Variables, labels, and instructions

Any word of the form $N\overline{k}$ is a numeric variable; $P\overline{k}$ is an alphabetic variable; $L\overline{k}$ is a label.
A basic instruction in $\mathcal{S}^{\Sigma}$ is one of the following words:

- $N\overline{k} \leftarrow N\overline{k} - 1$
- $N\overline{k} \leftarrow N\overline{k} + 1$
- $N\overline{k} \leftarrow N\overline{n}$
- $N\overline{k} \leftarrow 0$
- $P\overline{k} \leftarrow \curvearrowright P\overline{k}$
- $P\overline{k} \leftarrow P\overline{k}.a$
- $P\overline{k} \leftarrow P \leftarrow \varepsilon$
- $IF\ N\overline{k} \neq 0\ GOTO\ L\overline{n}$
- $IF\ P\overline{k}BEGINS\ a\ GOTO\ L\overline{n}$
- $GOTO\ L\overline{n}$
- SKIP

An instruction is any word of the form αI where $\alpha \in \{L\overline{n} : n \in \mathbb{N}\}$ and I is a basic instruction. We use Ins^{Σ} to denote the set of all instructions in $\mathcal{S}^{\Sigma}$. When $I = L\overline{n}J$ and J a basic instruction, we say $L\overline{n}$ is the label of J.

5.3 Programs in $\mathcal{S}^{\Sigma}$

A program in $\mathcal{S}^{\Sigma}$ is any word $I_1 \ldots I_n$, with $n \geq 1$, s.t. $I_k \in Ins^{\Sigma}$ for all $1 \leq k \leq n$ and the following property holds:

GOTO Law: For every $1 \leq i \leq n$, if $GOTOL\overline{m}$ is the end of Ii, then there is some $j, 1 \leq j \leq n$, s.t. I_j has label $L\overline{m}$.

Informally, a program is any chain of instructions satisfying that GOTO instructions map to actual labels in the program. We use Pro^{Σ} to denote the set of all programs in $\mathcal{S}^{\Sigma}$.

> *Observation.* Note that $Ins^{\Sigma} \not\subseteq Pro^{\Sigma}$. The reason is Ins^{Σ} contains chains of instructions that do not satisfy the GOTO law. For example, the single line $IFN1 \neq 0\ GOTOL1 \in Ins^{\Sigma}$ is not a program.

Theorem 26 *Let Σ a finite alphabet. Then*

- *If $I_1 \ldots I_n = J_1 \ldots J_n$, with $I_k, J_k \in Ins^{\Sigma}$, then $n = m$ and $I_k = J_k$ for all k.*
- *If $\mathcal{P} \in Pro^{\Sigma}$ then there is a unique set of instructions $I_1 \ldots I_n$ s.t. $\mathcal{P} = I_n \ldots I_n$.*

The theorem above establishes that any program in Pro^{Σ} is a *unique* concatenation of instructions. We use $n(\mathcal{P})$ to

denote the number of instructions that make up $\mathscr{P} \in Pro^{\Sigma}$. By convention, if $\mathscr{P} = I_1^{\mathscr{P}} \dots I_{n(\mathscr{P})}^{\mathscr{P}}$, then $I_j^{\mathscr{P}} = \varepsilon$ if $j \notin [1, n(\mathscr{P})]$. In other words, we understand that a program contains infinitely many empty symbols to the right and left (like in Turing machines).

Observation. $n(\alpha)$ and I_j^{α} are defined only when $\alpha \in Pro^{\Sigma}, i \in \omega$. This means the domain of $\lambda\alpha[n(\alpha)]$ is $Pro^{\Sigma} \subseteq \Sigma \cup \Sigma_p$ and that of $\lambda i\alpha[I_i^{\alpha}]$ is $\omega \times Pro^{\Sigma}$.

Problem 11 *Is is true that $Ins^{\Sigma} \cap Pro^{\Sigma} = \emptyset$? And is it true that $\lambda i\mathscr{P}[I_i^{\mathscr{P}}]$ has domain $\left\{(i, \mathscr{P}) \in \mathbb{N} \times Pro^{\Sigma} : i \leq n(\mathscr{P})\right\}$?*
Both statements are false. A single instruction in Ins^{Σ} can be a program (as long as it is not a GOTO statement to a non-existent label). Furthermore, $\lambda i\mathscr{P}[I_i^{\mathscr{P}}]$ is defined for $i = 0$ (it maps to ε) and for $i \geq n(\mathscr{P})$ (it also maps to ε).

Problem 12 *Prove: If $\mathscr{P}_1, \mathscr{P}_2 \in Pro^{\Sigma}$ then $\mathscr{P}_1\mathscr{P}_1 = \mathscr{P}_2\mathscr{P}_2 \Rightarrow \mathscr{P}_1 = \mathscr{P}_2$.*

This follows from the theorem that guarantees that any program $\mathscr{P} \in Pro^{\Sigma}$ is a *unique* concatenation of instructions. Let $\mathscr{P}_1 = I_1^{\mathscr{P}_1} \dots I_{n(\mathscr{P}_1)}^{\mathscr{P}_1}$ and $\mathscr{P}_2 = I_1^{\mathscr{P}_2} \dots I_{n(\mathscr{P}_2)}^{\mathscr{P}_2}$. Assume $\mathscr{P}_1\mathscr{P}_1 = \mathscr{P}_2\mathscr{P}_2$. Then

$$I_1^{\mathscr{P}_1} \dots I_{n(\mathscr{P}_1)}^{\mathscr{P}_1} I_1^{\mathscr{P}_1} \dots I_{n(\mathscr{P}_1)}^{\mathscr{P}_1} = I_2^{\mathscr{P}_2} \dots I_{n(\mathscr{P}_2)}^{\mathscr{P}_2} I_2^{\mathscr{P}_2} \dots I_{n(\mathscr{P}_2)}^{\mathscr{P}_2}$$

Then, from the last theorem follows that $I_k^{\mathscr{P}_1} = i_k^{\mathscr{P}_2}$. From this follows directly that $\mathscr{P}_1 = \mathscr{P}_2$. ■

5.4 States in programs of $\mathcal{S}^{\Sigma}$

.

We define $Bas : Ins^{\Sigma} \mapsto (\Sigma \cup \Sigma_p)^*$, the program that returns the substring of an instruction corresponding to its basic instruction, as

$$Bas(I) = \begin{cases} J & I = L\overline{k}J \\ I & \text{otherwise} \end{cases}$$

Recall that

$$\curvearrowright_\alpha = \begin{cases} [\alpha]_2 \dots \alpha_{|\alpha|}\alpha | & |\alpha| \geq 2 \\ \varepsilon & \text{otherwise} \end{cases}$$

We define $\omega^{[\mathbb{N}]} = \{(s_1, s_2, \dots) : \exists n \in \mathbb{N} : i > n \Rightarrow s_i = 0\}$. Similarly, $\Sigma^{*[\mathbb{N}]}$ denotes the set of infinite alphabetic tuples that contain only ε from some index onwards.

A **state** is a tuple $(\overrightarrow{s}, \overrightarrow{\sigma}) \in \omega^{\mathbb{N}} \times \Sigma^{*\mathbb{N}}$. If $i \geq i$ we say s_i has the value of the $N\bar{i}$ variable in the state, and σ_i the value of the $P\bar{i}$ variable in the state. Thus, a state is a pair of infinite tuples containing the values of the variables in a program.

We use

$$[[x_1, \dots x_n,\ \alpha_1, \dots, \alpha_m]]$$

to denote the state $((x_1, \dots, x_n, 0, 0, \dots), (\alpha_1, \dots, \alpha_m, \varepsilon, \varepsilon, \dots))$.

5.5 Instantaneous description of a program in $\mathcal{S}^\Sigma$

Since a program $\mathcal{P} \in Pro^\Sigma$ may contain GOTO instructions, it is not always the case that $I^{\mathcal{P}}_{k+1}$ is executed after $I^{\mathcal{P}}_k$. Thus, when running a program, we not only need to consider its state but the specific instruction to be executed. An instantaneous description is a mathematical object which describes all this information.

Formally, an instantaneous description is triple $(i, \overrightarrow{s}, \overrightarrow{\alpha}) \in \omega \times \omega^{\mathbb{N}} \times \Sigma^{*\mathbb{N}}$. This Cartesian product is the set of all possible instantaneous descriptions. The triple reads: The following instruction is $I^{\mathcal{P}}_i$ and the current state is $(\overrightarrow{s}, \overrightarrow{\sigma})$. Observe that if $i \notin [1, n(\mathcal{P})]$, then the description reads: We are in state $(\overrightarrow{s}, \overrightarrow{\sigma})$ and we must execute ε (nothing).

We define the successor function

$$S_{\mathcal{P}} : \omega \times \omega^{\mathbb{N}} \times \Sigma^{*\mathbb{N}} \mapsto \omega \times \omega^{\mathbb{N}} \times \Sigma^{*\mathbb{N}}$$

which maps an instantaneous description to the successor instantaneous description (the one after executing the instruction in the first).

5.6 Computation from a given state

Let $\mathscr{P} \in Pro^{\Sigma}$ and a state $(\overrightarrow{s}, \overrightarrow{\sigma})$. The *computation* of $\mathscr{P}$ from $(\overrightarrow{s}, \overrightarrow{\sigma})$ is defined as

$$\left((1, \overrightarrow{\sigma}, \overrightarrow{\sigma}), S_{\mathscr{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right), S_{\mathscr{P}}\left(S_{\mathscr{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right)\right), \ldots\right)$$

In other words, the *computation* of $\mathscr{P}$ is the infinite tuple whose ith element is the instantaneous description of $\mathscr{P}$ after $i-1$ instructions have been executed.
We say $S_{\mathscr{P}}\left(\ldots S_{\mathscr{P}}\left(S_{\mathscr{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right)\right)\right)$ is the instantaneous description obtained after t steps if the number of times $S_{\mathscr{P}}$ was executed is t.

Problem 13 *Give true or false for the following statements.*
Statement 1: If $S_{\mathscr{P}}(i, \overrightarrow{s}, \overrightarrow{\alpha}) = (i, \overrightarrow{s}, \overrightarrow{\alpha})$ *then* $i \notin [1, n(\mathscr{P})]$. The statement is false. It could be the case that $i \notin [1, n(\mathscr{P})]$, in which case we would say the program halted. However, consider the program

$$L1\ GOTO\ L1$$

Evidently, $S_{\mathscr{P}}(1, \overrightarrow{s}, \overrightarrow{\alpha}) = (1, \overrightarrow{s}, \overrightarrow{\alpha})$, and $1 \leq 1 \leq n(\mathscr{P})$.
Statement 2. Let $\mathscr{P} \in Pro^{\Sigma}$ *and* d *an instantaneous description whose first coordinate is* i. *If* $I_i^{\mathscr{P}} = N_2 \leftarrow N_2 + 1$, *then*

$$S_{\mathscr{P}}(d) = (i+1, (N_1, Suc(N_2), N_3, \ldots), (P_1, P_2, P_3, \ldots))$$

The statement is true via direct application of the $S_{\mathscr{P}}$ function.
Statement 3. Let $\mathscr{P} \in Pro^{\Sigma}$ *and* $(i, \overrightarrow{s}, \overrightarrow{\sigma})$ *an instantaneous description. If* $Bas(I_i^{\mathscr{P}}) = IF\ P_3\ BEGIN S\ a\ GOTO\ L_6$ *and* $[P_3]_1 = a$, *then* $S_{\mathscr{P}}(i, \overrightarrow{s}, \overrightarrow{\sigma}) = (j, \overrightarrow{s}, \overrightarrow{\sigma})$, *where* j *is the least number* l *s.t.* $I_l^{\mathscr{P}}$ *has label* L_6.
Because $[P_3]_1 = a$, the value of $S_{\mathscr{P}}(i, \overrightarrow{s}, \overrightarrow{\sigma})$ must indeed contain the instruction that has label L_6. This instruction is the jth instruction for some j, etc. The statement is true.

5.7 Halting

When the first coordinate of $S_{\mathcal{P}}\left(\ldots S_{\mathcal{P}}\left(S_{\mathcal{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right)\right)\right)$ with t steps is $n(\mathcal{P}) + 1$, we say $\mathcal{P}$ *halts after t steps when starting from* $(\overrightarrow{s}, \overrightarrow{\sigma})$.
If none of the first coordinates in the computation of $\mathcal{P}$,

$$\left((1, \overrightarrow{\sigma}, \overrightarrow{\sigma}), S_{\mathcal{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right), S_{\mathcal{P}}\left(S_{\mathcal{P}}\left(1, \overrightarrow{s}, \overrightarrow{\sigma}\right)\right), \ldots\right)$$

is $n(\mathcal{P})$, we say $\mathcal{P}$ does not halt starting from $(\overrightarrow{s}, \overrightarrow{\sigma})$.

5.8 Σ-computable functions

We give the model of a Σ-effectively computable function in the paradigm of von Neumann. Intuitively, f is Σ-computable if there is some $\mathcal{P} \in Pro^{\Sigma}$ that computes it.
Given $\mathcal{P} \in Pro^{\Sigma}$, for every pair $n, m \geq 0$, we define $\Psi_{\mathcal{P}}^{n,m,\#}$ as follows:

$$\mathcal{D}_{\Psi_{\mathcal{P}}^{n,m,\#}} = \left\{(\overrightarrow{x}, \overrightarrow{\alpha}) \in \omega^n \times \Sigma^{*m} : \mathcal{P} \text{ halts from } [[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]\right\}$$

$$\Psi_{\mathcal{P}}^{n,m,\#}(\overrightarrow{x}, \overrightarrow{\alpha}) = \text{Value of } N_1 \text{ in halting state from } [[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]$$

We analogously define $\Psi_{\mathcal{P}}^{n,m,*}$ for the alphabetic case, where the domain is the same and the value is that of P_1 in the halting state.

A Σ-mixed function, not necessarily total, is Σ-computable if there is a program $\mathcal{P} \in Pro^{\Sigma}$ s.t. $f \sim (n, m, \varphi) = \Psi_{\mathcal{P}}^{n,m,\varphi}$, with $\varphi \in \{\#, *\}$. We say f is computed by $\mathcal{P}$.

Theorem 27 *If f is Σ-computable, then it is Σ-effectively computable.*

The previous theorem should be obvious. Any program in $\mathcal{S}^{\Sigma}$ can be translated into an effective procedure with relative simplicity.

Problem 14 *Let* $\Sigma = \{@, !\}$. *Give a program that computes* $f : \{0, 1, 2\} \mapsto \omega$ *given by* $f(0) = f(1) = 0, f(2) = 5$.

Evidently $f \sim (1, 0, \#)$ and so we must find some $\mathcal{P} \in Pro^{\Sigma}$ s.t. $\Psi^{1,0,\#}_{\mathcal{P}}(x) = f(x)$. The program must let N_1 hold the value 0 if the starting state is either $[[0]]$ or $[[1]]$, and the value 5 if the starting state is $[[2]]$. In all other cases, it must not halt, to ensure that the domain of $\Psi^{1,0,\#}_{\mathcal{P}}$ is the same as that of f. The desired program is

$$
\begin{array}{ll}
 & N_2 \leftarrow N_1 \\
 & N_2 \leftarrow N_2 - 1 \\
 & IF\ N_2 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_4 \\
L_1 & N_2 \leftarrow N_2 - 1 \\
 & IF\ N_2 \neq 0\ GOTO\ L_2 \\
 & GOTO L_3 \\
L_2 & GOTO\ L_2 \\
L_3 & N_1 \leftarrow N_1 + 1 \\
 & N_1 \leftarrow N_1 + 1 \\
 & N_1 \leftarrow N_1 + 1 \\
 & GOTO\ L_5 \\
L_4 & N_1 \leftarrow 0 \\
L_5 & SKIP
\end{array}
$$

If $\mathcal{P}$ denotes this program, it is evident that $\mathcal{P}$ only halts for starting states $[[x_1]]$ with $x_1 \in \{0, 1, 2\}$. Thus, the domain of $\Psi^{1,0,\#}_{\mathcal{P}}$ is precisely $\mathcal{D}_f$. It is easy to verify that, more generally, $\Psi^{1,0,\#}_{\mathcal{P}} = f$.

Problem 15 *Using the same alphabet as in the previous problem, find* $\mathcal{P} \in Pro^{\Sigma}$ *that computes* $\lambda xy[x+y]$.

The desired program is

$$
\begin{array}{ll}
L_1 & IF\ N_2 = 0\ GOTO\ L_3 \\
 & N_1 \leftarrow N_1 + 1 \\
 & N_2 \leftarrow N_2 - 1 \\
 & GOTO\ L_1 \\
L_3 & SKIP
\end{array}
$$

Problem 16 *Same for* $C_0^{1,1}{}_{|\{0,1\}\times\Sigma^*}$

Since the domain of the constant function is restricted t o $\{0,1\}\times\Sigma^*$, we must ensure the program only halts for states $[[x_1, x_2, \alpha]]$ s.t. $x_1, x_2 \in \{0,1\}$. Thus, the program is

$$
\begin{aligned}
& N_1 \leftarrow N_1 - 1 \\
& N_2 \leftarrow N_2 - 1 \\
& IF N_2 \neq 0\ GOTO\ L_1 \\
& IF N_1 \neq 0\ GOTO\ L_1 \\
& GOTO\ L_2 \\
L_1\ & GOTO\ L_1 \\
L_2\ & SKIP
\end{aligned}
$$

Problem 17 *Same for* $\lambda i\alpha[[\alpha]_i]$ *(same alphabet).*

$$
\begin{aligned}
& IF\ N_0 \neq 0\ GOTO\ L_1 \\
& P_1 \leftarrow \varepsilon \\
& GOTO\ L_{100} \\
L_1\ & N_1 \leftarrow N_1 - 1 \\
L_2\ & N_1 \leftarrow N_1 - 1 \\
& P_1 \leftarrow {}^\frown P_1 \\
& IF\ N_1 \neq 0\ GOTO\ L_2 \\
& IF\ P_1\ STARTSWITH\ @\ GOTO\ L_2 \\
& IF\ P_1\ STARTSWITH\ !\ GOTOL_3 \\
& GOTOL_{100} \\
L_3\ & P_1 \leftarrow\ ! \\
L_2\ & P_1 \leftarrow @ \\
L_{100}\ & SKIP
\end{aligned}
$$

Example. Let $\alpha = @!!@@$. Assume we give $[[4, \alpha]]$. Since $4 \neq 0$ we go to L_1 immediately. Here N_1 is set to three. Then N_1 is set to two and P_1 is set to $!!@@$. Since $N_1 \neq 0$, N_1 is now set to 1 and P_1 to $!@@$. Once more, N_1 is now set to 0 and P_1 to $@@$. Since now $N_1 = 0$, we know the starting character of P_1 is the one we looked for. We set P_1 to be its first character (if $P_1 = \varepsilon$ it has no first character and nothings needs

to be done, because this means the input $[[x_1, \alpha]]$ had $x_1 > |\alpha|$). The other cases also work.

Problem 18 *Give a program that computes $s^{\leq}$ where @ <!.*

Recall that $s^{\leq} : \Sigma^* \mapsto \Sigma^*$ is defined as

$$\begin{aligned} s^{\leq}((a_n)^m) &= (a_1)^{m+1} && m \geq 0 \\ s^{\leq}(\alpha a_i (a_n)^m) &= \alpha a_{i+1}(a_1)^m && 1 \leq i < n, m \geq 0 \end{aligned}$$

In our case, this functions enumerates the language in question as follows:

ε, @, !, @@, @!, !@, !!, @@@, @@!, @!@, @!!, !@@, !@!, !!@, !!!, ...

5.9 Macros

A macro is the template of a program that computes a Σ-mixed function. There are two types:

- Those that assign that simulate setting the value of a variable to a function of others;
- Those that use IF statements that direct a program to a label if a predicate function of other variables is true.

A macro is not a program because it does not necessarily hold to **GOTO law**. The formal definition of a macro is hand-wavy and long; check the source. The variables of a macro that are only used within the macro are the *auxiliary variables*. The variables the receive the input (from within some program) are the *official variables*.

Theorem 28 *Let Σ a finite alphabet. Then if f a Σ-computable function, there is a macro $\left[Z\overline{n+1} \leftarrow f\,(V_1, \ldots, V\overline{n}, W_1, \ldots, W\overline{m})\right]$ with $Z \in \{V, W\}$ depending on the value of f.*

Example. The function $\mathcal{F} = \lambda xy[x + y]$ is Σ-computable. Then there is a macro that computes it. Such macro is:

$$
\begin{array}{ll}
 & V_4 \leftarrow V_2 \\
 & V_5 \leftarrow V_3 \\
 & V_1 \leftarrow V_4 \\
A_1 & IF\ V_5 \neq 0\ GOTO\ A_2 \\
 & GOTO\ A_3 \\
A_2 & V_5 \leftarrow V_5 - 1 \\
 & V_1 \leftarrow V_1 + 1 \\
 & GOTO\ A_1 \\
A_3 & SKIP
\end{array}
$$

We replace V_1 with that variable where the output is to be stored, V_2, V_3 with the variables the are to be summed, and this performs the sum of two variables. Now, to program $\lambda xy[x \cdot y]$ we can use the following:

$$
\begin{array}{ll}
L_1 & IF\ N_2 \neq 0\ GOTO\ L_2 \\
 & GOTO\ L_3 \\
L_2 & [N_3 \leftarrow \mathcal{F}(N_3, N_1)] \\
 & N_2 \leftarrow N_2 - 1 \\
 & GOTO\ L_1 \\
L_3 & N_1 \leftarrow N_3
\end{array}
$$

Problem 19 *Let* $\Sigma = \{@, !\}$ *and* $f \sim (0, 1, \#)$ *a* Σ*-computable function. Let* $L = \left\{\alpha \in \mathcal{D}_f : f(\alpha) = 1\right\}$. *Using the macro* $[V_1 \leftarrow f(W_1)]$*, give a program* $\mathcal{P} \in Pro^{\Sigma}$ *s.t.* $\mathcal{D}_{\Psi^{0,1,\#}_{\mathcal{P}}} = L$.

$\mathcal{D}_{\Psi^{0,1,\#}_{\mathcal{P}}} = L$ if and only if $\mathcal{P}$ halts only when starting from a state $[[\alpha \in L]]$ Such $\mathcal{P}$ may be

$$
\begin{array}{ll}
 & [N_1 \leftarrow f(P_1)] \\
 & IF\ N_1 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_2 \\
L_1 & GOTO\ L_1 \\
L_2 & SKIP
\end{array}
$$

Incidentally, it is easy to observe that $\Psi^{0,1,\#}_{\mathcal{P}} = f_{|L}$.

Problem 20 *Let* $\Sigma = \{@, !\}$ *and* $f \sim (1, 0, *)$ *a* Σ*-computable function. Using* $[W_1 \leftarrow f(V_1)]$*, give a program* $\mathcal{P} \in Pro^{\Sigma}$ *s.t.* $\mathcal{D}_{\Psi^{1,0,*}_{\mathcal{P}}} = Im_f$.

We require a program $\mathcal{P} \in Pro^{\Sigma}$ s.t. $\mathcal{P}$ halts only from a starting state of the form $[[\alpha \in Im_f]]$. Such a program may be

$$
\begin{array}{ll}
L_1 & [P_2 \leftarrow f(N_1)] \\
 & [IF\ P_1 = P_2\ GOTO\ L_2] \\
 & N_1 \leftarrow N_1 + 1 \\
 & GOTO\ L_1 \\
L_2 & Skip
\end{array}
$$

where $[IF\ W_1 = W_2\ GOTO\ A_1]$ is the macro

$$
\begin{array}{ll}
 & W_3 \leftarrow W_1 \\
 & W_4 \leftarrow W_2 \\
A_1 & IF\ W_3 BEGINS\ @\ GOTO\ A_2 \\
 & IF\ W_3 BEGINS\ !\ GOTO\ A_3 \\
 & A_2 \qquad\qquad\qquad\qquad\qquad IF\ W_4\ BEGINS\ @\ GOTO\ A_4 \\
 & GOTO\ A_{1000} \\
A_3 & IF\ W_4\ BEGINS\ !\ GOTO\ A_4 \\
A_4 & W_3 \leftarrow {}^{\frown}W_3 \\
 & W_4 \leftarrow {}^{\frown}W_4 \\
 & GOTO A_5 \\
A_{1000} & SKIP
\end{array}
$$

that checks if two *not-empty* strings are equal and jumps to the official label A_5 if the case is true.

5.10 Enumerable sets

A non-empty Σ-mixed set S is Σ-enumerable if and only if there are programs $\mathcal{P}_1, \ldots, \mathcal{P}_{n+m}$ s.t.

$$
\begin{array}{l}
\mathcal{D}_{\Psi^{n,m,\#}_{\mathcal{P}_1}} = \ldots = \mathcal{D}_{\Psi^{n,m,\#}_{\mathcal{P}_n}} = \omega \\
\mathcal{D}_{\Psi^{n,m,*}_{\mathcal{P}_{n+1}}} = \ldots = \mathcal{D}_{\Psi^{n,m,*}_{\mathcal{P}_{n+m}}} = \omega
\end{array}
$$

and

$$S = Im\ \left[\Psi^{n,m,\#}_{\mathcal{P}_1}, \ldots, \Psi^{n,m,\#}_{\mathcal{P}_n}, \Psi^{n,m,*}_{\mathcal{P}_{n+1}}, \ldots, \Psi^{n,m,*}_{\mathcal{P}_{n+m}}\right]$$

In other words, for each input $x \in \omega$, the ith program $\mathcal{P}_i$ computes the value of the ith element in a tuple of S. Another way to put this is

Theorem 29 *If S a non-empty Σ-mixed set, then it is equivalent to say:*
(1) S is Σ-enumerable.
(2) There is a $\mathcal{P} \in Pro^{\Sigma}$ satisfying the following two properties. a. For all $x \in \omega$, $\mathcal{P}$ halts from $[[x]]$ into a state of the form $[[x_1, \ldots, x_n, \alpha_1, \ldots,]]$ when $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_n) \in S$. b. For any tuple $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) \in S$, there is a $x \in \omega$ s.t. $\mathcal{P}$ halts starting from $[[x]]$ in a state of the form $[[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]$

When a program satisfies these properties, we say it *enumerates* S.

5.11 Σ-computable sets

A Σ-mixed set S is said to be Σ-computable if $\chi_S^{\omega^n \times \Sigma^{*m}}$ is Σ-computable. This is, S is Σ-computable if and only if there is a $\mathcal{P} \in Pro^{\Sigma}$ s.t. $\mathcal{P}$ commputes $\chi_S^{\omega^n \times \Sigma^{*m}}$.
Observe that this means that $\mathcal{P}$ halts with $N_1 = 1$ when starting from $[[\overrightarrow{x}, \overrightarrow{\alpha}]]$ if $(\overrightarrow{x}, \overrightarrow{\alpha}) \in S$, and halts with $N_1 = 0$ otherwise. We say $\mathcal{P}$ *decides* the belonging to S.
Observe that if $\chi_S^{\omega^n \times \Sigma^{*m}}$ is Σ-computable, then there is a macro

$$\left[IF\ \chi_S^{\omega^n \times \Sigma^{*m}}\ (V_1 \ldots, V\overline{n}, W_1, \ldots, W\overline{m})\ GOTO\ A_1\right]$$

We will write this macro as $[IF\ (V_1, \ldots V\overline{n}, W_1, \ldots, W\overline{m}) \in S\ GOTO\ A_1]$. Of course, this macro is only valid when S is a Σ-computable set.

Theorem 30 *In Godel's paradigm, S is* Σ*-p.r. iff it is the domain of a* Σ*-p.r. function. This statement does not hold in von Neumman's paradigm. There are sets that are domains of* Σ*-computable functions that are not* Σ*-computable themselves.*

6 Paradigm battles

6.1 Neumann triumphs over Godel

Theorem 31 *If h is Σ-recursive then it is Σ-computable.*

A corollary is that every Σ-recursive function has a corresponding macro.

Theorem 32 *If $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ are Σ-enumerable, then $S_1 \cup S_2$ is Σ-enumerable.*

> **Proof**. Via assumption there are $F : \omega \mapsto \omega^n \times \Sigma^{*m}$, $G : \omega \mapsto \omega^n \times \Sigma^{*m}$ s.t. $F_{(i)}, G_{(i)}$ are Σ-computable for every i and $Im_F = S_1$, $Im_G = S_2$. $\therefore F_{(i)}, G_{(i)}$ have associated macros.
> λx [x is even] is Σ-p.r. $\therefore$ it is Σ-computable. $\therefore$ λx [x is even] has an associated macro. $\therefore$ There is an IF macro which checks if a variable holds an even number.
> λx $[x/2]$ is Σ-p.r. $\therefore$ it is Σ-p.r. $\therefore$ it has an associated macro.
> Using the macros above, one can write a program $\mathcal{P} \in Pro^{\Sigma}$ that does the following:
>
> > If the input N_1 is even, set N_1 to its integer quotient by two and then let $N_1 = F_1(N_1), \ldots, N_n = F_n(N_n)$.
> > If the input N_1 is odd, set N_1 to its integer quotient by two minus one and then let $N_1 = G_1(N_1), \ldots, N_n = G_n(N_n)$.
>
> It is easy to see that this satisfies the theorem of the **Enumerable sets** section of the von Neumann paradigm.

Theorem 33 *If $S \subseteq \omega^n \times \Sigma^{*m}$ is Σ-computable, then S is enumerable.*

> **Proof.** Assume $S \subseteq \omega^n \times \Sigma^{*m}$ is Σ-computable. $\therefore \underset{S}{}^{n,m}$ is Σ-recursive and has an associated macro.
> Recall that $pr : \mathbb{N} \mapsto \omega$ is Σ-p.r. $\therefore$ pr is Σ-recursive and has an associated macro. The same is true of $\leq^*$, since its definition involves only string concatenations and string exponentiations (which are Σ-p.r.) on two mutually exclusive cases.
> Let

$$\mathcal{F} : [1,m] \times \mathbb{N} \mapsto \omega \cup \Sigma^*$$
$$(x,i) \mapsto \begin{cases} (x)_i & 1 \leq i \leq n \\ *^{\leq}((x)_i) & n+1 \leq i \leq m \end{cases}$$

The program $\mathcal{P}_i$ with input x, i defined as

$$\begin{array}{rl} & [N_2 \leftarrow p(N_2)] \\ L_1 & [IF \neg (N_2 \mid N_1)\ GOTO\ L_2] \\ & N_3 \leftarrow N_3 + 1 \\ & [N_2 \leftarrow N_2 \times N_2] \\ & GOTO\ L_1 \\ L_2 & [IF 1\ \leq N_1 \leq m\ GOTO\ L_3] \\ & [N_1 \leftarrow *^{\leq}(N_3)] \\ & GOTO\ L_{100} \\ L_3 & N_1 \leftarrow N_3 \\ L_{100} & SKIP \end{array}$$

when $n+1 \leq i \leq m$ computes $\mathcal{F}$. $\therefore$ There is a macro for $\mathcal{F}$.
Let $u = n + m + 1$ and $\mathcal{P}$ the following program:

$$\begin{array}{rl} & N\overline{u} \leftarrow N_1 \\ & [N_1 \leftarrow \mathcal{F}(N\overline{u}, 1)] \\ & [N_2 \leftarrow \mathcal{F}(N\overline{u}, 2)] \\ & \vdots \\ & [P_{\overline{m}} \leftarrow \mathcal{F}(N\overline{u}, n+m)] \\ & [IF\ \chi_s^{n,m}\ (N_1, \ldots, N_{\overline{n}}, P_{\overline{m}},)\ \ GOTO\ L_{100}] \\ L_0 & GOTO\ L_0 \\ L_{100} & SKIP \end{array}$$

a. For any $x \in \omega$, $\mathcal{P}$ halts at a state with instantaneous description $[[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]$ with $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_n) \in S$; *b.* for any $(\vec{x}, \vec{\alpha}) \in S$ there is some $x \in \omega$ s.t. $\mathcal{P}$ halts at $[[(\vec{x}, \vec{\alpha})]]$ when starting from x. $\therefore$ $\mathcal{P}$ enumerates S.

Problem 21 *Prove that, if $S \subseteq \Sigma^*$ is Σ-enumerable, then*

$$T = \{\alpha \in \Sigma^* : \exists \beta \in S : \alpha \textit{ is subchain of } \beta\}$$

is Σ*-enumerable.*

Let $\mathcal{F} : \omega \mapsto \Sigma^*$ denote the function which enumerates S, which is Σ-computable and has a macro. The key to the problem is to observe that, per each $\alpha \in S$, there are $|\alpha| - 1$ subchains of α in T. Thus, for $\mathcal{F}(i)$ we want to enumerate $|\mathcal{F}(i)| - 1$ subchains.

I provide a program that accomplishes this and then illustrate its operation on a finite alphabet. It is easy to verify that the program enumerates T and that this enumeration holds for infinite alphabets as well.

Let $\mathcal{P}$ with input $x \in \omega$ be

$$\begin{array}{rl}
 & [P_1 \leftarrow \mathcal{F}(N_3)] \\
 & [N_{50} \leftarrow |\mathcal{F}(N_3)|] \\
L_1 & [IF\ N_1 \geq N_{50}\ GOTO\ L_2] \\
 & N_{10} \leftarrow N_{50} - N_1 \\
 & N_{10} \leftarrow N_{10} - 1 \\
L_{11} & [P_1 \leftarrow {}^{\frown}P_1] \\
 & N_{10} \leftarrow N_{10} - 1 \\
 & [IF\ N_{10} = 0\ GOTO\ L_{100}] \\
 & GOTO\ L_{11} \\
L_2 & N_3 \leftarrow N_3 + 1 \\
 & [P_1 \leftarrow \mathcal{F}(N_3)] \\
 & [N_{50} \leftarrow N_{50} + |P_1|] \\
 & GOTO\ L_1 \\
L_{100} & SKIP
\end{array}$$

*Example.*Let $S = \{abab, bba, bbba, abb\}$. Then the program can be illustrated as follows:

$$\begin{array}{l}
 \overbrace{}^{N_{50} \leq N_1} \\
x = 0 \mid 0 \leq 4 \mid N_{10} = 3 \mapsto a \\
x = 1 \mid 1 \leq 4 \mid N_{10} = 2 \mapsto ab \\
x = 2 \mid 2 \leq 4 \mid N_{10} = 1 \mapsto aba \\
x = 3 \mid 3 \leq 4 \mid N_{10} = 0 \mapsto aba
\end{array}$$

When $x = 4$, we have $N_{50} \geq N_1$ and so P_1 becomes bba, N_{50} becomes $4 + 3 = 7$ and

$$\begin{aligned} x = 4 \mid \overbrace{4 \leq 7}^{N_{50} \leq N_1} \mid N_{10} = 2 \mapsto b \\ x = 5 \mid 5 \leq 7 \mid N_{10} = 1 \mapsto bb \\ x = 6 \mid 6 \leq 7 \mid N_{10} = 0 \mapsto bb \end{aligned}$$

When $x = 7$, the L_2 case is met twice, and so $P_1 = bbba$, $N_{50} = 4 + 3 + 4 = 11$ and

$$\begin{aligned} x = 7 \mid \overbrace{7 \leq 11}^{N_{50} \leq N_1} \mid N_{10} = 3 \mapsto b \\ x = 8 \mid 8 \leq 11 \mid N_{10} = 2 \mapsto bb \\ x = 9 \mid 9 \leq 11 \mid N_{10} = 1 \mapsto bbb \\ x = 10 \mid 10 \leq 11 \mid N_{10} = 0 \mapsto bbb \end{aligned}$$

etc. Thus, the program enumerates S and, per each $\alpha \in S$, it enumerates the substrings which make up α. The program is s.t.

$$\left.\begin{array}{l} 0 \mapsto \textit{The smallest substring of } \mathcal{F}(0) \\ 1 \mapsto \textit{The next smallest substring of } \mathcal{F}(0) \\ 2 \mapsto \textit{The next smallest substring of } \mathcal{F}(0) \\ 3 \mapsto \textit{The next smallest of } \mathcal{F}(0) \end{array}\right\} 0 \leq x \leq |\mathcal{F}(0)|$$

$$\left.\begin{array}{l} 4 \mapsto \textit{The smallest substring of } \mathcal{F}(1) \\ 5 \mapsto \textit{The next smallest } \mathcal{F}(1) \\ 6 \mapsto \textit{The next smallest substring of } \mathcal{F}(1) \end{array}\right\} |\mathcal{F}(0)| \leq x \leq |\mathcal{F}(0)\mathcal{F}(1)|$$

$$\vdots$$

6.2 Godel triumphs over Neumann

The syntax of $\mathcal{S}^{\Sigma}$ is $(\Sigma \cup \Sigma_p)$-recursive. Recall that $S : Num^* \mapsto Num!*$ was the (symbolic) successor satisfying $S(\overline{n}) = \overline{n+1}$. In particular, $S(\varepsilon) = 1$, $S(\alpha 0) = \alpha 1$, $S(\alpha 2) = \alpha 3, \ldots, S(\alpha 9) = S(\alpha)0$. It is obvious that the function is Num-p.r. The same can be shown of - : $\omega \mapsto Num^*$.

Theorem 34 *Let Σ an arbitrary alphabet. Then S and - are $(\Sigma \cup \Sigma_p)$-p.r.*

Recall that Ins^{Σ} is the set of instructions in $\mathcal{S}^{\Sigma}$. It can be proven that this set is the union of a series of $(\Sigma \cup \Sigma_p)$-p.r. sets.

Theorem 35 *Let Σ an arbitrary alphabet. Then Ins^{Σ} is $(\Sigma \cup \Sigma_p)$-p.r.*

The following three functions contain all the information relevant to $\mathcal{S}^{\Sigma}$. Assuming $n, m \in \omega$ are fixed,

$$\begin{aligned} i^{n,m} &\omega \times \omega^n \times \Sigma^{*m} x Pro^{\Sigma} \mapsto \omega \\ E^{n,m}_{\#} &\omega \times \omega^n \times \Sigma^{*m} x Pro^{\Sigma} \mapsto \omega^{[\mathbb{N}]} \\ E^{n,m}_{*} &\omega \times \omega^n \times \Sigma^{*m} x Pro^{\Sigma} \mapsto \Sigma^{*[\mathbb{N}]} \end{aligned}$$

For brevity, let $f^{n,m}(t, \vec{x}, \vec{a}, \mathcal{P}) = f(t)$ for $f \in \left\{i^{n,m}, E^{n,m}_{\#}, E^{n,m}_{*}\right\}$— this is, let as assume fixed $\vec{x}, \vec{\alpha}, \mathcal{P}$. Then

$$\begin{aligned} \left(i^{n,m}(0), E^{n,m}_{\#}(0), E^{n,m}_{*}(0)\right) &= \left(1, (x_1, \ldots, x_n, 0, \ldots), (\alpha_1, \ldots, \alpha_n, 0, \ldots)\right) \\ \left(i^{n,m}(t+1), E^{n,m}_{\#}(t+1), E^{n,m}_{*}(t+1)\right) &= S_{\mathcal{P}}\left(i^{n,m}(t), E^{n,m}_{\#}(t), E^{n,m}_{*}(t)\right) \end{aligned}$$

It is clear that $\left(i^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}), E^{n,m}_{\#}(t, \vec{x}, \vec{\alpha}, \mathcal{P}), E^{n,m}_{*}(t, \vec{x}, \vec{\alpha}, \mathcal{P})\right)$ is the instantaneous description of $\mathcal{P}$ after t steps starting from $[[\vec{x}, \vec{\alpha}]]$.
The E functions are not $(\Sigma \cup \Sigma_p)$-mixed, but if we define $E^{n,m}_{\varphi j}$, with $\varphi \in \{*, \#\}$, as the jth coordinate of $E^{n,m}_{\varphi}$, we find that $E^{n,m}_{\varphi j}$ is $(\Sigma \cup \Sigma_p)$-mixed.

Theorem 36 *The functions $i^{n,m}, E^{n,m}_{\varphi j}$ are $(\Sigma \cup \Sigma_p)$-mixed functions.*

Given $n, m \in \omega$, we define

$$Halt^{n,m} = \lambda t \vec{x} \vec{\alpha} \mathcal{P}\left[i^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}) = n(\mathcal{P}) + 1\right]$$

Observe that $\mathcal{D}_{Halt} = \omega \times \omega^n \times \Sigma^{*m} \times Pro^{\Sigma}$.

Theorem 37 *Pro^{Σ} is a $(\Sigma \cup \Sigma_p)$-p.r. set and $n(\mathcal{P}), I_j^{\mathcal{P}}$ are $(\Sigma \cup \Sigma_p)$-p.r. functions.*

Theorem 38 *The Halt function is Σ-p.r.*

Proof. Observe that

$$Halt^{n,m} = \lambda xy\,[x = y] \circ \left[i^{n,m}, \lambda \mathcal{P}\,[n(\mathcal{P})] \circ p^{1+n,m+1}_{n+m+2}\right]$$

We define $T^{n,m} = M(Halt^{n,m})$. Observe that

$$\mathcal{D}_T^{n,m} = \{(\vec{x}, \vec{\alpha}, \mathcal{P}) \in \omega^n \times \Sigma^{*m} : \mathcal{P} \text{ halts when starting from } [[(\vec{x}, \vec{\alpha})]]\}$$

Theorem 39 *$T^{n,m}$ is $(\Sigma \cup \Sigma_p)$-recursive but it is not $(\Sigma \cup \Sigma_p)$-p.r.*

The previous theorem follows from the fact that $T^{n,m}$ is a minimization over a Σ-p.r. predicate.

Theorem 40 *If f is a Σ-computable Σ-mixed function, then f is Σ-recursive.*

The previous theorem states that any function computable in von Neumman's paradigm is computable in Godel's paradigm.

Proof. The proof consists in showing that f is $(\Sigma \cup \Sigma_p)$-recursive. Then, due to the alphabet independence of recursion, it is Σ-recursive.

We define $\Phi^{n,m}_{\#} := \lambda \vec{x}\vec{\alpha}\mathcal{P}\left[\Psi^{n,m,\#}_{\mathcal{P}}(\vec{x}, \vec{\alpha})\right]$. The domain of the function is consists thus in all $(\vec{x}, \vec{\alpha}, \mathcal{P})$ s.t. $(\vec{x}, \vec{\alpha}) \in \mathcal{D}_{\Psi^{n,m,\#}_{\mathcal{P}}}$. We can think of this as the *execution function*, which takes some input and a program and returns the value of that program when executed on that input.
We similarly define $\Phi^{n,m,*}_{\mathcal{P}}$.

Theorem 41 *The functions $\Phi^{n,m,\#}_{\mathcal{P}}, \Phi^{n,m,*}_{\mathcal{P}}$ are $(\Sigma \cup \Sigma_p)$-recursive.*

Proof. Observe that $\mathcal{D}_{T^{n,m}} = \mathcal{D} = \Phi^{n,m,\#}_{\mathcal{P}}$. For any $(\vec{x}, \vec{\alpha}, \mathcal{P}) \in \mathcal{D}_{T^{n,m}}$ we have

$$\Phi^{n,m,\#}_{\mathcal{P}} = E^{n,m}_{\#!}\left(T^{n,m}(\vec{x},\vec{\alpha},\mathcal{P}),\vec{x},\vec{\alpha},\mathcal{P}\right)$$

which implies $\Phi^{n,m,\#}_{\mathcal{P}} = E^{n,m}_{\#1} \circ \left[T^{n,m}, p^{n,m+1}_1, \ldots, p^{n,m+1}_{n+m+1}\right]$.
$\therefore \Phi^{n,m,\#}_{\mathcal{P}}$ is $(\Sigma \cup \Sigma_p)$-p.r.

An important consequence of this theorem is that any Σ-computable function is Σ-recursive. From this follows an interesting theorem.

Theorem 42 *If $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$, with $\varphi = \omega$ or $\varphi = \Sigma^*$, then there is a Σ-p.r. predicate $P : \mathbb{N} \times \omega^n \times \Sigma^{*m} \mapsto \omega$ and a Σ-p.r. function $g : \mathbb{N} \mapsto \varphi$ s.t. $f = g \circ M(p)$.*

> Informally, the theorem establishes that any recursive function is some transformation g of the minimal natural number which satisfies some predicate P. The proof is straightforward. Let $\mathcal{P}_0$ a program which computes f and $\leq$ an order over Σ. Then

$$P := \lambda t \vec{x} \vec{\alpha} \left[Halt^{n,m}\left((t)_1,\vec{x},\vec{\alpha},\mathcal{P}_0\right) \wedge (t)_2 = \#^{\leq}\left(E^{n,m}_{*1}\left((t)_1,\vec{x},\vec{\alpha},\mathcal{P}_0\right)\right)\right]$$

> Now let $g := \lambda t \left[*^{\leq}\left((t)_2\right)\right]$
> Both P and g are Σ-p.r. and evidently $f = g \circ M(P)$.

6.3 Interacting programs

Using the results of the previous section, we can construct programs that depend on other programs. This is illustrated with an example. Say $\Sigma\{@,+\}$ and $\mathcal{P}_0 \in Pro^{\Sigma}$ is s.t. $0 \in \mathcal{D}_{\Psi^{1,0,\#}_{\mathcal{P}_0}}$ with $\Psi^{1,0,\#}_{\mathcal{P}_0}(0) = 2$. We will show

$$S = \left\{x \in \mathcal{D}_{\Psi^{1,0,\#}_{\mathcal{P}_0}} : \Psi^{1,0,\#}_{\mathcal{P}_0}(x) \neq 0\right\}$$

is Σ-enumerable.

The program will do the following. For each input x, it will test whether $\mathcal{P}_0$ halts when starting from $[[(x)_1]]$ in $(x)_2$ steps. If it doesn't it will simply return zero (since $0 \in S$). If it does, it will compute $\mathcal{P}_0$ starting from $[[(x)_1]]$; if the

output is not zero, it will return it. If it is zero, it will return 0.

Of course, the program will make use of the $Halt^{1,0}$ function, which is Σ-recursive and thus has an associated macro. It will also use $E_j^{1,0}$ to retrieve the output of $\mathscr{P}_0$ after $(x)_2$ steps. This function is also Σ-p.r. and thus it also has an associated macro. The program is

$$
\begin{array}{ll}
 & IF\ N_1 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_2 \\
L_1 & [N_3 \leftarrow (N_1)_1] \\
 & [N_4 \leftarrow (N_1)_2] \\
 & [IF\ Halt^{1,0}(N_4, N_3, \mathscr{P}_0)\ GOTO\ L_3] \\
 & GOTO\ L_2 \\
L_3 & \left[N_5 \leftarrow E_1^{1,0}(N_4, N_3, \mathscr{P}_0)\right] \\
 & [IF\ N_5 = 0\ GOTO\ L_2] \\
 & N_1 \leftarrow N_5 \\
 & GOTO\ L_4 \\
L_2 & N_1 \leftarrow 0 \\
L_4 & SKIP
\end{array}
$$

Thus, a program may use another program to operate.

Theorem 43 *If S a Σ-mixed set then these statements are equivalent:*

(1) S is Σ-enumerable

(2) $S = I_F$ for $F : \mathscr{D}_F \subseteq \omega^n \times \Sigma^{*m} \mapsto S$ s.t. $F_{(i)}$ is Σ-computable for each $i = 1, \ldots, n+m$.

(3) S is the domain of a Σ-computable function.

Proof. (1) $\Rightarrow$ (2) Assume S is Σ-enumerable. Then there are programs $\mathscr{P}_1, \ldots, \mathscr{P}_{n+m}$ s.t. S = Image of $\left[\Psi^{n,m,\#}_{\mathscr{P}_1}, \ldots, \Psi^{n,m,*}_{\mathscr{P}_{n+m}}\right]$. Let $f = \left[\Psi^{n,m,\#}_{\mathscr{P}_1}, \ldots, \Psi^{n,m,*}_{\mathscr{P}_{n+m}}\right]$. ■

(2) $\Rightarrow$ (3) Assume $S = I_F$ with F a function satisfying the specifications of the theorem. Let $\mathscr{P}_i \in Pro^{\Sigma}$ denote the program that computes $F_{(i)}$. It is possible to construct a program that, from an initial state $[[(\vec{x}, \vec{\alpha})]]$, does the following:

(0) Let $N_{n+1} \leftarrow 1$.
(1) Evaluate whether each $\mathcal{P}_i$ with initial state

$$[[(x_{n+1})_1, \ldots, (x_{n+1})_n, (*^{\leq}(\alpha_{m+1}))_1, \ldots, (*^{\leq}(\alpha_{m+1}))_m]]$$

halts in $t = (x_{n+1})_{n+1}$ steps.
(2) If the programs halt, proceed to step *(3)*. Else let $N_1 \leftarrow N_1 + 1$ and return to *(1)*.
(3) Let N_{n+i} store the output of $\mathcal{P}_i$ for $1 \leq i \leq n$, and P_{m+i} store the output of $\mathcal{P}_i$ for $n + 1 \leq i \leq m$.
(4) If $N_{n+i} = N_i \wedge P_{m+i} = P_i$ for all $1ß \leq n+m$ return 1. Else loop forever.

The reader may implement this program in the $\mathcal{S}^{\Sigma}$ language using the $Halt$ function. The program attempts to (and must eventually succeed at) generating an element of S using the $F_{(1)}, \ldots, F_{(n+m)}$ functions. It then compares whether the values of the input state are equal to the generated element of S. If this is true, then it outputs 1. Otherwise it loops forever. It is evident that the program computes $\left(\lambda \vec{x} \vec{\alpha}\, [1]\right)_{|S}$. ■
(3) $\Rightarrow$ (1) Assume S is the domain of a Σ-computable function f. One can construct a program that does the following: Given a fixed $w \in S$ and a starting state $[[x]]$,

(0) If $N_1 = 0$ set $N_1 \leftarrow w$ and finish.
(1) Check if $\mathcal{P}_f$ halts starting from

$$[[(N_1)_1, \ldots, (N_1)_n, *^{\leq}((N_1)_{n+1}), \ldots, *^{\leq}((N_1)_{n+m})]]$$

in $(N_1)_{n+m+1}$ steps. If it does not, set $N_1 \leftarrow N_1 + 1$ and go to *(1)*; else continue.
(2) Set $N_1, \ldots, N_n, P_1, \ldots, P_m$ so that the final state is

$$[[(N_1)_1, \ldots, (N_1)_n, *^{\leq}((N_1)_{n+1}), \ldots, *^{\leq}((N_1)_{n+m})]]$$

and finish.

This program is easy to implement in $\mathcal{S}^{\Sigma}$. It is evident that it enumerates $\mathcal{D}_f = S$.

Problem 22 *Let* $\Sigma = \{@, +\}$ *and* $f : \mathcal{D}_f \subseteq \Sigma^* \mapsto \omega$ *a* Σ*-computable function s.t.* $f(\varepsilon) = 1$. *Let*

$$L = \left\{\alpha \in D_f : f(\alpha) = 1\right\}$$

Provide a program $\mathbb{Q} \in Pro^{\Sigma}$ *that enumerates L.*
Let $\mathcal{P} \in Pro^{\Sigma}$ be the program that computes f. Our program $\mathbb{Q}$, given a starting state $[[x]]$, may operate as follows:

> *(0)* Check if the input is zero; if it is return ε. (This is important because we will generate strings with the $*^{\leq}$ function, which never maps to ε—without this condition our program would never enumerate ε!) *(1)* Compute $\beta := *^{\leq}((x)_1), (x)_2$.
> *(2)* If $\mathcal{P}$ halts in $(x)_2$ steps starting from β, compute it and go to next step; else return ε (since $\varepsilon \in L$).
> *(3)* If the output of $\mathcal{P}$ was 1 return β; else return ε.

$$\begin{array}{rl}
 & [IF\ N_1 = 0\ GOTO\ L_2] \\
 & \left[P_1 \leftarrow *^{\leq}((N_1)_1)\right] \\
 & [N_1 \leftarrow (N_1)_2] \\
 & [IF\ Halt^{0,1}(N_1, P_1, \mathcal{P})\ GOTO\ L_1] \\
 & GOTO\ L_2 \\
L_1 & \left[N_2 \leftarrow E^{0,1}_{*1}(N_1, P_1, \mathcal{P})\right] \\
 & [IF\ N_2 = 1\ GOTO\ L_{10}] \\
L_2 & P_1 \leftarrow \epsilon \\
L_{10} & SKIP
\end{array}$$

Problem 23 *Let* $\Sigma = \{@, +\}$ *and* $\mathcal{P}_0 \in Pro^{\Sigma}$. *Show that*

$$S = \left\{(x, \alpha) : \Psi^{1,0,\#}_{\mathcal{P}_0}(x) = \Psi^{0,1,\#}_{\mathcal{P}_0}(\alpha)\right\}$$

is Σ*-enumerable.*

The program may operate as follows. Given an initial state $[[x]]$,

(1) Find $(x)_1, (x)_2$.
(2) Evaluate whether $\mathcal{P}$ halts in $(x)_2$ steps from $[[(x)_1]]$. If it does not, let $x = x + 1$ and return to *(1)*. If it does go to *(3)*.
(3) Find $\alpha = *^{\leq}((x)_3), (x)_4$.
(4) Evaluate whether $\mathcal{P}$ halts in $(x)_4$ steps from $[[\alpha]]$. If it doesn't let $x = x + 1$ and to *(1)*; it it does go to *(5)*.
(5) Check whether the computation of $\mathcal{P}$ from $[[(x)_1]]$ is the same as the computation of $\mathcal{P}$ from $[[\alpha]]$. If it is not, let $x = x + 1$ and go to *(1)*, else go to (6).
(6) Set $N_1 \leftarrow (x)_1, P_1 \leftarrow \alpha$.

$$
\begin{array}{ll}
L_0 & [N_{11} \leftarrow (N_1)_1] \\
 & [N_{12} \leftarrow (N_1)_2] \\
 & [IF\ Halt^{1,0}(N_{12}, N_{11}, \mathcal{P})\ GOTO\ L_1] \\
 & N_1 \leftarrow N_1 + 1 \\
 & GOTO\ L_0 \\
L_1 & \left[P_{11} \leftarrow *^{\leq}((N_1)_3)\right] \\
 & [N_{14} \leftarrow (N_1)_4] \\
 & \left[IF\ Halt^{0,1}(N_{14}, P_{11}, \mathcal{P})\ GOTO\ L_2\right] \\
 & N_1 \leftarrow N_1 + 1 \\
 & GOTO\ L_0 \\
L_2 & \left[IF\ E^{1,0}_{\#1}(N_{12}, N_{11}, \mathcal{P}) = E^{0,1}_{\#,1}(N_{14}, P_{11}, \mathcal{P})\ GOTO\ L_3\right] \\
 & N_1 \leftarrow N_1 + 1 \\
 & GOTO\ L_0 \\
L_4 & N_1 \leftarrow N_{11} \\
 & P_1 \leftarrow P_{11}
\end{array}
$$

Problem 24 *If $S \subseteq \omega$ and $f : S \mapsto \omega$ is Σ-computable, then*

$$W = \{x \in S : x \text{ is even } \wedge x/2 \in S \wedge f(x) = f(x/2)\}$$

is Σ-enumerable.

There is some $\mathscr{P}_f \in Pro^{\Sigma}$ that computes f. $\therefore$ S is Σ-enumerable via some $\mathscr{P}_S \in Pro^{\Sigma}$. Let $w \in W$ an arbitrary element.

$$\begin{array}{rl}
 & [IF\ N_1 = 0\ GOTO\ L_{666}] \\
 & [N_2 \leftarrow (N_1)_1] \\
 & [N_3 \leftarrow (N_1)_2] \\
 & [N_4 \leftarrow \mathscr{P}_S(N_2)] \\
 & [N_5 \leftarrow \mathscr{P}_S(N_3)] \\
 & [IF\ N_4 \text{ is odd } GOTO\ L_{666}] \\
 & [IF \neg (N_5 = N_4/2)\ GOTO\ L_{666}] \\
 & [IF\ f(N_5) \neq f(N_4)\ GOTO\ L_{666}] \\
 & N_1 \leftarrow N_2 \\
 & GOTO\ L_1 \\
L_{666} & [N_1 \leftarrow w] \\
L_1 & SKIP
\end{array}$$

The program starting from $[[x]]$ generates two elements of $s_1, s_2 \in S$ using $(x)_1, (x)_2$—unless $x = 0$, where it just outputs w. If s_1 is even, and $s_2 = s_1/2$, and $f(s_1) = f(s_2)$, it outputs s_1. Else it outputs w. It is clear that the program enumerates W.

Problem 25 *Let $\Sigma = \{@, +\}$ and $\mathscr{P}_0 \in Pro^{\Sigma}$. Prove that*

$$W = \{(x, \alpha, \beta) \in \omega \times \Sigma^* \times \Sigma^* : \mathscr{P}_0 \text{ halts from } [[x, \alpha, \beta]]\}$$

is Σ-enumerable.

Observe that $W = \mathscr{D}_f$ where $f = \Psi^{1,2,\#}_{\mathscr{P}_0}$. We have already proven that the domain of a Σ-computable function is Σ-enumerable. Then W is Σ-enumerable.
If the program is still desired, let $(w, \alpha, \beta) \in \mathscr{D}_f$ an arbitrary element of the domain of f. Then

$[IF\ N_1 = 0\ GOTO\ L_{666}]$

$[N_2 \leftarrow (N_1)_1]$

$[P_1 \leftarrow *^{\leq}((N_1)_2)]$

$[P_2 \leftarrow *^{\leq}((N_1)_3)]$

$[N_3 \leftarrow (N_1)_4]$

$[IF\ \neg\left(Halt^{1,2}(N_3, N_2, P_1, P_2, \mathcal{P}_f)\right)\ GOTO\ L_{666}]$

$N_1 \leftarrow N_2$

$GOTO\ L_1$

L_{666} $[N_1 \leftarrow w]$

$[P_1 \leftarrow \alpha]$

$[P_2 \leftarrow \beta]$

L_1 $SKIP$

enumerates W.

Problem 26 *Let* $\Sigma = \{@, +\}$ *and* $\mathcal{P}_0 \in Pro^{\Sigma}$. *Let*

$$L = \left\{\alpha \in \Sigma^* : (\exists x \in \mathbb{N})\ \Psi^{1,1,\#}_{\mathcal{P}_0}(x^2, \alpha) = \Psi^{0,2,\#}_{\mathcal{P}_0}(\alpha, \alpha)\right\}$$

Provide a program $\mathcal{P} \in Pro^{\Sigma}$ *s.t.* $\mathcal{D}_{\Psi^{1,0,*}_{\mathcal{P}}} = \omega$ *and* $\mathcal{J}_{\Psi^{1,0,*}_{\mathcal{P}}} = L$.

Let $f = \Psi^{1,1,\#}_{\mathcal{P}_0}, g = \Psi^{0,2,\#}_{\mathcal{P}_0}$, which have associated macros by virtue of being Σ-computable. L is the set of strings $\alpha \in \Sigma^*$ s.t. some $x \in \mathbb{N}$ satisfies $f(x^2, \alpha) = g(\alpha, \alpha)$. Given an arbitrary $\varphi \in L$ and a starting state $[[x]]$we can construct $\mathcal{P}$ as follows:

(0) If $N_1 = 0$ return φ and finish.
(1) Let $N_1 \leftarrow (N_1)_1, P_1 \leftarrow *^{\leq}((N_1)_2)$.
(2) Let $N_3 \leftarrow (N_1)_3, N_4 \leftarrow (N_1)_4$.
(3) Evaluate whether $\mathcal{P}_0$ halts with input $[[N_1^2, P_1]]$ in N_3 steps. Evaluate whether $\mathcal{P}_0$ halts with input $[[P_1, P_1]]$ in N_4 steps. If both are true continue, else set $P_1 \leftarrow \varphi$ and finish.
(4) Evaluate whether $f(N_1^2, P_1) = g(P_1, P_1)$. If false, set $P_1 \leftarrow \varphi$ and finish. Else continue.
(5) Finish.

Since $\left((x)_1, *^{\leq}((x)_2), (x)_3, (x)_4\right)$ over $x = 1, 2, \ldots$ explores all possible tuples $(x, \alpha, t_1, t_2) \in \omega \times \Sigma^* \times \omega \times \omega$, the program enumerates L.

Since every $\mathscr{P} \in Pro^{\Sigma}$ is a word in $\Sigma \cup \Sigma_p$, we can enumerate sets of programs $\mathscr{P} \subseteq (\Sigma \cup \Sigma_p)^*$. The enumeration of a set of programs is no different to the enumeration of any set of words. For example, say we want to enumerate

$$\mathscr{P} = \left\{ \mathscr{P} \in Pro^{\Sigma} : \Psi^{n,m,\#}_{\mathscr{P}}(10) = 10 \right\}$$

Observe that $SKIP \in (\Sigma \cup \Sigma_p)$ and the program $\mathscr{P} = SKIP \in \mathscr{P}$. Thus, an enumeration starting at $[[x]]$ can proceed as follows:

> *(1)* If $x = 0$ let $P_1 \leftarrow SKIP$ and finish.
> *(2)* Take $\alpha = *^{\leq}((x)_1), \varphi = (x)_2$.
> *(3)* If $\alpha \in Pro^{\Sigma}$ and $\mathscr{P}$ halts at a state $[[10]]$ when starting from $[[10]]$ and in φ steps, let $P_1 \leftarrow \alpha$. Else let $P_1 \leftarrow SKIP$.

6.4 Church's thesis

Theorem 44 *If a Σ-mixed function f is Σ-Turing computable then it is Σ-recursive.*

Theorem 45 *If a Σ-mixed function is Σ-computable then it is Turing-computable.*

In virtue of the aforementioned theorems, we have that

Theorem 46 *The folllowing statements are equivalent:*

- *f is Σ-computable*
- *f is Σ-recursive*
- *f is Turing-computable*

The theorem above extends to set decidability and enumerability. Church's thesis is the following:

> **Church's thesis.** Every Σ-effectively computable function is Σ-recursive.

A formal proof of this thesis has not been given, but there's consensus around that fact that it is most likely true.

7 Extending Σ-recursion

We now extend the Σ-recursive paradigm with results which are easier to prove in the imperative paradigm.

Theorem 47 (Case division) *Assume* $f_i : \mathcal{D}_{f_i} \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$, *with* $\varphi = \omega$ *or* $\varphi = \Sigma^*$ *and* $i = 1, 2, \ldots, k$, *are* Σ*-recursive functions. If* $\mathcal{D}_{f_i} \neq \mathcal{D}_{f_j}$ *for any* $i \neq j, i, j \in [1, k]$, *then* $f_1 \cup \ldots \cup f_k$ *is* Σ*-recursive.*

Proof. Let $\mathcal{P}_1, \ldots, \mathcal{P}_k \in Pro^{\Sigma}$ the programs that compute $f_1, \ldots, f_k$. We define $H_i = \lambda t \vec{x} \vec{\alpha} \, [Halt^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}_i)]$.
Assume $\varphi = \omega$.
Since $Halt^{n,m}$ is $(\Sigma \cup \Sigma_p)$-p.r. it is Σ-p.r. Then it is Σ-computable and has an associated IF macro. Then the (pseudo)program

$$
\begin{array}{rl}
L_0 & :N\overline{n+1} \leftarrow N\overline{n+1} + 1 \\
 & \left[IF\ Halt^{n,m}\left(N\overline{n+1}, N_1, \ldots, N_{\overline{n}}, P_1, \ldots, P_{\overline{m}}, \mathcal{P}_1\right)\ GOTO\ L_2\right] \\
 & \vdots \\
 & \left[IF\ Halt^{n,m}\left(N\overline{n+1}, N_1, \ldots, N_{\overline{n}}, P_1, \ldots, P_{\overline{m}}, \mathcal{P}_k\right)\ GOTO\ L_2\right] \\
L_1 & [N_1 \leftarrow f_1(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
 & GOTO\ L\overline{n(\mathcal{P})} \\
L_2 & [N_1 \leftarrow f_2(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
 & GOTO\ L\overline{n(\mathcal{P})} \\
 & \vdots \\
L\overline{k} & [N_1 \leftarrow f_k(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
L\overline{n(\mathcal{P})} & SKIP
\end{array}
$$

Evidently the (pseudo) program computes $f_1 \cup \ldots \cup f_k$.
If $\varphi = \Sigma^*$ the change that is to be done is trivial. ■

Theorem 48 *Let* $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \varphi$, *with* $\varphi = \omega$ *or* $\varphi = \Sigma^*$, *a* Σ*-recursive function. If* $S \subseteq \mathcal{D}_f$ *is* Σ*-recursively enumerable, then* $f_{|S}$ *is* Σ*-recursive.*

Proof. We will not provide the explicit program but will describe what it should do.

S is Σ-recursively enumerable. $\therefore$ it is Σ-enumerable. $\therefore$ There are $F_{(1)}, \ldots, F_{(n+m)}$ s.t. $S = \mathcal{J}\left[F_{(1)}, \ldots, F_{(n+m)}\right]$ with $F_{(i)}$ Σ-enumerable for each $i \in [1, n+m]$. $\therefore$ There is a macro for each $F_{(i)}$.
A program that from a starting state $[[\vec{x}, \vec{\alpha}]]$

(1) Generates an element of S using the macros of $F_{(1)}, \ldots, F_{(n+m)}$ from input variable $N\overline{n+1}$
(2) If the coordinates of the generated element correspond to $(\vec{x}, \vec{\alpha})$, compute $f(\vec{x}, \vec{\alpha})$ and return this computation. Else continue.
(3) Let $N\overline{n+1} \leftarrow N\overline{n+1} + 1$ and go back to *(1)*

Evidently, this program computes $f_{|S}$.

Theorem 49 *Let S_1, S_2 be Σ-recursive. Prove that $S_1 \cap S_2, S_1 \cup S_2, S_1 - S_2$ are Σ-recursive.*

Proof. Complete.

Theorem 50 *Let $S \subseteq \omega^n \times \Sigma^{*m}$. If S and $\omega^n \times \Sigma^{*m} - S$ are Σ-recursively enumerable, then S is Σ-recursive.*

Corollary. If S is Σ-recursively enumerable, then it isn't necessarily Σ-recursive.

Proof. $\chi_S^{\omega^n \times \Sigma^{*m}} = C_1^{n,m}{}_{|S} \cup C_0^{n,m}{}_{|\omega^n \times \Sigma^{*m} - S}$ is Σ-recursive and by hypothesis $S, \omega^n \times \Sigma^{*m} - S$ are Σ-recursively enumerable. $\therefore$ S is Σ-recursive.

Problem 27 *Provide a proof of the following theorem in the imperative paradigm.*

Theorem 51 *Let $S \subseteq \omega^n \times \Sigma^{*m}$. The following statements are equivalent.*

(1) S is Σ-recursively enumerable.

(2) S is the domain of a Σ-recursive function f.

(3) S is the image of a function $F : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto S$ s.t. each $F_{(i)}$ is Σ-recursive.

Proof. We have proven before this holds for the imperative paradigm. Via paradigm equivalence then this holds for the recursive paradigm.

7.1 The Halting problem

When $\Sigma_p \subseteq \Sigma$, we define

$$AutoHalt^{\Sigma} = \lambda \mathcal{P} \left[(\exists t \in \omega)\, Halt^{0,1}(t, \mathcal{P}, \mathcal{P}) \right]$$

Evidently, $AutoHalt^{\Sigma} \mapsto 1$ with input $\mathcal{P}$ if $\mathcal{P}$ halts when inputted to itself.

Theorem 52 *$AutoHalt^{\Sigma}$ is not Σ-effectively computable. This is, there exists no Σ-effectively computable program that determines if a program halts with itself as input.*

> **Proof.** Assume $AutoHalt$ is Σ-computable. Then we can make $\mathcal{P}$
>
> $$L_1 \quad [IF\ AutoHalt^{\Sigma}(P_1)\ GOTO\ L_1]$$
>
> Let the starting state be $[[\mathcal{P}]]$ itself. Observe that $\mathcal{P}$ halts if $\mathcal{P}$ does not halt, and if it does not halt then it halts. $\perp$ This can be easily extended to effectively computable programs beyond the imperative paradigm.

8 Problems from final exams and the feared Tómbola

Problem 28 *If $\leq$ is an order over Σ and $|\Sigma| = n \geq 1$, then*

$$|\alpha| \leq x \iff \#^{\leq}(\alpha) \leq \sum_{i=0}^{i=x-1} nn^i$$

for whatever $x \in \mathbb{N}, \alpha \in \Sigma^*$.

Before proving that the statement is true, consider this example.

> Let $\Sigma = \{w, z\}$ with $w \leq z$; then the language list is
>
> $$\overbrace{w, z}^{2 \text{ with } |\alpha| \leq 1}, \overbrace{ww, wz, zw, zz}^{2^2 \text{ with } |\alpha| \leq 2}, \overbrace{www, wwz, wzw, wzz, zww, zwz, zzw, zzz}^{2^3 \text{ with } |\alpha| \leq 3}, \ldots$$
>
> It is easy to see that this is generalizable, because there are always n^k words of length k in an alphabet of n symbols.
>
> > **Proof.** Any word of length k is a combination of k symbols from a set of n that can be drawn with repetition. There are n^k words of length k.

($\Rightarrow$) Assume $|\alpha| \leq x$. There are $n^1 + \ldots + n^x$ words $\varphi \in \Sigma^*$ s.t. $|\varphi| \leq x$, to each of which correspond a position $\#^{\leq}(\varphi) \in \mathscr{W} := [1, \ldots, n^1 + \ldots + n^x]$. Since $\#^{\leq}(\alpha) \in \mathscr{W}$ and $\max \mathscr{W} = n^1 + \ldots + n^x$ we have $\#^{\leq}(\alpha) \leq \sum_{i=1}^{x} n^x$. ■

($\Leftarrow$) Assume $\#^{\leq}(\alpha) \leq n^1 + \ldots + n^x$ for $x \in \mathbb{N}$.

Assume $|\alpha| > x$.

$\therefore \#^{\leq}(\alpha) > \#^{\leq}(\beta)$ for any β s.t. $|\beta| \leq x$. There are $n_1 + \ldots + n^x$ words β s.t. $|\beta| \leq x$. Then $\#^{\leq}(\alpha) > n_1 + \ldots + n^x$, which is a contradiction.

$\therefore |\alpha| \leq x$.

Problem 29 *Let $\mathscr{P}_0 \in Pro^{\Sigma}$ and P be the predicate*

$$\lambda t\vec{x}\vec{\alpha}[Halt^{n,m}(t,\vec{x},\vec{\alpha},\mathscr{P}_0) \wedge (t)_1 = E^{n,m}_{\#1}(t,\vec{x},\vec{\alpha},\mathscr{P}_0))]$$

Then $\Psi^{n,m,\#}_{\mathscr{P}_0} = \lambda t\,[(t)_1] \circ M(P)$.

Two functions are the same if their domains are equal and their ranges are equal.
(1)

$\mathscr{D}_{\lambda t[(t)_1]\circ M(P)} = \mathscr{D}_{M(P)} = \{(\vec{x},\vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists t \in \omega)P(t,\vec{x},\vec{\alpha}) = 1\}$
$\mathscr{D}_{\Psi^{n,m,\#}_{\mathscr{P}_0}} = \{(\vec{x},\vec{\alpha}) \in \omega^n \times \Sigma^{*m} : \mathscr{P}_0 \text{ halts with starting state } [[\vec{x},\vec{\alpha}]]\}$

Any $(\vec{x},\vec{\alpha}) \in \mathscr{D}_{M(P)}$ satisfies that $\mathscr{P}_0$ halts starting from $(\vec{x},\vec{\alpha})$; this is, $\mathscr{D}_{M(P)} \subseteq \mathscr{D}_{\Psi^{n,m,\#}_{\mathscr{P}_0}}$. Now consider an arbitrary element $(\vec{x},\vec{\alpha}) \in \mathscr{D}_{\Psi^{n,m,\#}_{\mathscr{P}_0}}$. We know $\mathscr{P}_0$ halts at t steps, for some $t \in \omega$, when starting from $[[\vec{x},\vec{\alpha}]]$. This means it halts at $t+k$ steps for any $k \in \mathbb{N}$. Then, given that input $(\vec{x},\vec{\alpha})$, there are infinitely many $k \in \mathbb{N}$ s.t. the program halts at $t+k$ steps, and thus there is some $t' = t+k$ s.t. the output of the program is $(t')_1$. Then $\mathscr{D}_{\Psi^{n,m,\#}_{\mathscr{P}_0}} \subseteq \mathscr{D}_{M(P)}$.
$\therefore\ \mathscr{D}_{M(P)} = \mathscr{D}_{\Psi^{n,m,\#}_{\mathscr{P}_0}}$
(2) The value of $M(P)$ is the least t s.t. $\mathscr{P}_0$ halts in t steps and s.t. the output $\mathscr{P}_0$ at halt is $(t)_1$. It is trivial to see then that $\Psi^{n,m,\#}_{\mathscr{P}_0} = \lambda t\,[(t)_1] \circ M(P)$.
$\therefore$ The statement is true.

Problem 30 *If f is Σ-computable then $\mathscr{D}_f$ is Σ-recursive.*

Assume f is Σ-computable. We know that $\mathscr{D}_f$ is not necessarily Σ-computable itself. This directly implies $\mathscr{D}_f$ is not necessarily Σ-recursive.

Problem 31 *If $\mathscr{P} \in Pro^{\Sigma}$ s.t. $\mathscr{D}_{\Psi^{1,0,*}_{\mathscr{P}}} = \omega$ then $\mathscr{D}_{\Psi^{1,1,*}_{\mathscr{P}}} = \omega \times \Sigma^*$*

The statement is false. Observe that the program $\mathscr{P}_{\#}$ s.t. $\Psi^{1,0,*}_{\mathscr{P}_{\#}} = \#^{\leq}$ satisfies the antecedent (that the function has an associated program follows from the fact that it is Σ-p.r. and hence Σ-computable).

Add to the first line of $\mathscr{P}_{\#}$ the instruction L_1 $[IF\ P_1 \neq \varepsilon\ GOTO\ L_1]$, which is a valid macro expansion because $\lambda\alpha\,[\alpha \neq \epsilon]$ is Σ-r and then Σ-computable. It is evident that this program still satisfies the antecedent, but does not satisfy the consequent, since

$$\mathscr{D}_{\Psi^{1,1,*}_{\mathscr{P}_{\#}}} = \{(x, \alpha) \in \omega^n \times \Sigma^{*m} : \alpha = \varepsilon\}$$

Problem 32 *Let M a Turing machine that computes $p^{2,0}_1$. Then M computes $p^{1,0}_1$.*

The statement is not necessarily true. Consider the arbitrary instantaneous description

$$\lfloor q_0 B i^{x_1} B i^{x_2} \rfloor$$

The Turing machine may be programmed so that not having a i symbol after the second B symbol leads to a non-halting state. This could be done without loss of generality under the assumption that the Turing machine leads to a state $\left\lfloor q_f i^{x_1} \right\rfloor$. If this were true, then the Machine would not compute $p^{1,0}_1$.

Problem 33 *Let $f, g : \omega \mapsto \omega$ s.t. $g \in PR^{\Sigma}$ and $f \in PR_3^{\Sigma} - PR_2^{\Sigma}$. Then $f \circ g \subset PR_5^{\Sigma} - PR_3^{\Sigma}$.*

Recall that

$$PR_k^{\Sigma} = PR_{k-1}^{\Sigma} \cup \left\{ f \circ [f_1 \dots f_m] : f, f_i \in PR_{k-1}^{\Sigma} \right\} \cup \left\{ R(f,g) : f, g \in PR_{k-1}^{\Sigma} \right\}$$

For brevity, we will call the second and third sets in the union A_{k-1}, B_{k-1}
From the definition of PR_k^{Σ} follows that

$$\begin{aligned} PR_3^{\Sigma} - PR_2^{\Sigma} &= PR_2^{\Sigma} \cup A_2 \cup B_2 - PR_2^{\Sigma} \\ &= A_2 \cup B_2 \end{aligned}$$

and

$$\begin{aligned} PR_5^{\Sigma} - PR_3^{\Sigma} &= PR_4 \cup A_4 \cup B_4 - PR_3^{\Sigma} \\ &= (PR_3^{\Sigma} \cup A_3 \cup B_3) \cup A_4 \cup B_4 - PR_3^{\Sigma} \\ &= A_3 \cup B_3 \cup A_4 \cup B_4 \end{aligned}$$

From this follows that $g \in PR_3^{\Sigma}$, because $g \in A_2 \cup B_2$ is either a composition of functions from PR_2^{Σ} or a function built via primitive recursion with functions of PR_2^{Σ}.
Assume $f \in PR_k^{\Sigma}$ with $k > 5$. Then $f \circ g \in PR_6^{\Sigma}$. Then f is neither a composition of functions in $PR_3^{\Sigma} \cup PR_4^{\Sigma}$ nor built via primitive recursion by functions in $PR_3^{\Sigma} \cup PR_4^{\Sigma}$. In other words, $f \notin PR_5^{\Sigma} - PR_3^{\Sigma}$. This counter-example suffices to show that the statement is false.

Problem 34 *Assume* $\Sigma_p \subseteq \Sigma$. *Show that for any* $\mathcal{P} \in Pro^{\Sigma}$ *there is some* $\mathbb{Q} \in Pro^{\Sigma}$ *s.t.*

$$\Psi^{1,0,\#}_{\mathcal{P}} \circ \Psi^{1,0,\#}_{\mathbb{Q}} = id_{|Im(\Psi^{1,0,\#}_{\mathcal{P}})}$$

For any macro used give the predicate or function associated to it.

Let $\mathcal{P}$ an arbitrary program that computes a Σ-computable function f. Let $\mathbb{Q}$ be

$$\begin{array}{ll} & N_5 \leftarrow N_1 \\ L_1 & [N_2 \leftarrow f(N_3)] \\ & [IF\ N_2 = N_5\ GOTO\ L_2] \\ & N_3 \leftarrow N_3 + 1 \\ & GOTO\ L_1 \\ L_2 & N_1 \leftarrow N_3 \end{array}$$

where $[IF\ N_2 = N_5\ GOTO\ L_2]$ denotes (for practicity) the macro $[IF\ P(V_1, V_2)\ GOTO\ A_1]$ with $P = \lambda xy\ [x = y]$ (which is trivially Σ-computable) and $[V_1 \leftarrow f(V_2)]$ is the macro associated to the Σ-computable function f.

Let $f := \Psi^{1,0,\#}_{\mathcal{P}}, g := \Psi^{1,0,\#}_{\mathbb{Q}}$. We shall prove $f \circ g = id_{Im(f)}$.

(1) Observe that $\mathbb{Q}$ halts if and only if starting from $[[x]]$ with $x \in Im(f)$. The reason is that $\mathbb{Q}$ computes $f(N_3)$ with $N_3 = 0, 1, \ldots$ until $f(N_3) = N_5$. This will only happen if N_5, which stores the input, is in $Im(f)$.

(2) Observe that $\mathbb{Q}$ starting from $[[x]]$ returns a value y s.t. $f(y) = x$. In other words, $g(x) = y \Rightarrow f(y) = x$, or rather $(f \circ g)(x) = x$.

Points *(1)* and *(2)* entail that

$$(f \circ g) = Id_{Im(f)}\ \blacksquare$$

Problem 35 *Let* $\Sigma = \{@, +\}$ *and* $f : \omega^n \times \Sigma^{*m} \mapsto \omega$ *a* Σ*-p.r. function. Show that the set*

$$S = \left\{ (x, y, \alpha) : \exists z \in (\omega - \{1\}) : z \mid \sum_{k=y}^{x+y} f(k^y, \alpha)^x \right\}$$

is Σ*-p.r.*

The approach to the problem is to simplify S and then show that it is the domain of a Σ-p.r. function. Then, by virtue of the theorem which states that a set is Σ-p.r. iff it is the domain of a Σ-p.r. function, we shall have that S is Σ-p.r.

(1) Let $g = \lambda xy\alpha \left[\sum_{k=y}^{x+y} f(k^y, \alpha)^x \right]$. Let $\mathscr{D}_x = \{z \in \omega : z \neq 1 \wedge z \mid x\}$. Evidently, for any $x \neq 1$, $\mathscr{D}_x \neq \emptyset$. So,

$$\langle \exists z \in (\omega - 1) : z \neq 1 : z \mid x \rangle \equiv x \neq 1$$

It follows that $S = \{(x, y, \alpha) : g(x, y, \alpha) \neq 1\}$.
(2) We will prove g is Σ-p.r. Simply observe that by assumption f is Σ-p.r. Then

$$H(x, y, z, \alpha) := f(x^y, \alpha)^z = \lambda xz \left[x^z\right] \circ \left[f \circ \left[\lambda xz \left[x^z\right] \circ \left[p_1^{3,1}, p_2^{3,1}, \right], p_4^{3,1} \right], \ p_3^{3,1} \right]$$

is Σ-p.r. Then

$$g = \lambda abxy\alpha \left[\sum_{k=a}^{k=b} H(k, y, x, \alpha) \right] \circ \left[p_2^{2,1}, \lambda xy \left[x + y\right] \circ \left[p_1^{2,1}, p_2^{2,1} \right], p_1^{2,1}, p_2^{2,1}, p_3^{2,1} \right]$$

The general summation of a Σ-p.r. function is Σ-p.r., and we have proven g is a composition of this general summation and Σ-p.r. functions. Then g is Σ-p.r.
(3) Observe that $\mathscr{D}_{Suc \circ (Suc \circ g)}$ is Σ-p.r. because the function is Σ-p.r. Evidently, this set is closely related to S, insofar as

$$\mathscr{D}_{Suc \circ (Suc \circ g)} = \{(x, y, \alpha) : g(x, y, \alpha) \neq 1 \wedge g(x, y, \alpha) \neq 0\}$$

Now consider the set $S' = \{(x, y, \alpha) : g(x, y, \alpha) = 0\}$, potentially empty. The set is Σ-p.r. because

$$\chi_{S'}^{2,1} = \begin{cases} 1 & g(x,y,\alpha) = 0 \\ 0 & otherwise \end{cases}$$

is $\lambda xy\,[x = y] \circ \left[g \circ \left[p_1^{2,1}, p_2^{2,1}, p_3^{2,1}\right], C_0^{2,1}\right]$
Then, because the union of Σ-p.r. sets is Σ-p.r. we have

$$\mathcal{D}_{Suc\circ(Suc\circ g)} \cup S' = S$$

is Σ-p.r.

Problem 36 *Let*

$$L = \left\{ \mathscr{P} \in Pro^{\Sigma} : (\exists x \in \omega)\ \Psi^{1,1,*}_{\mathscr{P}}(x, \varepsilon) = \varepsilon \right\}$$

Find a $\mathbb{Q} \in Pro^{\Sigma}$ *s.t.* $Im(\Psi^{1,0,*}_{\mathbb{Q}}) = L$ *and* $\mathscr{D}_{\Psi^{1,0,*}_{\mathbb{Q}}} = \omega$

The problem essentially asks for a program $\mathbb{Q}$ that enumerates L. Let $\mathscr{P}$ be fixed, arbitrary element of L (For example $SKIP$, which evidently belongs to L). Using the fact that $(\Sigma \cup \Sigma_p)^*$ is Σ-enumerable (it is $(\Sigma \cup \Sigma_p)$-total), we can write $\mathbb{Q}$ to reproduce the following algorithm with an input x:

> *(1)* If $x = 0$ return $\mathscr{P}$ and stop. Else continue.
> *(2)* Enumerate $(\Sigma \cup \Sigma_p)$ with input $(x)_1$ and store the word in $\mathscr{P}'$.
> *(3)* If $\mathscr{P}'$, from a starting state $[[(x)_2, \varepsilon]]$, halts in $(x)_3$ steps with final state $[[\epsilon]]$, output $\mathscr{P}'$ and finish.
> *(4)* Output $\mathscr{P}$.

Let Φ the Σ-computable program which enumerates $(\Sigma \cup \Sigma_p)^*$ and $[V_1 \leftarrow \Phi(V_2)]$ its associated macro. Let $[W_1 \leftarrow \mathscr{P}]$ be the macro that assigns to W_1 the word $\mathscr{P} \in L$. Then the program $\mathbb{Q}$ given by

$$\begin{array}{ll}
 & IF\ N_1 \neq 0\ GOTO L_1 \\
 & GOTO\ L_4 \\
L_1 & [P_1 \leftarrow \Phi((N_1)_1)] \\
 & [IF\ Halt^{1,1}\ ((N_1)_3, (N_1)_2, \varepsilon, P_1)\ \ GOTO\ L_2] \\
 & GOTO\ L_4 \\
L_2 & [IF\ E^{1,1}_{*1}\ ((N_1)_3, (N_1)_2, \varepsilon, P_1) \neq \varepsilon\ GOTO\ L_4] \\
 & GOTO\ L_5 \\
L_4 & [P_1 \leftarrow \mathscr{P}] \\
L_5 & SKIP
\end{array}$$

enumerates L.

Note. Observe that $Halt$, $(x)_i$, $\lambda\alpha\beta\,[\alpha \neq \beta]$, $E^{1,1}_{*i}$ are all Σ-p.r. Then their compositions are Σ-p.r. and then Σ-computable. By alphabet independence they are $(\Sigma \cup \Sigma_p)$-computable $\therefore$ The macros used in the program are all valid.

Problem 37 *Let* $\Sigma = \{@, ?, \sim\}$. *Prove that*

$$S = \left\{(x, \alpha, \beta) \in \mathbb{N} \times \Sigma^* \times \{?, \sim\}^* : x = t^2 \text{ for some } t \in \mathbb{N} \vee \alpha = \beta\right\}$$

is Σ*-p.r.*

S is Σ-p.r. iff $\chi_S^{1,2}$ is Σ-p.r. Evidently

$$\chi_S^{1,2} = \lambda x \alpha \beta \left[\left((\exists t \in \omega)_{t \leq x}\, x = t^2 \ \vee \alpha = \beta\right) \wedge |\beta|_@ = 0\right]$$

We will show this function is Σ-p.r. by separating the predicates and showing that each is Σ-p.r. One must only be careful in ensuring that the predicates have the same domains.

(1) Let $P_1 = \lambda x \alpha \beta \left[(\exists t \in \omega)_{t \leq x}\, x = t^2\right]$. (Of course $x = t^2 \Rightarrow t \leq x$, hence the natural bound.) Of course, this is

$$\lambda u x \alpha \beta \left[(\exists t \in \omega)_{t \leq u}\, t = x^2\right] \circ \left[p_1^{1,2}, p_1^{1,2}, p_2^{1,2}, p_3^{1,2}\right]$$

It is a theorem that $\lambda x \vec{x} \vec{\alpha}\, [(\exists t \in \omega)_{t \leq x} Q(\vec{x}, \vec{\alpha})]$, with Q a predicate, is Σ-p.r. if Q is Σ-p.r. In our case it is evident that $t = x^2$ is Σ-p.r. and hence P is Σ-p.r.

(2) It is a theorem that $P_2 := \lambda x \alpha \beta\, [\alpha = \beta]$ is Σ-p.r.

(3) Let $f = C_0^{0,0}$ and $\mathcal{G}$ be the following indexed family of functions:

$$\begin{aligned} \mathcal{G} : \Sigma &\mapsto \left\{Suc \circ p_1^{1,1}, p_1^{1,1}\right\} \\ @ &\mapsto Suc \circ p_1^{1,1} \\ ? &\mapsto p_1^{1,1} \\ \sim &\mapsto p_1^{1,1} \end{aligned}$$

Then evidently $P_3 := \lambda x \alpha \beta\, [|\beta|_@] = R(f, \mathcal{G})$ is Σ-p.r.

Since all these predicates are Σ-p.r. and have identical domains, then

$$\chi_S^{1,2} = (P_1 \vee P_2) \wedge P_3$$

is well-defined and Σ-p.r. ■

Problem 38 *Assume* $f : \mathcal{D}_f \subseteq \Sigma^* \mapsto \omega$ *is* Σ*-effectively computable and* $g : \mathcal{D}_g \subseteq \Sigma \mapsto \omega$ *is* Σ*-effectively computable. Show*

$$L = \left\{\alpha \in \mathcal{D}_f \cap \mathcal{D}_g : f(\alpha) = g(\alpha)\right\}$$

is Σ*-effectively enumerable. Assume* $\varepsilon \in L$.

Let $\mathbb{P}_f, \mathbb{P}_g$ be the effective procedures which compute f, g respectively. Let $\mathbb{P}_L$ be the following effective procedure, with a unique input $x \in \omega$:

(1) If $x = 0$ go to step *(7)*; otherwise go to step *(2)*
(2) Let $\alpha = *^{\leq}((x)_1)$.
(3) If $|\alpha| > 1$ let $x = x + 1$ and go to step *(2)*. Otherwise, go to step *(4)*.
(4) Do $(x)_2$ steps of procedure $\mathbb{P}_f$ with input α. If the program finished, return its output in φ and go to *(5)*. If it didn't finish, let $x = x + 1$ and go to *(2)*.
(5) Do $(x)_3$ steps of procedure $\mathbb{P}_g$ with input α. If the program finished, store its output in ψ and go to *(7)*; otherwise let $x = x + 1$ and go to *(2)*.
(6) If $\varphi = \psi$ go to *(8)*
(7) Let $\alpha = \varepsilon$.
(8) Return α.

The procedure enumerates L.

Problem 39 *Determine true or false for the following statements.*

(1) $Num^* \neq \omega$.

Certainly true. In fact $Num^* \cap \omega = \emptyset$. The elements of Num are symbols denoting the natural numbers and zero, but *are not* these numbers.

(2) $\{(\omega, 2)$ is a function whose domain is $\omega\}$

Tuples containing sets are not defined. Hence $(\omega, 2)$ is an undefined and meaningless expression. The statement is false. However, observe that the set $\{(x, 2) : x \in \omega\}$ is the function $C_2^{1,0}$.

(3) If $(\vec{s}, \vec{\sigma})$ is a state of $\mathcal{S}^\Sigma$, then $Ti(\vec{s}) = n$-tuple for some n.

The statement is false. By definition, the state of a program in $\mathcal{S}^\Sigma$ is a 2-tuple whose first element is a member of $\omega^{[\mathbb{N}]}$. This is the set of infinite tuples that contain only zeros from some index onwards. Then $\vec{s}$ is infinite.

(4) Let $\Sigma = \{@, \sim\}$. Then $\lambda\alpha\,[@\alpha]$ is the function

$$R\left(C_@^{0,0}, \left\{(@, d_@ \circ p_2^{0,2}), (\sim, d_\sim \circ p_2^{0,2})\right\}\right)$$

Let $\mathcal{G}$ denote the indexed family of functions which the statement associates to R.
The first observation is that the domains are correct. Since $\lambda\alpha\,[@\alpha]$ does recursion over an alphabetic var, mapping to alphabetic values, and has type $(0, 1, *)$, then $f \sim (0, 0, *)$ and $\mathcal{G}_a \sim (0, 2, *)$ for all $a \in \Sigma$.
Evidently $R(f, g)(\epsilon) = @$ which makes the base case function correct. Now, the indexed family of functions maps

$$R(f, g)(\alpha w) = dw \circ R(\alpha)$$

We can prove this is not correct. Let $\alpha \in \Sigma^*$ be s.t. $|\alpha| = n$. Then

$$
\begin{aligned}
R(\alpha w) &= R(\alpha)w \\
&= R(\curvearrowleft\alpha)w[\alpha]_n \\
&= R\left(\curvearrowleft\curvearrowleft\alpha\right) w[\alpha]_n[\alpha]_{n-1} \\
&\vdots \\
&= R(\varepsilon)w[\alpha]_n[\alpha]_{n-1}\ldots[\alpha]_1 \\
&= @w[\alpha]_n[\alpha]_{n-1}\ldots[\alpha]_1
\end{aligned}
$$

Evidently $R(f, g)$ appends @ to the start of the reciprocal of α, which is not what $\lambda\alpha\,[@\alpha]$ does.

(5) For whatever $\alpha \in \Sigma^*$, we have $[\varepsilon\alpha]_1 \neq \varepsilon$.

This is false. Assume $\alpha = \varepsilon$. Then $\varepsilon\alpha = \varepsilon$ and $[\varepsilon\alpha]_1 = [\varepsilon\varepsilon]_1 = [\varepsilon]_1 = \varepsilon$.

(6) Let Σ a finite alphabet. Let $S = \left\{(x, y) \in \omega \times \mathbb{N} : \sqrt{x} \in \mathbb{Q}\right\}$. Then S is a Σ-mixed set.

By definition, a Σ-mixed set is any set W that satisfies $W \subseteq \omega^n \times \Sigma^{*m}$ for $n, m \geq 0$. Evidently, $S \subseteq \omega \times \mathbb{N}$ does not satisfy this.

(7) f is Σ-mixed iff $\exists n, m \geq 0$ s.t. $\mathscr{D}_f \subseteq \omega^n \times \Sigma^{*m}$ $\wedge \mathrm{I}_f \subseteq \omega \cup \Sigma^*$

The statement is false. The image of a Σ-mixed function must be *either* ω or Σ^*, but it cannot be the union of these.

(8) Let $M = (Q, \Sigma, \Gamma, \delta_0, q_0, B, F)$ a deterministic Turing machine and assume Q is an alphabet disjoint from Γ. Let $d, d' \in \mathbb{D}$. Then it is the case that

$$d \vdash^1 d' \Rightarrow d \neq d'$$

The statement is clearly false. We can provide a counter-example. Let Σ be an arbitrary language. Assume the machine has any non-zero number of states, some of which are final states. If $\delta(q_0, x) = \{(q_0, x, K)\}$ for any $x \in \Sigma$. Let d be the instantaneous description of the machine when it starts. Obviously, $d \vdash^n d$ for any $n \in \mathbb{N}$.

(9) Let M a Turing machine exactly like the one above. Let $d \in \mathbb{D}$. Then if $d \vdash d$ we have that M halts starting from d.

> The statement is false. We say a machine halts starting from d
>
> > *(1)* $d \vdash^* d'$
> > *(2)* $d' \nvdash d''$
>
> for $d', d'' \in \mathbb{D}$. The scenario described does not meet this definition, because $d \vdash d$ implies that it is not true that $d \nvdash d'$.

(1) $Ins \subseteq Pro^{\Sigma}$

The statement is false. Instructions in Ins^{Σ} may not satisfy the GOTO law.

(2) Let $\Sigma = \Sigma_p$. Then

$$\Psi^{2,1,*}_{\text{P1}\leftarrow\ \text{P1.SP1}\leftarrow\text{P1.KP1}\leftarrow\text{P1.IP1}\leftarrow\text{P1.P}} = \lambda\alpha\beta[\alpha\beta] \circ \left[p_1^{2,1}, C^{2,1}_{\text{SKIP}}\right]$$

The word $P_1 \leftarrow P_1.SP_1 \leftarrow P_1.KP_1 \leftarrow \ldots$ is not a program since it is not a concatenation of instructions.

9 Final 2019-07

Problem 40 *Let*

$$L = \left\{ \mathscr{P} \in Pro^{\Sigma_p} : (\exists x \in \mathbb{N})\ \Psi^{1,1,*}_{\mathscr{P}}(x, EBE) = EBE \right\}$$

Find a program $\mathbb{Q} \in Pro^{\Sigma_p}$ that enumerates L. For each macro used, give the associated Σ_p-computable function.

Observe that $SKIP \in L$. Observe that Σ_p^* is Σ_p-enumerable (because it is Σ_p-total). Furthermore, Pro^{Σ_p} is Σ-p.r. and therefore $\chi^{\Sigma_p}_{Pro^{\Sigma_p}}$ is Σ-recursive and hence Σ-computable. Let $\mathscr{F}(x)$ be the Σ-computable function which enumerates Σ_p^*. Then

$$\begin{array}{rl}
 & [IF\ N_1 = 0\ GOTO\ L_{666}] \\
 & [N_2 \leftarrow (N_1)_1] \\
 & [N_3 \leftarrow (N_1)_2] \\
 & [N_4 \leftarrow (N_1)_3] \\
 & [P_1 \leftarrow \mathscr{F}(N_2)] \\
 & [IF\ P_1 \in Pro^{\Sigma_p}\ GOTO\ L_1] \\
 & GOTO L_{666} \\
L_1 & [IF\ Halt^{1,1}\ (N_3, N_4, EBE, P_1)\ \ GOTO\ L_2] \\
 & GOTO\ L_{666} \\
 & [IF\ E^{1,1}_{*1}\ (N_3, N_4, EBE, P_1) = EBE\ GOTO\ L_3] \\
L_{666} & P_1 \leftarrow SKIP \\
L_3 & SKIP
\end{array}$$

enumerates L.
Since $E^{1,1}_{*1}$ is Σ_p-p.r. and $\lambda\alpha\beta\,[\alpha = \beta]$ is Σ-p.r. the macro $[IF\ E^{1,1}_{*1}\ \ldots]$ etc. is trivially associated to a Σ-computable. The same can be said of the macro that uses $Halt^{1,1}$.

Problem 41 *Let* $\Sigma = \{@, \sim\}$. *Let*

$$\mathscr{F} : \{x \in \omega : x\ is\ odd\ \} \times \Sigma^* \mapsto \omega$$

$$(x, \alpha) \mapsto \begin{cases} x & x \geq 3 \\ |\alpha| & otherwise \end{cases}$$

Prove that $\mathcal{F}$ is Σ-p.r.

We will present two different proofs of this fact.
(1) Observe that $\mathcal{D}_{\mathcal{F}} = \{x \in \omega : x \text{ is odd}\} \times \Sigma^*$. Since $\lambda xy\,[x \mid y]$ is Σ-p.r. we have $\lambda x\,[2|x]$ is Σ-p.r. The negation of aΣ-p.r. predicate is Σ-p.r. and then $\lambda x\,[2 \nmid x]$ is Σ-p.r. It is then evident that $\chi_{\mathcal{D}_{\mathcal{F}}}^{\omega \times \Sigma^*} = \lambda x\,[2 \nmid x]$ is Σ-p.r.
$\therefore$ $\mathcal{F}$ is Σ-p.r.
(2) Let $\mathcal{F}_1(x, \alpha) = x$ and $\mathcal{F}_2(x, \alpha) = |\alpha|$. It is trivial to see both are Σ-p.r. Furthermore, the set $S_1 := \{(x, \alpha) : x \text{ is odd} \wedge x \geq 3\}$ is Σ-p.r. (trivial) So is the set $S_2 := \{(x, \alpha) : x \text{ is odd } \wedge x < 3\}$. Then $\mathcal{F}_{1|S_1}, \mathcal{F}_{2|S_2}$ are both Σ-p.r. Since $S_1 \cap S_2 = \emptyset$, we have $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$.
$\therefore$ $\mathcal{F}$ is Σ-p.r.

Problem 42 *Determine true or false for the following statements.*

(1) $\{(\omega, w)\}$ is a function whose domain is ω.

The statement is false. In all rigour, $\{(\omega, w) = (\{0, 1, 2, \ldots\}, w)\}$ which is not a function.

(2) The function $x + 1$ is Σ-mixed for whatever alphabet Σ.

The expression $x+1$ is undefined. If x were defined over $\mathbb{Z}$, for example, the function would not be Σ-mixed.

(3) Let $\mathbb{P}$ an effective procedure which computes $f : \mathcal{D}_f \subseteq \omega \mapsto \omega$. Then the input set of $\mathbb{P}$ is $\mathcal{D}_f$

By definition we say $\mathbb{P}$ computes f if $\mathbb{P}$ halts from input $(\vec{x}, \vec{\alpha}) \in \mathcal{D}_f$ with output $f(\vec{x}, \vec{\alpha})$ and does not halt for $(\vec{x}, \vec{\alpha}) \notin \mathcal{D}_f$. However, the proper input set of the procedure is precisely $\omega^n \times \Sigma^{*m}$. So the statement is false.

(4) Let $\Sigma = \{@, *\}$. Then

$$R(p_1^{0,1}, \left\{(@, p_3^{0,3}), (*, d_* \circ p_3^{0,3})\right\})(@*, @*) = @ * **$$

Let $\mathcal{F} = R(f, g)$. Then by def.

$$\begin{aligned} \mathcal{F}(@*, @*) &= R(@*, @)* \\ &= R(@*, \varepsilon)* \\ &= @ * * \end{aligned}$$

The statement is false.

(5) If P_1, P_2, P_3 are Σ-p.r. predicates with equal domains, then $(P_1 \vee p_2 \wedge P_3)$ is Σ-p.r.

The statement is true by theory.

(6) Let Σ an alphabet and $\mathscr{P} \in Pro^{\Sigma}$. Then

$$\Phi_{*}^{n,m} \circ \left[p_1^{n,m}, \ldots, p_{n+m}^{n,m}, C_{\mathscr{P}}^{n,m}\right]$$

is a Σ-mixed function.

Φ is undefined and hence the statement is false.

(7) If M a Turing machine then $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R, K\}$

The statement is false, insofar as the domain of δ is $\mathscr{D}_{\delta} \subseteq Q \times \Gamma$ but may not be $Q \times \Gamma$.

(8) If $\mathscr{P} \in Pro^{\Sigma}$ is s.t. $GOTO$ is not a sub-word of $\mathscr{P}$, then $\Psi_{\mathscr{P}}^{1,0,\#}$ is Σ-p.r.

(1) Sea $\mathscr{P}_0 \in Pro^{\Sigma}$ un programa fijo que no tiene enunciados que involucren la sub-palabra $GOTO$. Podemos deducir que $\mathscr{P}_0$ se detiene siempre en $n(\mathscr{P}_0) + k$ pasos, con $k \geq 0$. En otras palabras, $\lambda tx \left[Halt^{1,0}(t, x, \mathscr{P}_0)\right] \geq n(\mathscr{P}_0)$. De esto se sigue por definición de $T^{1,0}$ que $T^{1,0}(x, \mathscr{P}_0) = n(\mathscr{P}_0)$ para cualquier $x \in \omega$, o más formalmente

$$\lambda x \left[T^{1,0}(x, \mathscr{P}_0)\right] = \lambda \mathscr{P} \left[n(\mathscr{P})\right] \circ C^{1,0}_{\mathscr{P}_0}$$

> *Duda.* Por teorema $\lambda \mathscr{P} \left[n(\mathscr{P})\right]$ es $(\Sigma \cup \Sigma_p)$-p.r. Luego, para nuestro $\mathscr{P}_0$ fijo, sucede que $\lambda x \left[T^{1,0}(x, \mathscr{P}_0)\right]$ es $(\Sigma \cup \Sigma_p)$-p.r. Pero esto no parece ser necesario para continuar la demostración. Es decir, todo lo que viene ahora prescinde de este hecho. ¿O me equivoco?

(2) Recuerde que $\lambda x \left[\Psi^{1,0,\#}_{\mathscr{P}_0}(x)\right]$ es, por definición, el valor que resulta en la primer variable numérica de $\mathscr{P}_0$ cuando termina partiendo de $[[x]]$. Puesto que sabemos que $\mathscr{P}_0$ termina en $n(\mathscr{P}_0)$ pasos, es evidente por la misma definición de $E^{1,0}_{\#j}$ que

$$\begin{aligned}\lambda x \left[\Psi^{1,0,\#}_{\mathscr{P}_0}(x)\right] &= \lambda x \left[E^{1,0}_{\#1}(n(\mathscr{P}_0), x, \mathscr{P}_0)\right] \\ &= \lambda tx\mathscr{P} \left[E^{1,0}_{\#1}(t, x, \mathscr{P})\right] \circ \left[\lambda \mathscr{P} \left[n(\mathscr{P})\right] \circ C^{1,0}_{\mathscr{P}_0}, p^{1,0}_1, C^{1,0}_{\mathscr{P}_0}\right]\end{aligned}$$

La función constante, la proyección, y $\lambda \mathscr{P} \left[n(\mathscr{P})\right]$ son $(\Sigma \cup \Sigma_p)$-p.r. Luego la función dada arriba es $(\Sigma \cup \Sigma_p)$-p.r. Por independencia del alfabeto, se sigue que es Σ-p.r.
$\therefore \lambda x \left[\Psi^{1,0,\#}_{\mathscr{P}_0}(x)\right]$ es Σ-p.r. ■

De esto se sigue trivialmente que $T^{1,0}(x, \alpha, \mathcal{P}) \leq n(\mathcal{P})$. Puesto que (por un teorema del teórico) la función $\lambda\mathcal{P}\,[n(\mathcal{P})]$ es $(\Sigma \cup \Sigma_p)$-p.r., tenemos que $T^{1,0}(x, \alpha, \mathcal{P})$ es menor o igual a una función $(\Sigma \cup \Sigma_p)$-p.r. Recordemos el teorema que establece que

> Si $P : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m}$ un predicado Σ-p.r.
> y $M(P) \leq f(\vec{x}, \vec{\alpha})$ con f una función Σ-p.r.,
> entonces $M(P)$ es Σ-p.r.

El predicado $Halt^{n,m}$ es $(\Sigma \cup \Sigma_p)$-p.r. para cualesquiera $n, m \in \omega$. Luego $T^{1,0}(x, \mathcal{P})$ es $(\Sigma \cup \Sigma_p)$-p.r.

(2) Denotemos $\mathcal{T} = \lambda x \left[T^{1,0}(x, \mathcal{P})\right]$ y $\mathcal{C} = \lambda x \left[E^{1,0}_{\#}(\mathcal{T}(x), x, \mathcal{P})\right]$. Es decir, $\mathcal{C}$ mapea un elemento x al valor de la primer variable numérica que resulta de ejecutar $\mathcal{P}$ desde el estado $[[x]]$, donde $\mathcal{P}$ termina con la cantidad mínima de pasos. Es evidente entonces que $\Psi^{1,0,\#}_{\mathcal{P}} = \mathcal{C}$. Pero dado que

$$\mathcal{T} = T^{1,0} \circ \left[p^{1,0}_1, C^{1,0}_{\mathcal{P}}\right]$$

es evidentemente $(\Sigma \cup \Sigma_p)$-p.r., y $E^{1,0}_{\#}$ es $(\Sigma \cup \Sigma_p)$-p.r., entonces

$$\mathcal{C} = E^{1,0}_{\#} \circ [\mathcal{T} \circ p^{1,0}_1, p^{1,0}_1, C^{1,0}_{\mathcal{P}}]$$

es $(\Sigma \cup \Sigma_p)$-p.r. Por independencia del alfabeto, $\mathcal{C}$ is Σ-p.r. Pero $\Psi^{1,0,\#}_{\mathcal{P}} = \mathcal{C}$. ■

10 Combos del listado de teoremas (2024 Enero)

10.1 Definiciones del listado

(1) El conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-recursivo.

Decimos que $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-recursivo cuando existe una función Σ-recursiva que decide la pertenencia de un elemento $x \in \omega^n \times \Sigma^{*m}$ respecto de S. En otras palabras, cuando la función característica $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-recursiva.

(2) $\langle s_1, s_2, \ldots \rangle$

Dada una infinitupla $(s_1, s_2, \ldots) \in \omega^{[\mathbb{N}]}$, definimos

$$\langle s_1, s_2, \ldots \rangle = \prod_{i=1}^{\infty} p(i)^{s_i}$$

(3) f es una función Σ-mixta.

Decimos que f es una función Σ-mixta si es una función cuyo dominio es un subconjunto de $\omega^n \times \Sigma^{*m}$ para algún $n, m \in \omega$ y cuya imagen es o bien ω o bien Σ^*.

(4) Familia indexada de funciones.

Una familia Σ-indexada de funciones $\mathcal{G}$ es una función cuyo dominio es un alfabeto Σ y cuya imagemn es un conjunto de funciones; es decir, $\mathcal{G}(a)$ es una función para toda $a \in \Sigma$.

(4) $R(f, \mathcal{G})$

$R(f, \mathcal{G})$ es una función definida recursivamente con caso base dependiente de f y caso recursivo dependiente de $\mathcal{G}$, y tal que la recursión se hace sobre una variable alfabética. Si $R(f, \mathcal{G}) : \omega^n \times \Sigma^{*m} \times \Sigma^* \mapsto \Sigma^*$, entonces

$$R(\vec{x}, \vec{\alpha}, \alpha) = \begin{cases} f(\vec{x}, \vec{\alpha}) & \alpha = \epsilon \\ \mathcal{G}_{[\alpha]_{|\alpha|}}(\vec{x}, \vec{\alpha}, \alpha^{\frown}, R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha^{\frown})) & \alpha \neq \epsilon \end{cases}$$

Si $R(f, \mathcal{G}) \mapsto \omega$ entonces

$$R(\vec{x}, \vec{\alpha}, \alpha) = \begin{cases} f(\vec{x}, \vec{\alpha}) & \alpha = \epsilon \\ \mathcal{G}_{[\alpha]_{|\alpha|}}(R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha^{\frown}), \vec{x}, \vec{\alpha}, \alpha^{\frown}) & \alpha \neq \epsilon \end{cases}$$

(1) $d \vdash^n d'$

Dadas dos descripciones instantáneas $d, d' \in Des$, decimos que $d \vdash^n d'$ si y solo si existen $d_1, \ldots, d_{n+1} \in Des$ tales que

- $d_1 = d$
- $d_i \vdash d_{i+1}$
- $d_{n+1} = d'$

Evidentemente, $d \vdash^0 d'$ sii $d = d'$.

(2) $L(M)$

Dada una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, usamos $L(M)$ para denotar el conjunto de palabras que son aceptadas por dicha máquina por alcance de estado final. Es decir,

$$L(M) = \{w \in \Sigma^* : (\exists d \in Des)\ \lfloor q_0 B w \rfloor \vdash^* d \wedge St(d) \in F\}$$

(3) $H(M)$

Dada una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, denotamos con $H(M)$ el conjunto de palabras que son aceptadas por detención por dicha máquina; es decir

$$H(M) = \{w \in \Sigma^* : (\exists d \in Des)\ \lfloor q_0 B w \rfloor \vdash d \wedge \neg(d \vdash d') \text{ para todo } d' \in Des\}$$

(4) f es de tipo (n, m, s)

Cuando $n, m \in \omega$ y $s \in \{\#, *\}$, decimos que una función Σ-mixed f es de tipo (n, m, s) si y solo si $\mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m}$ y $Im_f \subseteq g(s)$, con $g(s) = \Sigma^*$ si $s = *$ y $g(s) = \omega$ si $s = \#$.

(5) (x)

Dado un número $x \in \mathbb{N}$, tenemos que $(x) = (s_1, s_2, \ldots) \in \omega^{[\mathbb{N}]}$ la infinitupla tal que $x = \langle s_1, s_2, \ldots \rangle$.

(6) $(x)_i$

$(x)_i$ es una función con dominio $\mathbb{N}^2$, y denota el exponente del i-écimo primo en la descomposición prima de x.

(1) Defina cuando $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-r.e.

Decimos que $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-recursivamente enumerable si y solo si existe una función $F : \omega \mapsto \omega^n \times \Sigma^{*m}$ tal que $Im(F) = S$ y $F_{(i)}$ es Σ-recursiva.

(2) Defina $s^{\leq}$.

La función $s^{\leq} : \Sigma^* \mapsto \Sigma^*$ dado un orden $\leq$ sobre Σ mapea cada palabra $\alpha \in \Sigma^*$ con su sucesora en el orden. Se define de manera recursiva. En general, si $\Sigma = \{a_1, \ldots, a_n\}$, tenemos que

$$s^{\leq}(a_n^m) = a_1^{m+1}$$

para todo $m \in \omega$, y

$$s^{\leq}(\alpha a_i a_n^m) = \alpha_{i+1} a_1^m$$

con $\alpha \in \Sigma^*, 1 \leq i < n, m \in \omega$.
Un lenguaje puede entonces enumerarse de manera ordenada así: $\varepsilon, s^{\leq}(\varepsilon), s^{\leq}(s^{\leq}(\varepsilon)), \ldots$

(3) Defina $*^{\leq}$

Dado Σ un alfabeto y $\leq$ un orden sobre dicho alfabeto, la función Σ-mixta $*^{\leq} : \omega \mapsto \Sigma^*$ es una biyección entre ω y Σ^*. En particular, $*^{\leq}(0) = \varepsilon$ y

$$*^{\leq}(k+1) = s^{\leq}(*^{\leq}(i))$$

En general, le asigna a cada número i la i-écima palabra de acuerdo con el listado generado por $s^{\leq}$.

(4) Defina $\#^{\leq}$

La función $\#^{\leq}$ es la inversa de $*^{\leq}$, de lo cual se sigue que tiene dominio Σ^* e imagen ω. Si $\Sigma = \{a_1, \ldots, a_n\}$, su definición es: $\#^{\leq}(\varepsilon) = 0$,

$$\#^{\leq}(a_{i_k} \ldots a_{i_0}) = i_k n^k + \ldots + i_0 n^0$$

(1) Dada $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \omega$ con f Σ-efectivos computable, defina "el procedimiento $\mathbb{P}$ computa a f".

Decimos que el procedimiento $\mathbb{P}$ computa a $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \omega$ si y solo si:

- (1) El conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$
- El conjunto de datos de salida de $\mathbb{P}$ está contenido en ω
- Si el valor de entrada $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ pertenece al dominio de f, el procedimiento termina y devuelve $f(\vec{x}, \vec{\alpha})$.
- Si el valor de entrada $(\vec{x}, \vec{\alpha})$ no pertenece al dominio de f, el procedimiento nunca termina.

(1) Defina cuándo $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable y defina "el procedimiento $\mathbb{P}$ decide la pertenencia a S".

> Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable si y solo si $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-efectivamente computable.
> Decimos que $\mathbb{P}$ decida la pertenencia a S si y solo si
>
> - El conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$, siempre termina, y da como resultado un valor en $\{0, 1\}$
> - Si el input $\omega^n \times \Sigma^{*m}$ pertenece a S da como salida 1, de otro modo da como salida 0.

(1) Defina cuándo la función $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \omega$ es Σ-Turing computable y "la máquina de Turing M computa a f".

La función $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \omega$ es Σ-Turing computable si f es una función Σ-mixta y existe una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ que computa a f. Decimos que M comput a f si y solo si:
(a) Dado $(\vec{x}, \vec{\alpha}) \in \mathcal{D}_f$, sucede que

$$\lfloor q_0 B i^{x_1} B \ldots B i^{x_n} B \alpha_1 B \ldots B \alpha_m \rfloor \vdash^*_{(M)} \left\lfloor p B i^{f(\vec{x},\vec{\alpha})} \right\rfloor$$

con $p \in Q$ y $\neg \left(\lfloor p i^{f(\vec{x},\vec{\alpha})} \rfloor \vdash d \right)$ para todo $d \in Des$.
(b) Dado $(\vec{x}, \vec{\alpha}) \in (\omega^n \times \Sigma^{*m}) - \mathcal{D}_f$, entonces M no se detiene partiendo de $\lfloor q_0 B i^{x_1} B \ldots B i^{x_n} B \alpha_1 B \ldots B \alpha_m \rfloor$.

(1) $M(P)$

Dado un predicado $P : \omega \times \omega^n \times \Sigma^{*m} \mapsto \omega$, $M(P)$ denota la función

$$\lambda \vec{x}\vec{\alpha} \left[\min_t P(t, \vec{x}, \vec{\alpha}) \right]$$

El dominio de $M(P)$ es $\{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists t \in \omega)\ P(t, \vec{x}, \vec{\alpha}) = 1\}$. Su conjunto de llegada es naturalmente ω.

(2) Lt

$Lt : \mathbb{N} \mapsto \omega$ es la función que asocia un número natural x al mayor i tal que $(x)_i \neq 0$. Es decir,

$$Lt(x) = \begin{cases} \max_i (x)_i \neq 0 & x > 1 \\ 0 & x = 1 \end{cases}$$

(3) Conjunto rectangular.

Un conjunto A es rectangular si y solo si $A = S_1 \times \ldots S_n \times L_1 \times \ldots \times L_m$ con $S_i \subseteq \omega, L_i \subseteq \omega^n \times \Sigma^{*m}$. Por ser fruto de productos cartesianos, los conjuntos rectangulares satisfacen que si $(\vec{x}, \vec{\alpha}) \in A, (\vec{y}, \vec{\beta}) \in A$, entonces $(\vec{x}, \vec{\beta}) \in A$.

(4) S es un conjunto de tipo (n, m)

Decimos que S un conjunto Σ-mixto es un conjunto de tipo (n, m) si y solo si $S \subseteq \omega^n \times \Sigma^{*m}$. Observe que $\emptyset \subseteq \omega^n \times \Sigma^{*m}$ para todo $n, m \in \omega$; es decir, $\emptyset$ es de tipo $(0, 0), (0, 1), (1, 0), (1, 1), \ldots$ etc. Si $S \neq \emptyset$, existe un único par $n, m \in \omega$ tal que S es de tipo (n, m).

(1) I es una instrucción de $\mathcal{S}^{\Sigma}$.

Decimos que I es una instrucción de $\mathcal{S}^{\Sigma}$ si I es una palabra de $(\Sigma \cup \Sigma_p)$ que es o bien una instrucción básica o es de la forma αJ, con J una instrucción básica y $\alpha = L\overline{n}$ para algún $n \in \mathbb{N}$.

(2) $\mathcal{P}$ es un programa de $\mathcal{S}^{\Sigma}$

Decimos que $\mathcal{P}$ es un programa de $\mathcal{S}^{\Sigma}$ si $\mathcal{P}$ es una concatenación de instrucciones $I_1 \dots I_n$ y satisface la ley de los GOTO; es decir, si $GOTOL\overline{n}$ es tramo final de I_j para algunos $i, j \in \mathbb{N}$, entonces existe un I_i, $i \in \mathbb{N}$, tal que $L\overline{n}$ es el label de I_i.

(3) $I_i^{\mathcal{P}}$

$I_i\mathcal{P}$ es una función que, dado un programa $\mathcal{P} = I_{1n} \in Pro^{\Sigma}$ y un natural $i \in \mathbb{N}$, devuelve I_i si $1 \leq i \leq n$, y ε de otro modo.

(4) $n(\mathcal{P})$

Dado un programa $\mathcal{P} = I_1 \dots I_n \in Pro^{\Sigma}$, la expressión $n(\mathcal{P})$ denota n; es decir, la cantidad de instrucciones que participan en la concatenación de instrucciones $\mathcal{P}$.

(5) Bas

$Bas : Ins^{\Sigma} \mapsto (\Sigma \cup \Sigma_p)^*$ es una función que, dada una instrucción $I \in Ins^{\Sigma}$, devuelve solo la parte de I que constituye una instrucción básica. Formalmente,

$$Bas(J) = \begin{cases} I & J = L\overline{n}I \text{ para cierto } n \in \mathbb{N} \text{y} \mathrm{I} \in Ins^{\Sigma} \\ J & \text{de otro modo} \end{cases}$$

Todas las definiciones son relativas a $\mathcal{S}^{\Sigma}$.

(1) "estado"

Un estado es una $2-UPLA$ $(\vec{s},\vec{\sigma})$ tal que $\vec{s}\in\omega^{[\mathbb{N}]}$ y $\vec{\sigma}\in\Sigma^{*[\mathbb{N}]}$. Se habla generalmente del estado de un programa $\mathcal{P}$, y decir que $(\vec{s},\vec{\alpha})$ es el estado de $\mathcal{P}$ luego de t pasos equivale a decir que, luego de t pasos en la ejecución de $\mathcal{P}$, las variables $N_1, N_2, \ldots$ tienen valores $s_1, s_2, \ldots$, y las variables $P_1, P_2, \ldots$ tienen valores $\sigma_1, \sigma_2, \ldots$.

(2) Descripción instantánea

Una descripción instantánea es una 3-UPLA $(i,\vec{s},\vec{\sigma})$, con $i\in\omega, \vec{s}\in\omega^{[\mathbb{N}]}, \vec{\sigma}\in\Sigma^{*[\mathbb{N}]}$. Se habla en general de la descripción instantánea de un programa $\mathcal{P}$. Decir que $(i,\vec{s},\vec{\sigma})$ es la descripción instantánea de $\mathcal{P}$ equivale a decir que el estado de $\mathcal{P}$ es $(\vec{s},\vec{\sigma})$,y que está por ejecutarse la i-écima instrucción de $\mathcal{P}$. Cuando $i=0$ o $i>n(\mathcal{P})$, entendemos que la instrucción a realizarse es ε; es decir, no hacer nada.

(3) $S_{\mathcal{P}}$

$S_{\mathcal{P}}:\omega\times\omega^{[\mathbb{N}]}\times\Sigma^{*[\mathbb{N}]}\mapsto\omega\times\omega^{[\mathbb{N}]}\times\Sigma^{*[\mathbb{N}]}$ es una función que asocia una descripción instantánea $(i,\vec{s},\vec{\sigma})$ de $\mathcal{P}$ con la descripción instantánea $(j,\vec{s}',\vec{\sigma}')$ que describe al programa luego de ejecutar la i-écima instrucción desde el estado $(\vec{s},\vec{\sigma})$.

(4) Estado obtenido luego de t pasos partiendo del estado $(\vec{s},\vec{\sigma})$.

Dado un programa $\mathcal{P}\in Pro^{\Sigma}$, decimos que $(\vec{s}',\vec{\sigma}')$ es el estado obtenido luego de t pasos si y solo si

$$(j,\vec{s}',\vec{\sigma}')=\overbrace{S_{\mathcal{P}}(S_{\mathcal{P}}(\ldots(S_{\mathcal{P}}(1,\vec{s},\vec{\alpha})\ldots))}^{t\text{ veces}}$$

(5) $\mathcal{P}$ se detine luego de t pasos partiendo de $(\vec{s},\vec{\alpha})$.

Decimos que $\mathcal{P}$ se detiene luego de t pasos partiendo de $(\vec{s},\vec{\alpha})$ si y solo si la primera coordenada de

$$\overbrace{S_{\mathcal{P}}(S_{\mathcal{P}}\ldots S_{\mathcal{P}}}^{t\text{ veces}}(1,\vec{s},\vec{\alpha})\ldots)$$

es $n(\mathcal{P})+1$.

(1) $\Psi^{n,m,\#}_{\mathcal{P}}$

Dado un programa $\mathcal{P}$, la función $\Psi^{n,m,\#}_{\mathcal{P}}$: $\omega^n \times \Sigma^{*m} \mapsto \omega$ es tal que

$\Psi^{n,m,\#}_{\mathcal{P}}(\vec{x}, \vec{\alpha})$ = valor de N_1 en el estado final de $\mathcal{P}$ partiendo de $[[\vec{x}, \vec{\alpha}]]$

Es importante notar que

$\mathcal{D}_{\Psi^{n,m,\#}_{\mathcal{P}}} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : \mathcal{P}$ termina partiendo de $[[\vec{x}, \vec{\alpha}]]\}$

(2) f es Σ-computable

Decimos que una función Σ-mixta f es Σ-computable si y solo si existe un program $\mathcal{P}$ tal que $\Psi^{n,m,\#}_{\mathcal{P}} = f$.

(3) $\mathcal{P}$ computa a f

Si $\mathcal{P} \in Pro^{\Sigma}$ y f es Σ-mixta, decimos que $\mathcal{P}$ computa a f si $f = \Psi^{n,m,\#}_{\mathcal{P}}$.

(4) $M^{\leq}(P)$

Dado un predicado $P : \omega^n \times \Sigma^{*m} \times \Sigma^*$ y $\leq$ un orden sobre Σ, por definición

$$M^{\leq}(P) = \lambda \vec{x} \vec{\alpha} \left[\min^{\leq}_{\alpha} P(\vec{x}, \vec{\alpha}, \alpha) = 1 \right]$$

(1) Cuando $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-computable; cuando es Σ-enumerable; defina $\mathscr{P}$ enumera a S.

Decimos que $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-computable si y solo si $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-computable; es decir, existe un $\mathscr{P} \in Pro^{\Sigma}$ tal que $\chi_S^{\omega^n \times \Sigma^{*m}} = \Psi_{\mathscr{P}}^{n,m,\#}$.
Decimos que Σ es Σ-enumerable si es vacío o existe un programa existe una función Σ-mixta $F : \omega \mapsto \omega^n \times \Sigma^{*m}$ tal que

- $Im(F) = S$
- $F_{(i)}$ es Σ-computable para todo $1 \leq 1 \leq n + m$.

Decimos que $\mathscr{P}$ enumera a S si

- para cada $x \in \omega$, $\mathscr{P}$ se detiene partiendo de $[[x]]$ llegando a un estado $[[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]$ tal que $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) \in S$
- para cada $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) \in S$, existe un $x \in \omega$ tal que $\mathscr{P}$ se detiene partiendo de $[[x]]$ alcanzando el estado $[[x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m]]$.

10.2 Combo I

Problem 43 *Un conjunto es* Σ*-p.r. sii es el dominio de una función* Σ*-p.r. Hacer solo el caso de la composición.*

Lema a utilizar. Si f es una función Σ-mixta, existe una función Σ-total $\overline{f}$ que es Σ-p.r. y tal que $f = \overline{f}_{\mathcal{D}_f}$.
($\Rightarrow$) Asuma $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-p.r. Entonces $S = \mathcal{D}_{Pred \circ \chi_S^{\omega^n \times \Sigma^{*m}}}$.
($\Leftarrow$) Si $f = \emptyset$ no hay nada que probar. Tomemos el caso. El caso $f \in PR_0^\Sigma$ es trivial, y podemos asumir entonces que el postulado vale para PR_k^Σ. Sea $f \in PR_{k+1}^\Sigma$ de la forma $f = g \circ [g_1, \ldots, g_{n+m}]$, con

$$\begin{aligned} &\mathcal{D}_f : \omega^n \times \Sigma^{*m} \mapsto O \\ &g_i : \omega^a \times \Sigma^{*b} \mapsto \omega \qquad\qquad 1 \leq i \leq n \\ &g_i : \omega^a \times \Sigma^{*b} \mapsto \Sigma^* \qquad\qquad n+1 \leq i \leq m \end{aligned}$$

y $g, g_i \in PR_k^\Sigma$ para todo i.
Para cada g_i, sabemos que existe $\overline{g}_i$ Σ-p.r. y Σ-total tal que $g_i = \overline{g_i} \mid_{\mathcal{D}_{g_i}}$. Si $f = g \circ [g_1, \ldots, g_{n+m}]$, el dominio f es el conjunto

$$\mathcal{D}_f : \left\{(\vec{x}, \vec{\alpha}) \in \omega^a \times \Sigma^{*b} : (\vec{x}, \vec{\alpha}) \in \mathcal{D}_{g_i} \wedge g_i(\vec{x}, \vec{\alpha}) \in \mathcal{D}_g \text{ para toda } i\right\}$$

Por hipótesis inductiva, $\chi_{\mathcal{D}_g}^{\omega^n \times \Sigma^{*m}}$ es Σ-p.r. y $\chi_{\mathcal{D}_{g_i}}^{\omega^a \times \Sigma^{*b}}$ también para cada i. Más aún,

$$S = \bigcap_{i=1}^{n+m} \mathcal{D}_{g_i}$$

es Σ-p.r. Es fácil deducir de lo anterior que

$$\chi_{\mathcal{D}_f}^{\omega^a \times \Sigma^{*b}} = \left(\chi_{\mathcal{D}_g}^{\omega^n \times \Sigma^{*m}} \circ [\overline{g_1}, \ldots, \overline{g_{n+m}}] \wedge \chi_S^{\omega^a \times \Sigma^{*b}}\right)$$

Problem 44 *Si h is Σ-recursiva, entonces h es Σ-computable. (Solo el caso $R(f, \mathcal{G})$).*

Es fácil dar con programas que computen las funciones en R_0^Σ. Por lo tanto, asumimos que las funciones en PR_k^Σ son Σ-computables y probaremos que esto se cumple para PR_{k+1}^Σ. En particular, lo demostraremos para el caso en que $R \in PR_{k+1}^\Sigma$ es de la forma $R(f, \mathcal{G})$ con $f, \mathcal{G}_a \in PR_k^\Sigma$. Por hipótesis inductiva, f y $\mathcal{G}_a$ son Σ-computables (para cada $a \in \Sigma$).

Tenemos que $f \sim (n, m, *), \mathcal{G}_a \sim (n, m+2, *)$ para todo $a \in \Sigma$. Dada una evaluación $R(f, g)(\vec{x}, \vec{\alpha}, \alpha a)$, debemos dar con un programa que compute

$$\mathcal{G}_a\,(\vec{x}, \vec{\alpha}, \alpha, R(f, g)(\vec{x}, \vec{\alpha}, \alpha))$$

El programa debe ser tal que desde el estado $[[\vec{x}, \vec{\alpha}, \alpha]]$ deje en P_1 el valor de la función dada arriba. Es decir, partiendo de $[[\vec{x}, \vec{\alpha}, \alpha]]$, con $\alpha = \alpha_1 \ldots \alpha_n$, $\alpha_i \in \Sigma$, el programa deberá computar la recursión desde el caso base "hacia arriba", tal como esquematiza el siguiente procedimiento:

(1) Computar el caso base $\varphi_0 = f(\vec{x}, \vec{\alpha})$.
(2) Computar $\varphi_1 = \mathcal{G}_a(\vec{x}, \vec{\alpha}, \epsilon, \varphi_0)$ con a la palabra inicial de α.
(3) Computar $\varphi_2 = \mathcal{G}_a\,(\vec{x}, \vec{\alpha}, \alpha_1, \varphi_1)$
$\vdots$
(r+1) Computar y devolver $\varphi_r = \mathcal{G}(\vec{x}, \vec{\alpha}, \alpha_1 \ldots \alpha_{n-1}, \varphi_{r-1})$

Aquí se ve que los valores variables son φ_i y las subcadenas de α que van en cada evaluación. Guardaremos cada φ_i en $\overline{Pm+3}$ y cada subcadena en $\overline{Pm+2}$ (que adecuadamente empieza vacía). Entonces, si $\Sigma = \{a_1, \ldots, a_r\}$:

$$
\begin{array}{rl}
 & [P\overline{m+3} \leftarrow f(N1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
L\overline{r+1} & IF\ P\overline{m+1}\ BEGINS\ a_1\ GOTO\ L\overline{1} \\
 & \vdots \\
 & IF\ P\overline{m+1}\ BEGINS\ a_r\ GOTO\ L\overline{r+r} \\
 & GOTO\ L\overline{r1} \\
L\overline{1} & P\overline{m+1} \leftarrow {}^{\frown}P\overline{m+1} \\
 & \left[P\overline{m+3} \leftarrow \mathcal{G}_{a_1}\left(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m}, P\overline{m+2}, P\overline{m+3}\right)\right] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.a_1 \\
 & GOTO\ L\overline{r+1} \\
 & \vdots \\
L\overline{r} & P\overline{m+1} \leftarrow {}^{\frown}P\overline{m+1} \\
 & \left[P\overline{m+3} \leftarrow \mathcal{G}_{a_r}\left(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m}, P\overline{m+2}, P\overline{m+3}\right)\right] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.a_r \\
 & GOTO\ L\overline{r+1} \\
L\overline{r+r+1} & P_1 \leftarrow P\overline{m+3}
\end{array}
$$

10.3 Combo II

Problem 45 *Supongamos $f_i : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \Sigma^*$ con $i = 1, \ldots, k$. son funciones Σ-p.r con $\mathcal{D}_{f_i} \cap \mathcal{D}_{f_j} \neq \emptyset$ para cada $i \neq j$. Entonces $f_1 \cup \ldots \cup f_k$ es Σ-p.r.*

Sea $k = 2$. Existen $\overline{f_1}, \overline{f_2}$ Σ-p.r. y Σ-totales tal que f_1, f_2 son las restricciones de $\overline{f_1}, \overline{f_2}$ a sus respectivos dominios. Usaremos el hecho de que el dominio de una función Σ-p.r. es Σ-p.r. para construir la siguiente función:

$$H_f = \lambda x \alpha \, [\alpha^x] \circ \left[\chi_f^{\omega^n \times \Sigma^{*m}}, \overline{f} \right]$$

donde $f = \overline{f}_{\mathcal{D}_f}$ son Σ-p.r. Es claro que H_f es Σ-p.r. y que $\mathcal{D}_{H_f} = \omega^n \times \Sigma^{*m}$, y que

$$H_f(\vec{x}, \vec{\alpha}) = \begin{cases} \epsilon & (\vec{x}, \vec{\alpha}) \notin \mathcal{D}_f \\ \overline{f}(\vec{x}, \vec{\alpha}) & (\vec{x}, \vec{\alpha}) \in \mathcal{D}_f \end{cases}$$

Entonces vemos que

$$f_1 \cup f_2 = \left(\lambda \alpha \beta \, [\alpha \beta] \circ \left[H_{f_1}, H_{f_2} \right] \right)_{\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2}}$$

Si asumimos que el postulado vale para k arbitrario, entonces usamos el hecho de que $\bigcup_{i=1}^{k+1} f_i = \left(\bigcup_{i=1}^{k} f_i \right) \cup f_{k+1}$. Por HI, $F := \bigcup_{i=1}^{k} f_i$ es Σ-p.r. y por lo tanto H_F está bien definida. Luego

$$\bigcup_{i=1}^{k+1} f_i = F \cup f_{k+1} = \left(\lambda \alpha \beta \left[H_F, H_{f_{k+1}} \right] \right)_{\mathcal{D}_F \cup \mathcal{D}_{f_{k+1}}}$$

Problem 46 *Only give the idea of the following proof: Assume* $f : S \subseteq \omega^n \times \Sigma^{*m} \mapsto \Sigma^*$ *is* Σ*-Turing computable. Entonces* f *es* Σ*-recursiva. No es necesario probar que* $d \vdash^n d'$ *son* $(\Gamma \cup Q)$*-p.r. ni tampoco que P es* $(\Gamma \cup Q)$*-p.r.*

Teorema a recordar. Si Σ, Γ son alfabetos arbitrarios y f es Σ-mixta y Γ-mixta, entonces f es Σ-p.r. (o Σ-recursiva) sii es Γ-p.r. (o Γ-recursiva). Lo mismo aplica a conjuntos Σ o Γ-recursivos.

Lemmas a utilizar.

(1) Des es $(\Gamma \cup Q)$-p.r.; $St : Des \to Q$ es $(\Gamma \cup Q)$-p.r.
(2) $\lambda dd' [d \vdash d']$ es $(\Gamma \cup Q)$-p.r.
(3) Tanto $\lambda\alpha [\lfloor\alpha\rfloor]$ como $\lambda\alpha [^\frown\alpha] , \lambda\alpha [\alpha^\frown]$ son $(\Gamma \cup Q)$-p.r.

Un punto importante es comprender que nos manejaremos en el alfabeto $(\Gamma \cup Q)$ frecuentemente, pues debemos lidiar con descripciones instantáneas.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, i, F)$ una máquina de Turing determinística con unit que computa a f. La idea es dar un predicado $P : \omega \times \omega^n \times \Sigma^{*m}$ que sea $(\Gamma \cup Q)$-p.r. y equivalga a "M se detiene en $(x)_1$ pasos con input $(\vec{x}, \vec{\alpha})$ y con output $*^{\leq} ((x)_2)$". En el contexto de una máquina de Turing, este predicado es

$$(\exists q \in Q) \ \lfloor q_0 B i^{x_1} B \ldots B i^{x_n} B \alpha_1 B \ldots B \alpha_m \rfloor \vdash_M^{(x)_1} \left\lfloor qB *^{\leq} ((x)_2) \right\rfloor \wedge$$
$$(\forall d \in Des)_{|d| \leq |*^{\leq}((x)_2)|+2} \left\lfloor qB *^{\leq} ((x)_2) \right\rfloor \nvdash_M d$$

Este predicado es $(\Gamma \cup Q)$-p.r. La idea para probarlo sería primero observar que cada predicado en la conjunción tiene el mismo dominio. Segundamente, Q es finito y por lo tanto $(Q \cup \Gamma)$-p.r. y Des es $(Q \cup \Gamma)$-p.r. por lemma; es decir, que las cuantificaciones se hacen sobre conjuntos decidibles. Las funciones $*^{\leq}, \vdash_M, \vdash_M^n$, son $(\Gamma \cup Q)$-p.r. El hecho de que $(\exists q \in Q)$ no esté acotado no es un problema porque Q es finito; por lo tanto, puede recorrerse su espacio comprobando si algún q satisface el predicado.

Ahora que tenemos un predicado $P(x, \vec{x}, \vec{\alpha})$ equivalente a "M se detiene en $(x)_1$ pasos con output $*^{\leq}((x)_2)$", usamos el hecho de que M computa a f para observar que

si el predicado es verdadero, entonces $*^{\leq}((x)_2)$ es $f(\vec{x}, \vec{\alpha})$. Con esto, construimos una función recursiva que computa $f(\vec{x}, \vec{\alpha})$; a saber,

$$f = \lambda \vec{x} \vec{\alpha} \left[*^{\leq} \left((M(P))_2 \right) \right]$$

10.4 Combo III

Problem 47 *Si* $f : \mathcal{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto \Sigma^*$ *es* Σ*-computable, entonces es* Σ*-recursiva.*

Asuma que f es Σ-computable. Entonces existe un programa $\mathcal{P}_0 \in Pro^{\Sigma}$ que computa a f, es decir tal que

$$f = \Psi^{n,m,\#}_{\mathcal{P}_0}$$

Recordemos que $\Phi(\vec{x}, \vec{\alpha}, \mathcal{P}) := \Psi^{n,m,\#}_{\mathcal{P}}(\vec{x}, \vec{\alpha})$ es una función $(\Sigma \cup \Sigma_p)$-recursiva. Evidentemente,

$$f = \Phi^{n,m,\#}_{\mathcal{P}} \circ \left[p_1, \ldots, p_n, p_{n+1}, \ldots, p_{n+m}, C^{n,m}_{\mathcal{P}_0} \right]$$

donde $p_i, C^{n,m}_{\mathcal{P}^{n,m}_0}$ son dadas respecto al alfabeto $(\Sigma \cup S_p)$. Pues todas estas funciones son $(\Sigma \cup \Sigma_p)$-recursivas, f es $(\Sigma \cup \Sigma_p)$-recursiva. Por independencia del alfabeto, es Σ-recursiva.

Problem 48 *Sea Σ finito. Si $f : \omega \times S_1 \times \ldots \times S_n \times L_1 \times L_m \mapsto \omega$ es Σ-p.r. con $S_i \subseteq \omega$, $L_i \subseteq \Sigma^*$ no vacios, entonces*

$$F := \lambda xy\vec{x}\vec{\alpha}\left[\sum_{t=x}^{t=y} f(t, \vec{x}, \vec{y})\right]$$

es Σ-p.r.

Sea $G = \lambda tx\vec{x}\vec{\alpha}\left[\sum_{i=x}^{i=t} f(i, \vec{x}, \vec{\alpha})\right]$. Esta forma es deseable porque haremos recursión sobre el primer argumento t. Pero primero observemos que, dado que

$$F = G \circ \left[p_2^{n+2,m}, p_1^{n+2,m}, p_3^{n+2,m}, \ldots, p_{n+2+m}^{n+2,m}\right]$$

solo basta probar que G es Σ-p.r. La idea es dar con h, g tales que $G = R(h, g)$. Puesto que

$$G(0, x, \vec{x}, \vec{\alpha}) = \begin{cases} 0 & x > 0 \\ f(0, \vec{x}, \vec{\alpha}) & x = 0 \end{cases}$$

$$G(t+1, x, \vec{x}, \vec{\alpha}) = \begin{cases} 0 & x > t+1 \\ G(t, x, \vec{x}, \vec{\alpha}) + f(t+1, \vec{x}, \vec{\alpha}) & x \leq t+1 \end{cases}$$

Por supuesto, $G(t, x, \vec{x}, \vec{\alpha}) \in \omega$ y por lo tanto, si

$$h : \omega \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \mapsto \omega$$
$$(x, \vec{x}, \vec{\alpha}) \mapsto \begin{cases} 0 & x > 0 \\ f(0, \vec{x}, \vec{\alpha}) & x = 0 \end{cases}$$
$$g : \omega^3 \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \mapsto \omega$$
$$(A, t, x, \vec{x}, \vec{\alpha}) \mapsto \begin{cases} 0 & x > t+1 \\ A + f(t+1, \vec{x}, \vec{\alpha}) & x \leq t+1 \end{cases}$$

entonces $G = R(f, h)$. Solo nos queda probar que g, h son Σ-p.r.
(1) Sea $H_1 = \{(x, \vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : x > 0\}$ y $H_2 = \overline{H_1}$. Es claro que $h = C_0^{n+1,m} \mid_{H_1} \cup f \mid_{H_2}$. Que H_0, H_1 son Σ-p.r.

es trivial. Luego las restricciones en cuestión son Σ-p.r. Pues los dominios son disjuntos, la unión de las funciones restringidas es Σ-p.r. Luego h es Σ-p.r.
(2) Sea $D_1 = \left\{(A, t, x, \vec{x}, \vec{\alpha}) \in \omega^{n+3} \times \Sigma^{*m} : x > t + 1\right\}$ y $D_2 = \overline{D_1}$. Entonces

$$g = C_0^{n+3,m} \mid_{D_1} \cup \lambda A t x \vec{x} \vec{\alpha} \left[A + f(t + 1, \vec{x}, \vec{\alpha})\right] \mid_{D_2}$$

Es claro que

$$\lambda A t x \vec{x} \vec{\alpha} \left[A + f(t + 1, \vec{x}, \vec{\alpha})\right] = \lambda x y \left[x + y\right] \circ \left[p_1^{n+3,m}, f \circ \left[Suc \circ p_2^{n+3,m}, p_4^{n+3,m}, \ldots, p_{n+3+m}^{n+3,m}\right]\right.$$

Que D_1, D_2 son Σ-p.r. es trivial. Entonces g es Σ-p.r.

10.5 Combo V

Problem 49 *Sean $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ efectivamente enumerables. Entonces $S_1 \cap S_2$ es Σ-efectivamente enumerable.*

Pues S_1, S_2 son Σ-efectivamente enumerables, hay procedimientos efectivos $\mathbb{P}_1, \mathbb{P}_2$ que dado un input $x \in \omega$ devuelven un elemento de S_1, S_2, respectivamente. Podemos enumerar $S_1 \cap S_2$ de la siguiente manera.

Dado $\varphi \in S_1 \cap S_2$ arbitrario, el procedimiento efectivo es el siguiente: Dado un input $x \in \omega$,

> *(0)* Haga $\varrho = \varphi$.
> *(1)* Si $x = 0$ vaya a *(7)*; de otro modo, continúe.
> *(2)* Compute $(x)_1, (x)_2$.
> *(3)* Guarde en A el resultado de correr $\mathbb{P}_1$ con input $(x)_1$.
> *(4)* Guarde en B el resultado de correr $\mathbb{P}_2$ con input $(x)_2$
> *(5)* Si $A \neq B$ vaya a *(7)*; de otro modo, continúe.
> *(6)* Haga $\varrho = A$.
> *(7)* Devuelva ϱ.

Pues $\mathbb{P}_1, \mathbb{P}_2$ enumeran a S_1, S_2, y $((x)_1, (x)_2)$ recorren la totalidad del espacio ω^2, el programa enumera todos los pares posibles de elementos $\pi \in S_1, \zeta \in S_2$, y devuelve solo aquellos que están en ambos.

> **Note.** $((x)_1, (x)_2)$ da todos los pares posibles de valores porque todo $(a, b) \in \omega^2$ es alcanzado por el input $2^a 3^b$

Problem 50 *Sea* Σ *un alfabeto finito. Sea* $P : S \times S_1 \times \ldots S_n \times L_1 \times \ldots \times L_m \mapsto \omega$ *un predicado* Σ*-p.r. Asuma* $\overline{S} \subseteq S$ *es* Σ*-p.r. Entonces*

$$\lambda x \vec{x} \vec{\alpha} \left[(\forall t \in \overline{S})_{t \leq x} P(t, \vec{x}, \vec{\alpha}) \right]$$

es Σ*-p.r.*

Puesto que P es Σ-p.r. su dominio es Σ-p.r.

Un conjunto rectangular $S \times S_1 \times S_n \times L_1 \times \ldots \times L_m$ es Σ-p.r. sii cada S_i, L_i son Σ-p.r. Más aún, pues $\overline{S}$ es Σ-p.r. también lo es $\omega - \overline{S}$. Entonces

$$\begin{aligned} A &:= \overline{S} \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \\ B &:= (\omega - \overline{S}) \times S_1 \times \ldots \times S_n \times L_1 \times \ldots \times L_m \end{aligned}$$

son ambos Σ-p.r.

Definamos $\overline{P} = P \mid_A \cup C_1^{n+1,m} \mid_B$. Entonces $\overline{P}$ es claramente Σ-p.r. Observemos que

$$\begin{aligned} \lambda x \vec{x} \vec{\alpha} \left[(\forall t \in \overline{S})_{t \leq x} P(t, \vec{x}, \vec{\alpha}) \right] &= \lambda x \vec{x} \vec{\alpha} \left[\prod_{t=0}^{t=x} \overline{P}(x, \vec{x}, \vec{\alpha}) \right] \\ &= \lambda x y \vec{x} \vec{\alpha} \left[\prod_{t=x}^{t=y} \overline{P}(t, \vec{x}, \vec{\alpha}) \right] \circ \left[C_0^{n+1,m}, p_1^{n+1,m}, \ldots, p_{n+1+m}^{n+1,m} \right] \end{aligned}$$

Sabemos que esto es Σ-p.r.

Problem 51 *Si $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable, entonces S es Σ-efectivamente enumerable.*

Este es muy fácil. Se enumera $\omega^n \times \Sigma^{*m}$, que siempre es enumerable usando bajadas, y en cada enumeración se decide si el elemento pertenece a S o no. Si es así se lo devuelve.

Problem 52 *Solo probar* (2) $\Rightarrow$ (3). *Dado $S \subseteq \omega^n \times \Sigma^{*m}$, son equivalentes:*
(1) S es Σ-recursivamente enumerable.
(2) $S = I_F$ para alguna $F : \mathscr{D}_F \subseteq \omega^k \times \Sigma^{*l} \mapsto \omega^n \times \Sigma^{*m}$ tal que cada $F_{(i)}$ es Σ-recursiva.
(3) $S = \mathscr{D}_f$ para alguna f Σ-recursiva.

Asuma que $S = I_F$ para $F : \mathscr{D}_F \subseteq \omega^k \times \Sigma^{*l} \mapsto \omega^n \times \Sigma^{*m}$ con cada $F_{(i)}$ Σ-recursiva. Entonces existen programas $\mathscr{P}_1, \ldots, \mathscr{P}_k, \mathscr{P}_{k+1}, \ldots, \mathscr{P}_{k+l} \in Pro^{\Sigma}$ que computan cada $F_{(i)}$.
Haremos un programa que, dado un input $(\vec{x}, \vec{\alpha}) \in \omega^l \times \Sigma^{*k}$, termina si y solo si ese input está en S. Para esto, usaremos F para generar valores de S y compararlos con el input.
Definamos

$$H_i = \lambda t \vec{x} \vec{\alpha} \left[\neg Halt^{k,l}(t, \vec{x}, \vec{\alpha}, \mathscr{P}_i)\right]$$

Pues $Halt^{k,l}$ es $(\Sigma \cup \Sigma_p)$-p.r. y aquí solo estamos negando su composición con una constante, la función H_i es Σ-p.r.
Más aún, para $1 \leq i \leq l$ definimos

$$E_i = \lambda x t \vec{x} \vec{\alpha} \left[x \neq E_1^{k,l}(t, \vec{x}, \vec{\alpha}, \mathscr{P}_i)\right]$$

Y para $l + 1 \leq i \leq k$ definimos

$$E_i = \lambda x t \vec{x} \vec{\alpha} \left[x \neq E_{*1}^{k,l}(t, \vec{x}, \vec{\alpha}, \mathscr{P}_i)\right]$$

Como todas estas funciones son $(\Sigma \cup \Sigma_p)$-p.r. son Σ-p.r. y tienen macros asociados, que usaremos libremente. También usaremos macros para $\lambda x i\, [(x)_i]$ y $\lambda x i \left[*^{\leq}((x)_i)\right]$.
El programa, dado $[[x_1, \ldots, x_k, \alpha_1, \ldots, \alpha_l]]$, hará:

$$
\begin{array}{ll}
L_1 & N\overline{k+1} \leftarrow N\overline{k+1}+1 \\
& [N\overline{k+2} \leftarrow (N\overline{k+1})_1] \\
& \left[N\overline{k+3} \leftarrow (N\overline{k+1})_2\right] \\
& \vdots \\
& \left[N\overline{k+3+k-1} \leftarrow (N\overline{k+1})_{k+1}\right] \\
& \left[P\overline{l+1} \leftarrow *^{\leq}(N\overline{k+1})_{k+2}\right] \\
& \vdots \\
& \left[P\overline{l+1+l-1} \leftarrow (N\overline{k+1})_{k+l+1}\right] \\
& \left[IF\ \neg Halt^{k,l}(t, N\overline{k+3}, \ldots, N\overline{k+3+k-1}, P\overline{l+1}, \ldots, P\overline{l+1+l-1}, \mathscr{P}_1)\ GOTO\ L_1\right] \\
& \vdots \\
& \left[IF\ \neg Halt^{k,l}(t, N\overline{k+3}, \ldots, N\overline{k+3+k-1}, P\overline{l+1}, \ldots, P\overline{l+1+l-1}, \mathscr{P}_{k+l})\ GOTO\ L_1\right]
\end{array}
$$

10.6 Combo VII

Problem 53 *Sean $n, m \geq 0$. Sea $P : \mathcal{D}_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \mapsto \omega$ un predicado Σ-p.r. Entonces (1) $M(P)$ es Σ-recursiva y (2) Si existe $f : \omega^n \times \Sigma^{*m} \mapsto \omega$ Σ-p.r. tal que $M(P)(\vec{x}, \vec{\alpha}) = \min_t P(t, \vec{x}, \vec{\alpha}) \leq f(\vec{x}, \vec{\alpha})$ para toda $(\vec{x}, \vec{\alpha}) \in \mathcal{D}_{M(P)}$, entonces $M(P)$ es Σ-p.r.*

> **Idea.** Usar $\overline{P} = P \cup C_0^{n+1,m}$ para tener una función Σ-total t.q. $M(P) = M(\overline{P})$. Se demuestra que $M(\overline{P})$ es Σ-recursiva fácilmente. Expresar $M(P)(\vec{x}, \vec{\alpha}) = 1^0 \times 2^0 \times \ldots \times t^1$ con t el mínimo valor. Hacer esto usando un predicado en el exponente de cada valor t en una productoria. Se puede mostrar que esta productoria es Σ-p.r. al estar limitada por f.

(1) Sea $\overline{P} = P \cup C_0^{n+1,m} \mid_{\omega^{n+1} \times \Sigma^{*m} - \mathcal{D}_P}$. Evidentemente $\overline{P}$ es Σ-total y Σ-p.r. Observe que

$$\mathcal{D}_{M(\overline{P})} = \left\{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists t \in \omega)\ \overline{P}(t, \vec{x}, \vec{\alpha}) = 1\right\} = \mathcal{D}_{M(P)}$$

puesto que $P(t, \vec{x}, \vec{\alpha}) = 1 \iff \overline{P}(t, \vec{x}, \vec{\alpha}) = 1$. Además, $M(P)(\vec{x}, \vec{\alpha}) = M(\overline{P})(\vec{x}, \vec{\alpha})$. Luego $M(P) = M(\overline{P})$. Pues $\overline{P}$ es Σ-recursive y Σ-total, por definición pertenece a R^Σ. Pues recordemos que uno de los conjuntos en la unión de R_{k+1}^Σ es

$$\left\{M(P) : P \text{ es } \Sigma\text{-total y } P \in PR_k^\Sigma\right\}$$

(2) Pues $M(P) = M(\overline{P})$ basta probar que $M(\overline{P})$ es Σ-p.r. Y pues $M(P)(\vec{x}, \vec{\alpha}) \leq f(\vec{x}, \vec{\alpha})$, tenemos que

$$\chi_{\mathcal{D}_{M(\overline{P})}}^{\omega^n \times \Sigma^{*m}} = \left\{(\exists t \in \omega)_{t \leq f(\vec{x}, \vec{\alpha})}\ \overline{P}(t, \vec{x}, \vec{\alpha}) = 1\right\}$$

que es una composición de la cuantificación con f, que es Σ-p.r., sobre un conjunto Σ-p.r. (ω) y un predicado Σ-p.r. $(\overline{P})$. Luego el dominio de $M(\overline{P})$ es Σ-p.r.

Ahora bien, sea

$$P_1 = \lambda t \vec{x} \vec{\alpha} \left[\overline{P}(t, \vec{x}, \vec{\alpha}) \wedge (\forall j \in \omega)_{j \leq t} \; j = t \vee \overline{P}(j, \vec{x}, \vec{\alpha}) = 0 \right]$$

Es fácil ver que P_1 es Σ-total y Σ-p.r., y que

$$M(\overline{P})(\vec{x}, \vec{\alpha}) = t \iff (\vec{x}, \vec{\alpha}) \in \mathcal{D}(M(\overline{P})) \wedge P_1(t, \vec{x}, \vec{\alpha}) = 1$$

Entonces

$$M(\overline{P}) = \lambda \vec{x} \vec{\alpha} \left(\left[\prod_{i=0}^{f(\vec{x}, \vec{\alpha})} t^{P_1(t, \vec{x}, \vec{\alpha})} \right] \right)_{\mathcal{D}_{M(\overline{P})}}$$

Problem 54 *Si $f : \mathscr{D}_f \subseteq \omega^n \times \Sigma^{*m} \mapsto O$ es Σ-recursiva y $S \subseteq \mathscr{D}_f$ es Σ-r.e. entonces $f\mid_S$ es Σ-recursiva.*

> **Idea.** Dar un programa $\mathscr{P}$ que enumere a S, y solo compute f cuando la enumeración coincida con el input.

Pues S es Σ-r.e. hay una $F : \omega \mapsto \omega^n \times \Sigma^{*m}$ tal que $Im(F) = S$ y $F_{(1)}, \ldots, F_{(n+m)}$ son Σ-recursivas y por lo tanto Σ-computables. Haremos uso de los macros de cada una de estas $F_{(i)}$. Entonces

$$
\begin{array}{ll}
L_1 & [N\overline{n+1} \leftarrow F_1(N\overline{n+m+1})] \\
 & \vdots \\
 & [N\overline{n+n} \leftarrow F_n(N\overline{n+m+1})] \\
 & [P\overline{m+1} \leftarrow F_{n+1}(N\overline{n+m+1})] \\
 & \vdots \\
 & [P\overline{n+m} \leftarrow F_{n+m}(N\overline{n+m+1})] \\
 & [IF\ N_1 \neq N\overline{n+1}\ GOTO\ L_2] \\
 & \vdots \\
 & [IF\ N\overline{n} \neq N\overline{n+n}\ GOTO\ L_2] \\
 & [IF\ P_1 \neq P\overline{m+1}\ GOTO\ L_2] \\
 & \vdots \\
 & [IF\ P\overline{m} \neq P\overline{n+m}\ GOTO\ L_2] \\
 & GOTO\ L_3 \\
L_2 & N\overline{n+m+1} \leftarrow N\overline{n+m+1} + 1 \\
 & GOTO\ L_1 \\
L_3 & [N_1 \leftarrow f(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})]
\end{array}
$$

10.7 Combo XVIII

Problem 55 *Sea* $\Sigma \supseteq \Sigma_p$. *Entonces* $AutoHalt^{\Sigma}$ *no es* Σ*-recursivo.*

Recuerde que $AutoHalt^{\Sigma} = \lambda \mathscr{P} \left[(\exists t \in \omega)\ Halt^{0,1}(t, \mathscr{P}, \mathscr{P}) \right]$. El dominio es Pro^{Σ} y $AutoHalt^{\Sigma}(\mathscr{P}) \iff \mathscr{P}$ termina partiendo de $[[\mathscr{P}]]$.
Suponga que $AutoHalt^{\Sigma}$ es Σ-recursivo. Entonces hay un macro

$$[IF\ AutoHalt^{\Sigma}(W_1)\ GOTO\ A_1]$$

Sea $\mathscr{P}_0$ el siguiente programa:

$$A_1 \quad [IF\ AutoHalt^{\Sigma}(P_1)\ GOTO\ A_1]$$

Dado input $[[\mathscr{P}_0]]$, si $\mathscr{P}_0$ termina partiendo de sí mismo, entonces no termina; y si no termina partiendo de sí mismo, entonces termina. La contradicción surge de asumir que $AutoHalt^{\Sigma}$ es Σ-recursivo.

Problem 56 *Suponga* $\Sigma \supseteq \Sigma_p$. *Entonces* $AutoHalt^{\Sigma}$ *no es* Σ*-efectivamente computable.*

Si fuera Σ-efectivamente computable, la tesis de Church nos diría que es Σ-computable, lo cual contradice el resultado anterior.

Problem 57 *Suponga* $\Sigma \supseteq \Sigma_p$. *Entonces*

$$A = \left\{ P \in Pro^{\Sigma} : AutoHalt^{\Sigma}(\mathscr{P}) = 1 \right\}$$

es Σ*-r.e. y no es* Σ*-recursivo, y*

$$N = \left\{ \mathscr{P} \in Pro^{\Sigma} : AutoHalt^{\Sigma}(\mathscr{P})(0) \right\}$$

no es Σ*-r.e.*

(1) Usamos el resultado que establece que un conjunto es Σ-recursivamente enumerable si es el dominio de una función Σ-recursiva. Considere

$$P = \lambda t \mathcal{P} \left[Halt^{0,1}(t, \mathcal{P}, \mathcal{P}) \right]$$

Pues $Halt^{n,m}$ es Σ-p.r. tenemos que P es Σ-p.r. y por lo tanto $M(P)$ es Σ-recursiva. Pero el dominio de $M(P)$ es A. ■

(2) Asuma que N es Σ-r.e. Observe que

$$AutoHalt^{\Sigma} = C_1^{0,1} \mid_A \cup C_0^{0,1} \mid_N$$

donde cada función en la unión es Σ-recursiva por ser restricción de función Σ-recursiva a un conjunto Σ-r.e. Pero entonces $AutoHalt^{\Sigma}$ es Σ-recursiva, por lema de división por casos, lo cual contradice el primer resultado. Luego N no es Σ-r.e.

(3) Solo queda probar que A no es Σ-recursivo. Pero si A fuera Σ-recursivo, entonces $(\Sigma^* - A) \cap Pro^{\Sigma}$ debería serlo. Pero este conjunto es N, que no puede ser Σ-recursivo pues sería enumerable.

Problem 58 (Neumann vence a Godel) *Si h es Σ-recursiva entonces h es Σ-computable (Solo el caso $h = M(P)$).*

Probaremos que si h es Σ-computable pertenece a algún R_k^{Σ} haciendo inducción en k. El caso $k = 0$ puede hacerse fácilmente. Asumimos que el postulado vale para un k arbitrario y mostramos que vale para $k + 1$.

Asuma $h \in R_{k+1}^{\Sigma} - R_k^{\Sigma}$ y $h = M(P)$, con $P : \omega \times \omega^n \times \Sigma^{*m} \mapsto \omega$. Pues P es por definición de R_k^{Σ} un predicado Σ-recursivo, es Σ-computable y tiene un macro asociado. Entonces el programa

$$\begin{array}{ll} & A_1[IF\ P(N\overline{n+1}, N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})\ GOTO\ A_2] \\ & \quad N\overline{n+1} \leftarrow N\overline{n+1} + 1 \\ & \quad GOTO\ A_1 \\ A_2 & N_1 \leftarrow N\overline{n+1} \end{array}$$

Evidentemente el programa computa a $h = M(P)$.

10.8 Combo IX

Problem 59 *Suponga* $f_i : \mathcal{D}_{f_i} \subseteq \omega^n \times \Sigma^{*m} \mapsto O$, *con* $i = 1, \ldots, k$, *son funciones* Σ*-recursivas tales que* $D_{f_i} \cap D_{f_j} = \emptyset$ *si* $i \neq j$. *Entonces* $f_1 \cup \ldots \cup f_k$ *es* Σ*-recursiva.*

Sean $\mathcal{P}_1, \ldots, \mathcal{P}_k \in Pro^{\Sigma}$ los programas que computan $f_1, \ldots, f_k$, respectivamente. Sabemos que $H_i := \lambda t \vec{x} \vec{\alpha} \, [Halt^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}_i)]$ es $(\Sigma \cup \Sigma_p)$-p.r. y por lo tanto Σ-p.r. Entonces, asumiendo $O = \omega$, el programa

$$
\begin{array}{ll}
L_1 & N\overline{n+1} \leftarrow N+1 \\
 & [IF\ H_1(N\overline{n+1}, N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})\ GOTO\ L2] \\
 & \vdots \\
 & [IF\ H_k(N\overline{n+1}, N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})\ GOTO\ L\overline{2+k-1}] \\
L_2 & [N_1 \leftarrow f_1(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
 & GOTO\ L\overline{2+k} \\
 & \vdots \\
L_{\overline{2+k-1}} & [N_1 \leftarrow f_k(N_1, \ldots, N\overline{n}, P_1, \ldots, P\overline{m})] \\
 & GOTO\ L\overline{2+k} \\
L\overline{2+k} & SKIP
\end{array}
$$

computa $\bigcup f_i$. Para el caso $O = \Sigma^*$ solo se debe cambiar $N_1 \leftarrow f_i(\ldots)$ etc. por $P_1 \leftarrow f_i(\ldots)$ etc.

Problem 60 $Halt^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-p.r.*

Recordemos que $Halt^{n,m} = \lambda t \vec{x} \vec{\alpha} \mathcal{P} \, [i^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}) = n(\mathcal{P}) + 1]$. Recordemos además que $i^{n,m}$ y $\lambda \mathcal{P} \, [n(\mathcal{P})]$ son $(\Sigma \cup \Sigma_p)$-p.r. Entonces

$$
Halt^{n,m} = \lambda xy \, [x = y] \circ \left[i^{n,m} \circ \left[p_1^{n+1,m+1}, \ldots, p_{n+m+2}^{n+1,m+1} \right], Suc \circ \left[n \circ p_{n+m+2}^{n+1,m+1} \right] \right]
$$

Por independencia del alfabeto, es Σ-p.r.

Problem 61 $T^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-recursiva.*

Es un resultado teórico que si P es un predicado Σ-p.r. entonces $M(P)$ es Σ-recursivo.

11 Finales

11.1 Final 2020-02

Problem 62 *Let* $\Sigma = \{\sim, @\}$ *and*

$$L = \left\{ \mathscr{P} \in Pro^{\Sigma} : \exists \alpha \in \Sigma^* : \Psi^{1,1,\#}_{\mathscr{P}}(5, \alpha @) = |\mathscr{P}| \right\}$$

Provide a program $\mathbb{Q} \in Pro^{\Sigma \cup \Sigma_p}$ *that enumerates* L.

Observe that $\varphi := N1 \leftarrow N1 \in L$, since $|N1 \leftarrow N1| = 5$ and hence $\Psi^{1,1,\#}_{\varphi}(5, \alpha) = 5 = |\varphi|$ for all $\alpha \in \Sigma^*$.
Let $\mathscr{F}_{\Sigma \cup \Sigma_p}$ be the function which enumerates $(\Sigma \cup \Sigma_p)^*$ (which is enumerable by virtue of being $(\Sigma \cup \Sigma_p)$-total) and $\mathscr{F}_{\Sigma}$ that which enumerates Σ^* (enumerable by the same reason).
The desired program is then

$$\begin{array}{rl}
 & IF\ N_1 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_{666} \\
L_1 & [N_2 \leftarrow (N_1)_1] \\
 & [N_3 \leftarrow 5] \\
 & [N_4 \leftarrow (N_1)_2] \\
 & [N_5 \leftarrow (N_1)_3] \\
 & [P_1 \leftarrow \mathscr{F}_{\Sigma \cup \Sigma_p}(N_5)] \\
 & [P_2 \leftarrow \mathscr{F}_{\Sigma}(N_4)] \\
 & P_2 \leftarrow P_2.@ \\
 & [IF\ P_1 \in Pro^{\Sigma}\ GOTO\ L_2] \\
 & GOTO\ L_{666} \\
L_2 & [IF\ Halt^{1,1}(N_2, N_3, P_2, P_1)\ GOTO\ L_3] \\
 & GOTO\ L_{666} \\
L_3 & [N_6 \leftarrow |P_1|] \\
 & [IF\ E^{1,1}_{*1}(N_2, N_3, P_2, P_1) = N_6\ GOTO\ L_4] \\
 & GOTO\ L_{666} \\
L_{666} & [P_1 \leftarrow \varphi] \\
L_4 & SKIP
\end{array}$$

Problem 63 *Let* $\Sigma = \{@, *\}$ *and* $S = \{(x, \alpha) \in \omega \times \Sigma^* : [\alpha]_1 = *\}$. *Prove that the function*

$$F(x,\alpha) = \begin{cases} \prod_{i=x}^{i=x^2} x^i & x \leq |\alpha| \\ |\alpha|^2 & otherwise \end{cases}$$

is Σ*-p.r.*

We shall use division by cases. First, observe that

$$\mathscr{D}_f = \{(x,\alpha) \in \omega \times \Sigma^* : x \leq |\alpha|\} \cup \{(x,\alpha) \in \omega \times \Sigma^* : x > |\alpha|\}$$

Call each term in the union of $\mathscr{D}_f$ S_1, S_2 (left to right).
It is evident that $S_1 \cap S_2 = \emptyset$.
(1) Let $F_1 := \lambda x\alpha \left[\prod_{i=x}^{i=x^2} x^i\right]$. It is easy to observe that

$$F_1 = \lambda abx\alpha \left[\prod_{i=a}^{i=b} x^i\right] \circ \left[p_1^{1,1}, \lambda x \left[x^2\right] \circ p_1^{1,1}, p_1^{1,1}, p_1^{1,1}, p_2^{1,1}\right]$$

The general productorial $\lambda abx\alpha \left[\prod_{i=a}^{i=b} x^i\right]$ is Σ-p.r. if and only if $\lambda xi \left[x^i\right]$ is Σ-p.r. We know that exponentiation is Σ-p.r. Then the general productorial is Σ-p.r. Then F_1 is Σ-p.r.
(2) It is trivial to observe that $F_2 = \lambda x\alpha \left[|\alpha|^2\right]$ is Σ-p.r. Simply observe that

$$\lambda x\alpha[|\alpha|^2] = \lambda xy\,[x^y] \circ \left[\lambda\alpha\,[|\alpha|] \circ p_2^{1,1}, p_1^{1,1}\right]$$

(3) Both S_1, S_2 are Σ-recursive. This is simple to prove. Simply observe that $\chi_{S_1}^{\omega\times\Sigma^*} = \lambda x\alpha\,[x \leq |\alpha|] = \lambda xy\,[x < y] \circ \left[p_1^{1,1}, \lambda\alpha\,[|\alpha|] \circ p_2^{1,1}\right]$ which is evidently Σ-p.r. That $\chi_{S_2}^{\omega\times\Sigma^*}$ is Σ-p.r. is proven similarly.
(4) It is a theorem that a Σ-p.r. function restricted to a Σ-p.r. set is Σ-p.r. Then $F_{1|S_1}, F_{2|S_2}$ are Σ-p.r. It is obvious that $F = F_{1|S_1} \cup F_{2|S_2}$.
$\therefore$ By case division, F is Σ-p.r.

11.2 Final 2022-07

Problem 64 *Let* $\Sigma = \{@, !\}$ *and*

$$L = \left\{ \mathscr{P} \in Pro^{\Sigma} : @@ \in Im\left(\Psi^{2,1,*}_{\mathscr{P}}\right) \right\}$$

Find a program that belongs to L *and then give* $\mathbb{Q} \in Pro^{\Sigma\cup\Sigma_p}$ *s.t.* $\mathscr{D}_{\mathbb{Q}} = \omega$ *and* $Im(\mathbb{Q}) = L$.

(1) $SKIP \in L$ since $\Psi^{2,1,*}_{SKIP}(x, y, @@) = @@$ for any $x, y \in \omega$.
(2) $Pro^{\Sigma\cup\Sigma_p}$ es Σ-p.r. in accordance to a lemma. Then it is Σ-computable and there is a macro $[IF\ \chi^{(\Sigma\cup\Sigma_p)^*}_{Pro^{\Sigma\cup\Sigma_p}}(V_1)\ GOTO\ A_1]$, which we will call $\left[IF\ V_1 \in Pro^{\Sigma\cup\Sigma_p}\ GOTO\ A_1\right]$ for simplicity.
(3) We know by theory that $\lambda xi\,[(x)_i]$, $\lambda x\left[*^{\leq}(x)\right]$ are Σ-recursive. By alphabet independence they are then $(\Sigma \cup \Sigma_p)$-recursive. The same is true of $Halt^{2,1}$ and $E^{2,1}_{j*}$.
Since $E^{2,1}_{j*}$ is $(\Sigma\cup\Sigma_p)$-p.r., $\lambda txy\alpha\mathscr{P}\beta\left[E^{2,1}_{*1}(t, x, y, \alpha, \mathscr{P}) = \beta\right]$ is $(\Sigma \cup \Sigma_p)$-p.r. (trivial to show).
(4) Let $\mathbb{Q}$ be

$$\begin{array}{rl}
 & IF\ N_1 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_{666} \\
L_1 & [N_2 \leftarrow (N_1)_1] \\
 & [N_3 \leftarrow (N_1)_2] \\
 & [N_4 \leftarrow (N_1)_3] \\
 & [P_2 \leftarrow *^{\leq}((N_1)_4)] \\
 & [P_3 \leftarrow *^{\leq}((N_1)_5)] \\
 & [IF\ P_3 \in Pro^{\Sigma\cup\Sigma_p}\ GOTO\ L_2] \\
 & GOTO\ L_{666} \\
L_2 & [IF\ Halt^{2,1}(N_2, N_3, N_4, P_2, P_3)\ GOTO\ L_3] \\
 & GOTO\ L^{666} \\
L_3 & [IF\ E^{2,1}_{*1}(N_2, N_3, N_4, P_2, P_3) = @@\ GOTO\ L_4] \\
L_{666} & P_1 \leftarrow SKIP \\
 & GOTO\ L_{10} \\
L_4 & P_1 \leftarrow P_3 \\
L_{100} & SKIP
\end{array}$$

Observe that here $*^{\leq}$ is defined over $(\Sigma \cup \Sigma_p)^*$. Starting from $[[x]]$ the program obtains the $(x)_4$th and $(x)_5$th words from $(\Sigma \cup \Sigma_p)^*$ and values $(x)_1, (x)_2, (x)_3$. If the $(x)_5$th word is a program, it determines whether it halts in $(x)_1$ steps with input $(x)_2$, $(x)_3$ and $*^{\leq}((x)_4)$. If it halts it checks whether the output is @@. If it is it returns the program obtained. If any of the checks fail, it returns $SKIP$.

Problem 65 *Let* $\Sigma = \{@, !\}$. *Show that*

$$S = \left\{(x, y, \alpha, \beta) \in \omega^2 \times \Sigma^{*2} : x \geq 5 \wedge (\exists \gamma \in \Sigma^*)\ @b = \gamma^y\right\}$$

is Σ*-p.r.*

Observe that

$$\chi_S^{\omega^2 \times \Sigma^{*2}}(x, y, \alpha, \beta) = \lambda xy\alpha\beta\,[x \geq 5] \wedge \lambda xy\alpha\beta\,[(\exists \in \Sigma^*)@b = \gamma^y]$$

Consider $F = \lambda xy\alpha\beta\,[(\exists \gamma \in \Sigma^*)\ @b = \gamma^y]$. It is evident that any $\gamma \in \Sigma^*$ s.t. $@b = \gamma^y$ for some $y \in \omega$ is s.t. $|\gamma| = 1$.

Problem 66 *Sea $\mathscr{P}_0$ un programa fijo y $f : \omega \mapsto \omega$ una función. Asuma que para toda $x \in \omega$ el programa $\mathscr{P}_0$ termina partiendo de $[[x]]$. Más aún, asuma que termina en $t \leq f(x)$ pasos. Demuestre que $\Psi^{1,0,\#}_{\mathscr{P}_0}$ es Σ-p.r.*

No veo como encarar el problema sin asumir algún tipo de computablidad para f. Vamos a ver cómo se ve el problema *(1)* asumiendo que f es Σ-p.r. y *(2)* asumiendo que es Σ-recursiva.
Sea g la función computada por $\mathscr{P}_0$. Considere el conjunto

$$G = \{x \in \omega : (x, g(x))\}$$

(1) Asuma f es Σ-p.r. Por las definiciones de $\Psi^{1,0,\#}_{\mathscr{P}}$, $E^{1,0}_{\#1}$, tenemos que

$$\Psi^{1,0,\#}_{\mathscr{P}_0} = E^{1,0}_{\#1} \circ \left[f \circ p^{1,0}_1, p^{1,0}_1, C^{1,0}_{\mathscr{P}_0} \right]$$

Esto se sigue de que, al saber que $\mathscr{P}_0$ desde $[[x]]$ terminará en $t \leq f(x)$ pasos, el estado de $\mathscr{P}_0$ al terminar es el mismo que el estado de $\mathscr{P}_0$ tras $f(x)$ pasos. Ahora bien, $E^{1,0}_{\#j}$, las proyecciones y la función constante, son $(\Sigma \cup \Sigma_p)$-p.r. Luego, por independencia del alfabeto, son Σ-p.r. Como f es Σ-p.r. tenemos que $\Psi^{1,0,\#}_{\mathscr{P}_0}$ es Σ-p.r.

(2) Asuma que f es Σ-recursiva. Se sigue que es o bien Σ-p.r. o bien $f = M(P)$ para algún predicado $P : \omega^2 \mapsto \omega$ que es Σ-recursivo. Si fuera Σ-p.r. la demostración del punto *(1)* resuelve el problema. Asumamos que $f = M(P_0)$ para algún predicado $P_0 : \omega^2 \mapsto \omega$, donde P_0 es Σ-recursivo.
Como f es Σ-recursiva, es Σ-computable y tiene un macro $[V1 \leftarrow f(V2)]$ asociado.

$$[N_2 \leftarrow f(N_1)]$$
$$[N_1 \leftarrow E_1^{1,0}(N_2, N_1, \mathscr{P}_0)]$$

Problem 67 *Let*

$$L = \left\{ \mathscr{P} \in Pro^{\Sigma_p} : (\exists n \in \mathbb{N})\ \Psi_{\mathscr{P}}^{1,1,*}(x, EBE) = EBE \right\}$$

Find a program $Q \in Pro^{\Sigma_p}$ that enumerates L.

Observe that $SKIP \in L$. The program in question will make effective the following procedure with input $x \in \omega$:

(1) If $x = 0$ go to *(7)*.
(2) Compute $(x)_1, (x)_2, (x)_3, (x)_4$.
(3) Let $\alpha = *^{\leq}((x)_4), \beta = *^{\leq}((x)_3)$.
(4) If $\alpha \notin Pro^{\Sigma_p}$ go to *(7)*.
(5) Evaluate whether α halts in $(x)_1$ steps with input $(x)_2, \beta$. If it does not, go to *(7)*.
(6) Evaluate whether the output of α after $(x)_1$ steps with input $(x)_2, \beta$ is EBE. If it is, go to *(8)*.
(7) Return $SKIP$ and stop.
(8) Return α and stop.

The program is

$$\begin{array}{rl}
 & IF\ N_1 \neq 0\ GOTO\ L_1 \\
 & GOTO\ L_{666} \\
L_1 & [N_2 \leftarrow (N_1)_1] \\
 & [N_3 \leftarrow (N_1)_2] \\
 & [N_4 \leftarrow (N_1)_3] \\
 & [N_5 \leftarrow (N_1)_4] \\
 & [P_1 \leftarrow *^{\leq}(N_5)] \\
 & [P_2 \leftarrow *^{\leq}(N_4)] \\
 & [IF\ P_1 \in Pro^{\Sigma_p}\ GOTO\ L_2] \\
 & GOTO\ L_{666} \\
L_2 & [IF\ Halt^{1,1}(N_2, N_3, P_2, P_1)\ GOTO\ L_3] \\
 & GOTO\ L_{666} \\
L_3 & [IF\ E_*^{1,1}(N_2, N_3, P_2, P_1) = EBE\ GOTO\ L_4] \\
L_{666} & P_1 \leftarrow SKIP \\
L_4 & SKIP
\end{array}$$

The macros we use correspond to $(x)_i$ the ith prime function, which is Σ-p.r. We $Halt^{n,m}$ function is Σ^* p.r. etc.

Problem 68 *Let* $\Sigma = \{@, !\}$ *and*

$$P : \{x \in \omega : x \; odd \} \times \Sigma^* \mapsto \omega$$
$$(x, \alpha) \mapsto \begin{cases} x & x \geq 3 \\ |\alpha| & otherwise \end{cases}$$

Prove that P *is* Σ*-p.r.*

(1) Let $f_1 := \lambda x \alpha \, [x] \, , f_2 : \lambda x \alpha \, [|\alpha|]$. Both functions are Σ-p.r.
(2) The set of tuples (x, α) with x odd is Σ-p.r. Simply observe that if S denotes that set, $\chi_S^{\omega \times \Sigma^*} = \lambda x \alpha \, [\neg \, x \text{ is even}]$, which is Σ-p.r.
(3) Let $S' = \{(x, \alpha) \in S : x \geq 3\}$. This is trivially a Σ-p.r. set. Then it is clear that

$$P = f_{1 \, | S'} \cup f_{2 | \overline{S'}}$$

Then P is a union of Σ-p.r. functions with disjoint domains restricted to Σ-p.r. sets. Then P is Σ-p.r.

Problem 69 *True or false?*

Problem 70 *Let* $\Sigma = \{\%, !\}$ *and*

$$S = \{(x, x+1, x+2, \%\%!!) : x \in \omega\}$$

Give a program that enumerates S without using macros.

Since the fourth element in the tuples of S is constantly %%!! and all other elements are functions of a single natural number $x \in \omega$, we enumerate S with the following program $\mathcal{P}$:

$$\begin{aligned}
P_1 &\leftarrow P_1.\% \\
P_1 &\leftarrow P_1.\% \\
P_1 &\leftarrow P_1.! \\
P_1 &\leftarrow P_1.! \\
N_2 &\leftarrow N_1 + 1 \\
N_3 &\leftarrow N_2 + 1
\end{aligned}$$

Then, for any $x \in \omega$, $\mathcal{P}$ starting from $[[x]]$ halts at a state $[[x, x+1, x+2, \%\%!!]]$. Furthermore, for any $(x, x+1, x+2, \%\%!!) \in S$, running the program with input x leads to a final state $[[x, x+1, x+2, \%\%!!]]$. Then condition *(2)* of the previous proposition holds. Then $\mathcal{P}$ enumerates S.

Problem 71 *Sea Σ finito y tal que $\Sigma_p \subseteq \Sigma$, y sean $n, m, l, k \in \omega$. Existe una función Σ-p.r. $S_{n,l,m,k} : \omega^l \times \Sigma^{*k} \times Pro^{\Sigma} \mapsto Pro^{\Sigma}$ tal que*

$$\mathbf{\Phi}(\vec{x}, \vec{y}, \vec{\alpha}, \vec{\beta}, \mathcal{P}) \approx \mathbf{\Phi}\left(\vec{x}, \vec{\alpha}, S_{n,l,m,k}(\vec{y}, \vec{\beta}, \mathcal{P})\right)$$

Parte 1. Sea $\Sigma = \{a_1, \ldots, a_r\}$. Definamos $\mathbb{Q}$ una familia Σ-indexada de funciones con dominio $\mathbb{N}$ y tal que para todo $i \in \mathbb{N}$, $\mathbb{Q}_i : \omega \mapsto Pro^{\Sigma}$ se define como

$$\mathbb{Q}_i := \lambda\alpha\beta\,[\alpha\beta] \circ \left[C^{1,0}_{N\bar{i}\leftarrow 0}, \lambda x\alpha\,[\alpha^x] \circ \left[p^{1,0}_1, C^{1,0}_{N\bar{i}\leftarrow N\bar{i}+1}\right]\right]$$

Evidentemente, para toda $i \in \mathbb{N}$ tenemos que $\mathbb{Q}_i$ es Σ-p.r. y es tal que $\mathbb{Q}_i(x)$ concatena $N\bar{i} \leftarrow 0$ con x veces $N\bar{i} \leftarrow N\bar{i}+1$. Es decir, $\mathbb{Q}_i$ devuelve un programa que guarda en $N\bar{i}$ el valor x.

Definamos también $\mathcal{R}$ una familia Σ-indexada de funciones con dominio $\mathbb{N}$ y tal que $\mathcal{R}_i : \Sigma^* \mapsto Pro^{\Sigma}$ se define como $\mathcal{R}_i = R(f, \mathcal{G})$ para

$$\begin{aligned} f &:= C^{0,0}_{SKIP} \\ \mathcal{G}_a &:= \lambda\alpha\beta\,[\alpha\beta] \circ \left[p_2, {}^{0,2}, d_a \circ C^{0,2}_{P\bar{i}\leftarrow P\bar{i}.}\right] \end{aligned}$$

Es claro que, para todo i, $\mathcal{R}_i$ es Σ-p.r. y que $\mathcal{R}_i(\alpha)$ devuelve un programa que guarda en $P\bar{i}$ la palabra α.

> **Nota.** Para dar un ejemplo (y chequear!), hacemos la computación de $\mathcal{R}_3(word)$ sobre $\Sigma = \{w, o, r, d\}$:
>
> $$\begin{aligned} \mathcal{R}_3(word) &= \mathcal{G}_d(wor, R(f, \mathcal{G})(wor)) \\ &= R(f, \mathcal{G})(wor)(P_3 \leftarrow P_3.d) \\ &\vdots \\ &SKIP(P_3 \leftarrow P_3.w)(P_3 \leftarrow P_3.o)(P_3 \leftarrow P_3.r)(P_3 \leftarrow P_3.d) \end{aligned}$$
>
> que es lo que esperábamos.

El resultado de esta primera parte es que contamos con $\mathbb{Q}, \mathcal{R}$ familias Σ-indexadas de funciones. Las funciones de estas familias devuelven programas que agregan un valor arbitrario a una variable numérica o alfabética, respectivamente. Además, son todas Σ-p.r.

Parte 2. Dados $n, l, m, k \in \omega$, definimos $S_{n,l,m,k} : \omega^l \times \Sigma^{*k} \mapsto Pro^{\Sigma}$ de la siguiente manera:

$$S_{n,l,m,k}(\vec{y}, \vec{\beta}, \mathcal{P}) = \mathbb{Q}_{n+1}(y_1) \dots \mathbb{Q}_{n+l}(y_l)\mathcal{R}_{m+1}(\beta_1) \dots \mathcal{R}_{m+k}(\beta_k)\mathcal{P}$$

Cada $\mathbb{Q}_i, \mathcal{R}_i$ son Σ-p.r. y sus evaluaciones son palabras; la concatenación de palabras es Σ-p.r. Luego $S_{n,l,m,k}$ es Σ-p.r. (¿Debería darle más rigor a esto?)

Entonces, veamos que $\Phi_{\#}^{n,m}\left(\vec{x}, \vec{\alpha}, S_{n,l,m,k}(\vec{y}, \vec{\beta}, \mathcal{P})\right)$ "ejecuta" el programa

$$\mathbb{Q}_{n+1}(y_1) \dots \mathbb{Q}_{n+l}(y_l)\mathcal{R}_{m+1}(\beta_1) \dots \mathcal{R}_{m+k}(\beta_k)\mathcal{P}$$

partiendo del estado

$$[[x_1, \dots, x_n, \alpha_1, \dots, \alpha_m]]$$

Eventualmente, tras las ejecuciones de cada $\mathbb{Q}_i, \mathcal{R}_i$, la computación conducirá a un estado

$$[[x_1, \dots, x_n, y_1, \dots, y_l, \alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_k]]$$

y solo entonces comenzará la computación de $\mathcal{P}$ (desde ese estado). Pues computará a $\mathcal{P}$ desde ese estado, es claro que

$$\Phi_{\#}^{n,m}\left(\vec{x}, \vec{\alpha}, S_{n,l,m,k}(\vec{y}, \vec{\beta})\right) \approx \Phi_{\#}^{n,l,m,k}(\vec{x}, \vec{y}, \vec{\alpha}, \vec{\beta}, \mathcal{P})$$

Problem 72 *Sea* $f : S \subseteq \omega^n \times \Sigma^{*m} \times Pro^{\Sigma} \mapsto \omega$ *una función* Σ*-computable. Entonces existe* $\mathscr{P}_0 \in Pro^{\Sigma}$ *tal que*

$$f(\vec{x}, \vec{\alpha}, \mathscr{P}_0) \approx \Phi_{\#}^{n,m}(\vec{x}, \vec{\alpha}, \mathscr{P}_0)$$

Definamos

$$\mathscr{F} = \lambda \vec{x} x \vec{\alpha} \left[f\left(\vec{x}, \vec{\alpha}, S^{\Sigma}_{n,1,m,0}(x, *^{<}(x))\right)\right]$$

Sea $\mathbb{Q}_0$ el programa que computa $\mathscr{F}$ y $\mathscr{P}_0 = S^{\Sigma}_{n,1,m,0}(\#^{<}(\mathbb{Q}_0), \mathbb{Q}_0)$. Veamos que

$$\Phi_{\#}^{n,m}(\vec{x}, \vec{\alpha}, \mathscr{P}_0) \approx \Phi_{\#}^{n+1,m}\left(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}, \mathbb{Q}_0\right) \tag{1}$$

Proof. Por definición de $\mathscr{P}_0$,

$$\begin{aligned}\Phi_{\#}^{n,m}(\vec{x}, \vec{\alpha}, \mathscr{P}_0) &\approx \Phi_{\#}^{n,m}\left(\vec{x}, \vec{\alpha}, S^{\Sigma}_{n,1,m,0}(\#^{<}(\mathbb{Q}_0), \mathbb{Q}_0)\right) \\ &\approx \Phi_{\#}^{n+1,m}\left(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}, \mathbb{Q}_0\right)\end{aligned}$$

por el teorema del parámetro. (El uso de $\approx$ está claramente justificado porque las expresiones son iguales por definición de $\mathscr{P}_0$; es decir, sólo estamos reemplazando un símbolo por su definición explícita.)

Ahora veamos que

$$\Phi_{\#}^{n+1,m}\left(\vec{x}, \#^{<}\left(\mathbb{Q}_0\right), \vec{\alpha}, \mathbb{Q}_0\right) \approx f\left(\vec{x}, \vec{\alpha}, \mathscr{P}_0\right) \tag{2}$$

Proof. Asuma por ahora que las expresiones en *(2)* están simultáneamente definidas. Pues $\mathbb{Q}_0$ computa $\mathscr{F}$,

$$\begin{aligned}
\Phi_{\#}^{n+1,m}\left(\vec{x}, \#^{<}\left(\mathbb{Q}_0\right), \vec{\alpha}, \mathbb{Q}_0\right) &= \mathscr{F}\left(\vec{x}, \#^{<}\left(\mathbb{Q}_0\right), \vec{\alpha}\right) \\
&= f\left(\vec{x}, \vec{\alpha}, S_{n,1,m,0}^{\Sigma}\left(\#^{<}(\mathbb{Q}_0), *^{<}\left(\#^{<}\left(\mathbb{Q}_0\right)\right)\right)\right) \\
&= f\left(\vec{x}, \vec{\alpha}, S_{n,1,m,0}^{\Sigma}\left(\#^{<}(\mathbb{Q}_0), \mathbb{Q}_0\right)\right) \\
&= f\left(\vec{x}, \vec{\alpha}, \mathscr{P}_0\right)
\end{aligned}$$

De esto se sigue que, de estar simultáneamente definidas, las expresiones en la ecuación dos toman el mismo valor. Probaremos que en efecto están simultáneamente definidas.
Observe que la expresión que involucra a Φ en *(2)* solo está definida si $(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}) \in D_{\mathscr{F}}$, por ser $\mathscr{F}$ la función que computa a $\mathbb{Q}_0$. Ahora bien, por la definición de f, tenemos que

$$D_{\mathscr{F}} = \left\{(\vec{x}, x, \vec{\alpha}) : *^{<}(x) \in Pro^{\Sigma} \wedge (\vec{x}, \vec{\alpha}, *^{<}(x)) \in S\right\}$$

Pues # y $*$ son funciones inversas, y por hipótesis $\mathbb{Q}_0 \in Pro^{\Sigma}$, tenemos que $*^{<}(\#(\mathbb{Q}_0)) = \mathbb{Q}_0 \in Pro^{\Sigma}$, y por lo tanto $(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}) \in D_{\mathscr{F}} \iff (\vec{x}, \vec{\alpha}, \mathbb{Q}_0) \in S$. Es decir que $\Phi(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}, \mathbb{Q}_0)$ y $f(\vec{x}, \vec{\alpha}, \mathbb{Q}_0)$ están definidas para el mismo conjunto de valores; a saber, los $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ tales que $(\vec{x}, \vec{\alpha}, \mathbb{Q}_0) \in S$.

Usando la igualdad *(2)* en la *(1)* obtenemos

$$\Phi_{\#}^{n,m}\left(\vec{x}, \vec{\alpha}, \mathscr{P}_0\right) \approx f\left(\vec{x}, \vec{\alpha}, \mathscr{P}_0\right) \quad \blacksquare \tag{3}$$

Observación. Sean $f : \omega^n \times \Sigma^{*m} \mapsto O$, $g : \omega^n \times \Sigma^{*m} \mapsto O$, con $O \in \{\omega, \Sigma^*\}$. Si para todo $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ sucede $f(\vec{x}, \vec{\alpha}) \approx g(\vec{x}, \vec{\alpha})$, entonces $f = g$. Inversamente, $f = g \Rightarrow f(\vec{x}, \vec{\alpha}) \approx g(\vec{x}, \vec{\alpha})$ para todo $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$.

> **Proof.** ($\Rightarrow$) La notación $f(\vec{x}, \vec{\alpha}) \approx g(\vec{x}, \vec{\alpha})$ denota que o bien ambas expresiones no están definidas, o ambas están definidas y toman el mismo valor. (Es decir, fortalece la idea de que decir que dos expresiones toman el mismo valor es decir *dos* cosas: que sus computaciones ambas terminan, y que terminan dando el mismo output.) Entonces, si $f(\vec{x}, \vec{\alpha}) \approx g(\vec{x}, \vec{\alpha})$ para todo $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$, tenemos que $\mathcal{D}_f = \mathcal{D}_g$. Y pues siempre que están definidas toman el mismo valor, sucede $f = g$. La prueba de $\Leftarrow$ es similar.

Usaré esta observación en los siguientes ejercicios.

Problem 73 *Probar que existe $\mathcal{P} \in Pro^{\Sigma}$ tal que $\Psi^{1,0,\#}_{\mathcal{P}}(x) = x + |\mathcal{P}|$ para cada $x \in \omega$.*

Sea $f : \omega \times Pro^{\Sigma} \mapsto \omega$ definida como $f := \lambda x \mathcal{P}\,[x + |\mathcal{P}|]$. Es fácil ver que la función es Σ-p.r. (recuerde que asumimos $\Sigma \supseteq \Sigma_p$) y por lo tanto Σ-computable. Por aplicación directa del teorema de la recursión, existe un programa $\mathcal{P}_0 \in Pro^{\Sigma}$ tal que $f(x, \mathcal{P}_0) \approx \Phi(x, \mathcal{P}_0) \approx \Psi^{1,0,\#}_{\mathcal{P}_0}(x)$. Observe que $f(x, \mathcal{P}_0)$ está definida para toda $x \in \omega$. Por lo tanto,

$$f(x, \mathcal{P}_0) = x + |\mathcal{P}_0| = \Psi^{1,0,\#}_{\mathcal{P}_0}(x)$$

para cada $x \in \omega$.

Problem 74 *Pruebe que hay un $\mathcal{P} \in Pro^{\Sigma}$ tal que $\Psi^{1,0,\#}_{\mathcal{P}} = C^{1,0}_{|\mathcal{P}|}$.*

Sea $f : \omega \times Pro^{\Sigma} \mapsto \omega$ definida como $f := \lambda x \mathcal{P}\,[|\mathcal{P}|]$. Claramente la función es Σ-computable. Por teorema de la recursión, existe $\mathcal{P}_0 \in Pro^{\Sigma}$ tal que $f(x, \mathcal{P}_0) \approx \Psi^{1,0,\#}_{\mathcal{P}_0}(x)$. Ahora bien, $f(x, \mathcal{P}_0) = f \circ \left[p^{1,0}_1, C^{1,0}_{\mathcal{P}_0}\right] = C^{1,0}_{|\mathcal{P}_0|}$. Es decir que, $C^{1,0}_{|\mathcal{P}_0|}(x) \approx \Psi^{1,0,\#}_{\mathcal{P}_0}(x)$. Pues $C^{1,0}_{|\mathcal{P}_0|}$ está definida para toda $x \in \omega$ (es Σ-total), la igualdad anterior vale para todo $x \in \omega$. Usando la observación dada al principio, resulta entonces que $C^{1,0}_{|\mathcal{P}_0|} = \Psi^{1,0,\#}_{\mathcal{P}_0}$.

Problem 75 *Enuncie el teorema de la recursión para f con imagen contenida en Σ^* y analice la prueba.*

El teorema se enuncia como sigue:

> Sea $f : S \subseteq \omega^n \times \Sigma^{*m} \times Pro^{\Sigma} \mapsto \Sigma^*$ una función Σ-computable.
> Entonces existe $\mathcal{P}_0 \in Pro^{\Sigma}$ tal que

$$f(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx \Phi_*^{n,m}(\vec{x}, \vec{\alpha}, \mathcal{P}_0)$$

La prueba es prácticamente idéntica. Se define $\mathbb{Q}_0$ como el programa que computa

$$\mathcal{F} = \lambda \vec{x} x \vec{\alpha} \left[f\left(\vec{x}, \vec{\alpha}, S^{\Sigma}_{n,1,m,0}(x, *^{<}(x))\right)\right]$$

y se define $\mathcal{P}_0 = S^{\Sigma}_{n,1,m,0}(\#^{<}(\mathbb{Q}_0), \mathbb{Q}_0)$. Aplicando el teorema del parámetro, se ve que $\Phi_{\#}^{n,m}(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx \Phi_{\#}^{n+1,m}(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}, \mathbb{Q}_0)$
Se muestra entonces que

$$\Phi_{\#}^{n+1,m}\left(\vec{x}, \#^{<}(\mathbb{Q}_0), \vec{\alpha}, \mathbb{Q}_0\right) \approx f(\vec{x}, \vec{\alpha}, \mathcal{P}_0)$$

usando la definición de $\mathcal{F}$ y teorema del parámetro. Por transitividad de $\approx$ (la transitividad de la relación es clara por su definición), se obtiene entonces

$$\Phi_*^{n,m}(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx f(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \quad \blacksquare \tag{4}$$

Problem 76 *Pruebe que hay un $\mathcal{P} \in Pro^{\Sigma}$ tal que $\Psi^{2,2,*}_{\mathcal{P}} = C^{2,2}_{\mathcal{P}}$.*

Sea $f := \lambda x y \alpha \beta \mathcal{P}\left[C^{2,2}_{\mathcal{P}}(x, y, \alpha, \beta)\right]$. Claramente, f es una función constante y por lo tanto Σ-computable. Por aplicación directa del teorema de la recursión, obtenemos que existe un programa $\mathcal{P}_0 \in Pro^{\Sigma}$ tal que $f(x, y, \alpha, \beta, \mathcal{P}_0) \approx \Phi^{2,2}_*(x, y, \alpha, \beta, \mathcal{P}_0)$. Pero $f(x, y, \alpha, \beta, \mathcal{P}_0) = f \circ \left[p^{2,2,}_1, p^{2,2}_2, p^{2,2}_3, p^{2,2}_4, C^{2,2}_{\mathcal{P}_0}\right] = C^{2,2}_{\mathcal{P}_0}$. Por lo tanto, lo que hemos obtenido es que $C^{2,2}_{\mathcal{P}_0}(x, y, \alpha, \beta) \approx \Psi^{2,2,*}_{\mathcal{P}_0}(x, y, \alpha, \beta)$ para todo $(x, y, \alpha, \beta) \in \omega^2 \times \Sigma^{*2}$. Luego $C^{2,2}_{\mathcal{P}_0} = \Psi^{2,2,\#}_{\mathcal{P}_0}$.

Problem 77 *Sea $f : Pro^{\Sigma} \mapsto Pro^{\Sigma}$ una función Σ-computable. Dados $n, m \in \omega$, existe $\mathcal{P} \in Pro^{\Sigma}$ tal que $\Psi^{n,m,\#}_{\mathcal{P}} = \Psi^{n,m,\#}_{f(\mathcal{P})}$*

Sea $G : \omega^n \times \Sigma^{*m} \times Pro^{\Sigma} \mapsto \omega$ definida como

$$G = \lambda \vec{x}\vec{\alpha}\mathcal{P}\left[\Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, f(\mathcal{P}))\right] = \Phi^{n,m}_{\#} \circ \left[p^{n,m+1}_{1}, \ldots, p^{n,m+1}_{n+m}, f \circ p^{n,m+1}_{n+m+1}\right]$$

Claramente, G es Σ-computable, por ser composición de las funciones $\Phi^{n,m}_{\#}$, $p^{n,m+1}_{j}$ y f, todas Σ-computables. Luego G satisface las condiciones del teorema de la recursión, y por lo tanto existe un $\mathcal{P}_0 \in Pro^{\Sigma}$ tal que

$$G(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx \Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \tag{5}$$

Sea $\mathcal{G}_0$ el programa que computa a $G(\vec{x}, \vec{\alpha}, \mathcal{P}_0)$. Entonces, la ecuación (5) puede escribirse

$$\Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, \mathcal{G}_0) \approx \Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \tag{6}$$

Por definición de G, tenemos que $G(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx \Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, f(\mathcal{P}_0))$; es decir, que

$$\Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, \mathcal{G}_0) \approx \Phi^{n,m}(\vec{x}, \vec{\alpha}, f(\mathcal{P}_0)) \tag{7}$$

Por lo tanto, usando (6) y (7), $\Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, \mathcal{P}_0) \approx \Phi^{n,m}_{\#}(\vec{x}, \vec{\alpha}, f(\mathcal{P}_0))$ lo cual implica

$$\Psi^{n,m,\#}_{\mathcal{P}_0} = \Psi^{n,m,\#}_{f(\mathcal{P}_0)}$$