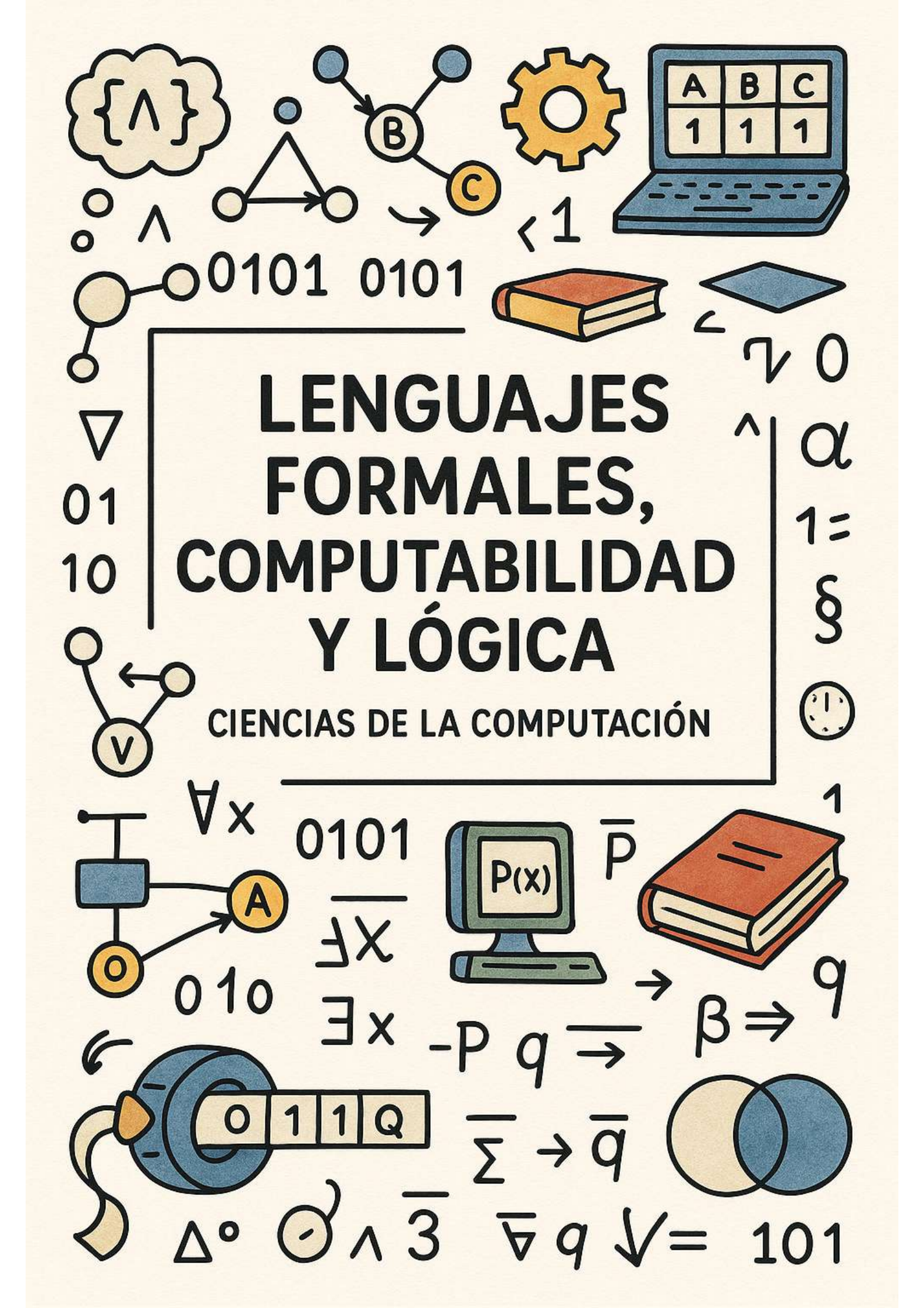
LENGUAJES FORMALES, COMPUTABILIDAD Y LÓGICA
CIENCIAS DE LA COMPUTACIÓN

GUIA 1 DE LENGUAJES: NOTACION Y CONCEPTOS BASICOS

Usaremos $\mathbf{R}$ para denotar el conjunto de los numeros reales, $\mathbf{Z}$ para denotar el conjunto de los numeros enteros, $\mathbf{N}$ para denotar el conjunto de los numeros naturales y ω para denotar al conjunto $\mathbf{N} \cup \{0\}$.

Dado un conjunto A, usaremos $\mathcal{P}(A)$ para denotar el conjunto formado por todos los subconjuntos de A, es decir:

$$\mathcal{P}(A) = \{S : S \subseteq A\}$$

Si A es un conjunto finito, entonces $|A|$ denotara la cantidad de elementos de A.

Para $x, y \in \omega$, definamos

$$x \dot{-} y = \begin{cases} x - y & \text{si } x \geq y \\ 0 & \text{caso contrario} \end{cases}$$

Dados $x, y \in \omega$ diremos que x *divide a* y cuando haya un $z \in \omega$ tal que $y = z.x$. Notar que 0 divide a 0, 3 divide a 0 y 0 no divide a 23. Escribiremos $x \mid y$ para expresar que x divide a y. Si bien no hay una definicion natural en matematica de cuanto vale 0^0 (0 elevado a la 0), por convencion para nosotros $0^0 = 1$

Producto carteciano

Dados conjuntos $A_1, ..., A_n$, con $n \geq 2$, usaremos $A_1 \times ... \times A_n$ para denotar el *producto Cartesiano* de $A_1, ..., A_n$, es decir el conjunto formado por todas las n-uplas $(a_1, ..., a_n)$ tales que $a_1 \in A_1, ..., a_n \in A_n$. Si $A_1 = ... = A_n = A$, con $n \geq 2$, entonces escribiremos A^n en lugar de $A_1 \times ... \times A_n$. Para $n = 1$, definimos $A^n = A$, es decir $A^1 = A$. Usaremos $\Diamond$ para denotar la unica 0-upla. Definimos entonces $A^0 = \{\Diamond\}$. Si A es un conjunto denotaremos con $A^{\mathbf{N}}$ al conjunto formado por todas las infinituplas $(a_1, a_2, ...)$ tales que $a_i \in A$ para cada $i \in \mathbf{N}$. Por ejemplo

$$(1, 2, 3, 4, ...) \in \omega^{\mathbf{N}}$$

donde $(1, 2, 3, 4, ...)$ es una forma intuitiva de denotar la infinitupla cuyo i-esimo elemento es el numero natural i.

Si $(A_1, A_2, ...)$ es una infinitupla de conjuntos, entonces usaremos $\bigcup_{i=1}^{\infty} A_i$ o $\bigcup_{i \geq 1} A_i$ para denotar al conjunto

$$\{a : a \in A_i, \text{ para algun } i \in \mathbf{N}\}$$

Conjuntos

Supondremos que el lector sabe las nociones basicas sobre conjuntos, aunque resaltaremos algunas de las mas importantes para que el lector las repase.

La propiedad de *extensionalidad* nos dice que, dados conjuntos A, B, se tiene que $A = B$ si y solo si para cada objeto x se da que

$$x \in A \text{ si y solo si } x \in B$$

Esta propiedad es importante metodologicamente ya que a la hora de probar que dos conjuntos A, B son iguales, extensionalidad nos asegura que basta con ver que se dan las dos inclusiones $A \subseteq B$ y $B \subseteq A$.

Otro tema importante es manejar correctamente la notacion cuando definimos un conjunto usando llaves y mediante propiedades que caracterizan la pertenencia al mismo. Algunos ejercicios para entrenar esta notacion:

Ejercicio 1: Entender en forma precisa que conjunto se esta denotando en cada uno de los siguientes casos

(a) $\{x \in \mathbf{N} : x = 1 \text{ o } x \geq 5\}$
(b) $\{x : x \in \mathbf{R} \text{ y } x^2 \geq 100\}$
(c) $\{x : x = 100\}$
(d) $\{x^2 + 1 : x \in \omega\}$
(e) $\{x + y + z : x, y, z \in \{1, 2\}\}$

Ejercicio 2: V o F o I, justifique.

(a) $\{x.y : x, y \in \omega\} = \omega$
(b) $|\{x.y : x, y \in \omega \text{ y } 1 \leq x, y \leq 5\}| = 25$
(c) Dados $A, B \subseteq \omega$, se tiene que $\{a \in A \text{ y } b \in B : a + b = 1000\} \subseteq A \times B$
(d) $\{a \in \mathbf{N}, a \geq 3\} \subseteq \omega$
(e) $\{x + 1 : x \in \{1, 2, 3\}\} = \{1, 2, 3, 4\}$

Alfabetos

Un *alfabeto* es un conjunto finito de simbolos. Notese que $\emptyset$ es un alfabeto. Si Σ es un alfabeto, entonces Σ^* denotara el conjunto de todas las palabras formadas con simbolos de Σ. Las palabras de longitud 1 son exactamente los elementos de Σ, en particular esto nos dice que $\Sigma \subseteq \Sigma^*$. La unica palabra de longitud 0 es denotada con ε. Ya que en ε no ocurren simbolos, tenemos que $\varepsilon \in \Sigma^*$, para cualquier alfabeto, mas aun notese que $\emptyset^* = \{\varepsilon\}$. Usaremos $|\alpha|$ para denotar la longitud de la palabra α. Si $\alpha \in \Sigma^*$ y $\sigma \in \Sigma$, usaremos $|\alpha|_\sigma$ para denotar la cantidad de ocurrencias del simbolo σ en α. Notese que funciones, n-uplas y palabras son objetos de distinto tipo, por lo cual $\emptyset$, $\Diamond$ y ε son tres objetos matematicos diferentes.

Si $\alpha_1, ..., \alpha_n \in \Sigma^*$, con $n \geq 0$, usaremos $\alpha_1 ... \alpha_n$ para denotar la *concatenacion* de las palabras $\alpha_1, ..., \alpha_n$ (notese que cuando $n = 0$, resulta que $\alpha_1 ... \alpha_n = \varepsilon$). Si $\alpha_1 = ... = \alpha_n = \alpha$, entonces escribiremos α^n en lugar de $\alpha_1 ... \alpha_n$. O sea que $\alpha^0 = \varepsilon$.

Un *lenguaje sobre* Σ sera un subconjunto de Σ^*. Si L es un lenguaje sobre Σ, entonces denotaremos con L^+ al conjunto formado por todas las concatenaciones de sucesiones finitas no nulas de lementos de L. Es decir:

$$L^+ = \{\alpha_1 ... \alpha_n : \alpha_1, ..., \alpha_n \in L \text{ y } n \geq 1\}$$

Notese que en particular obtenemos que $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Diremos que α *es subpalabra (propia) de* β cuando ($\alpha \notin \{\varepsilon, \beta\}$ y) existan palabras δ, γ tales que $\beta = \delta\alpha\gamma$. Diremos que β es un *tramo inicial (propio)* de α si hay una palabra γ tal que $\alpha = \beta\gamma$ (y $\beta \notin \{\varepsilon, \alpha\}$). En forma similar se define *tramo final (propio)*.

Dados $i \in \omega$ y $\alpha \in \Sigma^*$ definamos

$$[\alpha]_i = \begin{cases} i\text{-esimo elemento de } \alpha & \text{si } 1 \leq i \leq |\alpha| \\ \varepsilon & \text{caso contrario} \end{cases}$$

Dada $\gamma \in \Sigma^*$, definamos

$$\gamma^R = \begin{cases} [\gamma]_{|\gamma|}[\gamma]_{|\gamma|-1} ... [\gamma]_1 & \text{si } |\gamma| \geq 1 \\ \varepsilon & \text{caso contrario} \end{cases}$$

La palabra γ^R es llamada la *resiproca* de γ. Dada $\alpha \in \Sigma^*$, definamos

$$\curvearrowright\alpha = \begin{cases} [\alpha]_2 \dots [\alpha]_{|\alpha|} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases} \qquad \alpha^{\curvearrowleft} = \begin{cases} [\alpha]_1 \dots [\alpha]_{|\alpha|-1} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases}$$

Ocurrencias. Dadas palabras $\alpha, \beta \in \Sigma^*$, con $|\alpha|, |\beta| \geq 1$, y un natural $i \in \{1, ..., |\beta|\}$, se dice que α *ocurre a partir de* i *en* β cuando se de que existan palabras δ, γ tales que $\beta = \delta\alpha\gamma$ y $|\delta| = i - 1$. Intuitivamente hablando α ocurre a partir de i en β cuando se de que si comensamos a leer desde el lugar i-esimo de β en adelante, leeremos la palabra α completa y luego posiblemente seguiran otros simbolos.

Notese que una palabra α puede ocurrir en β, a partir de i, y tambien a partir de j, con $i \neq j$. En virtud de esto, hablaremos de las distintas ocurrencias de α en β. Por ejemplo hay dos ocurrencias de la palabra *aba* en la palabra

$$ccccccabaccccabaccccc$$

y tambien hay dos ocurrencias de la palabra *aba* en la palabra

$$ccccccababaccccccccc$$

En el primer caso diremos que dichas ocurrencias de *aba* son *disjuntas* ya que ocupan espacios disjuntos dentro de la palabra. En cambio en el segundo caso puede apreciarse que las dos ocurrencias se superponen en una posicion. A veces diremos que una ocurrencia esta *contenida* o *sucede* dentro de otra. Por ejemplo la segunda ocurrencia de *ab* en $babbbfabcccfabccc$ esta contenida en la primer ocurrencia de $fabc$ en $babbbfabcccfabccc$.

No definiremos en forma matematica precisa el concepto de ocurrencia pero el lector no tendra problemas en comprenderlo y manejarlo en forma correcta.

Reemplazos de ocurrencias. Tambien haremos *reemplazos* de ocurrencias por palabras. Por ejemplo el resultado de reemplazar la primer ocurrencia de *abb* en $ccabbgfgabbgg$ por *oolala* es la palabra $ccoolalagfgabbgg$. Cuando todas las ocurrencias de una palabra α en una palabra β sean disjuntas entre si, podemos hablar del resultado de *reeplazar simultaneamente cada ocurrencia de* α *en* β *por* γ. Por ejemplo si tenemos

$$\alpha = yet$$
$$\beta = ghsyetcjjjyetbcpyeteabc$$
$$\gamma = \%\%$$

entonces $ghs\%\%cjjj\%\%bcp\%\%eabc$ es el resultado de reemplazar simultaneamente cada ocurrencia de α en β por γ. Es importante notar que los reemplazos se hacen simultaneamente y no secuencialmente (i.e. reemplazando la primer ocurrencia de α por γ y luego al resultado reemplazarle la primer ocurrencia de α por γ y asi sucesivamente). Obviamente el reemplazo secuencial puede dar un resultado distinto al simultaneo (que es el que usaremos en general) e incluso puede suceder que en el reemplazo secuencial el proceso se pueda iterar indefinidamente. Dejamos al lector armar ejemplos de estas cituaciones.

Tambien se pueden hacer reemplazos simultaneos de distintas palabras en una palabra dada. Supongamos tenemos palabras $\alpha_1, ..., \alpha_n$ (con $\alpha_i \neq \alpha_j$, para $i \neq j$) las cuales tienen la propiedad de que las distintas ocurrencias de ellas en la palabra β son siempre disjuntas de a pares, y tenemos ademas palabras $\gamma_1, ..., \gamma_n$. Entonces hablaremos del resultado de reemplazar simultaneamente:

- cada ocurrencia de α_1 en β, por γ_1
- cada ocurrencia de α_2 en β, por γ_2

$$\vdots$$

- cada ocurrencia de α_n en β, por γ_n

Por ejemplo si tomamos

$$\begin{aligned}\alpha_1 &= gh\\ \alpha_2 &= yet\\ \alpha_3 &= ana\\ \beta &= ghbbbyetbbgh\%\%ana\#\#ana!!!ana\\ \gamma_1 &= AA\\ \gamma_2 &= BBBB\\ \gamma_3 &= CCC\end{aligned}$$

entonces $AAbbbBBBBbbAA\%\%CCC\#\#CCC!!!CCC$ es el resultado de reemplazar simultaneamente:

- cada ocurrencia de α_1 en β, por γ_1
- cada ocurrencia de α_2 en β, por γ_2
- cada ocurrencia de α_3 en β, por γ_3

Matematica orientada a objetos

Nuestro estilo o enfoque matematico pondra enfasis en los objetos, es decir haremos matematica prestando atencion a los distintos objetos matematicos involucrados, los cuales siempre seran definidos en forma precisa en terminos de objetos mas primitivos. Hay ciertos objetos matematicos los cuales no definiremos y supondremos que el lector tiene una idea clara y precisa de los mismos. Por ejemplo un tipo de objeto matematico, quizas el mas famoso, son los *numeros*. No diremos que es un numero pero supondremos que el lector tiene una intuicion clara acerca de este tipo de objetos y de sus propiedades basicas. Otro tipo de objeto que no definiremos y que sera clave para nuestro enfoque son los *conjuntos*. Nuevamente, no diremos que es un conjunto pero supondremos que el lector tiene una intuicion clara acerca de estos objetos y sus propiedades basicas. Es importante que en nuestro enfoque, numeros y conjuntos son objetos de distinta naturaleza por lo cual nunca un numero es un conjunto ni un conjunto es un numero. En particular esto nos dice que el numero 0 y el conjunto $\emptyset$ son objetos distintos. Otro tipo de objeto matematico muy importante para la matematica discreta son los *simbolos*. No discutiremos que es un simbolo sino que aceptaremos este concepto en forma primitiva. Tambien constituyen un tipo de objeto matematico las *palabras*, las cuales intuitivamente hablando son juxtaposiciones de simbolos. Otro tipo de objeto matematico muy importante son los *pares ordenados* o 2-*uplas*, es decir los objetos de la forma (a, b), donde a y b son objetos matematicos cualesquiera. Tambien son objetos matematicos y de distinta naturaleza las 3-*uplas*, las 4-*uplas* y en general las n-*uplas* para n un numero natural mayor o igual a 2. Cabe destacar que en nuestro enfoque no habra 1-uplas. Sin envargo, si bien hay una sola 0-*upla*, ella constituye un tipo de objeto matematico distinto a los antes mencionados. El ultimo tipo de objeto matematico que consideraremos es aquel de las *infinituplas*.

Tenemos entonces dividido nuestro universo matematico en las distintas categorias o tipos de objetos:

$$\begin{array}{c} \text{NUMERO} \\ \text{CONJUNTO} \\ \text{PALABRA} \\ 0-\text{UPLA} \\ 2-\text{UPLA} \\ 3-\text{UPLA} \\ \vdots \\ \text{INFINITUPLA} \end{array}$$

(Notar que los simbolos quedan contenidos en la categoria de las palabras). Es importante entender que las anteriores categorias o tipos de objetos son disjuntas entre si, es decir nunca un numero sera una palabra o una palabra sera una 3-upla etc. Esto nos permite definir una funcion Ti la cual a un objeto matematico le asigna su tipo de objeto matematico segun la lista anterior. Por ejemplo:

$$\begin{aligned} Ti(\pi) &= \text{NUMERO} \\ Ti(\mathbf{N}) &= \text{CONJUNTO} \\ Ti(\mathcal{P}(\mathbf{N})) &= \text{CONJUNTO} \\ Ti((1,2,3)) &= 3-\text{UPLA} \\ Ti(\emptyset) &= \text{CONJUNTO} \\ Ti(\varepsilon) &= \text{PALABRA} \\ Ti(\Diamond) &= 0-\text{UPLA} \\ Ti(\alpha) &= \text{PALABRA, si } \alpha \text{ es un simbolo} \end{aligned}$$

Escarabajo y Mariposa

Para hacer divertido nuestro desarrollo crearemos dos personajes que con su personalidad nos dejaran analizar las distintas maneras de hacer matematica. El *escarabajo* es netamente sintactico y mecanico, no le gusta pensar los conceptos matematicos imaginando los objetos involucrados en los mismos, simplemente quiere avanzar (sin mucho pensamiento) aplicando reglas sintacticas que le permitan obtener nuevas ecuaciones o expresiones matematicas. El vive en el mundo plano de las palabras y ahi se siente apasionado por sus habiles movimientos mecanicos. El escarabajo es en algun sentido una maravilla de la destreza mecanico-simbolica. La *mariposa* es todo lo contrario, le interesa pensar en los objetos matematicos como si fueran reales dentro de su imaginacion fantastica y con su habilidad de volar contempla el universo matematico desde un punto de vista conceptual e independiente de la manipulacion sintactica. Reconoce como objetos reales de su universo matematico no solo a los objetos finitarios (palabras, numeros, etc) sino tambien a toda la gama de objetos sofisticados transfinitos que se pueden construir con la imaginacion y la especulacion matematica. Mantiene este universo matematico en su imaginacion dandole existencia y coherencia mas alla del mundo limitado de los

objetos sintacticos que usa para expresarse. La mariposa es en algun sentido una maravilla de la imaginacion matematica coherente.

A modo de ejemplo, al escarabajo la matematica orientada a objetos descripta en la seccion anterior no le hace mucha gracia ya que es perezoso para imaginar los objetos asociados a los simbolos. Al contrario, la mariposa solo concibe la matematica orientada a objetos, los cuales piensa e imagina con un grado de realidad exacerbado.

En general la matematica se enseña con un estilo muy escarabajo y aun en las universidades esto persiste, manifestandose mas en las carreras de ciencias de la computacion que en las vinculadas a la matematica clasica. Es natural que esto suceda ya que los informaticos manipulan constantemente objetos sintacticos usando reglas mecanicas y a veces los objetos subyacentes (semantica) pasan a un segundo plano.

El concepto de funcion

Asumiremos que el lector tiene una idea intuitiva del concepto de funcion. Daremos aqui una definicion matematica de dicho concepto. Una *funcion* es un conjunto f de pares ordenados con la siguiente propiedad

(F) Si $(x, y) \in f$ y $(x, z) \in f$, entonces $y = z$.

Por ejemplo, si tomamos $f = \{(x, x^2) : x \in \omega\}$ se puede ver facilmente que f cumple la propiedad (F). Dada una funcion f, definamos

$$D_f = \text{ dominio de } f = \{x : (x, y) \in f \text{ para algun } y\}$$
$$I_f = \text{ imagen de } f = \{y : (x, y) \in f \text{ para algun } x\}$$

A veces escribiremos $\mathrm{Dom}(f)$ y $\mathrm{Im}(f)$ para denotar, respectivamente, el dominio y la imagen de una funcion f. Notese que $\emptyset$ es una funcion y que $D_\emptyset = I_\emptyset = \emptyset$. Si $f = \{(x, x^2) : x \in \omega\}$ se tiene que $D_f = \omega$ y $I_f = \{y : y = x^2 \text{ para algun } x \in \omega\}$. Si $f = \{(x, 1) : x \in \{1, 6, 18\}\}$, entonces $D_f = \{1, 6, 18\}$ y $I_f = \{1\}$.

Convencion Notacional 1: Como es usual, dado $x \in D_f$, usaremos $f(x)$ para denotar al unico $y \in I_f$ tal que $(x, y) \in f$. Por ejemplo, si $f = \{(x, x^2) : x \in \omega\}$ se tiene que $f(x) = x^2$, para cada $x \in D_f = \omega$.

Convencion Notacional 2: Escribiremos $f : S \subseteq A \to B$ para expresar que f es una funcion tal que $D_f = S \subseteq A$ y $I_f \subseteq B$. Tambien escribiremos $f : A \to B$ para expresar que f es una funcion tal que $D_f = A$ y $I_f \subseteq B$. En tal contexto llamaremos a B *conjunto de llegada*. Por supuesto B no esta determinado por f ya que solo debe cumplir $I_f \subseteq B$. Es decir que cualquier conjunto B que contenga a I_f puede ser considerado conjunto de llegada de f.

Convencion Notacional 3: Muchas veces para definir una funcion f, lo haremos dando su dominio y su regla de asignacion, es decir especificaremos en forma precisa que conjunto es el dominio de f y ademas especificaremos en forma presisa quien es $f(x)$ para cada x de dicho dominio. Obviamente esto determina por completo a la funcion f ya que siempre se da que $f = \{(x, f(x)) : x \in D_f\}$. Por ejemplo si decimos que f es la funcion dada por:

$$D_f = \omega$$
$$f(x) = 23x^2$$

nos estaremos refiriendo a la funcion $\{(x, 23x^2) : x \in \omega\}$. Tambien escribiremos

$$\begin{array}{rcl} f : \omega & \to & \omega \\ x & \to & 23x^2 \end{array}$$

para describir a f. Es decir, a veces para hacer mas intuitiva aun la descripcion de la funcion, tambien incluiremos un conjunto de llegada de dicha funcion y a la regla de asignacion la escribiremos usando una flecha. Para dar otro ejemplo, si escribimos sea f dada por:

$$\begin{array}{rcl} f : \mathbf{N} & \to & \omega \\ x & \to & \begin{cases} x+1 & \text{si } x \text{ es par} \\ x^2 & \text{si } x \text{ es impar} \end{cases} \end{array}$$

estaremos diciendo que f es la funcion

$$\{(x, x+1) : x \text{ es par y } x \in \mathbf{N}\} \cup \{(x, x^2) : x \text{ es impar y } x \in \mathbf{N}\}$$

Funcion identidad. Dado un conjunto A, a la funcion

$$\begin{array}{rcl} A & \to & A \\ a & \to & a \end{array}$$

La denotaremos con Id_A y la llamaremos la funcion *identidad sobre* A. Notese que $Id_A = \{(a, a) : a \in A\}$.

Igualdad de funciones. Sean f y g dos funciones. Ya que las mismas son conjuntos, tendremos que f sera igual a g si y solo si para cada par (a, b), se tiene que $(a, b) \in f$ sii $(a, b) \in g$. Muchas veces sera util el siguiente criterio de igualdad de funciones:

Lema 1. *Sean f y g funciones. Entonces $f = g$ sii $D_f = D_g$ y para cada $x \in D_f$ se tiene que $f(x) = g(x)$*

.

Ejercicio 3: Pruebe el lema anterior (en el apunte esta probado)

Ejercicio 4: V o F o I, justifique.

(a) Si

$$\begin{array}{rcl} f : \mathbf{N} & \to & \omega \\ x & \to & x^3 \end{array} \qquad \begin{array}{rcl} g : \mathbf{N} & \to & \mathbf{R} \\ x & \to & x^3 \end{array}$$

entonces $f = g$

(b) Si

$$\begin{array}{rcl} f : \mathbf{N} & \to & \omega \\ x & \to & x^3 \end{array} \qquad \begin{array}{rcl} g : \mathbf{N} & \to & \omega \\ x & \to & x^4/x \end{array}$$

entonces $f = g$

(c) Si f es una funcion y $z \in D_f$, entonces $Ti(z) =$ CONJUNTO

(d) $\text{Dom}((1, 2)) = \{1\}$

(e) $\text{Dom}(\{(1, 2)\}) + 1 = 2$

(f) Si f es una funcion, entonces $D_f = \{a : (a, b) \in f\}$

(g) Si $f : A \to B$, entonces $D_f \subseteq A$

(h) Si $f : A \to B$, entonces $I_f = B$

(i) Si f es una función y $g \subseteq f$, entonces g es una función

Funcion caracteristica de un subconjunto. Sea X un conjunto cualquiera y sea $S \subseteq X$. Usaremos χ_S^X para denotar la funcion

$$\begin{array}{rcl} \chi_S^X : X & \to & \omega \\ x & \to & \begin{cases} 1 \text{ si } x \in S \\ 0 \text{ si } x \notin S \end{cases} \end{array}$$

Llamaremos a χ_S^X la *funcion caracteristica de S con respecto a X*. Muchas veces cuando el conjunto X este fijo y sea claro el contexto, escribiremos χ_S en lugar de χ_S^X.

Restriccion de una funcion. Dada una funcion f y un conjunto $S \subseteq D_f$, usaremos $f|_S$ para denotar la *restriccion* de f al conjunto S, i.e. $f|_S = f \cap (S \times I_f)$. Notese que $f|_S$ es la funcion dada por

$$\begin{aligned} D_{f|_S} &= S \\ f|_S(e) &= f(e), \text{ para cada } e \in S \end{aligned}$$

Notese que cualesquiera sea la funcion f tenemos que $f|_\emptyset = \emptyset$ y $f|_{D_f} = f$.

Funciones Σ-mixtas

Sea Σ un alfabeto finito. Dados $n, m \in \omega$, usaremos $\omega^n \times \Sigma^{*m}$ para abreviar la expresion

$$\overbrace{\omega \times ... \times \omega}^{n \text{ veces}} \times \overbrace{\Sigma^* \times ... \times \Sigma^*}^{m \text{ veces}}$$

Por ejemplo, $\omega^3 \times \Sigma^{*4}$ sera una forma abreviada de escribir $\omega \times \omega \times \omega \times \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^*$. Debe quedar claro que estamos haciendo cierto abuso notacional ya que en principio si no hacemos esta convencion notacional, $\omega^3 \times \Sigma^{*4}$ denota un conjunto de pares y $\omega \times \omega \times \omega \times \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^*$ es un conjunto de 7-uplas.

Notese que:

- Cuando $n = m = 0$, tenemos que $\omega^n \times \Sigma^{*m}$ denota el conjunto $\{\Diamond\}$
- Si $m = 0$, entonces $\omega^n \times \Sigma^{*m}$ denota el conjunto ω^n
- Si $n = 0$, entonces $\omega^n \times \Sigma^{*m}$ denota el conjunto Σ^{*m}
- Cuando $\Sigma = \emptyset$, tenemos que $\Sigma^* = \{\varepsilon\}$. O sea que por ejemplo

$$\omega^n \times \Sigma^{*5} = \{(x_1, ..., x_n, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon) : x_1, ..., x_n \in \omega\}$$

Es decir que tenemos que tener cuidado cuando leemos esta notacion y no caer en la confucion de interpretarla mal.

Con esta convencion notacional, un elemento generico de $\omega^n \times \Sigma^{*m}$ es una $(n+m)$-upla de la forma $(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$. Para abreviar, escribiremos $(\vec{x}, \vec{\alpha})$ en lugar de $(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$.

Definicion de funcion Σ-mixta. Sea Σ un alfabeto finito. Dada una funcion f, diremos que f es Σ-*mixta* si cumple las siguientes propiedades

(M1) Existen $n, m \geq 0$, tales que $D_f \subseteq \omega^n \times \Sigma^{*m}$
(M2) Ya sea $I_f \subseteq \omega$ o $I_f \subseteq \Sigma^*$

Algunos ejemplos:

(E1) Sea $\Sigma = \{\square, \%, \blacktriangle\}$. La funcion

$$\begin{array}{rcl} f : \{(1, \square\%\%), (100, \%\blacktriangle\blacktriangle)\} & \to & \omega \\ (x, \alpha) & \to & x + |\alpha| \end{array}$$

es Σ-mixta ya que se cumple (M1) con $n = m = 1$ y tambien cumple (M2). Notese que f no es $\{\square, \%\}$-mixta ya que no cumple (M1) respecto del alfabeto $\{\square, \%\}$. Sin envargo note que f es $\{\square, \%, \blacktriangle, @\}$-mixta

(E2) La funcion

$$\begin{array}{rcl} \omega^4 & \to & \omega \\ (x, y, z, w) & \to & x + y \end{array}$$

es Σ-mixta cualesquiera sea el alfabeto Σ

(E3) Sea $\Sigma = \{\square, @\}$. La funcion

$$\begin{array}{rcl} \{\square\square\square, @@\} & \to & \omega \\ \alpha & \to & |\alpha| \end{array}$$

es Σ-mixta ya que se cumple (M1) (con $n = 0$ y $m = 1$) y (M2)

(E4) Supongamos $\Sigma = \emptyset$. Tenemos entonces que $\Sigma^* = \{\varepsilon\}$. Por ejemplo

$$\begin{array}{rcl} D & \to & \omega \\ (x, \varepsilon, \varepsilon, \varepsilon) & \to & x^2 \end{array}$$

donde $D = \{(x, \varepsilon, \varepsilon, \varepsilon) : x \text{ es impar}\}$, es Σ-mixta (con $n = 1$ y $m = 3$ en (M1)). Tambien notese que

$$\begin{array}{rcl} \{(\varepsilon, \varepsilon)\} & \to & \{\varepsilon\} \\ (\varepsilon, \varepsilon) & \to & \varepsilon \end{array}$$

es Σ-mixta (con $n = 0$ y $m = 2$ en (M1)).

Dejamos al lector la facil prueba del siguiente resultado basico.

Lema 2. *Supongamos $\Sigma \subseteq \Gamma$ son alfabetos finitos. Entonces si f es una funcion Σ-mixta, f es Γ-mixta*

Una funcion Σ-mixta f es Σ-*total* cuando haya $n, m \in \omega$ tales que $D_f = \omega^n \times \Sigma^{*m}$. El lema anterior nos dice que si $\Sigma \subseteq \Gamma$, entonces toda funcion Σ-mixta es Γ-mixta. Sin envargo una funcion puede ser Σ-total y no ser Γ-total, cuando $\Sigma \subseteq \Gamma$. Por ejemplo tomemos $\Sigma = \{\square, \%, \blacktriangle\}$ y $\Gamma = \{\square, \%, \blacktriangle, !\}$, y consideremos la funcion

$$\begin{array}{rcl} f : \omega \times \Sigma^* & \to & \omega \\ (x, \alpha) & \to & x + |\alpha| \end{array}$$

Es claro que f es Σ-mixta y Σ-total. Tambien es Γ-mixta ya que $D_f \subseteq \omega \times \Gamma^*$ y $I_f \subseteq \omega$, por lo cual cumple (M1) y (M2). Sin envargo f no es Γ-total ya que D_f no es igual a $\omega^n \times \Gamma^{*m}$, cualesquiera sean n y m.

Como hemos visto recien, una funcion f puede ser Σ-mixta y Γ-mixta para dos alfabetos distintos Σ y Γ e incluso es facil construir un ejemplo en el cual Σ y Γ sean incomparables como conjuntos, es decir que ninguno incluya al otro. Dejamos al lector convencerse de que si f es una funcion que es Σ-mixta para algun alfabeto Σ, entonces hay un alfabeto Σ_0 el cual es el menor de todos los alfabetos respecto de los cuales f es mixta, es decir Σ_0 cumple que f es Σ_0-mixta y si Γ es tal que f es Γ-mixta, entonces $\Sigma_0 \subseteq \Gamma$.

A continuacion daremos algunas funciones Σ-mixtas basicas las cuales seran frecuentemente usadas.

Funciones Suc **y** $Pred$**.** La *funcion sucesor* es definida por

$$\begin{array}{rcl} Suc : \omega & \to & \omega \\ n & \to & n+1 \end{array}$$

La *funcion predecesor* es definida por

$$\begin{array}{rcl} Pred : \mathbf{N} & \to & \omega \\ n & \to & n-1 \end{array}$$

Las funciones d_a**.** Sea Σ un alfabeto no vacio. Para cada $a \in \Sigma$, definamos

$$\begin{array}{rcl} d_a : \Sigma^* & \to & \Sigma^* \\ \alpha & \to & \alpha a \end{array}$$

La funcion d_a es llamada la funcion *derecha sub* a, respecto del alfabeto Σ.

Las funciones $p_i^{n,m}$**.** Sea Σ un alfabeto. Para $n, m \in \omega$ e i tal que $1 \leq i \leq n$, definamos

$$\begin{array}{rcl} p_i^{n,m} : \omega^n \times \Sigma^{*m} & \to & \omega \\ (\vec{x}, \vec{\alpha}) & \to & x_i \end{array}$$

Para $n, m \in \omega$ e i tal que $n+1 \leq i \leq n+m$, definamos

$$\begin{array}{rcl} p_i^{n,m} : \omega^n \times \Sigma^{*m} & \to & \Sigma^* \\ (\vec{x}, \vec{\alpha}) & \to & \alpha_{i-n} \end{array}$$

Las funciones $p_i^{n,m}$ son llamadas *proyecciones*. La funcion $p_i^{n,m}$ es llamada la *proyeccion* n, m, i, respecto del alfabeto Σ. Notese que esta definicion requiere que $n+m \geq 1$ ya que i debe cumplir $1 \leq i \leq n+m$. Ademas notese que siempre la funcion $p_i^{n,m}$ aplicada a una $(n+m)$-upla da la coordenada i-esima de dicha $(n+m)$-upla. Por ejemplo:

$$\begin{array}{rcl} p_3^{1,3} : \omega \times \Sigma^* \times \Sigma^* \times \Sigma^* & \to & \Sigma^* \\ (x, \alpha_1, \alpha_2, \alpha_3) & \to & \alpha_2 \end{array}$$

Las funciones $C_k^{n,m}$ **y** $C_\alpha^{n,m}$**.** Sea Σ un alfabeto. Para $n, m, k \in \omega$, y $\alpha \in \Sigma^*$, definamos

$$\begin{array}{rcl} C_k^{n,m} : \omega^n \times \Sigma^{*m} & \to & \omega \\ (\vec{x}, \vec{\alpha}) & \to & k \end{array} \qquad \begin{array}{rcl} C_\alpha^{n,m} : \omega^n \times \Sigma^{*m} & \to & \Sigma^* \\ (\vec{x}, \vec{\alpha}) & \to & \alpha \end{array}$$

Por ejemplo:

$$\begin{array}{rcl} C_3^{1,3} : \omega \times \Sigma^* \times \Sigma^* \times \Sigma^* & \to & \omega \\ (x, \alpha_1, \alpha_2, \alpha_3) & \to & 3 \end{array} \qquad \begin{array}{rcl} C_\varepsilon^{1,3} : \omega \times \Sigma^* \times \Sigma^* \times \Sigma^* & \to & \omega \\ (x, \alpha_1, \alpha_2, \alpha_3) & \to & \varepsilon \end{array}$$

Notese que $C_k^{0,0} : \{\Diamond\} \to \{k\}$ y que $C_\alpha^{0,0} : \{\Diamond\} \to \{\alpha\}$.

Ejercicio 5: V o F o I, justifique.

(a) La funcion $x+1$ es $\emptyset$-mixta

(b) La función

$$\begin{array}{rcl} \left\{(x, \alpha) \in \omega \times \{\#, \&, @\}^* : |\alpha|_\# = 0\right\} & \to & \omega \\ (x, \alpha) & \to & |\alpha| \, . x \end{array}$$

es $\{\&, @\}$-mixta

(c) f es Σ-mixta si existen $n, m \geq 0$, tales que $D_f \subseteq \omega^n \times \Sigma^{*m}$ y $I_f \subseteq \omega \cup \Sigma^*$

(d) Sea $\begin{array}{rcl} f:\omega & \to & \omega \\ x & \to & C_2^{1,0} \end{array}$. Entonces $f(5)=2$

El tipo de una funcion mixta. Dada una funcion Σ-mixta f, si $n,m\in\omega$ son tales que $D_f\subseteq\omega^n\times\Sigma^{*m}$ y ademas $I_f\subseteq\omega$, entonces diremos que f *es una funcion de tipo* $(n,m,\#)$. Similarmente si $n,m\in\omega$ son tales que $D_f\subseteq\omega^n\times\Sigma^{*m}$ y ademas $I_f\subseteq\Sigma^*$, entonces diremos que f *es una funcion de tipo* $(n,m,*)$. Notese que si $f\neq\emptyset$, entonces hay unicos $n,m\in\omega$ y $s\in\{\#,*\}$ tales que f es una funcion de tipo (n,m,s). Sin envargo $\emptyset$ es una funcion de tipo (n,m,s) cualesquiera sean $n,m\in\omega$ y $s\in\{\#,*\}$. De esta forma, cuando $f\neq\emptyset$ hablaremos de "el tipo de f" para refererirnos a esta unica terna (n,m,s). Notese que Suc es de tipo $(1,0,\#)$ y d_a es de tipo $(0,1,*)$.

Ejercicio 6: Hacer

(a) De que tipo es cada una de las siguientes funciones

(i) $C_\varepsilon^{1,2}$

(ii) $\begin{array}{rcl} \left\{(x,\alpha)\in\omega\times\{\#,\&,@\}^*:|\alpha|_\#=0\right\} & \to & \omega \\ (x,\alpha) & \to & |\alpha|\,.x \end{array}$

(iii) Id_ω

(iv) Id_{Σ^*}

(v) $\begin{array}{rcl} \Sigma^* & \to & \omega \\ \alpha & \to & |\alpha| \end{array}$

(vi) $\begin{array}{rcl} \{(\varepsilon,\varepsilon)\} & \to & \{\varepsilon\} \\ (\varepsilon,\varepsilon) & \to & \varepsilon \end{array}$

(vii) $\begin{array}{rcl} \{\Diamond\} & \to & \omega \\ \Diamond & \to & 0 \end{array}$

(b) Que significa la frase

- la relacion "f es una funcion de tipo (n,m,s)" no depende del alfabeto Σ

Intente expresar esto en forma matematica

Predicados Σ-mixtos. Un *predicado Σ-mixto* es una funcion f la cual es Σ-mixta y ademas cumple que $I_f\subseteq\{0,1\}$. Por ejemplo

$$\begin{array}{rcl} \omega\times\omega & \to & \omega \\ (x,y) & \to & \begin{cases} 1 \text{ si } x=y \\ 0 \text{ si } x\neq y \end{cases} \end{array} \qquad \begin{array}{rcl} \{1,2,3,4,5\}\times\Sigma^* & \to & \omega \\ (x,\alpha) & \to & \begin{cases} 1 \text{ si } x=|\alpha| \\ 0 \text{ si } x\neq|\alpha| \end{cases} \end{array}$$

Operaciones logicas entre predicados. Dados predicados $P:S\subseteq\omega^n\times\Sigma^{*m}\to\{0,1\}$ y $Q:S\subseteq\omega^n\times\Sigma^{*m}\to\{0,1\}$, con el mismo dominio, definamos nuevos predicados $(P\vee Q)$, $(P\wedge Q)$ y $\neg P$ de la siguiente manera

$$\begin{array}{rcl} (P\vee Q):S & \to & \omega \\ (\vec{x},\vec{\alpha}) & \to & \begin{cases} 1 & \text{si } P(\vec{x},\vec{\alpha})=1 \text{ o } Q(\vec{x},\vec{\alpha})=1 \\ 0 & \text{caso contrario} \end{cases} \end{array}$$

$$\begin{array}{rcl} (P\wedge Q):S & \to & \omega \\ (\vec{x},\vec{\alpha}) & \to & \begin{cases} 1 & \text{si } P(\vec{x},\vec{\alpha})=1 \text{ y } Q(\vec{x},\vec{\alpha})=1 \\ 0 & \text{caso contrario} \end{cases} \end{array}$$

$$\begin{array}{rcl} \neg P : S & \to & \omega \\ (\vec{x}, \vec{\alpha}) & \to & \begin{cases} 1 & \text{si } P(\vec{x}, \vec{\alpha}) = 0 \\ 0 & \text{si } P(\vec{x}, \vec{\alpha}) = 1 \end{cases} \end{array}$$

Composicion de funciones

Dadas funciones f y g definamos la funcion $f \circ g$ de la siguiente manera:

$$D_{f\circ g} = \{e \in D_g : g(e) \in D_f\}$$
$$f \circ g(e) = f(g(e))$$

Notese que

$$f \circ g = \{(e, f(g(e))) : e \in D_g \text{ y } g(e) \in D_f\}$$

(ver Convencion Notacional 3). Veamos un ejemplo. Si f es dada por

$$\begin{array}{rcl} f : \mathbf{N} & \to & \{@, \%\}^* \\ x & \to & @\%^x@ \end{array}$$

y g es dada por

$$\begin{array}{rcl} g : \mathbf{R} & \to & \mathbf{R} \\ x & \to & x^2 \end{array}$$

entonces tenemos que $f \circ g$ es la funcion

$$\begin{array}{rcl} f \circ g : \{x \in \mathbf{R} : x^2 \in \mathbf{N}\} & \to & \{@, \%\}^* \\ x & \to & @\%^{x^2}@ \end{array}$$

Notar que $f \circ g = \{(u, v) : \text{existe } z \text{ tal que } (u, z) \in g \text{ y } (z, v) \in f\}$ (por una prueba ver el apunte).

Ejercicio 7: V o F o I, justifique

(a) $Pred = Pred \circ (Pred \circ Suc)$
(b) $Pred \circ (Suc \circ Pred) = Pred$
(c) $Pred \circ (Suc \circ \{(x, x) : x \in \mathbf{N}\}) = Pred \circ Suc$
(d) $\emptyset \circ f = f \circ \emptyset = \emptyset$ cualquiera sea la funcion f
(e) Sea Σ un alfabeto finito. Si $x_1, x_2, x_3, x_4, x_5 \in \omega$ se tiene que $(Suc \circ p_2^{5,0})(x_1, x_2, x_3, x_4, x_5) = x_3$
(f) Sea Σ un alfabeto finito. Entonces $Suc \circ Pred = p_1^{1,0}$
(g) $Suc \circ x = Suc$
(h) $Suc \circ 4 = 5$
(i) Sea Σ un alfabeto finito. Entonces $\emptyset = Pred \circ C_0^{0,0}$
(j) Si $f : D_f \subseteq \omega \to \omega$ y $g : D_g \subseteq \omega \to \omega$, entonces $D_{f\circ g} = \{x \in \omega : x \in D_g \text{ y } I_g \subseteq D_f\}$

A la hora de probar enunciados acerca de funciones hay una regla o idea basica que si la tenemos en cuenta nos facilitara la construccion de la prueba.

Regla Pertenecer a la Imagen: Si f es una funcion y ud sabe que $b \in I_f$, entonces escriba a b en la forma $f(a)$ donde a denotara un elemento fijo de D_f tal que $f(a) = b$

Muchas veces tener esta regla en mente es de suma utilidad al hacer pruebas. Por ejemplo el lector puede usarla para hacer una prueba rigurosa del ($\Leftarrow$) del enunciado del siguiente ejercicio. Esa regla aqui es simplemente un consejo o sugerencia pero

gana su existencia material en un entorno de inteligencia artificial al transformarse en parte de la estructura de un probador automatico de teoremas!

Ejercicio 8: Pruebe que $f \circ g \neq \emptyset$ si y solo si $I_g \cap D_f \neq \emptyset$ (esta probado en el apunte). Notese que este resultado nos dice que que muchas veces sucedera que $f \circ g = \emptyset$.

Funciones de la forma $[f_1, ..., f_n]$

Dadas funciones $f_1, ..., f_n$, con $n \geq 2$, definamos la funcion $[f_1, ..., f_n]$ de la siguiente manera:

$$D_{[f_1,...,f_n]} = D_{f_1} \cap ... \cap D_{f_n}$$
$$[f_1, ..., f_n](e) = (f_1(e), ..., f_n(e))$$

Notese que $I_{[f_1,...,f_n]} \subseteq I_{f_1} \times \cdots \times I_{f_n}$. Por conveniencia notacional (que el lector entendera mas adelante) definiremos $[f_1] = f_1$. Es decir que hemos definido para cada sucecion de funciones $f_1, ..., f_n$, con $n \geq 1$, una nueva funcion la cual denotamos con $[f_1, ..., f_n]$.

Ejercicio 9: V o F o I, justifique

(a) Sea Σ un alfabeto y supongamos $\# \in \Sigma$. Entonces
$$p_4^{2,3} \circ [p_1^{1,1}, p_1^{1,1}, p_2^{1,1}, C_{\#\#}^{1,1}, p_2^{1,1}] = C_{\#\#}^{1,1}$$
(b) Si $f : \omega^2 \to \omega$, entonces $f = f \circ [x, y]$
(c) $[p_2^{2,3}, Suc] = \emptyset$
(d) Supongamos $f_i : \omega \to \omega$, para $i \in \{1, ..., n\}$, con $n \geq 2$. Entonces $I_{[f_1,...,f_n]} = I_{f_1} \times \cdots \times I_{f_n}$

Funciones inyectivas, suryectivas y biyectivas

Una funcion f es *inyectiva* cuando no se da que $f(a) = f(b)$ para algun par de elementos distintos $a, b \in D_f$. Dicho de otra manera, f sera inyectiva cuando se de la implicacion

$$f(a) = f(b) \text{ implica } a = b$$

cualesquiera sean $a, b \in D_f$. Dada una funcion $f : A \to B$ diremos que f es *suryectiva* cuando $I_f = B$. Debe notarse que el concepto de suryectividad depende de un conjunto de llegada previamente fijado, es decir que no tiene sentido hablar de la suryectividad de una funcion f si no decimos respecto de que conjunto de llegada lo es. Muchas veces diremos que una funcion f es *sobre* para expresar que es suryectiva.

Dada una funcion $f : A \to B$ diremos que f es *biyectiva* cuando f sea inyectiva y suryectiva. Tambien diremos que f es una *biyeccion de A en B* cuando $f : A \to B$ sea biyectiva. Notese que si $f : A \to B$ es biyectiva, entonces para cada $b \in B$ hay un unico $a \in A$ tal que $f(a) = b$. Entonces cuando $f : A \to B$ es biyectiva podemos definir una nueva funcion $f^{-1} : B \to A$, de la siguiente manera:

$$f^{-1}(b) = \text{ unico } a \in A \text{ tal que } f(a) = b$$

La funcion f^{-1} sera llamada la *inversa de f*. Notese que $f \circ f^{-1} = Id_B$ y $f^{-1} \circ f = Id_A$. El siguiente lema muestra que esta ultima propiedad caracteriza la inversa.

Ejercicio 10: V o F o I, justifique.

(a) Una función f es inyectiva si $f(x) = f(y)$ cada vez que $x = y$

(b) $F : A \to B$ es suryectiva sii para cada $a \in A$ existe un $b \in B$ tal que $b = F(a)$

Ejercicio 11: Hacer:

(a) Dar una biyeccion entre $\mathbf{N}$ y ω. Idem entre ω y $\{x \in \omega : x \text{ es par}\}$

(b) Dar una funcion inyectiva de ω^2 en ω

(c) Dar una funcion sobreyectiva de ω en ω^5

Lema 3. *Supongamos $f : A \to B$ y $g : B \to A$ son tales que $f \circ g = Id_B$ y $g \circ f = Id_A$. Entonces f y g son biyectivas, $f^{-1} = g$ y $g^{-1} = f$.*

Ejercicio 12: Haga una prueba del lema anterior (esta probado en el apunte)

Conjuntos Σ-mixtos

Un conjunto S es llamado Σ-*mixto* si existen $n, m \in \omega$ tales que $S \subseteq \omega^n \times \Sigma^{*m}$. Por ejemplo,

$$\{(x, \alpha) \in \omega \times \{▲, !\}^* : |\alpha| = x\}$$
$$\{(0, ▲▲▲, \varepsilon), (1, \%▲\%, ▲▲)\}$$

son conjuntos $\{▲, \%, !\}$-mixtos. Tambien notese que $\emptyset$ y $\{\Diamond\}$ son conjuntos Σ-mixtos, cualesquiera sea el alfabeto Σ. Por ultimo el conjunto

$$\{(x, \varepsilon, \varepsilon, \varepsilon) : x \in \omega \text{ y } x \text{ es impar}\}$$

es $\emptyset$-mixto (con $n = 1$ y $m = 3$).

Ejercicio 13: V o F o I, justifique.

(a) Un conjunto S es Σ-mixto sii $S = D_f$ para alguna función Σ-mixta f

(b) $\{(1, 2, \varepsilon), (1, 2)\}$ es un conjunto Σ-mixto, cualesquiera sea el alfabeto finito Σ

El tipo de un conjunto mixto. Dado un conjunto Σ-mixto S, si $n, m \in \omega$ son tales que $S \subseteq \omega^n \times \Sigma^{*m}$, entonces diremos que S *es un conjunto de tipo* (n, m). Notese que si $S \neq \emptyset$, entonces hay unicos $n, m \in \omega$ tales que S es un conjunto de tipo (n, m). De esta forma, cuando $S \neq \emptyset$ hablaremos de "el tipo de S" para refererirnos a este unico par (n, m). Tambien es importante notar que de la definicion anterior sale inmediatemante que $\emptyset$ es un conjunto de tipo (n, m) cualesquiera sean $n, m \in \omega$, por lo cual cuando hablemos de EL tipo de un comjunto deberemos estar seguros de que dicho conjunto es no vacio.

Notese que ω es de tipo $(1, 0)$ y Σ^* es de tipo $(0, 1)$.

Ejercicio 14: Hacer

(a) De que tipo es cada uno de los siguientes conjuntos

(i) $\left\{(x, \alpha) \in \omega \times \{\#, \&, @\}^* : |\alpha|_{\#} = 0\right\}$

(ii) $\{1, 2, 3\}$

(iii) $\{\varepsilon\}$

(iv) $\{\Diamond\}$

(v) $\{(1, \varepsilon)\}$
(vi) $\{(\varepsilon, \varepsilon)\}$

(b) Para el caso $\Sigma = \emptyset$, describa para un $m \in \omega$ dado, como son los conjuntos no vacios de tipo $(0, m)$.
(c) Que significa la frase
- la relacion "S es un conjunto de tipo (n, m)" no depende del alfabeto Σ

Intente expresar esto en forma matematica

Notacion lambda

Usaremos la notacion lambda de Church en la forma que se explica a continuacion. Esta notacion siempre depende de un alfabeto finito previamente fijado. En general en nuestro lenguaje matematico utilizamos diversas expresiones las cuales involucran variables que una vez fijadas en sus valores hacen que la expresion tambien represente un determinado valor u objeto.

En el contexto de la notacion lambda solo se podran utilizar expresiones con caracteristicas muy especiales por lo cual a continuacion iremos describiendo que condiciones tienen que cumplir las expresiones para que puedan ser usadas en la notacion lambda.

(1) Solo utilizaremos expresiones que involucran variables numericas, las cuales se valuaran en numeros de ω, y variables alfabeticas, las cuales se valuaran en palabras del alfabeto previamente fijado. Las variables numericas seran seleccionadas de la lista

$$x, y, z, w, n, m, k, ...$$
$$x_1, x_2, ...$$
$$y_1, y_2, ...$$
$$etc$$

Las variables alfabeticas seran seleccionadas de la lista

$$\alpha, \beta, \gamma, \eta, ...$$
$$\alpha_1, \alpha_2, ...$$
$$\beta_1, \beta_2, ...$$
$$etc$$

(2) Por ejemplo la expresion:

$$x + y + 1$$

tiene dos variables numericas x e y (y ninguna alfabetica). Si le asignamos a x el valor 2 y a y el valor 45, entonces la expresion $x + y + 1$ produce o representa el valor $48 = 2 + 45 + 1$.

(3) Otro ejemplo, consideremos la expresion

$$|\alpha\beta| + |\alpha|^x$$

la cual tiene una variable numerica x y dos variables alfabeticas α y β. Supongamos ademas que el alfabeto previamente fijado es $\{@, \%\}$. Si le asignamos a x el valor 2, a α el valor @@ y a β el valor %%%, entonces la expresion $|\alpha\beta| + |\alpha|^x$ produce o representa el valor $|@@\%\%\%| + |@@|^2 = 9$.

(4) Para ciertas valuaciones de sus variables la expresion puede no estar definida en el sentido que cuando reemplazamos en ella dichos valores la palabra obtenida no representa en forma precisa un objeto. Por ejemplo la expresion

$$Pred(|\alpha|)$$

cuando le asignamos a α la palabra ε, nos queda la expresion $Pred(|\varepsilon|)$ la cual no representa un objeto matematico en forma precisa. Otro ejemplo, consideremos la expresion

$$x/(y - |\alpha|)^2$$

Esta expresion no esta definida o no asume valor para aquellas asignaciones de valores a sus variables en las cuales el valor asignado a y sea igual a la longitud del valor asignado a α. Un ultimo ejemplo, la expresion

$$x/y/z$$

no esta definida en forma precisa cuando le damos a x, y, z los valores $100, 10, 5$ ya que nos queda la expresion $100/10/5$ la cual es imprecisa puesto que es ambigua ya que no sabemos en que orden dividir.

(5) En los ejemplos anteriores las expresiones producen valores numericos pero tambien trabajaremos con expresiones que producen valores alfabeticos. Por ejemplo la expresion

$$\beta^y$$

tiene una variable numerica, y, una variable alfabetica, β, y una vez valuadas estas variables produce un valor alfabetico, a saber el resultado de elevar el valor asignado a la variable β, a el valor asignado a y.

(6) Tambien consideraremos expresiones en las cuales no ocurren variables, es decir ellas representan un valor concreto. Por ejemplo la expresion

$$5$$

siempre produce el valor 5. O la expresion

$$17 + 11$$

siempre produce el valor 28. Tambien la expresion

$$1/0$$

no tiene variables y ademas es siempre indefinida. Es decir no representa un valor u objeto.

(7) Una expresion E para poder ser utilizada en la notacion lambda relativa a un alfabeto Σ debera cumplir alguna de las dos siguientes propiedades
   (a) los valores que asuma E cuando hayan sido asignados valores de ω a sus variables numericas y valores de Σ^* a sus variables alfabeticas deberan ser siempre elementos de ω
   (b) los valores que asuma E cuando hayan sido asignados valores de ω a sus variables numericas y valores de Σ^* a sus variables alfabeticas deberan ser siempre elementos de Σ^*

   Cabe destacar que la expresion E puede, para alguna asignacion de valores de ω a sus variables numericas y valores de Σ^* a sus variables alfabeticas, no estar definida o representar en forma precisa algun objeto matematico.

(8) Por ejemplo la expresion
$$x/2$$
no cumple la propiedad dada en (7) ya que para ciertos valores de ω asignados a la variable x, la expresion da valores numericos que se salen de ω por lo cual no cumple ni (a) ni (b).

(9) Otro ejemplo, si el alfabeto fijado es $\Sigma = \{@, \%\}$, entonces la expresion
$$@^x\$^y$$
no cumple la propiedad dada en (7) ya que por ejemplo cuando le asignamos a x el valor 2 y a y el valor 6, la expresion nos da la palabra @@$$$$$$ la cual no pertenece a Σ^* por lo cual no cumple ni (a) ni (b).

(10) No necesariamente las expresiones que usaremos en la notacion lambda deben ser hechas como combinacion de operaciones matematicas conocidas. Muchas veces usaremos expresiones que involucran incluso lenguaje coloquial castellano. Por ejemplo la expresion
$$\text{el menor numero primo que es mayor que } x$$
Es claro que esta expresion para cada valor de ω asignado a la variable x produce o representa un valor concreto de ω. Otro ejemplo:
$$\text{el tercer simbolo de } \alpha$$
notese que esta expresion, una ves fijado un alfabeto Σ, estara definida o producira un valor solo cuando le asignamos a α una palabra de Σ^* de longitud mayor o igual a 3.

(11) **Expresiones Booleanas.** A las expresiones Booleanas tales como
$$x = y + 1 \text{ y } |\alpha| \leq 22$$
las pensaremos que asumen valores del conjunto $\{0, 1\} \subseteq \omega$. Por ejemplo la expresion anterior asume o produce el valor 1 cuando le asignamos a x el valor 11, a y el valor 10 y a α la palabra ε. Las expresiones Booleanas pensadas de esta forma podran ser utilizadas en la notacion lambda si es que tambien cumplen con las anteriores condiciones. Otro ejemplo
$$11 = 17$$
es una expresion Booleana que no tiene variables y siempre produce el valor 0.

(12) Seremos muy estrictos en lo que respecta a cuando "una expresion E esta definida (o representa o produce) en forma precisa un valor, para una valuacion dada de sus variables". El criterio sera similar al usado en la tombola con los vofois en el sentido que la mas minima impresicion ya implicara que para esa valuacion la expresion no esta definida. Algunos ejemplos:

(a) Consideremos la expresion
$$0.Suc(z, z)$$
Uno podria pensar que cualquiera sea el valor de ω asignado a la variable z, la expresion produce el valor 0, ya que cualquiera sea el valor de $Suc(z, z)$, al multiplicarlo por 0 nos dara 0. Sin embargo para nosotros la expresion $0.Suc(z, z)$ no estara definida en forma precisa cualquiera sea el valor que le asignemos a z y esto es porque $Suc(z, z)$ tiene una imprecision ya que Suc no se aplica a pares ordenados.

(b) Consideremos la expresion

$$0.(4/x/2)$$

Esta expresion no estara definida en forma precisa para ningun valor de x ya que la expresion $4/x/2$ es ambigua puesto que no aclara en que orden se realizan las divisiones. Notese que aunque al multiplicar por 0 podriamos pensar que la expresion produce siempre 0, optamos por convenir que la expresion nunca produce en forma precisa valores u objetos.

(c) Consideremos la expresion Booleana

$$Pred(0) = 1 \text{ o } 5 \leq x$$

Uno podria pensar que esta expresion produce el valor 1 cuando le asignamos a x un valor mayor o igual a 5 ya que independientemente de que signifique $Pred(0) = 1$ se hace verdadero que $5 \leq x$ por lo cual sera verdadero el "o" de ambos enunciados. Sin embargo esta expresion no esta definida en forma precisa para ninguna asignacion de valores a x ya que $Pred(0)$ es una imprecision que es parte de la expresion. (O sea sucede lo mismo que en los vofois).

(d) Consideremos la expresion Booleana

$$(\forall t \in \emptyset)\ Pred(x).t \text{ es impar}$$

Uno podria pensar que esta expresion produce el valor 1 independientemente de cuanto vale x ya que debemos chequear que $Pred(x).t$ sea impar para 0 valores posibles de t. Sin embargo esta expresion no esta definida en forma precisa para el caso en que hacemos valer 0 a x ya que en este caso $Pred$ no esta definida. Obviamente para cualquier valor de $\mathbf{N}$ que asignemos a x la expresion produce en forma precisa el valor 1

Expresiones lambdificables con respecto a Σ. Dado un alfabeto Σ a las expresiones que cumplan las caracteristicas dadas anteriormente las llamaremos *lambdificables con respecto a* Σ. Notese que este concepto es intuitivo y no un concepto matematicamente definido en forma precisa. Mas aun el concepto de expresion tampoco ha sido definido matematicamente (aunque obviamente si sabemos que una expresion es una palabra de cierto alfabeto). Esto no nos traera problemas para el uso notacional que las utilizaremos. Recien en las secciones de logica veremos la matematizacion de ciertas expresiones (no las lambdificables) y nos servira de ejemplo para imaginar como podriamos matematizar el concepto de expresion.

Algunos ejemplos:

(E1) $x/2$ no es lambdificable con respecto a Σ cualesquiera sea Σ

(E2) $@^x\$^y$ es lambdificable con respecto a $\{@, \$\}$ y no es lambdificable con respecto a $\{@, \#, \%\}$

(E3) $x = y + 1$ es lambdificable con respecto a Σ cualesquiera sea Σ

(E4) la expresion

$$\text{el menor numero primo que es mayor que } x^{|\beta|}$$

es lambdificable con respecto a Σ cualesquiera sea Σ

(E5) la expresion

$$5$$

es lambdificable con respecto a Σ cualesquiera sea Σ

(E6) la expresion

$$Suc(x/20)$$

es lambdificable con respecto a Σ cualesquiera sea Σ. Notese que esta expresion no esta definida o no asume valor para aquellas asignaciones de valores a x en las cuales $x/20$ no sea un elemento de ω ya que en estos casos $x/20$ no pertenece al dominio de Suc. Mas concretamente dicha expresion esta definida o produce un valor cuando le asignamos a x un valor multiplo de 20. Notese que sin enbargo, la expresion

$$(x/20)+1$$

no es lambdificable con respecto a Σ cualesquiera sea Σ (por que?).

Definicion de $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$**.** Supongamos ya hemos fijado un alfabeto finito Σ y supongamos E es una expresion la cual es lambdificable con respecto a Σ. Sea $x_1,...,x_n,\alpha_1,...,\alpha_m$ (con $n,m\in\omega$) una lista de variables todas distintas tal que las variables numericas que ocurren en E estan todas contenidas en la lista $x_1,...,x_n$ y las variables alfabeticas que ocurren en E estan en la lista $\alpha_1,...,\alpha_m$ (puede suceder que haya variables de la lista $x_1,...,x_n,\alpha_1,...,\alpha_m$ las cuales no ocurran en E).g Entonces

$$\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$$

denotara la funcion definida por:

(L1) El dominio de $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$ es el conjunto de las $(n+m)$-uplas $(k_1,...,k_n,\beta_1,...,\beta_m)\in\omega^n\times\Sigma^{*m}$ tales que E esta definida en forma precisa cuando le asignamos a cada x_i el valor k_i y a cada α_i el valor β_i.

(L2) $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]\,(k_1,...,k_n,\beta_1,...,\beta_m)=$ valor que asume, produce o representa E cuando le asignamos a cada x_i el valor k_i y a cada α_i el valor β_i.

Notese que por tener E la propiedad (7) de mas arriba, la funcion $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$ es Σ-mixta de tipo (n,m,s) para algun $s\in\{\#,*\}$. Tambien notese que cuando $n=m=0$ la expresion E debera no tener variables y $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$ pasara a ser $\lambda\,[E]$. Ademas $\lambda\,[E]$ sera la funcion vacia cuando E no produsca o represente un valor y en caso contrario $\lambda\,[E]$ tendra dominio igual a $\{\Diamond\}$. Algunos ejemplos:

(a) Supongamos fijamos el alfabeto $\Sigma=\{@,?,¡\}$. Entonces $\lambda x\alpha\left[\alpha^{2x}\right]$ es la funcion

$$\begin{array}{rcl}\omega\times\{@,?,¡\}^* & \to & \{@,?,¡\}^*\\ (x,\alpha) & \to & \alpha^{2x}\end{array}$$

Aqui el lector puede notar la dependencia de la notacion lambda respecto del alfabeto fijado. Si en lugar de fijar $\Sigma=\{@,?,¡\}$ hubieramos fijado $\Sigma=\{\%\}$, entonces $\lambda x\alpha\left[\alpha^{2x}\right]$ denotaria otra funcion, a saber

$$\begin{array}{rcl}\omega\times\{\%\}^* & \to & \{\%\}^*\\ (x,\alpha) & \to & \alpha^{2x}\end{array}$$

(b) Supongamos fijamos el alfabeto $\Sigma = \{@, ?, ¡\}$. Entonces $\lambda x \alpha\,[5]$ es la funcion

$$\begin{array}{rcl} \omega \times \{@, ?, ¡\}^* & \to & \omega \\ (x, \alpha) & \to & 5 \end{array}$$

(c) Supongamos fijamos el alfabeto $\Sigma = \{\%, !\}$. Entonces $\lambda \alpha \beta\,[\alpha\beta]$ es la funcion

$$\begin{array}{rcl} \{\%, !\}^* \times \{\%, !\}^* & \to & \{\%, !\}^* \\ (\alpha, \beta) & \to & \alpha\beta \end{array}$$

Tambien tenemos que $\lambda \beta \alpha\,[\alpha\beta]$ es la funcion

$$\begin{array}{rcl} \{\%, !\}^* \times \{\%, !\}^* & \to & \{\%, !\}^* \\ (\beta, \alpha) & \to & \alpha\beta \end{array}$$

Notese que estas funciones son distintas. Por ejemplo $\lambda \alpha \beta\,[\alpha\beta]\,(\%, !) = \%!$ y $\lambda \beta \alpha\,[\alpha\beta]\,(\%, !) = !\%$

(d) Independientemente de quien sea Σ el alfabeto previamente fijado, tenemos que $\lambda xy[x + y]$ es la funcion

$$\begin{array}{rcl} \omega^2 & \to & \omega \\ (x, y) & \to & x + y \end{array}$$

Tambien $\lambda xyzw[x + w]$ es la funcion

$$\begin{array}{rcl} \omega^4 & \to & \omega \\ (x, y, z, w) & \to & x + w \end{array}$$

(e) Supongamos fijamos el alfabeto $\Sigma = \{@, ?, ¡\}$. Entonces por la clausula (L1) tenemos que el dominio de la funcion $\lambda xy\alpha\beta\,[Pred(|\alpha|) + Pred(y)]$ es

$$D = \left\{(x, y, \alpha, \beta) \in \omega^2 \times \Sigma^{*2} : |\alpha| \geq 1 \text{ y } y \geq 1\right\}$$

Es decir que $\lambda xy\alpha\beta\,[Pred(|\alpha|) + Pred(y)]$ es la funcion

$$\begin{array}{rcl} D & \to & \omega \\ (x, y, \alpha, \beta) & \to & Pred(|\alpha|) + Pred(y) \end{array}$$

(f) Atentos a (11) de mas arriba, la funcion $\lambda xy\,[x = y]$ es el predicado

$$\begin{array}{rcl} \omega \times \omega & \to & \omega \\ (x, y) & \to & \begin{cases} 1 \text{ si } x = y \\ 0 \text{ si } x \neq y \end{cases} \end{array}$$

y $\lambda x \alpha\,[Pred(x) = |\alpha|]$ es el predicado

$$\begin{array}{rcl} \mathbf{N} \times \Sigma^* & \to & \omega \\ (x, \alpha) & \to & \begin{cases} 1 \text{ si } Pred(x) = |\alpha| \\ 0 \text{ si } Pred(x) \neq |\alpha| \end{cases} \end{array}$$

Tambien $\lambda \alpha \beta\,[\alpha = \beta]$ es el predicado

$$\begin{array}{rcl} \Sigma^* \times \Sigma^* & \to & \omega \\ (\alpha, \beta) & \to & \begin{cases} 1 \text{ si } \alpha = \beta \\ 0 \text{ si } \alpha \neq \beta \end{cases} \end{array}$$

(g) Notar que para $S \subseteq \omega^n \times \Sigma^{*m}$ se tiene que $\chi_S^{\omega^n \times \Sigma^{*m}} = \lambda x_1 ... x_n \alpha_1 ... \alpha_m\,[(\vec{x}, \vec{\alpha}) \in S]$

(h) Como dijimos, la notacion lambda depende del alfabeto previamnete fijado, aunque para el caso en que la lista de variables que sigue a la letra λ no tenga variables alfabeticas, la funcion representada no depende del alfabeto.

(i) La funcion $\lambda x\,[Suc(x/20)]$ es la siguiente funcion:

$$\begin{array}{rcl} \{x \in \omega : 20 \text{ divide a } x\} & \to & \omega \\ x & \to & x+1 \end{array}$$

(j) La funcion $\lambda\,[5]$ es la funcion

$$\begin{array}{rcl} \{\Diamond\} & \to & \omega \\ \Diamond & \to & 5 \end{array}$$

(k) La funcion $\lambda\,[1/0]$ es la funcion vacia, es decir $\lambda\,[1/0] = \emptyset$

(l) La funcion $\lambda\,[11 = 17]$ es la funcion

$$\begin{array}{rcl} \{\Diamond\} & \to & \omega \\ \Diamond & \to & 0 \end{array}$$

Algunos ejemplos sutiles.

(a) La expresion

$$Suc$$

no es lambdificable respecto de cualquier alfabeto Σ. Esto es porque si bien cualesquiera sea el valor asignado a las variables, ella asume el valor Suc (ya que no tiene variables), no cumple (6) de mas arriba ya que Suc no es un elemento de ω ni tampoco una palabra (es una funcion!)

(b) La expresion

$$Suc + (|\beta| + 1)$$

es lambdificable con respecto a Σ cualesquiera sea Σ. Por ejemplo $\lambda x\beta[Suc+(|\beta|+1)]$ es la funcion $\emptyset$, ya que la expresion $Suc + (|\beta| + 1)$ cualesquiera sean los valores de x y β no esta definida.

(c) La expresion

$$Suc + 1$$

es lambdificable con respecto a Σ cualesquiera sea Σ ya que no esta definida nunca y obtenemos entonces que $\lambda x_1...x_n\alpha_1...\alpha_m[Suc + 1]$ es la funcion $\emptyset$, cualesquiera sean las variables $x_1...x_n\alpha_1...\alpha_m$. En particular $\lambda[Suc+1] = \emptyset$.

Ejercicio 15: V o F o I, justifique

(a) La expresion

$$|\alpha\#@@| + x$$

no es lambdificable con respecto a $\{\#, \%\}$

(b) La expresion

$$x + 1 = 1/3$$

es lambdificable con respecto a Σ cualesquiera sea Σ

(c) La expresion

$$\lambda x[x^2] + (|\beta| + 1)$$

es lambdificable con respecto a Σ cualesquiera sea Σ

Ejercicio 16: V o F o I, justifique

(a) $\lambda xy[x + y] = \lambda yx[x + y]$

(b) Si $f : \Sigma^{*2} \to \omega$, entonces $\lambda\alpha\beta[f(\alpha, \beta)] = \lambda\beta\alpha[f(\beta, \alpha)]$

(c) $\lambda xy\alpha\beta\left[Pred(|\alpha|) + Pred(y)\right]$ es la función

$$\begin{array}{rcl} \left\{(x,y,\alpha,\beta) \in \omega^2 \times \Sigma^{*2} : |\alpha|\,.y \neq 0\right\} & \to & \omega \\ (x,y,\alpha,\beta) & \to & (|\alpha| + y) - 2 \end{array}$$

(d) $D_{\lambda xy[x^2]} = \omega$
(e) $\lambda x[Pred(x).0] = C_0^{1,0}$
(f) $Suc = \lambda x[Suc]$
(g) $\lambda xy[x.y] \circ [\lambda xy[x.y], C_1^{2,0}] = \lambda xy[x.y]$
(h) Sea $\Sigma = \{\bigtriangledown, \square\}$. Entonces $\lambda\alpha\beta[\alpha = \square\beta] = \lambda\alpha\beta[\alpha = \beta] \circ \left[p_1^{0,2}, \lambda\alpha\beta[\alpha\beta] \circ \left[d_\square \circ C_\varepsilon^{0,0}, p_2^{0,2}\right]\right]$

Ordenes totales

Sea A un conjunto. Recordemos que una *relacion binaria sobre* A es un subconjunto de A^2. Algunos ejemplos:

(E1) Sea $R = \{(1,2),(2,3)\}$. Entonces R es una relacion binaria sobre $\mathbf{N}$.
(E2) Sea $R = \{(x,y) \in \omega^2 : x \text{ divide a } y\}$. Entonces R es una relacion binaria sobre ω.
(E3) Sea $R = \{(r,t) \in \mathbf{R}^2 : r \leq t\}$. Entonces R es una relacion binaria sobre $\mathbf{R}$
(E4) $\emptyset$ es una relacion binaria sobre A, cualesquiera sea el conjunto A.
(E5) Sea $R = \{(x,y) \in \omega^2 : x < y \text{ o } y = 0\}$. Entonces R es una relacion binaria sobre ω

Notese que si R es una relacion binaria sobre A y $A \subseteq B$ entonces R es una relacion binaria sobre B. Por ejemplo las relaciones dadas en los ejemplos (E1), (E2), (E4) y (E5) tambien son relaciones binarias sobre $\mathbf{R}$

Como es usual, cuando R sea una relacion binaria sobre un conjunto A, algunas veces escribiremos aRb en lugar de $(a,b) \in R$.

Recordemos que una relacion binaria R sobre un conjunto A es llamada un *orden parcial sobre* A si cumple las siguientes tres propiedades:

Reflexividad xRx, cualesquiera sea $x \in A$
Transitividad xRy y yRz implica xRz, cualesquiera sean $x, y, z \in A$
Antisimetria xRy y yRx implica $x = y$, cualesquiera sean $x, y \in A$

Algunos ejemplos:

(E1) $\{(r,t) \in \mathbf{R}^2 : r \leq t\}$ es un orden parcial sobre $\mathbf{R}$, llamado el orden usual de $\mathbf{R}$
(E2) Sea $R = \{(1,2),(1,3),(1,1),(2,2),(3,3)\}$. Entonces R es un orden parcial sobre $\{1,2,3\}$
(E3) Sea $R = \{(S,T) \in \mathcal{P}(\omega)^2 : S \subseteq T\}$. Entonces R es un orden parcial sobre $\mathcal{P}(\omega)$
(E4) $\{(x,y) \in \omega^2 : x \leq y\}$ es un orden parcial sobre ω.
(E5) Sea $R = \{(1,1)\}$. Entonces R es un orden parcial sobre $\{1\}$.
(E6) $\{(a,b) \in A^2 : a = b\}$ es un orden parcial sobre A, cualesquira sea el conjunto A
(E7) Sea $\leq = \{(n,m) \in \mathbf{N}^2 : n \mid m\}$. Es facil ver que $\leq$ es un orden parcial sobre $\mathbf{N}$

Notese que las relaciones dadas en (E1) y (E4) son distintas, ademas

Ejercicio 17: Es la relacion dada en (E4) un orden parcial sobre $\mathbf{R}$?

Muchas veces denotaremos con $\leq$ a una relacion binaria que sea un orden parcial. Esto hace mas intuitiva nuestra escritura pero siempre hay que tener en cuenta que $\leq$ en estos casos esta denotando cierto conjunto de pares ordenados previamente definido.

Usaremos la siguiente

Convencion notacional Si hemos denotado con $\leq$ a cierto orden parcial sobre un conjunto A, entonces

- Denotaremos con $<$ a la relacion binaria $\{(a,b) \in A^2 : a \leq b \text{ y } a \neq b\}$. Es decir que $< = \{(a,b) \in A^2 : a \leq b \text{ y } a \neq b\}$. Cuando se de $a < b$ diremos que a *es menor que* b o que b *es mayor que* a (*respecto de* $\leq$)

Por ejemplo, si $A = \{1,2,3,4\}$ y $\leq = \{(1,2),(2,3),(1,3),(1,1),(2,2),(3,3),(4,4)\}$, entonces $< = \{(1,2),(2,3),(1,3)\}$.

Ahora sí estamos en condiciones de definir orden total. Sea A un conjunto cualquiera. Por un *orden total sobre* A entenderemos un orden parcial $\leq$ sobre A el cual cumple:

(C) $a \leq b$ o $b \leq a$, cualesquiera sean $a, b \in A$

Ejercicio 18: Decida cuáles ordenes parciales de la lista de ejemplos (E1)-(E7) son ordenes totales.

Supongamos A es finito, no vacio, y que $\leq$ es un orden total sobre A. La propiedad (C) nos permite probar que para cada conjunto no vacio $S \subseteq A$, hay un elemento $s \in S$ el cual cumple $s \leq s'$ para cada $s' \in S$. Por supuesto, s es unico (por que?) y habitualmente es llamado el *menor elemento de* S, ya que es menor que todo otro elemento de S.

Si A es finito, no vacio, y $\leq$ es un orden total sobre A, podemos definir recursivamente una funcion $f : \{1, ..., |A|\} \to A$ de la siguiente manera:

- $f(1) =$ menor elemento de A
- Si $i \in \{1, ..., |A| - 1\}$, entonces
  - $f(i+1) =$ menor elemento de $A - \{f(1), ..., f(i)\}$

Como es habitual, $f(i)$ es llamado el *i-esimo elemento de* A.

Muchas veces para dar un orden total sobre un conjunto finito A, daremos simplemente sus elementos en forma creciente ya que esto determina el orden por completo. Por ejemplo si $A = \{1,2,3\}$, el orden total dado por $2 < 1 < 3$ es la relacion $\leq = \{(2,1),(1,3),(2,3),(1,1),(2,2),(3,3)\}$.

Ejercicio 19: Sea A un conjunto finito de n elementos, con $n > 0$. Encuentre una biyección entre $\{R : R \text{ es un orden total sobre } A\}$ y $\{(a_1, \ldots, a_n) \in A^n : a_i \neq a_j \text{ para } i \neq j\}$. ¿Por qué este resultado se puede considerar informático?

GUIA 2 DE LENGUAJES: DOS CODIFICACIONES IMPORTANTES

Veremos dos formas de codificar objetos matematicos con numeros. En la primera codificaremos ciertas infinituplas de elementos de ω con numeros naturales y en la segunda codificaremos con elementos de ω a las palabras de un alfabeto finito Σ para el cual tenemos un orden total dado.

Codificacion de infinituplas de numeros

Usaremos $\omega^{\mathbf{N}}$ para denotar el conjunto de todas las infinituplas con coordenadas en ω. Es decir

$$\omega^{\mathbf{N}} = \{(s_1, s_2, ...) : s_i \in \omega, \text{ para cada} i \geq 1\}.$$

Definamos el siguiente subconjunto de $\omega^{\mathbf{N}}$

$$\omega^{[\mathbf{N}]} = \left\{(s_1, s_2, ...) \in \omega^{\mathbf{N}} : \text{ hay un } n \in \mathbf{N} \text{ tal que } s_i = 0, \text{ para } i \geq n\right\}.$$

Notese que $\omega^{\mathbf{N}} \neq \omega^{[\mathbf{N}]}$, por ejemplo las infinituplas

$$(10, 20, 30, 40, 50, ...)$$
$$(1, 0, 1, 0, 1, 0, 1, 0, ...)$$

no pertenecen a $\omega^{[\mathbf{N}]}$. Notese que $(s_1, s_2, ...) \in \omega^{[\mathbf{N}]}$ si y solo si solo una cantidad finita de coordenadas de $(s_1, s_2, ...)$ son no nulas (i.e. $\{i : s_i \neq 0\}$ es finito).

Definamos

$$\begin{array}{rccl} pr : \mathbf{N} & \rightarrow & \omega \\ n & \rightarrow & n\text{-esimo numero primo} \end{array}$$

Nótese que $pr(1) = 2$, $pr(2) = 3$, $pr(3) = 5$, etc.

Es bien conocido que todo numero natural es expresable como producto de primos. Por ejemplo si tomamos $x = 57596$ tenemos que $x = 2.2.7.11.11.17$. Tambien es un hecho conocido que dicha representacion en producto de primos es unica, si escribimos a los factores primos de menor a mayor, tal como lo hicimos recien con el numero 57596. El Teorema Fundamental de la Aritmetica justamente acevera esta propiedad de factorisacion unica de todo numero natural. Trataremos de escribir este teorema de una forma un poco mas "cheta".

Ya que $57596 = 2.2.7.11.11.17$, podemos escribir

$$57596 = pr(1)^2.pr(4)^1.pr(5)^2.pr(7)^1$$

Notese que ahora cada primo que interviene en la factorizacion de 57596 figura con un exponente que nos dice cuantas veces ocurre en dicha factorizacion. Hay muchos primos que no ocurren en esta factorizacion, es decir ocurren 0 veces en la misma. Pero podemos escribir

$$57596 = pr(1)^2.pr(2)^0.pr(3)^0.pr(4)^1.pr(5)^2.pr(6)^0.pr(7)^1.pr(8)^0.pr(9)^0.pr(10)^0....$$

y la igualdad no se altera ya que agregamos factores iguales a 1 (una cantidad infinita!). De esta manera hemos logrado que cada primo intervenga en la factorizacion. Ademas si vemos la infinitupla de exponentes de esta nueva factorizacion,

es decir

$$(2, 0, 0, 1, 2, 0, 1, 0, 0, 0, ...)$$

obtenemos un elemento de $\omega^{[\mathbf{N}]}$.

Por supuesto esto lo podemos hacer con cualquier numero natural y siempre la infinitupla de exponentes sera un elemento de $\omega^{[\mathbf{N}]}$. Ademas es facil notar, basandose en el Teorema Fundamental de la Aritmética, que estas representaciones "chetas" tambien resultan unicas. Mas concretamente tenemos:

Teorema 1 (Teorema Fundamental de la Aritmetica (version cheta))**.** *Para cada $x \in \mathbf{N}$, hay una unica infinitupla $(s_1, s_2, ...) \in \omega^{[\mathbf{N}]}$ tal que*

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

(Tiene sentido escribir $\prod_{i=1}^{\infty} pr(i)^{s_i}$, ya que en esta productoria solo una cantidad finita de factores son no iguales a 1.)

Ejercicio 1: Pruebe la existencia en el teorema anterior. (Hint: Induccion completa.)

Como podra notarse despues de hacer el ejercicio anterior, la existencia de dicha infinitupla para un numero x en general, no es un hecho dificil de probar. En realidad la potencia del Teorema Fundamental de la Aritmética radica en el hecho de que dicha infinitupla es unica.

Ejercicio 2: Use el teorema anterior para probar que
(a) $17^{1045} \neq 13^{2000}$
(b) $5^{55}.13^{90}.17^{1045} \neq 5^{55}.3^{122}.31^{400}$
(c) $2^{90}.3^{20}.17^{1045} \neq 2^{100}.3^{12}.17^{1044}$

Ejercicio 3: Enuncie en forma precisa que significa la "unicidad" en el teorema anterior.

A continuacion un poco de notacion. Dada una infinitupla $(s_1, s_2, ...) \in \omega^{[\mathbf{N}]}$ usaremos $\langle s_1, s_2, ...\rangle$ para denotar al numero $\prod_{i=1}^{\infty} pr(i)^{s_i}$.

Dado $x \in \mathbf{N}$, usaremos (x) para denotar a la unica infinitupla $(s_1, s_2, ...) \in \omega^{[\mathbf{N}]}$ tal que

$$x = \langle s_1, s_2, ...\rangle = \prod_{i=1}^{\infty} pr(i)^{s_i}$$

Ademas para $i \in \mathbf{N}$, usaremos $(x)_i$ para denotar a s_i de dicha infinitupla. Es decir que

(1) $(x) = ((x)_1, (x)_2, ...)$
(2) $(x)_i$ es el exponente de $pr(i)$ en la (unica posible) factorizacion de x como producto de primos

Se le suele llamar la "bajada *i-esima de x*" al numero $(x)_i$. La idea de este nombre es que para obtener $(x)_i$ debemos bajar el exponente de $pr(i)$ en la factorizacion de x. Claramente entonces

(3) $\langle (x)_1, (x)_2, ...\rangle = x$, para cada $x \in \mathbf{N}$

(4) Para cada $(s_1, s_2, ...) \in \omega^{[\mathbf{N}]}$, se tiene que

$$(\langle s_1, s_2, ...\rangle)_i = s_i, \text{ para } i \in \mathbf{N}$$

Es decir que

$$(\langle s_1, s_2, ...\rangle) = (s_1, s_2, ...)$$

Ejercicio 4: Justifique con palabras las propiedades (3) y (4)

Ejercicio 5: Pruebe que si $x, y \in \mathbf{N}$, entonces

(a) $(x)_i \leq x$, para cada $i \in \mathbf{N}$
(b) $(x.y)_i = (x)_i + (y)_i$, para cada $i \in \mathbf{N}$
(c) $x|y$ si y solo si $(x)_i \leq (y)_i$, para cada $i \in \mathbf{N}$

Tenemos entonces el siguiente resultado fundamental

Teorema 2. *Las funciones*

$$\begin{array}{lcl} \mathbf{N} & \to & \omega^{[\mathbf{N}]} \\ x & \to & (x) = ((x)_1, (x)_2, ...) \end{array} \qquad \begin{array}{rcl} \omega^{[\mathbf{N}]} & \to & \mathbf{N} \\ (s_1, s_2, ...) & \to & \langle s_1, s_2, ...\rangle \end{array}$$

son biyecciones una inversa de la otra.

Proof. Llamemos f a la funcion de la izquierda y g a la de la derecha. Notese que el Lema 3 de la Guia 1 nos dice que basta con probar que $f \circ g = Id_{\omega^{[\mathbf{N}]}}$ y $g \circ f = Id_{\mathbf{N}}$. Pero (3) justamente nos dice que $g \circ f = Id_{\mathbf{N}}$ y (4) nos dice que $f \circ g = Id_{\omega^{[\mathbf{N}]}}$. ■

Tal como se hace en la escuela primaria, el siguiente lema nos permite calcular $(x)_i$.

Lema 3. *Dados $x, i \in \mathbf{N}$, se tiene que*

$$(x)_i = \max_t \left(pr(i)^t \textit{ divide a } x\right)$$

Ejercicio 6: Use el Teorema Fundamental de la Aritmetica para probar el lema anterior

Definamos la funcion $Lt : \mathbf{N} \to \omega$ de la siguiente manera:

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

Se tienen las siguientes propiedades basicas

Lema 4. *Para cada $x \in \mathbf{N}$:*

(1) $Lt(x) = 0$ *sii* $x = 1$
(2) $x = \prod_{i=1}^{Lt(x)} pr(i)^{(x)_i}$

Ejercicio 7: Dar una prueba del lema anterior.

Ejercicio 8: Encuentre el dominio de las siguientes funciones:

(a) $\lambda ix[(x)_i]$
(b) $\lambda x[Lt(x)]$
(c) $\lambda x[\langle (x)_1, (x)_2, (x)_3, 0, 0, \ldots\rangle]$

Ordenes naturales sobre Σ^*

Daremos biyecciones naturales entre Σ^* y ω, para cada alfabeto no vacio Σ. Dichas biyecciones dependen de tener asociado a Σ un orden total, el cual nos permite de manera "lexicografica" listar con el tipo de orden de ω a todos los elementos de Σ^*. Recomendamos antes repasar el concepto de orden total que esta desarrollado en la ultima seccion de la Guia 1.

Primero haremos un caso particular pero que tiene un interes extra ya que esta emparentado con nuestra notacion decimal clasica de los numeros de ω.

Notacion decimal sin 0**.** Usaremos Num para denotar el conjunto de numerales. Notese que $Num \cap \omega = \emptyset$. Es decir, no debemos confundir los simbolos que usualmente denotan los primeros diez numeros enteros con los numeros que ellos denotan. De hecho en china o japon los primeros diez numeros enteros se denotan con otros simbolos. Similarmente las palabras pertenecientes a Num^* denotan (notacion decimal) a los numeros de ω pero debemos tener en cuenta que $Num^* \cap \omega = \emptyset$. Cuando tratamos con palabras de Num^*, debemos ser cuidadosos ya que muchas veces en nuestro discurso matematico (es decir las guias, el apunte, lo que escriben los profesores en el pizarron, etc) representamos dos objetos diferentes de la misma forma. Por ejemplo 45 puede estar denotando al numero entero cuarenta y cinco o tambien 45 puede estar denotando la palabra de longitud 2 cuyo primer simbolo es el numeral 4 y cuyo segundo simbolo es el numeral 5, es decir en este caso la palabra 45 se denota a ella misma. Por dar otro ejemplo, el simbolo 1 en nuestro discurso algunas veces se denotara a si mismo y otras veces denotara al numero uno.

Es bien conocido que, en notacion decimal, las siguientes palabras del alfabeto Num, denotan, de menor a mayor, a los numeros de ω

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...$$

Por supuesto esta lista de palabras es infinita pero asumimos que el lector sabe como obtener la palabra siguiente a cada miembro de la lista (i.e. sumar 1 en notacion decimal), lo cual determina por completo la lista conociendo que la misma comienza con la palabra 0.

Cabe destacar que debido a la presencia del numeral 0 en la lista, la n-esima palabra representa (o denota) al numero $n-1$. Dicho de otra forma, el numero $n \in \omega$ es representado por la $(n+1)$-esima palabra de la lista.

Un detalle de la representacion decimal de numeros de ω mediante palabras de Num^* es que la misma no nos da una biyeccion entre Num^* y ω ya que por ejemplo las palabras 00016 y 16 representan el mismo numero. Dicho de otra forma en la lista anterior no figuran todas las palabras de Num^*, a saber estan omitidas todas las palabras que comienzan con el simbolo 0 y tienen longitud mayor que uno. A continuacion daremos una representacion de los numeros de ω mediante palabras, la cual no tendra este problema. El alfabeto que usaremos tendra todos los numerales menos el 0 y ademas tendra un simbolo para denotar al numero diez, a saber el simbolo d. Es decir

$$\widetilde{Num} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, d\}$$

Representaremos a los numeros de ω con la siguiente lista infinita de palabras de $\widetilde{Num}$

$$\varepsilon, 1, 2, 3, 4, 5, 6, 7, 8, 9, d,$$

$11, 12, ..., 1d, 21, 22, ..., 2d, ..., 91, 92, ..., 9d, d1, d2, ..., dd,$
$111, 112, ..., 11d, 121, 122, ..., 12d, ...$

El lector ya se habra dado cuenta de que el siguiente a una palabra α de la lista anterior se obtiene aplicando las siguientes clausulas

- si $\alpha = d^n$, con $n \geq 0$ entonces el siguiente de α es 1^{n+1}
- si α no es de la forma d^n, con $n \geq 0$, entonces el siguiente de α se obtiene de la siguiente manera:
  (a) buscar de derecha a izquierda el primer simbolo no igual a d
  (b) reemplazar dicho simbolo por su siguiente en la lista $1, 2, 3, 4, 5, 6, 7, 8, 9, d$
  (c) reemplazar por el simbolo 1 a todos los simbolos iguales a d que ocurrian a la derecha del simbolo reemplazado

Para hacer las cosas mas formalmente, definamos $s : \widetilde{Num}^* \to \widetilde{Num}^*$ de la siguiente manera

- $s(d^n) = d^{n+1}$, para cada $n \geq 0$
- $s(\alpha 1 d^n) = \alpha 2 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 2 d^n) = \alpha 3 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 3 d^n) = \alpha 4 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 4 d^n) = \alpha 5 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 5 d^n) = \alpha 6 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 6 d^n) = \alpha 7 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 7 d^n) = \alpha 8 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 8 d^n) = \alpha 9 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$
- $s(\alpha 9 d^n) = \alpha d 1^{n+1}$, cada vez que $\alpha \in \widetilde{Num}^*$ y $n \geq 0$

Notese que la definicion de s es correcta ya que una palabra de $\widetilde{Num}^*$ ya sea es de la forma d^n, con $n \geq 0$, o es de la forma $\alpha \delta d^n$, con $\alpha \in \widetilde{Num}^*$, $\delta \in \widetilde{Num} - \{d\}$ y $n \geq 0$; y estos casos posibles son mutuamente excluyentes.

Claramente se tiene entonces que la lista anterior puede ser escrita de la siguiente manera

$$\varepsilon, s(\varepsilon), s(s(\varepsilon)), s(s(s(\varepsilon))), s(s(s(s(\varepsilon)))), ...$$

Y como ya dijimos, pensamos que las palabras de la lista van representando en forma creciente los elementos de ω:

- El numero 0 es representado en la lista anterior con la palabra ε
- El numero 1 es representado en la lista anterior con la palabra 1

$\vdots$

- El numero 9 es representado en la lista anterior con la palabra 9
- El numero 10 es representado en la lista anterior con la palabra d
- El numero 11 es representado en la lista anterior con la palabra 11

$\vdots$

- El numero 19 es representado en la lista anterior con la palabra 19
- El numero 20 es representado en la lista anterior con la palabra $1d$
- El numero 21 es representado en la lista anterior con la palabra 21

- El numero 22 es representado en la lista anterior con la palabra 22

$$\vdots$$

Como puede notarse en estos primeros veinte y pico numeros solo dos (el 0 y el 20) se representan en forma distinta a la reprentacion decimal clasica. Es natural que ε denote al numero 0 y ademas notese que la palabra $1d$ (que en la lista representa el 20) puede leerse como "diecidiez" (es decir la palabra que sigue a "diecinueve") que justamente es 20. Por supuesto con esta manera de pensar la palabra $2d$ deberiamos leerla como "ventidiez" y si nos fijamos en la lista ella representa al numero treinta lo cual nuevamente es muy natural. Otro ejemplo: a $6d$ deberiamos leerla como "sesentidiez" y es natural ya que en la lista representa al setenta. Tambien, la palabra $9d$ puede leerse noventidiez ya que representa en la lista al numero 100.

La lista anterior va representando los numeros de ω en forma muy natural pero, aunque nuestra intuicion nos diga que no, en principio podria pasar que una misma palabra del alfabeto $\widetilde{Num}$ ocurra dos veces en la lista y esto nos diria que una misma palabra estaria representando a dos numeros distintos. Tambien, en principio podria suceder que haya una palabra del alfabeto $\widetilde{Num}$ la cual nunca figure en la lista. Mas abajo daremos una serie de ejercicios que muestran que estas dos posibilidades no suceden, es decir muestran que

(S) Toda palabra de $\widetilde{Num}^*$ aparece en la lista
(I) Ninguna palabra de $\widetilde{Num}^*$ aparece mas de una ves

Notese que la propiedad (S) nos dice que la funcion

$$\begin{array}{rcl} * : \omega & \to & \widetilde{Num}^* \\ n & \to & (n+1)\text{-esimo elemento de la lista} = \overbrace{s(...s(}^{n \text{ veces}}s(\varepsilon))...) \end{array}$$

es sobreyectiva y la propiedad (I) nos garantiza que dicha funcion es inyectiva, por lo cual entre las dos nos garantizan que dicha representacion establece una biyeccion entre ω y $\widetilde{Num}^*$.

Por supuesto, la pregunta que inmediatamente surge es como calcular la inversa de $*$. Llamemos $\#$ a la inversa de $*$. Notese que dada una palabra $\alpha \in \widetilde{Num}^*$, el numero $\#(\alpha)$ es justamente el numero representado por la palabra α, o dicho de otra forma $\#(\alpha)$ es la posicion que ocupa α en la lista, contando desde el 0 (es

decir α es la $(\#(\alpha)+1)$-esima palabra de la lista). Por ejemplo:

$$\begin{aligned}
\#(\varepsilon) &= 0\\
\#(1) &= 1\\
&\vdots\\
\#(9) &= 9\\
\#(d) &= 10\\
\#(11) &= 11\\
\#(12) &= 12\\
&\vdots\\
\#(19) &= 19\\
\#(1d) &= 20
\end{aligned}$$

Aqui hay que tener cuidado como leemos las igualdades anteriores. Por ejemplo en la igualdad

$$\#(1) = 1$$

la primera ocurrencia del simbolo 1 se refiere al numeral uno, es decir denota una palabra y la segunda ocurrencia se esta refiriendo al numero uno, es decir denota un numero.

Dejamos al lector el ejercicio de ganar intuicion con ejemplos hasta que se convensa de que tal como en el caso de la notacion decimal, el numero $\#(\alpha)$ se expresa como una suma de potencias de 10, con los coeficientes dados en funcion de los simbolos de α. Mas concretamente si $\alpha = s_1 s_2 ... s_k$ con $k \geq 1$ y $s_1, s_2, ..., s_k \in \widetilde{Num}$, entonces

$$\#(\alpha) = \#(s_1).10^{k-1} + \#(s_2).10^{k-2} + ... + \#(s_k).10^0$$

No daremos ahora una prueba formal de este hecho pero para ganar intuicion sobre el mismo el lector puede hacer los Ejercicios 13, 14, 15 y 16. Algunos ejemplos

$$\begin{aligned}
\#(1d) &= 1.10^1 + 10.10^0 = 10 + 10 = 20\\
\#(dd) &= 10.10^1 + 10.10^0 = 100 + 10 = 110\\
\#(111) &= 1.10^2 + 1.10^1 + 1.10^0 = 100 + 10 + 1 = 111\\
\#(1d3d) &= 1.10^3 + 10.10^2 + 3.10^1 + 10.10^0
\end{aligned}$$

Ahora que sabemos que las palabras de $\widetilde{Num}$ representan los numeros como suma de potencias de diez, en forma analoga a la notacion decimal clasica, podemos refozar aun mas la analogia poniendo nombres adecuados que, tal como en el caso clasico, nos permitan leer las palabras de $\widetilde{Num}$ describiendo su suma de potencias asociada. Por ejemplo podriamos llamar "decenta" al numero 100, por analogia a "treinta", "cuarenta",...,"noventa". O sea una decenta es diez veces diez. De esta forma la palabra $d1$ se leera "decenta y uno" y esto es natural ya que en la lista representa al 101. La palabra dd se leera "decenta y diez" y esto describe a la perfeccion el numero que representa, i.e. el $10.10 + 10 = 110$. La palabra que sigue en la lista a dd es 111 la cual representa al 111, es decir aqui como en los otros casos vistos en los cuales no hay ocurrencias del simbolo d la palabra representa al mismo numero que representa en la notacion decimal clasica. Por dar otro ejemplo,

la palabra $59d3$ se leera "cinco mil novecientos decenta y tres" y representara al numero 6003.

Para seguir debemos ponerle nombre a "diez veces cien", es decir, "decientos" (por analogia con "novecientos = nueve veces cien") denotara al numero 1000 = 10.100. De esta forma la palabra $d51$ se leera "decientos cincuenta y uno" y esto es natural ya que pensando un rato se puede ver que ella representa al 1051. Tambien, la palabra ddd se leera "decientos decenta y diez" y representara al numero 1110. En la pagina granlogico.com hay un programa hecho por Guido Ivetta el cual hace lo siguiente:

- Dado un número n expresado en notación clásica decimal da explicitamente la palabra $*(n)$ y la forma de pronunciarla o leerla a dicha palabra

Prueba de las propiedades (S) e (I). Dado que el siguiente a un elemento α de la lista es de la misma longitud que α o tiene longitud igual a $|\alpha| + 1$, podemos representar la lista anterior de la siguiente manera:

$$B_0; B_1; B_2; B_3; B_4; ...$$

donde cada B_n es, por definicion, la parte de la lista en la cual las palabras tienen longitud exactamente n. Por ejemplo:

- B_0 es ε
- B_1 es $1, 2, 3, 4, 5, 6, 7, 8, 9, d$
- B_2 es $11, 12, ..., 1d, 21, 22, ..., 2d, ..., 91, 92, ..., 9d, d1, d2, ..., dd$

Notese que hasta el momento nada nos asegura que no suceda que para algun n se de que B_n sea una lista infinita, lo cual ademas nos diria que los bloques $B_{n+1}, B_{n+2}, ...$ son todos vacios. Es decir podria pasar que la lista se estanque en una longitud n y nunca aparezca una palabra de longitud mayor que n. Esto por supuesto obligaria a que se repitan muchas veces palabras de dicha longitud n ya que hay una cantidad finita de las mismas (10^n).

Por supuesto nuestra intuicion nos dice que en el bloque B_n estan listadas sin repeticion todas las palabras de $\widetilde{Num}^*$ de longitud n, pero debemos justificar esto con argumentos solidos. Algunas propiedades basicas que se pueden probar facilmente son:

(1) Si $B_n = \alpha_1, ..., \alpha_k$, entonces $\alpha_1 = 1^n$ y $\alpha_k = d^n$
(2) d^n ocurre en B_n solo en la ultima posicion

estas propiedades son consecuencias inmediatas de como se calcula el elemento siguiente a uno dado en la lista.

Ejercicio 9: (Opcional) Justifique con palabras la propiedad (1)
Ejercicio 10: (Opcional) Justifique con palabras la propiedad (2)

Otra propiedad importante es la siguiente

(3) Si $B_n = \alpha_1, ..., \alpha_k$, entonces $B_{n+1} = 1\alpha_1, ..., 1\alpha_k, 2\alpha_1, ..., 2\alpha_k, ..., d\alpha_1, ..., d\alpha_k$

Para probar (3) es muy util el siguiente resultado obvio

Lema 5. *Sea $\sigma \in \widetilde{Num}$ y supongamos $\alpha \in \widetilde{Num}^*$ no es de la forma d^n. Entonces $s(\sigma\alpha) = \sigma s(\alpha)$.*

Ejercicio 11: (Opcional) Use (1), (2) y el lema anterior para dar una explicacion solida con palabras de por que es cierta la propiedad (3)

Ahora es facil usando (3) probar inductivamente que

(4) B_n es una lista sin repeticiones de todas las palabras de longitud n

Ejercicio 12: (Opcional) Pruebe (4)

Pero claramente de (4) se desprenden en forma obvia las propiedades (S) y (I).

El caso general. Sea Σ un alfabeto no vacio y supongamos $\leq$ es un orden total sobre Σ. Supongamos que $\Sigma = \{a_1, ..., a_n\}$, con $a_1 < a_2 < ... < a_n$. Inspirados en la lista dada anteriormente de las palabras de $\widetilde{Num}^*$, podemos dar la siguiente lista de palabras de Σ^*:

$$\begin{array}{l}
\varepsilon, a_1, a_2, ..., a_n, \\
a_1a_1, a_1a_2, ..., a_1a_n, a_2a_1, a_2a_2, ..., a_2a_n, ..., a_na_1, a_na_2, ..., a_na_n, \\
a_1a_1a_1, a_1a_1a_2, ..., a_1a_1a_n, a_1a_2a_1, a_1a_2a_2, ..., a_1a_2a_n, ..., a_1a_na_1, a_1a_na_2, a_1a_na_n, \\
a_2a_1a_1, a_2a_1a_2, ..., a_2a_1a_n, a_2a_2a_1, a_2a_2a_2, ..., a_2a_2a_n, ..., a_2a_na_1, a_2a_na_2, a_2a_na_n, \\
\vdots \\
a_na_1a_1, a_na_1a_2, ..., a_na_1a_n, a_na_2a_1, a_na_2a_2, ..., a_na_2a_n, ..., a_na_na_1, a_na_na_2, a_na_na_n, \\
a_1a_1a_1a_1, a_1a_1a_1a_2, ...
\end{array}$$

El objetivo es probar que la lista anterior enumera sin repeticiones todas las palabras de Σ^*, i.e. produce naturalmente una biyeccion entre ω y Σ^*. Pero antes debemos definir mas formalmente la lista. Para esto definamos $s^{\leq} : \Sigma^* \to \Sigma^*$ de la siguiente manera

- $s^{\leq}((a_n)^m) = (a_1)^{m+1}$, para cada $m \geq 0$
- $s^{\leq}(\alpha a_i(a_n)^m) = \alpha a_{i+1}(a_1)^m$, cada vez que $\alpha \in \Sigma^*$, $1 \leq i < n$ y $m \geq 0$

Notese que la definicion de $s^{\leq}$ es correcta ya que toda palabra de Σ^* es de la forma $(a_n)^m$, con $m \geq 0$, o es de la forma $\alpha a_i(a_n)^m$, con $\alpha \in \Sigma^*$, $1 \leq i < n$ y $m \geq 0$; y estos dos casos posibles son mutuamente excluyentes (convencerse fuertemente de que esto es asi).

Ejercicio 13: Sea $\Sigma = \{\%, !, @\}$ y sea $\leq$ el orden total sobre Σ dado por $\% < ! < @$ (es decir que aqui $a_1 = \%$, $a_2 = !$ y $a_3 = @$). Escriba los primeros elementos de la lista y describa para este caso particular la funcion $s^{\leq}$, sin hablar de los a_i, i.e. solo haciendo referencia a los simbolos $\%, !, @$.

Ejercicio 14: Pruebe para el caso general que

(a) $\varepsilon \neq s^{\leq}(\alpha)$, para cada $\alpha \in \Sigma^*$

(b) Si $\alpha \neq \varepsilon$, entonces $\alpha = s^{\leq}(\beta)$ para algun β

Ejercicio 15: (a) Convensace de que valen las siguientes ecuaciones

$$s^{\leq}(\varepsilon) = a_1$$

$$s^{\leq}(\alpha a_i) = \alpha a_{i+1},\ i < n$$

$$s^{\leq}(\alpha a_n) = s^{\leq}(\alpha) a_1$$

(b) Note que sucesivas aplicaciones de las ecuaciones anteriores determinan por completo el valor de $s^{\leq}$ en una palabra α previamente fijada. Explique esto con palabras.
(c) Pruebe formalmente que valen las ecuaciones de (a)

Por supuesto, la lista anterior puede ser escrita de la siguiente manera

$$\varepsilon, s^{\leq}(\varepsilon), s^{\leq}(s^{\leq}(\varepsilon)), s^{\leq}(s^{\leq}(s^{\leq}(\varepsilon))), s^{\leq}(s^{\leq}(s^{\leq}(s^{\leq}(\varepsilon)))), ...$$

Con esta definicion formal de la lista, podemos probar de la misma forma en la que lo hicimos arriba que:

(S) Toda palabra de Σ^* aparece en la lista
(I) Ninguna palabra de Σ^* aparese mas de una ves en la lista

Ejercicio 16: (Opcional) Pruebe (S) e (I). Hint: use las mismas ideas que se usaron para probar (S) e (I) para el caso de $\Sigma = \widetilde{Num}$ y $\leq$ dado por $1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < d$.

Definamos $*^{\leq} : \omega \to \Sigma^*$ recursivamente de la siguiente manera:

- $*^{\leq}(0) = \varepsilon$
- $*^{\leq}(i+1) = s^{\leq}(*^{\leq}(i))$

Es claro que entonces $*^{\leq}(i)$ nos da el $(i+1)$-esimo elemento de la lista, o lo que es lo mismo, el i-esimo elemento de la lista contando desde el 0. O sea que las propiedades (S) y (I) nos garantizan que la funcion $*^{\leq}$ es biyectiva. A continuacion describiremos su inversa. Primero un lema obvio pero muy importante.

Lema 6. *Sea Σ un alfabeto no vacio y supongamos $\leq$ es un orden total sobre Σ. Supongamos que $\Sigma = \{a_1, ..., a_n\}$, con $a_1 < a_2 < ... < a_n$. Entonces para cada $\alpha \in \Sigma^* - \{\varepsilon\}$ hay unicos $k \in \omega$ y $i_0, i_1, ..., i_k \in \{1, ..., n\}$ tales que*

$$\alpha = a_{i_k} ... a_{i_0}$$

Notar que el k del lema anterior es $|\alpha| - 1$ y los numeros $i_k, ..., i_0$ van dando el numero de orden de cada simbolo de α yendo de izquierda a derecha. Por ejemplo si $\Sigma = \{\%, !, @\}$ y $\leq$ es el orden total sobre Σ dado por $\% <! < @$ (es decir que aqui $a_1 = \%$, $a_2 =!$ y $a_3 = @$) entonces para la palabra !%@%@ tenemos $k = 4$ y $i_4 = 2$, $i_3 = 1$, $i_2 = 3$, $i_1 = 1$ y $i_0 = 3$. Sin envargo si hubieramos tomado el orden dado por $@ < \% <!$, para la misma palabra hubieramos tenido $i_4 = 3$, $i_3 = 2$, $i_2 = 1$, $i_1 = 2$ y $i_0 = 1$.

Ahora podemos definir la funcion $\#^{\leq}$ de la siguiente manera

$$\begin{array}{rcl} \#^{\leq} : \Sigma^* & \to & \omega \\ \varepsilon & \to & 0 \\ a_{i_k} ... a_{i_0} & \to & i_k n^k + ... + i_0 n^0 \end{array}$$

Ejercicio 17: Si $\leq$ es el orden de $\{@, \&\}$ dado por $@ < \&$, entonces $\#^{\leq}(@\&@\&@) = 2^4 + 2^4 + 2^2 + 2^2 + 1$ y $\#^{\leq}(@\&@\&@) = 2^5 + 2^3 + 1$

Ejercicio 18: Si $\leq$ es el orden de $\{@, \&\}$ dado por $@ < \&$, entonces $\#^{\leq}(\alpha@) = \#^{\leq}(\alpha).2 + 1$, para todo $\alpha \in \{@, \&\}^*$

Ejercicio 19: Sea Σ un alfabeto finito no vacio y sea $\leq$ un orden total sobre Σ. Inspirese en el ejercicio anterior para dar una "definicion recursiva" de la funcion $\#^{\leq}$.

Aceptaremos sin prueba el siguiente resultado fundamental

Lema 7. *La funcion $\#^{\leq}$ es la inversa de $*^{\leq}$*

Cabe destacar que dada una palabra α, el numero $\#^{\leq}(\alpha)$ nos dice en que posicion se hubica α en la lista, es decir α es la $(\#^{\leq}(\alpha)+1)$-esima palabra de la lista.

De los desarrollos hechos se desprende el siguiente interesante resultado

Lema 8. *Sea $n \geq 1$ fijo. Entonces cada $x \geq 1$ se escribe en forma unica de la siguiente manera:*

$$x = i_k n^k + i_{k-1} n^{k-1} + ... + i_0 n^0,$$

con $k \geq 0$ y $1 \leq i_k, i_{k-1}, ..., i_0 \leq n$.

Ejercicio 20: Pruebe el lema anterior

Ejercicio 21: Si $\leq$ es un orden total sobre un alfabeto no vacio Σ, entonces $s^{\leq} = *^{\leq} \circ Suc \circ \#^{\leq}$

Como hemos visto las biyecciones dadas producen una "identificacion" entre numeros de ω y palabras del alfabeto Σ. Es decir, en algun sentido identificamos palabras y numeros ya que se corresponden biunivocamente. Supongamos que α es una palabra de $\Sigma^* - \{\varepsilon\}$ y queremos "verla como un numero". Entonces en ves de ver sus simbolos vemos los ordenes de aparicion en Σ de los mismos y miramos la suma de potencias asociada.

Supongamos ahora que x es un numero de $\omega - \{0\}$ y ademas supongamos que somos super inteligentes y que cuando vemos a x vemos la secuencia unica de numeros $i_k, i_{k-1}, ..., i_0$ que nos permite expresarlo como suma de potencias segun el lema anterior. Entonces si queremos ver a x como una palabra simplemente miramos la secuencia $i_k, i_{k-1}, ..., i_0$ como palabra, reemplazando cada i_j por el simbolo i_j-esimo de Σ.

Ejercicio 22: Mastique mastique e imagine hasta que entienda con claridad el ultimo parrafo. Luego sera mas mariposa.

Extension del orden total de Σ a Σ^*. Dado un orden total $\leq$ sobre Σ, podemos extender $\leq$ a Σ^* de la siguiente manera

- $\alpha \leq \beta$ sii $\#^{\leq}(\alpha) \leq \#^{\leq}(\beta)$

Es decir $\alpha \leq \beta$ sii $\alpha = \beta$ o α ocurre antes que β en la lista. Llamaremos a este orden el *orden total de Σ^* inducido por $\leq$*. Obviamente este orden es un orden total sobre Σ^*. Tal como en el caso de la notacion decimal clasica (con 0) el orden total de Σ^* inducido por $\leq$ puede ser caracterizado "lexicograficamente" en forma similar al orden de los diccionarios. Ya que se vuelve mas facil de escribir daremos a continuacion la caracterizacion lexicografica del orden $<$ asociado al orden total de Σ^* inducido por $\leq$. Lo aceptaremos sin demostracion.

Lema 9 (Orden lexicografico de Σ^*)**.** *Sea $\leq$ un orden total sobre Σ. Dados $\alpha, \beta \in \Sigma^*$ tenemos que:*

(1) *Si $|\alpha| \neq |\beta|$, entonces $\alpha < \beta$ si y solo si $|\alpha| < |\beta|$*
(2) *Si $|\alpha| = |\beta|$, entonces $\alpha < \beta$ si y solo si existen $\rho, \gamma_1, \gamma_2 \in \Sigma^*$ y $i, j \in \omega$ tales que*

$$\alpha = \rho a_i \gamma_1 \text{ y } \beta = \rho a_j \gamma_2 \text{ y } i < j$$

Proof. Ver el apunte. ■

Cabe destacar que si bien la condicion del item (2) del lema anterior es de tipo descriptiva, tiene un caracter algotitmico bien marcado ya que cuando tenemos dos palabras no iguales a ε, de la misma longitud, podemos ver el primer par de dijitos en los que difieren y la relacion de orden que tienen dichos simbolos es la que tienen dichas palabras.

Tambien es interesante tener caracterizado el orden sin dividir por casos:

Lema 10. *Sea $\leq$ un orden total sobre Σ. Sean $\alpha, \beta \in \Sigma^*$. Entonces $\alpha < \beta$ si y solo si $|\alpha| < |\beta|$ o $|\alpha| = |\beta|$ y existen $\rho, \gamma_1, \gamma_2 \in \Sigma^*$ y $i, j \in \omega$ tales que*

$$\alpha = \rho a_i \gamma_1 \text{ y } \beta = \rho a_j \gamma_2 \text{ y } i < j$$

Proof. Es un corolario directo del lema anterior. ■ Tambien se puede caracterizar la relacion $<$ asociada al orden total de Σ^* inducido por $\leq$ de manera recursiva:

Lema 11. *Sea $\leq$ un orden total sobre Σ. Sean $\alpha, \beta \in \Sigma^*$. Entonces $\alpha < \beta$ si y solo si se da alguna de las siguientes*

- $|\alpha| < |\beta|$
- $|\alpha| = |\beta|$ y $[\alpha]_1 < [\beta]_1$
- $|\alpha| = |\beta|$, $[\alpha]_1 = [\beta]_1$ y $^\frown\alpha < {}^\frown\beta$

($^\frown\alpha$ es definido en la Guia 1)

Proof. Es un corolario directo del lema anterior. ■

Deberia ser intuitivamente claro que el orden recien definido sobre Σ^* posee las mismas propiedades matematicas que el orden usual de ω. Por ejemplo:

Lema 12. *Si $S \subseteq \Sigma^*$ es no vacio, entonces existe $\alpha \in S$ tal que $\alpha \leq \beta$, para cada $\beta \in S$.*

Ejercicio 23: Pruebe el lema anterior (Hint: notar que el conjunto $\{\mu \in S : |\mu| \leq |\alpha| \text{ para cada } \alpha \in S\}$ es finito y no vacio)

GUIA 3 DE LENGUAJES: PROCEDIMIENTOS EFECTIVOS

Un concepto importante en ciencias de la computacion es el de *procedimiento* o *metodo* para realizar alguna tarea determinada. Nos interesan los procedimientos que estan definidos en forma precisa e inambigua, es decir aquellos en los cuales en cada paso a seguir, la tarea a realizar esta objetivamente descripta. Tambien deben ser repetibles, en el sentido de que si realizamos un procedimiento dos veces con el mismo dato de entrada, entonces ambas ejecuciones deben ser identicas, es decir se realizaran las mismas tareas y en el mismo orden.

Nos interesan los procedimientos $\mathbb{P}$ que posean las siguientes caracteristicas:

(1) El interprete o ejecutante de $\mathbb{P}$ es una persona que trabajara con papel y lapiz (ambos recursos disponibles en forma ilimitada).

(2) Cada paso o tarea que $\mathbb{P}$ encomiende a realizar debe ser simple y facil de realizar en forma *efectiva* por cualquier persona.

(3) El procedimiento $\mathbb{P}$ comienza a funcionar siempre a partir de cierto dato de entrada y una ves que haya comensado, siempre sucedera una de las dos siguientes posibilidades
   (a) $\mathbb{P}$ se detiene y da cierto dato de salida
   (b) $\mathbb{P}$ nunca se detiene, es decir a medida que se van realizando las instrucciones o tareas, $\mathbb{P}$ siempre direcciona a realizar nuevas tareas y lo hace sucesiva e indefinidamente.

   En el caso (a) diremos que $\mathbb{P}$ se detiene partiendo del dato de entrada en cuestion y en el caso (b) diremos que $\mathbb{P}$ no se detiene partiendo de dicho dato.

(4) Hay $n, m \in \omega$ y un alfabeto Σ tales que el conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$. Cabe aclarar que para ciertas $(n+m)$-uplas de $\omega^n \times \Sigma^{*m}$ el procedimiento $\mathbb{P}$ se detendra y para ciertas otras no lo hara.

Llamaremos *procedimientos efectivos* a aquellos procedimientos que posean las caracteristicas arriba mencionadas.

El *conjunto de datos de salida de* $\mathbb{P}$ es el conjunto de todos los datos que el procedimiento $\mathbb{P}$ dara como salida en alguna de las posibles ejecuciones al variar todos los datos de entrada posibles. Si bien siempre el conjunto de datos de entrada sera de la forma $\omega^n \times \Sigma^{*m}$, puede ser muy dificil o imposible, en general, conocer con precision el conjunto de datos de salida de un procedimiento (esto lo justificaremos mas adelante).

Ya que el interprete de $\mathbb{P}$ es una persona dotada de lapiz y papel, supondremos que los elementos de ω que intervienen en los datos de entrada y de salida estaran representados por palabras de Num usando la notacion decimal.

Quisas el procedimiento efectivo mas famoso de la matematica es aquel que se enseña en los colegios para sumar dos numeros naturales expresados en notacion decimal. Notar que el conjunto de datos de entrada de dicho procedimiento es ω^2 y el conjunto de datos de salida es el conjunto formado por todas las sumas posibles de pares de elementos de ω, es decir ω. Por supuesto este procedimiento

solo usa lapiz, papel y pasos extremadamente simples a seguir en cada momento de la computacion, es decir, en algun sentido, no es necesario "entender que es lo que se esta haciendo" para llegar al final y obtener la palabra que representa en notacion decimal a la suma de los numeros iniciales. Dejamos al lector repasar este procedimiento asi como el que calcula dado un numero x no nulo de ω, al numero $x-1$, los cuales nos haran falta mas adelante en los ejemplos.

Funciones Σ-efectivamente computables

Una funcion Σ-mixta $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ sera llamada Σ-*efectivamente computable* si hay un procedimiento efectivo $\mathbb{P}$ tal que

(1) El conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$
(2) El conjunto de datos de salida esta contenido en ω.
(3) Si $(\vec{x}, \vec{\alpha}) \in D_f$, entonces $\mathbb{P}$ se detiene partiendo de $(\vec{x}, \vec{\alpha})$, dando como dato de salida $f(\vec{x}, \vec{\alpha})$.
(4) Si $(\vec{x}, \vec{\alpha}) \in (\omega^n \times \Sigma^{*m}) - D_f$, entonces $\mathbb{P}$ no se detiene partiendo desde $(\vec{x}, \vec{\alpha})$

Analogamente una funcion Σ-mixta $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$ sera llamada Σ-*efectivamente computable* si hay un procedimiento efectivo $\mathbb{P}$ tal que

(1) El conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$
(2) El conjunto de datos de salida esta contenido en Σ^*.
(3) Si $(\vec{x}, \vec{\alpha}) \in D_f$, entonces $\mathbb{P}$ se detiene partiendo de $(\vec{x}, \vec{\alpha})$, dando como dato de salida $f(\vec{x}, \vec{\alpha})$.
(4) Si $(\vec{x}, \vec{\alpha}) \in (\omega^n \times \Sigma^{*m}) - D_f$, entonces $\mathbb{P}$ no se detiene partiendo desde $(\vec{x}, \vec{\alpha})$

Cuando un procedimiento $\mathbb{P}$ cumpla los items (1), (2), (3) y (4) de arriba diremos que $\mathbb{P}$ *computa* a la funcion f o que f es *computada* por $\mathbb{P}$.

Veamos algunos ejemplos:

(E1) La funcion $\lambda xy\,[x+y]$ es Σ-efectivamente computable, cualquiera sea el alfabeto Σ ya que el procedimiento enseñado en la escuela primaria para sumar numeros en notacion decimal es efectivo y computa esta funcion.
(E2) La funcion $C_3^{1,2}$ es Σ-efectivamente computable ya que el siguiente procedimiento $\mathbb{P}$ con conjunto de datos de entrada $\omega \times \Sigma^{*2}$ la computa:
- Independientemente de quien sea el dato de entrada $(x_1, \alpha_1, \alpha_2)$, detenerse dando como salida el numero 3

(E3) La funcion $p_3^{2,3}$ es Σ-efectivamente computable ya que el siguiente procedimiento con conjunto de datos de entrada $\omega^2 \times \Sigma^{*3}$ la computa:
- Dado el dato de entrada $(x_1, x_2, \alpha_1, \alpha_2, \alpha_3)$, detenerse y dar como salida la palabra α_1

(E4) *Pred* es Σ-efectivamente computable. Para realizar el procedimiento efectivo que compute a *Pred* necesitaremos el procedimiento de la escuela primaria que dado un numero no nulo x, expresado en notacion decimal, calcula el numero $x-1$, en notacion decimal. Llamemos $\mathbb{P}_{-1}$ a dicho procedimiento. El siguiente procedimiento (con conjunto de datos de entrada igual a ω) computa a *Pred*:

Dado como dato de entrada un elemento $x \in \omega$, realizar lo siguiente:
Etapa 1
Si $x = 0$, entonces ir a Etapa 3, en caso contrario ir a Etapa 2.
Etapa 2

Correr $\mathbb{P}_{-1}$ con dato de entrada x obteniendo y como dato de salida. Detenerse y dar y como dato de salida.

Etapa 3

Si $x = 0$, entonces ir a Etapa 1.

Como puede notarse el procedimiento anterior es efectivo ya que como todos sabemos, los sucesivos pasos efectuados al correr $\mathbb{P}_{-1}$ en la Etapa 2 son todos simples y efectivamente realizables. Por supuesto si uno quisiera ser mas prolijo, deberia reemplazar la Etapa 2 por las distintas instrucciones de $\mathbb{P}_{-1}$, referidas a x.

(E5) El predicado $\lambda xy[x < y]$ es Σ-efectivamente computable cualquiera sea el alfabeto Σ. Describiremos con palabras un procedimiento $\mathbb{P}_{<}$ que computa a $\lambda xy[x < y]$. Su conjunto de datos de entrada es ω^2. Dado un par $(x, y) \in \omega^2$, el procedimiento primero compara las longitudes de las palabras que en notacion decimal representan a x y y. Por supuesto esto lo hace borrando de a un simbolo y viendo si alguna se termina primero. Si resultan de distinta longitud, es facil darse cuenta como sigue. En caso de que las palabras resulten de igual longitud, entonces se hace el procedimiento clasico de ir comparando digitos de izquierda a derecha.

(E6) Veamos que la funcion $\lambda\alpha[|\alpha|]$ es Σ-efectivamente computable. Como en los lenguajes de programacion, usaremos variables y asignaciones para diseñar el procedimiento. Ademas llamemos $\mathbb{P}_{+1}$ a el procedimiento de la escuela primaria que dado un numero no nulo x, expresado en notacion decimal, calcula el numero $x + 1$, en notacion decimal. Sea $\mathbb{P}_{|\ |}$ el siguiente procedimiento.

Dado como dato de entrada un elemento $\alpha \in \Sigma^*$, realizar lo siguiente:

Etapa 1: Hacer las siguientes asignaciones

$$A \leftarrow \alpha$$
$$B \leftarrow 0$$

e ir a Etapa 2.

Etapa 2: Si A no es ε, ir a Etapa 3. En caso contrario detenerse y dar como salida B.

Etapa 3: Correr $\mathbb{P}_{+1}$ con dato de entrada igual al contenido de B, obteniendo y como salida. Hacer las siguientes asignaciones

$$A \leftarrow \text{resultado de remover el 1er simbolo de } A$$
$$B \leftarrow y$$

e ir a Etapa 2.

Ejercicio 1: Convensase que el uso de asignaciones puede realizarse usando solo lapiz y papel. Imagine como lo haria en el caso anterior y corrobore que dicho procedimiento es efectivo y ademas computa a $\lambda\alpha[|\alpha|]$.

Ejercicio 2: Use los procedimientos $\mathbb{P}_{<}$ y $\mathbb{P}_{|\ |}$ de los dos ejemplos anteriores para diseñar usando asignaciones un procedimiento que compute a la funcion $\lambda i\alpha[[\alpha]_i]$

Un ultimo ejemplo:

(E7) Si $\leq$ es el orden total sobre $\Sigma = \{\blacktriangle, \%\}$ dado por $\blacktriangle < \%$, entonces veremos que la funcion $s^{\leq}$ es Σ-efectivamente computable. En un par de ejercicios de la Guia 2 vimos que cualquiera sea $\alpha \in \Sigma^*$, se cumple

$$s^{\leq}(\varepsilon) = \blacktriangle$$
$$s^{\leq}(\alpha\blacktriangle) = \alpha\%$$
$$s^{\leq}(\alpha\%) = s^{\leq}(\alpha)\blacktriangle$$

y que estas ecuaciones permiten calcular el valor de $s^{\leq}$. Usaremos esta idea para dar un procedimiento efectivo (con conjunto de datos de entrada igual a Σ^*) que compute a $s^{\leq}$. Como en los lenguajes de programacion, usaremos variables y asignaciones para diseñar el procedimiento.

Etapa 1: Dado el dato de entrada $\alpha \in \Sigma^*$, hacer las siguientes asignaciones

$$A \leftarrow \alpha$$
$$B \leftarrow \varepsilon$$
$$F \leftarrow \blacktriangle$$

e ir a Etapa 2.

Etapa 2: Si A comiensa con $\blacktriangle$, entonces hacer las siguientes asignaciones

$A \leftarrow$ resultado de remover el 1er simbolo de A

$F \leftarrow B\%$

$B \leftarrow B\blacktriangle$

e ir a la Etapa 2. En caso contrario ir a la Etapa 3.

Etapa 3: Si A comiensa con $\%$, entonces hacer las siguientes asignaciones

$A \leftarrow$ resultado de remover el 1er simbolo de A

$F \leftarrow F\blacktriangle$

$B \leftarrow B\%$

e ir a la Etapa 2. En caso contrario ir a la Etapa 4.

Etapa 4: Dar como salida F

Observacion: Notese que la definicion de funcion Σ-efectivamente computable para el caso $f = \emptyset$ tiene a priori cierta ambiguedad ya que cualesquiera sean $n, m \in \omega$ y $O \in \{\omega, \Sigma^*\}$ tenemos que $\emptyset : \emptyset \subseteq \omega^n \times \Sigma^{*m} \to O$ ya que $D_\emptyset = \emptyset$ y $I_\emptyset = \emptyset$. De todas maneras, cualesquiera sean los n, m y O elejidos, siempre hay un procedimiento efectivo que computa a $f = \emptyset$, i.e. un procedimiento que nunca se detiene, cualesquiera sea el dato de entrada de $\omega^n \times \Sigma^{*m}$. Es decir que la funcion $\emptyset$ es Σ-efectivamente computable cualesquiera sea el alfabeto Σ. Cabe destacar que para el caso de una funcion $f \neq \emptyset$, nuestra definicion es inambigua ya que hay unicos $n, m \in \omega$ y $O \in \{\omega, \Sigma^*\}$ tales que $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$.

Ejercicio 3: Sea $\mathbb{P}_+$ un procedimiento efectivo que compute a la funcion $\lambda xy\,[x + y]$. Basado en $\mathbb{P}_+$ diseñe un procedimiento efectivo que compute a $\lambda xy\,[x.y]$

Ejercicio 4: Sea $\leq$ es el orden total sobre $\Sigma = \{\blacktriangle, \%\}$ dado por $\blacktriangle < \%$. Usando que $s^{\leq}$ es Σ-efectivamente computable diseñe un procedimiento efectivo que compute a $*^{\leq} : \omega \to \Sigma^*$

Ejercicio 5: Sea $\Sigma = \{▲, \%\}$ y sea $f : D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega$ dada por:

$$\begin{aligned} D_f &= \{(0,1,\varepsilon), (55,54,▲\%▲\%▲), (1,1,▲▲)\} \\ f(0,1,\varepsilon) &= 1 \\ f(55,54,▲\%▲\%▲) &= 2 \\ f(1,1,▲▲) &= 3 \end{aligned}$$

Pruebe que f es Σ-efectivamente computable.

Nota Importante: en lo que sigue muchas veces nos permitiremos trabajar con procedimientos que son efectivos en terminos de otros que ya se han dado, es decir daremos procedimientos que en principio no es claro que sean efectivos pero los cuales se volverian efectivos si reemplazaramos ciertas instrucciones por la manera efectiva de simularlas. Para hacer mas dinamico el discurso no distinguiremos entre este tipo de procedimientos (efectivisables) y los efectivos propiamente dichos.

Ejercicio 6: De un procedimiento efectivo (efectivisable) que compute la funcion $\lambda ix[(x)_i]$

Ejercicio 7: Sean

$$\begin{aligned} f &: D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega \\ f_1 &: D_{f_1} \subseteq \omega \times \Sigma^* \to \omega \\ f_2 &: D_{f_2} \subseteq \omega \times \Sigma^* \to \omega \\ f_3 &: D_{f_3} \subseteq \omega \times \Sigma^* \to \Sigma^* \end{aligned}$$

(a) Es verdad que $D_{f \circ [f_1,f_2,f_3]} = D_{f_1} \cap D_{f_2} \cap D_{f_3}$?
(b) De una descripcion del dominio de $f \circ [f_1, f_2, f_3]$.
(c) Si f, f_1, f_2, f_3 son Σ-efectivamente computables, entonces $f \circ [f_1, f_2, f_3]$ lo es.

Ejercicio 8: V o F o I. Justifique

(a) Si $\mathbb{P}$ es un procedimiento efectivo que computa una funcion no vacia $f : D_f \subseteq \omega \to \omega$ entonces el conjunto de datos de salida de $\mathbb{P}$ es ω
(b) Si $\mathbb{P}$ y $\mathbb{Q}$ son procedimientos efectivos, entonces $\mathbb{PQ}$ lo es.
(c) Denotemos con e a la cantidad de veces que estornudó Alan Turing a lo largo de su vida. Sea $f : \omega \to \omega$, dada por $f(x) = x^e$. Entonces f es $\emptyset$-efectivamente computable.
(d) Si f y g son Σ-efectivamente computables entonces $f.g$ lo es

Ejercicio 9: Si $f : D_f \subseteq \omega \times \omega \times \{@, \uparrow\}^* \to \omega$ es tal que D_f es finito, entonces f es $\{@, \uparrow\}$-efectivamente computable

Ejercicio 10: Sean $f : D_f \subseteq \omega \times \Sigma^* \to \omega$ y $g : D_g \subseteq \omega \times \Sigma^* \to \omega$ funciones Σ-efectivamente computables tales que $D_f \cap D_g = \emptyset$. Sea $F : D_f \cup D_g \to \omega$ dada por

$$e \quad \to \quad \begin{cases} f(e) & \text{si } e \in D_f \\ g(e) & \text{si } e \in D_g \end{cases}$$

Pruebe que F es Σ-efectivamente computable.

Conjuntos Σ-efectivamente computables

Sea X un conjunto cualquiera y sea $S \subseteq X$. Usaremos χ_S^X para denotar la funcion

$$\begin{array}{rcl} \chi_S^X : X & \to & \omega \\ x & \to & \begin{cases} 1 \text{ si } x \in S \\ 0 \text{ si } x \notin S \end{cases} \end{array}$$

Llamaremos a χ_S^X la *funcion caracteristica de S con respecto a X*.

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*efectivamente computable* cuando la funcion $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ-efectivamente computable. O sea S es Σ-efectivamente computable sii hay un procedimiento efectivo $\mathbb{P}$, el cual computa $\chi_S^{\omega^n \times \Sigma^{*m}}$, es decir:

- El conjunto de datos de entrada de $\mathbb{P}$ es $\omega^n \times \Sigma^{*m}$, siempre termina y da como dato de salida un elemento de $\{0, 1\}$.
- Dado $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$, $\mathbb{P}$ da como salida al numero 1 si $(\vec{x}, \vec{\alpha}) \in S$ y al numero 0 si $(\vec{x}, \vec{\alpha}) \notin S$.

Notese que la definicion de conjunto Σ-efectivamente computable, para el caso $S = \emptyset$, tiene a priori cierta ambiguedad ya que cualesquiera sean $n, m \in \omega$ tenemos que $\emptyset \subseteq \omega^n \times \Sigma^{*m}$. De todas maneras, cualesquiera sean los n, m elejidos, siempre hay un procedimiento efectivo que computa a $\chi_\emptyset^{\omega^n \times \Sigma^{*m}}$, i.e. un procedimiento que siempre da como salida 0, cualesquiera sea el dato de entrada de $\omega^n \times \Sigma^{*m}$. Es decir que el conjunto $\emptyset$ es Σ-efectivamente computable cualesquiera sea el alfabeto Σ. Cabe destacar que para el caso de un conjunto $S \neq \emptyset$, nuestra definicion es inambigua ya que hay unicos $n, m \in \omega$ tales que $S \subseteq \omega^n \times \Sigma^{*m}$.

Si $\mathbb{P}$ es un procedimiento efectivo el cual computa a $\chi_S^{\omega^n \times \Sigma^{*m}}$, diremos que $\mathbb{P}$ *decide la pertenecia a S*, con respecto al conjunto $\omega^n \times \Sigma^{*m}$.

Ejercicio 11: $\omega^3 \times \Sigma^{*2}$ es Σ-efectivamente computable

Ejercicio 12: Sea $\Sigma = \{▲, \%\}$ y sea

$$S = \{(0, 1, \varepsilon), (55, 54, ▲\%▲\%▲), (1, 1, ▲▲)\}$$

Pruebe que S es Σ-efectivamente computable.

Ejercicio 13: Convensase que todo conjunto finito $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable.

Ejercicio 14: $\{(x, \alpha) \in \omega \times \Sigma^* : |\alpha|^x \text{ es par}\}$ es Σ-efectivamente computable

Ejercicio 15: Sean $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ conjuntos Σ-efectivamente computables. Entonces $S_1 \cup S_2, S_1 \cap S_2$ y $(\omega^n \times \Sigma^{*m}) - S_1$ son Σ-efectivamente computables.

Ejercicio 16: Sean $S \subseteq \omega$ y $L \subseteq \{@, \uparrow\}^*$ tales que $(0, \varepsilon) \in S \times L$. Entonces S y L son ambos $\{@, \uparrow\}$-efectivamente computables sii $S \times L$ es $\{@, \uparrow\}$-efectivamente computable

Conjuntos Σ-efectivamente enumerables

Supongamos que $k, l, n, m \in \omega$ y que $F : D_F \subseteq \omega^k \times \Sigma^{*l} \to \omega^n \times \Sigma^{*m}$. Supongamos ademas que $n + m \geq 1$. Entonces denotaremos con $F_{(i)}$ a la funcion $p_i^{n,m} \circ F$. Notar que

$$F_{(i)} : D_F \subseteq \omega^k \times \Sigma^{*l} \to \omega, \text{ para cada } i = 1, ..., n$$

$$F_{(i)} : D_F \subseteq \omega^k \times \Sigma^{*l} \to \Sigma^*, \text{ para cada } i = n + 1, ..., n + m$$

Por lo cual cada una de las funciones $F_{(i)}$ son Σ-mixtas.

Ejercicio 17: Sea $F : \{(x, \alpha) \in \omega \times \Sigma^* : |\alpha|^x \text{ es par}\} \subseteq \omega \times \Sigma^* \to \omega^2 \times \Sigma^*$ dada por $F(x, \alpha) = (x, x^2 + |\alpha|, \varepsilon)$. Diga quienes son $F_{(1)}$,$F_{(2)}$ y $F_{(3)}$. Que funcion es $[F_{(1)}, F_{(2)}, F_{(3)}]$?

Ejercicio 18: En la definicion anterior, para el caso $n + m = 1$, quien es $F_{(1)}$?

Ejercicio 19: Pruebe que si $F : D_F \subseteq \omega^k \times \Sigma^{*l} \to \omega^n \times \Sigma^{*m}$, con $k, l, n, m \in \omega$, entonces $F = [F_{(1)}, ..., F_{(n+m)}]$

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*efectivamente enumerable* cuando sea vacio o haya una funcion $F : \omega \to \omega^n \times \Sigma^{*m}$ tal que $I_F = S$ y $F_{(i)}$ sea Σ-efectivamente computable, para cada $i \in \{1, ..., n + m\}$.

Observacion: Notese que para el caso $n = m = 0$, la condicion anterior de que $F_{(i)}$ sea Σ-efectivamente computable, para cada $i \in \{1, ..., n + m\}$ se cumple vacuamente y por lo tanto la definicion anterior nos dice que un conjunto $S \subseteq \omega^0 \times \Sigma^{*0} = \{\Diamond\}$ sera Σ-efectivamente enumerable sii es vacio o hay una funcion $F : \omega \to \{\Diamond\}$, tal que $I_F = S$. Por supuesto, esto nos dice que $\emptyset$ y $\{\Diamond\}$ son Σ-efectivamente enumerables.

Para entender mejor la idea de esta definicion consideremos el siguiente resultado.

Lema 1. *Un conjunto no vacio $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente enumerable sii hay un procedimiento efectivo $\mathbb{P}$ tal que*

(1) *El conjunto de datos de entrada de $\mathbb{P}$ es ω*
(2) *$\mathbb{P}$ se detiene para cada $x \in \omega$*
(3) *El conjunto de datos de salida de $\mathbb{P}$ es igual a S. (Es decir, siempre que $\mathbb{P}$ se detiene, da como salida un elemento de S, y para cada elemento $(\vec{x}, \vec{\alpha}) \in S$, hay un $x \in \omega$ tal que $\mathbb{P}$ da como salida a $(\vec{x}, \vec{\alpha})$ cuando lo corremos con x como dato de entrada)*

Ejercicio 20: Pruebe el lema anterior. (Hacer el caso $n = 1$, $m = 2$).

Cuando un procedimiento $\mathbb{P}$ cumpla (1), (2) y (3) del lema anterior, diremos que $\mathbb{P}$ *enumera* a S. O sea que S es Σ-efectivamente enumerable sii es vacio o hay un procedimiento efectivo el cual enumera a S.

Dicho de otra forma un conjunto no vacio S es Σ-efectivamente enumerable sii hay un procedimiento efectivo $\mathbb{P}$ el cual tiene conjunto de datos de entrada ω y ademas para los sucesivos datos de entrada $0, 1, 2, 3, ...$, el procedimiento $\mathbb{P}$ produce respectivamente los datos de salida $e_0, e_1, e_2, e_3, ...$ de manera que $S = \{e_0, e_1, e_2, ...\}$. Cabe destacar aqui que puede suceder que $e_i = e_j$, para ciertos i, j, con $i \neq j$.

Algunos ejemplos:

(E1) Un procedimiento efectivo que enumera $\omega \times \omega$ es el siguiente:
Etapa 1
Si $x = 0$, dar como salida el par $(0, 0)$ y terminar. Si $x \neq 0$, calcular $x_1 = (x)_1$ y $x_2 = (x)_2$.
Etapa 2

Dar como dato de salida el par (x_1, x_2) y terminar

Como puede notarse el procedimiento es efectivo (efectivisable) y ademas el conjunto de datos de salida es $\omega \times \omega$ ya que si tomamos un par cualquiera $(a, b) \in \omega \times \omega$, el procedimiento lo dara como dato de salida para la entrada $x = 2^a 3^b$.

(E2) Veamos que $\omega^2 \times \Sigma^{*3}$ es Σ-efectivamente enumerable cualquiera sea el alfabeto no vacio Σ. Sea $\leq$ un orden total para el alfabeto Σ. Utilisando el orden $\leq$ podemos diseñar el siguiente procedimiento para enumerar $\omega^2 \times \Sigma^{*3}$:

Etapa 1

Si $x = 0$, dar como salida $(0, 0, \varepsilon, \varepsilon, \varepsilon)$ y terminar. Si $x \neq 0$, calcular

$x_1 = (x)_1$

$x_2 = (x)_2$

$\alpha_1 = *^{\leq}((x)_3)$

$\alpha_2 = *^{\leq}((x)_4)$

$\alpha_3 = *^{\leq}((x)_5)$

Etapa 2

Dar como dato de salida la 5-upla $(x_1, x_2, \alpha_1, \alpha_2, \alpha_3)$.

Ejercicio 21: Explique por que es efectivo el procedimiento de (E2)

Ejercicio 22: Pruebe que el procedimiento de (E2) tiene conjunto de datos de salida igual a $\omega^2 \times \Sigma^{*3}$.

Ejercicio 23: Sea $S = \{(x, \uparrow^x) : x \text{ es par}\}$. Pruebe que S es $\{@, \uparrow\}$-efectivamente enumerable.

Ejercicio 24: Sea $S = \{(x, \alpha) \in \omega \times \{@, \uparrow\}^* : \alpha \text{ tiene exactamente } x \text{ ocurrencias de } @\}$. Pruebe que S es $\{@, \uparrow\}$-efectivamente enumerable.

Ejercicio 25: Sean $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ conjuntos Σ-efectivamente enumerables. Entonces $S_1 \cup S_2$ es Σ-efectivamente enumerable

Ejercicio 26: Sean $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ conjuntos Σ-efectivamente enumerables. Entonces $S_1 \cap S_2$ es Σ-efectivamente enumerable. (En el apunte esta probado en forma de lema.)

Ejercicio 27: (Explicado en video en granlogico.com) Si $\mathbb{P}$ es un procedimiento efectivo cuyo conjunto de datos de entrada es $\omega \times \Sigma^*$ entonces el conjunto

$$\{(x, \alpha) \in \omega \times \Sigma^* : \mathbb{P} \text{ termina partiendo de } (x, \alpha)\}$$

es Σ-efectivamente enumerable. Saque como conclusion que el dominio de una funcion Σ-efectivamente computable es Σ-efectivamente enumerable.

Lema 2. *Si $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable entonces S es Σ-efectivamente enumerable.*

Proof. Supongamos $S \neq \emptyset$. Sea $(\vec{z}, \vec{\gamma}) \in S$, fijo. Sea $\mathbb{P}$ un procedimiento efectivo que compute a $\chi_S^{\omega^n \times \Sigma^{*m}}$. Ya vimos en el ejemplo anterior que $\omega^2 \times \Sigma^{*3}$ es Σ-efectivamente enumerable. En forma similar se puede ver que $\omega^n \times \Sigma^{*m}$ lo es. Sea $\mathbb{P}_1$ un procedimiento efectivo que enumere a $\omega^n \times \Sigma^{*m}$. Entonces el siguiente procedimiento enumera a S:

Dato de entrada: $x \in \omega$

Etapa 1

Realizar $\mathbb{P}_1$ con x de entrada para obtener como salida un $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$.

Etapa 2

Realizar $\mathbb{P}$ con $(\vec{x}, \vec{\alpha})$ de entrada para obtener el valor Booleano e de salida.

Etapa 3

Si $e = 1$ dar como dato de salida $(\vec{x}, \vec{\alpha})$. Si $e = 0$ dar como dato de salida $(\vec{z}, \vec{\gamma})$. ■

Como veremos mas adelante en la Guia 9, hay conjuntos que son Σ-efectivamente enumerables y no Σ-efectivamente computables. Sin envargo tenemos el siguiente interesante resultado:

Teorema 3. *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Son equivalentes*

(a) *S es Σ-efectivamente computable*
(b) *S y $(\omega^n \times \Sigma^{*m}) - S$ son Σ-efectivamente enumerables*

Proof. (a)⇒(b). Por el lema anterior tenemos que S es Σ-efectivamente enumerable. Notese ademas que, dado que S es Σ-efectivamente computable, $(\omega^n \times \Sigma^{*m}) - S$ tambien lo es (por que?). Es decir que aplicando nuevamente el lema anterior tenemos que $(\omega^n \times \Sigma^{*m}) - S$ es Σ-efectivamente enumerable.

(b)⇒(a). Si $S = \emptyset$ o $S = \omega^n \times \Sigma^{*m}$ es claro que se cumple (a). O sea que podemos suponer que ni S ni $(\omega^n \times \Sigma^{*m}) - S$ son igual al conjunto vacio. Sea $\mathbb{P}_1$ un procedimiento efectivo que enumere a S y sea $\mathbb{P}_2$ un procedimiento efectivo que enumere a $(\omega^n \times \Sigma^{*m}) - S$. Es facil ver que el siguiente procedimiento computa el predicado $\chi_S^{\omega^n \times \Sigma^{*m}}$:

Dato de entrada: $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$

Etapa 1

Darle a la variable T el valor 0.

Etapa 2

Realizar $\mathbb{P}_1$ con el valor de T como entrada para obtener de salida la upla $(\vec{y}, \vec{\beta})$.

Etapa 3

Realizar $\mathbb{P}_2$ con el valor de T como entrada para obtener de salida la upla $(\vec{z}, \vec{\gamma})$.

Etapa 4

Si $(\vec{y}, \vec{\beta}) = (\vec{x}, \vec{\alpha})$, entonces detenerse y dar como dato de salida el valor 1. Si $(\vec{z}, \vec{\gamma}) = (\vec{x}, \vec{\alpha})$, entonces detenerse y dar como dato de salida el valor 0. Si no suceden ninguna de las dos posibilidades antes mensionadas, aumentar en 1 el valor de la variable T y dirijirse a la Etapa 2. ■

Ejercicio 28: Sea $\Sigma = \{@, !, \%\}$. Si $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-efectivamente computable, entonces $\mathrm{Im}(f)$ es Σ-efectivamente enumerable

Ejercicio 29: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \times \Sigma^* \to \Sigma^*$ es Σ-efectivamente computable. Suponga ademas que $(1, @@) \in S$ y $f(1, @@) = @!!$. Pruebe que el conjunto

$$\{(x, \alpha) \in S : f(x, \alpha) = @!!\}$$

es Σ-efectivamente enumerable.

Ejercicio 30: Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. Pruebe que el conjunto

$$\{(x, y) \in S : (f(x, y), y) \in S \text{ y } f(f(x, y), y) = 0\}$$

es Σ-efectivamente enumerable.

Ejercicio 31: (Explicado en video en granlogico.com) Si $f : S \subseteq \omega \to \omega$ es Σ-efectivamente computable, entonces el conjunto

$$\{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

es Σ-efectivamente enumerable.

Ejercicio 32: Si $f : S \subseteq \omega \to \omega$ es Σ-efectivamente computable, entonces el conjunto

$$\{x \in S : x^2 \in S \text{ y } f(x) = f(x^2)\}$$

es Σ-efectivamente enumerable.

Ejercicio 33: Sea $\Sigma = \{@, !, \%\}$. Supongamos $L_1, L_2 \subseteq \Sigma^*$ son Σ-efectivamente enumerables. Entonces

$$\{\alpha \in L_1 : \text{hay un } \beta \in L_2 \text{ tal que } |\beta| = |\alpha|\}$$

es Σ-efectivamente enumerable.

Ejercicio 34: Sean $S \subseteq \omega$ y $L \subseteq \{@, \uparrow\}^*$ tales que $(0, \varepsilon) \in S \times L$. Entonces $S \times L$ es $\{@, \uparrow\}$-efectivamente enumerable si y solo si ambos S y L son $\{@, \uparrow\}$-efectivamente enumerables

Ejercicio 35: Sea $f : S \subseteq \Sigma^* \to \Sigma^*$ una funcion Σ-efectivamente computable. Supongamos que $\varepsilon \in S$ y que $f(\varepsilon) = \varepsilon$. Sea

$$L = \{\alpha \in S : f(\alpha) \in S \text{ y } f(f(\alpha)) = \alpha\}$$

Notar que $\varepsilon \in L$. Pruebe que L es Σ-efectivamente enumerable

Ejercicio 36: V o F o I. Justifique

(a) Denotemos con e a la cantidad de veces que estornudó Alan Turing a lo largo de su vida. Sea $S = \{e\}$. Entonces S es un conjunto $\emptyset$-efectivamente computable

(b) $(2, 4, 6, 8, 10, ...)$ es Σ-efectivamente enumerable

(c) Si $\mathbb{P}$ es un procedimiento efectivo, entonces $I_{\mathbb{P}}$ es Σ-efectivamente enumerable

(d) Sea $S \subseteq \omega$ y supongamos $\mathbb{P}$ es un procedimiento efectivo el cual enumera a S. Entonces el siguiente procedimiento (con dato de entrada $x \in \omega$) enumera a $\{u \in S : u \text{ es par}\}$:

  (i) Etapa 1: Guardar en la variable X el valor x.

  (ii) Etapa 2: Correr $\mathbb{P}$ con dato de entrada X y guardar en la variable E el dato de salida dado por $\mathbb{P}$

  (iii) Etapa 3: Si E es par, dar como salida E y detenerse. Caso contrario aumentar en 1 el valor de X e ir a Etapa 2

Ejercicios de examen

No se puede usar que el dominio (o la imagen) de una funcion Σ-efectivamente computable es Σ-efectivamente enumerable. Los procedimientos efectivos pueden ser escritos en forma "efectivisable" , por ejemplo puede usar las funciones $\#^{\leq}$, $*^{\leq}$, $\lambda ix[(x)_i]$, $\lambda xy[x \leq y]$, $\lambda\alpha[|\alpha|]$, $\lambda xy[x = y]$, $\lambda\alpha\beta\,[\alpha = \beta]$, $\lambda xy[x + y]$, $\lambda xy[x.y]$, $\lambda xy\left[x \dot{-} y\right]$, $\lambda x\alpha\,[\alpha^x]$, $\lambda xy\,[x^y]$, etc sin explicar en su procedimiento como haria para calcularlas en forma efectiva.

(1) Sean

$$f : D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega$$
$$f_1 : D_{f_1} \subseteq \omega \times \Sigma^* \to \omega$$
$$f_2 : D_{f_2} \subseteq \omega \times \Sigma^* \to \omega$$
$$f_3 : D_{f_3} \subseteq \omega \times \Sigma^* \to \Sigma^*$$

   (a) Es verdad que $D_{f \circ [f_1, f_2, f_3]} = D_{f_1} \cap D_{f_2} \cap D_{f_3}$?
   (b) De una descripcion del dominio de $f \circ [f_1, f_2, f_3]$.
   (c) Si f, f_1, f_2, f_3 son Σ-efectivamente computables, entonces $f \circ [f_1, f_2, f_3]$ lo es.

(2) Sean

$$f : D_f \subseteq \omega \times \Sigma^* \to \omega$$
$$f_1 : D_{f_1} \subseteq \omega \to \omega$$
$$f_2 : D_{f_2} \subseteq \omega \to \Sigma^*$$

   (a) Es verdad que $D_{f \circ [f_1, f_2]} = D_{f_1} \cap D_{f_2}$?
   (b) De una descripcion del dominio de $f \circ [f_1, f_2]$.
   (c) Si f, f_1, f_2 son Σ-efectivamente computables, entonces $f \circ [f_1, f_2]$ lo es.

(3) Sean

$$f : D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega$$
$$f_1 : D_{f_1} \subseteq \Sigma^* \to \omega$$
$$f_2 : D_{f_2} \subseteq \Sigma^* \to \omega$$
$$f_3 : D_{f_3} \subseteq \Sigma^* \to \Sigma^*$$

   (a) Es verdad que $D_{f \circ [f_1, f_2, f_3]} = D_{f_1} \cap D_{f_2} \cap D_{f_3}$?
   (b) De una descripcion del dominio de $f \circ [f_1, f_2, f_3]$.
   (c) Si f, f_1, f_2, f_3 son Σ-efectivamente computables, entonces $f \circ [f_1, f_2, f_3]$ lo es.

(4) Sean

$$f : D_f \subseteq \omega \times \omega \to \omega$$
$$f_1 : D_{f_1} \subseteq \omega \times \omega \times \Sigma^* \times \Sigma^* \to \omega$$
$$f_2 : D_{f_2} \subseteq \omega \times \omega \times \Sigma^* \times \Sigma^* \to \omega$$

   (a) Es verdad que $D_{f \circ [f_1, f_2]} = D_{f_1} \cap D_{f_2}$?
   (b) De una descripcion del dominio de $f \circ [f_1, f_2]$.
   (c) Si f, f_1, f_2 son Σ-efectivamente computables, entonces $f \circ [f_1, f_2]$ lo es.

(5) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : L_1 \to L_2$, con $L_1, L_2 \subseteq \Sigma^*$, es Σ-efectivamente computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) Pruebe que f^{-1} es Σ-efectivamente computable.

(6) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S_1 \subseteq \omega \times \Sigma^*$ y $S_2 \subseteq \omega$. Supongamos ademas que $f : S_1 \to S_2$ es Σ-efectivamente computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) Sean $g_1 : S_2 \to \omega$ y $g_2 : S_2 \to \Sigma^*$ funciones tales que $f^{-1} = [g_1, g_2]$. Pruebe que g_1 y g_2 son Σ-efectivamente computables.

(7) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S \subseteq \omega \times \Sigma^*$ y $L \subseteq \Sigma^*$. Supongamos ademas que $f : S \to L$ es Σ-efectivamente computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) Sean $g_1 : L \to \omega$ y $g_2 : L \to \Sigma^*$ funciones tales que $f^{-1} = [g_1, g_2]$. Pruebe que g_1 y g_2 son Σ-efectivamente computables.

(8) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S_1 \subseteq \omega$ y $S_2 \subseteq \omega \times \Sigma^*$. Supongamos ademas que $F : S_1 \to S_2$ es biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion F^{-1}.
   (b) Pruebe que si $F_{(1)}$ y $F_{(2)}$ son Σ-efectivamente computables, entonces F^{-1} es Σ-efectivamente computable.

(9) Un conjunto no vacio $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente enumerable sii hay un procedimiento efectivo $\mathbb{P}$ tal que
   (a) El conjunto de datos de entrada de $\mathbb{P}$ es ω
   (b) $\mathbb{P}$ se detiene para cada $x \in \omega$
   (b) El conjunto de datos de salida de $\mathbb{P}$ es igual a S. (Es decir, siempre que $\mathbb{P}$ se detiene, da como salida un elemento de S, y para cada elemento $(\vec{x}, \vec{\alpha}) \in S$, hay un $x \in \omega$ tal que $\mathbb{P}$ da como salida a $(\vec{x}, \vec{\alpha})$ cuando lo corremos con x como dato de entrada)

   (Hacer el caso $n = 1$, $m = 2$).

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

(11) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0, 0, \varepsilon) \in S$ y $f(0, 0, \varepsilon) = 0$. Pruebe que el conjunto
$$\{(x, y, \alpha) \in S : f(x, y, \alpha) = 0\}$$
es Σ-efectivamente enumerable.

(12) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. Pruebe que el conjunto
$$\{(x, y) \in S : f(x, y) = x\}$$
es Σ-efectivamente enumerable.

(13) Sea $\Sigma = \{@, \&\}$. Supongamos $f : L \subseteq \Sigma^* \times \Sigma^* \to \Sigma^*$ es Σ-efectivamente computable. Suponga ademas que $(@, \&) \in L$ y $f(@, \&) = @@@$. Pruebe que el conjunto
$$\{(\alpha, \beta) \in L : f(\alpha, \beta) = \alpha\alpha\alpha\}$$
es Σ-efectivamente enumerable.

(14) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. Pruebe que el conjunto
$$\{(x, y) \in S : (y, x) \in S \text{ y } f(x, y) = f(y, x)\}$$
es Σ-efectivamente enumerable.

(15) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. Pruebe que el conjunto
$$\{(x, y) \in S : (f(x, y), y) \in S \text{ y } f(f(x, y), y) = 0\}$$
es Σ-efectivamente enumerable.

(16) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0,0) \in S$ y $f(0,0) = 0$. Pruebe que el conjunto

$$\{x \in \omega : \text{ hay un } y\text{tal que } (x,y) \in S \text{ y } f(x,y) = 0\}$$

es Σ-efectivamente enumerable.

(17) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0,0) \in S$ y $f(0,0) = 0$. Pruebe que el conjunto

$$\{(x,x) : (x,x) \in S \text{ y } f(x,x) = x\}$$

es Σ-efectivamente enumerable.

(18) Sea $\Sigma = \{@, \&\}$. Supongamos $f : L \subseteq \Sigma^* \times \Sigma^* \to \Sigma^*$ es Σ-efectivamente computable. Suponga ademas que $(@, @) \in L$. Pruebe que el conjunto

$$\{(\alpha, \beta) \in L : (\beta, \alpha) \in L \text{ y } f(\alpha, \beta) = f(\beta, \alpha)\}$$

es Σ-efectivamente enumerable.

(19) Sea $\Sigma = \{@, \&\}$. Sea $f : S \subseteq \Sigma^* \to \omega$ una funcion Σ-efectivamente computable. Supongamos que $\varepsilon \in S$ y que $f(\varepsilon) = 0$. Sea

$$L = \{\alpha \in S : @^{f(\alpha)} \in S \text{ y } f(@^{f(\alpha)}) = 0\}$$

Notar que $\varepsilon \in L$. Pruebe que L es Σ-efectivamente enumerable

(20) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-efectivamente computable. Suponga ademas que $(0,0,\varepsilon) \in S$ y $f(0,0,\varepsilon) = 0$. Pruebe que el conjunto

$$\{(x,y,\alpha) \in S : (y,x,\alpha) \in S \text{ y } f(x,y,\alpha) = f(y,x,\alpha)\}$$

es Σ-efectivamente enumerable.

(21) Si $\mathbb{P}_1$ y $\mathbb{P}_2$ son procedimientos efectivos ambos con conjunto de datos de entrada igual a $\omega \times \Sigma^*$, entonces el conjunto

$$\{(\alpha, \beta) \in \Sigma^* \times \Sigma^* : \mathbb{P}_1 \text{ termina partiendo de } (0, \alpha) \text{ y } \mathbb{P}_2 \text{ termina partiendo de } (0, \beta)\}$$

es Σ-efectivamente enumerable.

(22) Sea $\Sigma = \{@, !, \%\}$. Si $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-efectivamente computable, entonces S es Σ-efectivamente enumerable

(23) Sea $\Sigma = \{@, !, \%\}$. Si $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-efectivamente computable, entonces $\mathrm{Im}(f)$ es Σ-efectivamente enumerable

(24) Sea $\Sigma = \{@, !, \%\}$. Si $f : L \subseteq \Sigma^* \times \Sigma^* \to \omega$ es Σ-efectivamente computable, entonces $\mathrm{Im}(f)$ es Σ-efectivamente enumerable

(25) Sea $\Sigma = \{@, !, \%\}$. Si $f_1 : S_1 \subseteq \omega \times \omega \times \Sigma^* \to \omega$ y $f_2 : S_2 \subseteq \omega \times \omega \times \Sigma^* \to \Sigma^*$ son Σ-efectivamente computables, entonces $\mathrm{Im}([f_1, f_2])$ es Σ-efectivamente enumerable

(26) Sea $\Sigma = \{@, !, \%\}$. Si $f : S \subseteq \omega \to \omega$ y $g : L \subseteq \Sigma^* \to \omega$ son Σ-efectivamente computables, entonces

$$\{x : x \in S,\ @^x \in L \text{ y } f(x) = g(@^x)\}$$

es Σ-efectivamente enumerable

(27) Sea $\Sigma = \{@, !, \%\}$. Si $f : S \subseteq \omega \to \omega$ y $g : L \subseteq \Sigma^* \to \omega$ son Σ-efectivamente computables, entonces

$$\{\alpha \in L : g(\alpha) \in S \text{ y } f(g(\alpha)) = 0\}$$

es Σ-efectivamente enumerable.

(28) Sea $\Sigma = \{@, !, \%\}$. Si $f : \omega \to \Sigma^*$ es Σ-efectivamente computable y biyectiva, entonces f^{-1} es Σ-efectivamente computable

(29) Sea $\Sigma = \{@, !, \%\}$. Si $f : L \subseteq \Sigma^* \to \omega$ y $g : S \subseteq \omega \to \Sigma^*$ son Σ-efectivamente computables, entonces $f \circ g$ es Σ-efectivamente computable

(30) Si $f : S \subseteq \omega \to \omega$ es Σ-efectivamente computable, entonces el conjunto

$$\{x \in S : x^2 \in S \text{ y } f(x) = f(x^2)\}$$

es Σ-efectivamente enumerable.

(31) Si $f : S \subseteq \omega \to \omega$ es Σ-efectivamente computable, entonces el conjunto

$$\{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

es Σ-efectivamente enumerable.

(32) Sea $\Sigma = \{@, !, \%\}$. Supongamos $L_1, L_2 \subseteq \Sigma^*$ son Σ-efectivamente enumerables. Entonces

$$\{\alpha \in L_1 : \text{hay un } \beta \in L_2 \text{ tal que } |\beta| = |\alpha|\}$$

es Σ-efectivamente enumerable.

(33) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \omega \times \omega \times \Sigma^*$ y $L \subseteq \Sigma^*$ son Σ-efectivamente enumerables. Entonces

$$\{(x, y, \alpha) \in S : \text{hay un } \beta \in L \text{ tal que } |\beta| = |\alpha|\}$$

es Σ-efectivamente enumerable.

(34) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \omega \times \omega$ y $S' \subseteq \omega$ son Σ-efectivamente enumerables. Entonces

$$\{(x, y) : (x, y) \in S \text{ y } x + y \in S'\}$$

es Σ-efectivamente enumerable.

(35) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \Sigma^* \times \Sigma^*$ y $L \subseteq \Sigma^*$ son Σ-efectivamente enumerables. Entonces

$$\{(\alpha, \beta) \in S : \text{hay un } \gamma \in L \text{ tal que } \alpha = \beta\gamma\}$$

es Σ-efectivamente enumerable.

(36) Sean $f : D_f \subseteq \omega \times \Sigma^* \to \omega$ y $g : D_g \subseteq \omega \times \Sigma^* \to \omega$ funciones Σ-efectivamente computables tales que $D_f \cap D_g = \emptyset$. Sea $F : D_f \cup D_g \to \omega$ dada por

$$e \quad \to \quad \begin{cases} f(e) & \text{si } e \in D_f \\ g(e) & \text{si } e \in D_g \end{cases}$$

Pruebe que F es Σ-efectivamente computable.

(37) Sea $\Sigma = \{@, !, \%\}$. Supongamos $L_1, L_2 \subseteq \Sigma^*$ son Σ-efectivamente enumerables. Entonces

$$\{\alpha \in L_1 : \text{hay un } \beta \in L_2 \text{ tal que } \beta^3 = \alpha\}$$

es Σ-efectivamente enumerable.

(38) Sean $S \subseteq \omega$ y $L \subseteq \{@, \uparrow\}^*$ tales que $(0, \varepsilon) \in S \times L$. Entonces $S \times L$ es $\{@, \uparrow\}$-efectivamente enumerable si y solo si ambos S y L son $\{@, \uparrow\}$-efectivamente enumerables

GUIA 4 DE LENGUAJES: EL PARADIGMA DE TURING

Tres paradigmas matematicos de la computabilidad efectiva

Ya que el concepto de procedimiento efectivo es un concepto intuitivo, impresiso y a priori no expresado en el formalismo matematico, los conceptos de

- Funcion Σ-efectivamente computable
- Conjunto Σ-efectivamente computable
- Conjunto Σ-efectivamente enumerable

tambien son impresisos y estan fuera del formalismo matematico, debido a que los tres se definen en terminos de la existencia de procedimientos efectivos. Por supuesto, los tres conceptos son fundamentales en el estudio teorico de la computabilidad por lo que es muy importante poder dar un modelo o formalizacion matematica de estos conceptos. Pero notese que los dos ultimos se definen en funcion del primero por lo que una formalizacion matematica precisa del concepto de funcion Σ-efectivamente computable, resuelve el problema de modelizar en forma matematica a estos tres conceptos.

En esta materia daremos las tres formalizaciones matematicas mas clasicas del concepto de funcion Σ-efectivamente computable. La primera y la mas apegada a la idea intuitiva de procedimiento efectivo es la dada por Alan Turing via la matematizacion del concepto de maquina. Llamaremos a esta modelizacion el *paradigma de Turing* y lo desarrollaremos en esta guia. La segunda, es la dada por Godel en su estudio de sistemas formales de la logica de primer orden. Llamaremos a esta modelizacion el *paradigma de Godel o el paradigma funcional o el paradigma recursivo*. Por ultimo veremos una formalizacion via un lenguaje de programacion imperativo. En honor a la influencia que tuvo Von Neumann en el diseño de la primer computadora de caracter universal (i.e. programable de proposito general), llamaremos a este paradigma el *paradigma de Neumann o el paradigma imperativo*. Dada la naturaleza filosofica e imprecisa del concepto de procedimiento efectivo y de sus conceptos derivados (i.e. funcion Σ-efectivamente computable, etc) a este conjunto de conceptos fundamentales para las ciencias de la computacion lo llamaremos el *paradigma filosofico*. En honor al filosofo y matematico Gottfried Leibniz llamaremos tambien al paradigma filosofico, el *paradigma de Leibniz*. Cabe destacar que Leibniz creo la primera maquina de calcular, llamada la Stepped Reckoner.

Con esta manera de hablar notese que los paradigmas matematicos de Turing, Godel y Neumann intentan modelizar al paradigma de Leibniz. Para darle un toque de humor expresaremos esto diciendo que Turing, Godel y Neumann intentan vencer a Leibniz.

Descripcion informal de las maquinas de Turing

Comenzaremos estudiando el concepto de maquina de Turing, el cual fue introducido por Alam Turing para formalizar o modelizar matematicamente la idea de procedimiento efectivo. Una vez definidas las maquinas podremos dar una modelizacion matematica precisa del concepto de funcion Σ-efectivamente computable.

Llamaremos a estas funciones Σ-Turing computables y seran aquellas que (en algun sentido que sera bien precisado matematicamente) pueden ser computadas por una maquina de Turing. Por supuesto, la fidedignidad de este concepto, es decir cuan buena es la modelizacion matematica dada por Turing, puede no ser clara al comienzo pero a medida que vayamos avanzando en nuestro estudio y conozcamos ademas los otros paradigmas y su relacion, quedara claro que el modelo de Turing es acertado.

Vivimos en un mundo plagado de maquinas (ascensores, celulares, relojes, taladros, etc). Una caracteristica comun a todas las maquinas es que tienen distintos estados posibles. Un estado es el conjunto de caracteristicas que determinan un momento concreto posible de la maquina cuando esta funcionando. Por ejemplo un estado posible de un ascensor seria:

- esta en el tercer piso, con la primer puerta abierta y la otra cerrada, esta apretado el boton de ir al sexto piso, etc

donde ponemos etc porque dependiendo del tipo de ascensor (si es con memoria, a que pisos puede ir, etc) habra mas datos que especificar para determinar un estado concreto.

Otra caracteristica comun de las maquinas es que interactuan de distintas formas con el usuario o mas generalmente su entorno. Dependiendo de que accion se ejecute sobre la maquina y en que estado este, la maquina realizara alguna tarea y ademas cambiara de estado. En general las maquinas son *deterministicas* en el sentido que siempre que esten en determinado estado y se les aplique determinada accion, realizaran la misma tarea y pasaran al mismo estado.

Las maquinas de Turing son un modelo abstracto de maquina con una cantidad finita de estados la cual trabaja sobre una cinta de papel dividida en cuadros e interactua o recibe acciones externas por medio de una cabeza lectora la cual lee de a un cuadro de la cinta a la ves y ademas puede borrar el contenido del cuadro leido y escribir en el un simbolo. Tambien la cabeza lectora puede moverse un cuadro hacia la izquierda o hacia la derecha. La cinta tiene un primer cuadro hacia su izquierda pero hacia la derecha puede extenderse todo lo necesario. En un cuadro de la cinta podra haber un simbolo o un cuadro puede simplemente estar en blanco. Es decir que habra un alfabeto Γ el cual consiste de todos los simbolos que pueden figurar en la cinta. Esto sera parte de los datos o caracteristicas de cada maquina, es decir, Γ puede cambiar dependiendo de la maquina. La maquina, en funcion del estado en que se encuentre y de lo que vea su cabeza lectora en el cuadro escaneado, podra modificar lo que encuentre en dicho cuadro (borrando y escribiendo algun nuevo simbolo), moverse a lo sumo un cuadro (izquierda, derecha o quedarse quieta), y cambiar de estado (posiblemente al mismo que tenia). Para simplificar supondremos que hay en Γ un simbolo el cual si aparece en un cuadro de la cinta, significara que dicho cuadro esta sin escribir o en blanco. Esto nos permitira describir mas facilmente el funcionamiento de la maquina. En gral llamaremos B a tal simbolo. Tambien por lo general llamaremos Q al conjunto de estados de la maquina.

Tambien cada maquina tendra un estado especial el cual sera llamado su estado inicial, generalmente denotado con q_0, el cual sera el estado en el que estara la maquina al comenzar a trabajar sobre la cinta. Hay otras caracteristicas que tendran las maquinas de Turing pero para dar un primer ejemplo ya nos basta. Describiremos una maquina de Turing M que tendra $\Gamma = \{@, a, b, B\}$ y tendra dos

estados, es decir $Q = \{q_0, q_1\}$. Obviamente q_0 sera su estado inicial y ademas el "comportamiento o personalidad" de M estara dado por las siguientes clausulas:

- Estando en estado q_0 si ve ya sea b o B o @, entonces se queda en estado q_0 y se mueve a la derecha
- Estando en estado q_0 si ve a entonces reescribe @, se mueve a la izquierda y cambia al estado q_1
- Estando en estado q_1 si ve a o b o B o @ entonces lo deja como esta, se mueve a la izquierda y queda en estado q_1

Supongamos ahora que tomamos una palabra $\alpha \in \Gamma^*$ y la distribuimos en la cinta dejando el primer cuadro en blanco y luego poniendo los simbolos de α en los siguientes cuadros. Supongamos ademas que ponemos la maquina en estado q_0 y con su cabeza lectora escaneando el primer cuadro de la cinta. Esto lo podemos representar graficamente de la siguiente manera

$$\begin{array}{cccccccc} B & \alpha_1 & ... & \alpha_n & B & B & B & ... \\ \uparrow & & & & & & & \\ q_0 & & & & & & & \end{array}$$

donde $\alpha_1, ..., \alpha_n$ son los sucesivos simbolos de α. Supongamos ademas que a ocurre an α. Dejamos al lector ir aplicando las clausulas de M para convencerse que luego de un rato de funcionar M, la situacion sera

$$\begin{array}{cccccccc} B & \beta_1 & ... & \beta_n & B & B & B & ... \\ \uparrow & & & & & & & \\ q_1 & & & & & & & \end{array}$$

donde $\beta_1 ... \beta_n$ es el resultado de reemplazar en α la primer ocurrencia de a por @.

Ejercicio 1: Que sucede cuando a no ocurre en α?

Una cosa que puede pasar es que para un determinado estado p y un $\sigma \in \Gamma$, la maquina no tenga contemplada ninguna accion posible. Por ejemplo sea M la maquina de Turing dada por $Q = \{q_0\}$, $\Gamma = \{@, \$, B\}$ y por la siguiente clausula:

- Estando en estado q_0 si ve ya sea @ o B, entonces se queda en estado q_0 y se mueve a la derecha

Es facil ver que si partimos de una situacion

$$\begin{array}{cccccccc} B & \alpha_1 & ... & \alpha_n & B & B & B & ... \\ \uparrow & & & & & & & \\ q_0 & & & & & & & \end{array}$$

donde $\alpha_1, ..., \alpha_n \in \Gamma$, entonces si ningun α_i es igual a \$, la maquina se movera indefinidamente hacia la derecha y en caso contrario se movera i pasos a la derecha y se detendra, donde i es el menor l tal que $\alpha_l = \$$.

Otro caso posible de detencion de una maquina de Turing es cuando esta escaneando el primer cuadro de la cinta y su unica accion posible implica moverse un cuadro a la izquierda. Tambien en estos casos diremos que la maquina se detiene ya que la cinta no es extensible hacia la izquierda.

Otra caracteristica de las maquinas de Turing es que poseen un *alfabeto de entrada* el cual esta contenido en el alfabeto Γ y en el cual estan los simbolos que se usaran para formar la configuracion inicial de la cinta (exepto B). En general lo denotaremos con Σ al alfabeto de entrada y los simbolos de $\Gamma - \Sigma$ son considerados

auxiliares. Tambien habra un conjunto F contenido en el conjunto Q de los estados de la maquina, cuyos elementos seran llamados *estados finales*.

El lenguaje $L(M)$. Diremos que una palabra $\alpha = \alpha_1 ... \alpha_n \in \Sigma^*$ es *aceptada por M por alcance de estado final* si partiendo de

$$\begin{array}{cccccccc} B & \alpha_1 & ... & \alpha_n & B & B & B & ... \\ \uparrow & & & & & & & \\ q_0 & & & & & & & \end{array}$$

en algun momento de la computacion M esta en un estado de F. Llamaremos $L(M)$ al conjunto formado por todas las palabras que son aceptadas por alcance de estado final

Ejercicio 2: Para cada uno de los siguientes conjuntos, encuentre una máquina de Turing M (con alfabeto de entrada $\Sigma = \{a, b\}$) tal que $L(M)$ sea dicho conjunto.
(a) $\{\alpha \in \{a, b\}^* : a \text{ ocurre en } \alpha\}$
(b) $\{ab\}$
(c) $\{a^n b^n : n \geq 2\}$ (explicada en video en granlogico.com)
(d) $\{\alpha \in \{a, b\}^* : |\alpha|_a \text{ es par y } |\alpha|_b \text{ es impar}\}$

Definicion matematica de maquina de Turing

Una *maquina de Turing* es una 7-upla $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ donde
- Q es un conjunto finito cuyos elementos son llamados *estados*
- Γ es un alfabeto que contiene a Σ
- Σ es un alfabeto llamado el *alfabeto de entrada*
- $B \in \Gamma - \Sigma$ es un simbolo de Γ llamado el *blank symbol*
- $\delta : D_\delta \subseteq Q \times \Gamma \to Q \times \Gamma \times \{L, R, K\}$
- q_0 es un estado llamado el *estado inicial* de M
- $F \subseteq Q$ es un conjunto de estados llamados *finales*

Notese que la funcion δ da la "personalidad" de la maquina. Describiremos esto informalmente y despues cuando definamos la relacion $\vdash$ mas abajo esta idea quedara precisada en forma matematica.

- $\delta(p, \sigma) = (q, \gamma, L)$ significara que: si M esta en estado p y su cabezal esta escaneando una casilla distinta de la primera, en la cual esta dibujado el simbolo σ, entonces la maquina hara lo siguiente:
  (a) borrara σ y escribira γ en su lugar, luego
  (b) se movera un cuadro a la izquierda y luego
  (c) pasara al estado q
- $\delta(p, \sigma) = (q, \gamma, K)$ significara que: si M esta en estado p y su cabezal esta escaneando una casilla en la cual esta dibujado el simbolo σ, entonces la maquina hara lo siguiente:
  (a) borrara σ y escribira γ en su lugar, luego
  (b) pasara al estado q
  (es decir que el cabezal se queda kieto)
- $\delta(p, \sigma) = (q, \gamma, R)$ significara que: si M esta en estado p y su cabezal esta escaneando una casilla en la cual esta dibujado el simbolo σ, entonces la maquina hara lo siguiente:

(a) borrara σ y escribira γ en su lugar, luego
(b) se movera un cuadro a la derecha y luego
(c) pasara al estado q

Los casos arriba descriptos son los unicos en los cuales la maquina M trabajara. O sea que si $(p, \sigma) \in (Q \times \Gamma) - D_\delta$ entonces M estando en estado p y con el cabezal leyendo el simbolo σ no puede hacer nada es decir solo permanecer quieta en el lugar de la cinta que este. Tambien si $\delta(p, \sigma) = (q, \gamma, L)$, entonces cuando M este en estado p, con su cabezal escaneando la primer casilla y en la cual este dibujado el simbolo σ, M no podra hacer nada, lo cual es razonable ya que la cinta no es extensible hacia la izquierda.

Asumsion Inocua: Si bien en nuestra definicion de maquina de Turing no hay ninguna restriccion acerca de la naturaleza de los elementos de Q, para continuar nuestro analisis asumiremos en lo que sigue de esta guia que Q es un alfabeto disjunto con Γ. Esto nos permitira dar definiciones matematicas precisas que formalizaran el funcionamiento de las maquinas de Turing en el contexto de las funciones mixtas. Deberia quedar claro que el hecho que solo trabajemos con maquinas en las cuales Q es un alfabeto disjunto con Γ, no afectara la profundidad y generalidad de nuestros resultados y definiciones.

Ejercicio 3: V o F o I, justifique.
(a) Si $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ es una maquina de Turing, entonces δ es una funcion $(Q \cup \Gamma \cup \{L, K, R\})$-mixta
(b) Si $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ es una maquina de Turing, entonces D_δ es un conjunto $(Q \cup \Gamma)$-mixto

Descripciones instantaneas. Una *descripcion instantanea* sera una palabra de la forma $\alpha q \beta$, donde $\alpha, \beta \in \Gamma^*$, $[\beta]_{|\beta|} \neq B$ y $q \in Q$. Notese que la condicion $[\beta]_{|\beta|} \neq B$ nos dice que $\beta = \varepsilon$ o el ultimo simbolo de β es distinto de B. La descripcion instantanea $\alpha_1 ... \alpha_n q \beta_1 ... \beta_m$, con $\alpha_1, ..., \alpha_n$, $\beta_1, ..., \beta_m \in \Gamma$, $n, m \geq 0$ representara la siguiente situacion

$$\begin{array}{cccccccccccc} \alpha_1 & \alpha_2 & ... & \alpha_n & \beta_1 & \beta_2 & ... & \beta_m & B & B & B & ... \\ & & & & \uparrow & & & & & & & \\ & & & & q & & & & & & & \end{array}$$

Notese que aqui n y m pueden ser 0. Por ejemplo si $n = 0$ tenemos que $\alpha_1 ... \alpha_n q \beta_1 ... \beta_m = q \beta_1 ... \beta_m$ y representa la siguiente situacion

$$\begin{array}{cccccccc} \beta_1 & \beta_2 & ... & \beta_m & B & B & B & ... \\ \uparrow & & & & & & & \\ q & & & & & & & \end{array}$$

Si $m = 0$ tenemos que $\alpha_1 ... \alpha_n q \beta_1 ... \beta_m = \alpha_1 ... \alpha_n q$ y representa la siguiente situacion

$$\begin{array}{cccccccccc} \alpha_1 & \alpha_2 & ... & \alpha_n & B & B & ... & & & ... \\ & & & & \uparrow & & & & & \\ & & & & q & & & & & \end{array}$$

Si ambos n y m son 0 entonces tenemos que $\alpha_1...\alpha_n q\beta_1...\beta_m = q$ y representa la siguiente situacion

$$\begin{array}{cccc} B & B & B & ... \\ \uparrow & & & \\ q & & & \end{array}$$

La condicion de que en una descripcion instantanea $\alpha q\beta$ deba suceder que $[\beta]_{|\beta|} \neq B$ es para que haya una correspondencia biuniboca entre descripciones instantaneas y situaciones de funcionamiento de la maquina. Dejamos al lector meditar sobre esto hasta convenserse de su veracidad.

Usaremos Des para denotar el conjunto de las descripciones instantaneas. Definamos la funcion $St : Des \to Q$, de la siguiente manera

$$St(d) = \text{unico simbolo de } Q \text{ que ocurre en } d$$

Ejercicio 4: V o F o I, justifique.

(a) Si d es una descripcion instantanea, entonces $Ti(d) = 3-\text{UPLA}$
(b) Si d es una descripcion instantanea, entonces $St(d) = d \cap Q$

La relacion $\vdash$. Dado $\alpha \in (Q \cup \Gamma)^*$, definamos $\lfloor \alpha \rfloor$ de la siguiente manera

$$\begin{aligned} \lfloor \varepsilon \rfloor &= \varepsilon \\ \lfloor \alpha\sigma \rfloor &= \alpha\sigma, \text{ si } \sigma \neq B \\ \lfloor \alpha B \rfloor &= \lfloor \alpha \rfloor \end{aligned}$$

Es decir $\lfloor \alpha \rfloor$ es el resultado de remover de α el tramo final mas grande de la forma B^n. Dada cualquier palabra α definimos

$$\curvearrowright\alpha = \begin{cases} [\alpha]_2 ... [\alpha]_{|\alpha|} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases} \qquad \alpha^{\curvearrowleft} = \begin{cases} [\alpha]_1 ... [\alpha]_{|\alpha|-1} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases}$$

Observacion: Notese que si $\alpha p\beta$ es una descripcion instantanea, entonces en la situacion real que ella describe la cabeza lectora de la maquina esta leyendo el simbolo $[\beta B]_1$ (o sea el 1er simbolo de β si $\beta \neq \varepsilon$ y B en caso contrario). Esta forma cheta de describir que simbolo lee la cabeza lectora nos sera util para definir a continuacion la relacion $\vdash$.

Dadas $d_1, d_2 \in Des$, escribiremos $d_1 \vdash d_2$ cuando existan $\sigma \in \Gamma$, $\alpha, \beta \in \Gamma^*$ y $p, q \in Q$ tales que se cumple alguno de los siguientes casos

Caso 1.

$$\begin{aligned} d_1 &= \alpha p\beta \\ \delta(p, [\beta B]_1) &= (q, \sigma, R) \\ d_2 &= \alpha\sigma q^{\curvearrowright}\beta \end{aligned}$$

Caso 2.

$$\begin{aligned} d_1 &= \alpha p\beta \\ \delta(p, [\beta B]_1) &= (q, \sigma, L) \text{ y } \alpha \neq \varepsilon \\ d_2 &= \left\lfloor \alpha^{\curvearrowleft} q\, [\alpha]_{|\alpha|}\, \sigma^{\curvearrowright}\beta \right\rfloor \end{aligned}$$

Caso 3.

$$d_1 = \alpha p \beta$$
$$\delta(p, [\beta B]_1) = (q, \sigma, K)$$
$$d_2 = \lfloor \alpha q \sigma^{\frown} \beta \rfloor$$

Escribiremos $d \nvdash d'$ para expresar que no se da $d \vdash d'$. Para $d, d' \in Des$ y $n \geq 0$, escribiremos $d \overset{n}{\vdash} d'$ si existen $d_1, ..., d_{n+1} \in Des$ tales que

$$d = d_1$$
$$d' = d_{n+1}$$
$$d_i \vdash d_{i+1}, \text{ para } i = 1, ..., n.$$

Notese que $d \overset{0}{\vdash} d'$ sii $d = d'$. Finalmente definamos

$$d \overset{*}{\vdash} d' \text{ sii } (\exists n \in \omega)\ d \overset{n}{\vdash} d'.$$

Ejercicio 5: V o F o I, justifique.

(a) $d \vdash d$, para cada $d \in Des$
(b) Si $\alpha p \beta \nvdash d$ para toda descripción instantánea d entonces $(p, [\beta B]_1) \notin D_\delta$
(c) Si $\delta(p, a) = (p, a, L)$ entonces $pa \nvdash d$ para toda descripción instantánea d
(d) Dadas $d, d' \in Des$, se tiene que si $d \vdash d'$, entonces $|d| \leq |d'| + 1$

Ejercicio 6: Sea $\Sigma = \{@, \%\}$.

(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que para cada $\alpha \in \Sigma^*$ haya un $q \in Q$ tal que

$$\lfloor q_0 B B \alpha \rfloor \overset{*}{\vdash} \lfloor q B \alpha \rfloor$$

(b) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que para cada $\alpha \in \Sigma^*$ haya un $q \in Q$ tal que

$$\lfloor q_0 B \alpha \rfloor \overset{*}{\vdash} \lfloor q B B \alpha \rfloor$$

Detencion (definicion matematica ahora). Dada $d \in Des$, diremos que M *se detiene partiendo de* d si existe $d' \in Des$ tal que

- $d \overset{*}{\vdash} d'$
- $d' \nvdash d''$, para cada $d'' \in Des$

Ejercicio 7: Estudie los dos posibles casos de detencion:

(a) estando el cabezal sobre el primer cuadro de la cinta
(b) estando el cabezal en un cuadro que no es el primero

Ejercicio 8: V o F o I, justifique.

(a) Sea $d \in Des$. Entonces existe una infinitupla $(d_1, d_2, ...)$ tal que $d \vdash d_1 \vdash d_2 \vdash d_3 \vdash d_4 \vdash \cdots$ si y solo si M no se detiene partiendo de d
(b) Supongamos que para cada $(p, \sigma) \in Q \times \Gamma$ la tercera coordenada de $\delta(p, \sigma)$ es igual a L. Entonces M se detiene partiendo de cada $d \in Des$

El lenguaje $L(M)$ (definicion matematica ahora). Diremos que una palabra $\alpha \in \Sigma^*$ es *aceptada por M por alcance de estado final* cuando

$$\lfloor q_0 B\alpha \rfloor \overset{*}{\vdash} d, \text{ con } d \text{ tal que } St(d) \in F.$$

El *lenguage aceptado por M por alcance de estado final* se define de la siguiente manera

$$L(M) = \{\alpha \in \Sigma^* : \alpha \text{ es aceptada por } M \text{ por alcance de estado final}\}.$$

Ejercicio 9: Para cada uno de los siguientes conjuntos, encuentre una máquina de Turing M tal que $L(M)$ sea dicho conjunto
(a) $\{\alpha \in \{@, \%\}^* : |\alpha|_@ = 2\,|\alpha|_\%\}$
(b) $\left\{\alpha \in \{@, \%\}^* : \alpha = \alpha^R\right\}$ (palabras capicuas)

Ejercicio 10: Sea $\Sigma = \{@, \%\}$. Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que para cada $\alpha, \beta \in \Sigma^*$, con $|\alpha| = |\beta|$, haya un $q \in Q$ tal que

$$\lfloor q_0 B\alpha\beta \rfloor \overset{*}{\vdash} \lfloor B\alpha q\beta \rfloor \text{ y } \lfloor B\alpha q\beta \rfloor \nvdash d, \text{ para cada } d \in Des$$

Ejercicio 11: Sea $\Sigma = \{@, \%\}$. Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que

$$L(M) = \{\beta\beta : \beta \in \{@, \%\}^+\}$$

Ejercicio 12: Sea $\Sigma = \{@, \%\}$.
(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que

$$L(M) = \{@^x(@\%)^y : x \neq y \text{ y } x, y \in \mathbf{N}\}$$

(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.
(i) $\alpha = @@\%$
(ii) $\alpha = \%\%\%$
(iii) $\alpha = \varepsilon$
(iv) $\alpha = @@\%@\%$
(Note que dicha sucesion puede ser finita o infinita)

Ejercicio 13: V o F o I, justifique.
(a) Si q_2 es un estado final de la máquina M, $\delta(q_0, B) = (q_1, B, R)$ y $\delta(q_1, a) = (q_2, b, R)$ entonces $a \in L(M)$.
(b) Si q_2 es un estado final de la máquina M, $\delta(q_0, B) = (q_1, B, R)$ y $\delta(q_1, a) = (q_2, b, R)$ entonces $b \in L(M)$.
(c) $\alpha \notin L(M)$ si y solo si existe una infinitupla $(d_1, d_2, ...)$ tal que
(i) $St(d_i) \notin F$, para cada $i = 1, 2, ...$
(ii) $\lfloor q_0 B\alpha \rfloor \vdash d_1 \vdash d_2 \vdash d_3 \vdash d_4 \vdash \cdots$

Funciones Σ-Turing computables

Para poder computar funciones mixtas con una maquina de Turing necesitaremos un simbolo para representar numeros sobre la cinta. Llamaremos a este simbolo *unit* y lo denotaremos con ı. Mas formalmente una *maquina de Turing con unit* es una 8-upla $M = (Q, \Sigma, \Gamma, \delta, q_0, B, ı, F)$ tal que $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ es una maquina de Turing y ı es un simbolo distingido perteneciente a $\Gamma - (\{B\} \cup \Sigma)$.

Diremos que una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$ es Σ-*Turing computable* si existe una maquina de Turing con unit, $M = (Q, \Sigma, \Gamma, \delta, q_0, B, \shortmid, F)$ tal que:

(1) Si $(\vec{x}, \vec{\alpha}) \in D_f$, entonces hay un $p \in Q$ tal que

$$\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor \overset{*}{\vdash} \lfloor pBf(\vec{x}, \vec{\alpha}) \rfloor$$

y $\lfloor pBf(\vec{x}, \vec{\alpha}) \rfloor \nvdash d$, para cada $d \in Des$

(2) Si $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} - D_f$, entonces M no se detiene partiendo de

$$\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor .$$

En forma similar, una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$, es llamada Σ-*Turing computable* si existe una maquina de Turing con unit, $M = (Q, \Sigma, \Gamma, \delta, q_0, B, \shortmid, F)$, tal que:

(1) Si $(\vec{x}, \vec{\alpha}) \in D_f$, entonces hay un $p \in Q$ tal que

$$\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor \overset{*}{\vdash} \left\lfloor pB \shortmid^{f(\vec{x}, \vec{\alpha})} \right\rfloor$$

y $\left\lfloor pB \shortmid^{f(\vec{x}, \vec{\alpha})} \right\rfloor \nvdash d$, para cada $d \in Des$

(2) Si $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} - D_f$, entonces M no se detiene partiendo de

$$\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor$$

Cabe destacar que la condicion

$$\lfloor pBf(\vec{x}, \vec{\alpha}) \rfloor \nvdash d, \text{ para cada } d \in Des$$

es equivalente a que (p, B) no este en el dominio de δ o que si lo este y que la tercer coordenada de $\delta(p, B)$ sea L.

Cuando una maquina de Turing con unit M cumpla los items (1) y (2) de la definicion anterior, diremos que M *computa* a la funcion f o que f es *computada* por M. Por supuesto esta definicion no tendria sentido como modelo matematico del concepto de funcion Σ-efectivamente computable si no sucediera que toda funcion Σ-Turing computable fuera Σ-efectivamente computable. Este hecho es intuitivamente claro y lo presentamos a continuacion en forma de proposicion. En algun sentido esto nos dice que el paradigma filosofico es mas amplio (o igual) al paradigma de Turing. Para darle un toque de humor expresaremos esto diciendo que Leibniz vence a Turing.

Proposición 1 (Leibniz vence a Turing)**.** *Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$, con $O \in \{\omega, \Sigma^*\}$, es computada por una maquina de Turing con unit $M = (Q, \Sigma, \Gamma, \delta, q_0, B, \shortmid, F)$, entonces f es Σ-efectivamente computable.*

Proof. Haremos el caso $O = \Sigma^*$. Sea $\mathbb{P}$ el siguiente procedimiento efectivo.

- Conjunto de datos de entrada de $\mathbb{P}$ igual a $\omega^n \times \Sigma^{*m}$
- Conjunto de datos de salida de $\mathbb{P}$ contenido en O
- Funcionamiento: Hacer funcionar paso a paso la maquina M partiendo de la descripcion instantanea $\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor$. Si en alguna instancia M termina, dar como salida el resultado de remover de la descripcion instantanea final los dos primeros simbolos.

Notese que este procedimiento termina solo en aquelos elementos $(\vec{x}, \vec{\sigma}) \in \omega^n \times \Sigma^{*m}$ tales que la maquina M termina partiendo desde

$$\lfloor q_0 B \shortmid^{x_1} B...B \shortmid^{x_n} B\alpha_1 B...B\alpha_m \rfloor$$

por lo cual termina solo en los elementos de D_f ya que M computa a f. Ademas es claro que en caso de terminacion el procedimiento da como salida $f(\vec{x}, \vec{\sigma})$. ■

Sin envargo el modelo Turingniano podria a priori no ser del todo correcto ya que podria pasar que haya una funcion que sea computada por un procedimiento efectivo pero que no exista una maquina de Turing que la compute. En otras palabras el modelo podria ser incompleto. La completitud de este modelo puede no ser clara al comienzo pero a medida que vayamos avanzando en nuestro estudio y conozcamos ademas los otros paradigmas y su relacion, quedara claro que el modelo de Turing es acertado, es decir que tambien pasa que Turing vence a Leibniz.

Ejercicio 14: Sea $\Sigma = \{a, b\}$. Para cada una de las siguientes funciones Σ-mixtas dar una máquina de Turing $(Q, \Gamma, \Sigma, \delta, q_0, B, \shortmid, \emptyset)$ que la compute

(a) $\begin{array}{rcl} Suc : \omega & \to & \omega \\ n & \to & n+1 \end{array}$

(b) $\begin{array}{rcl} Pred : \mathbf{N} & \to & \omega \\ n & \to & n-1 \end{array}$

(c) $\begin{array}{rcl} p_2^{1,1} : \omega \times \Sigma^* & \to & \Sigma^* \\ (x, \alpha) & \to & \alpha \end{array}$

(explicada en video en granlogico.com)

(d) $\begin{array}{rcl} C_2^{1,1} : \omega \times \Sigma^* & \to & \omega \\ (x, \alpha) & \to & 2 \end{array}$

(e) $\lambda xy[x + y]$

(f) $\lambda xy[x.y]$

(g) $\begin{array}{rcl} Q : \omega \times \mathbf{N} & \to & \omega \\ (x, y) & \to & \text{cociente de la division de } x \text{ por } y \end{array}$

(h) $\begin{array}{rcl} R : \omega \times \mathbf{N} & \to & \omega \\ (x, y) & \to & \text{resto de la division de } x \text{ por } y \end{array}$

Ejercicio 15: Sea $\Sigma = \{@, \%\}$. Sea

$$\begin{array}{rcl} f : \{(x, \alpha) \in \omega \times \Sigma^* : |\alpha| \text{ es impar}\} & \to & \omega \\ (x, \alpha) & \to & x + |\alpha| \end{array}$$

(a) De el diagrama de una maquina de Turing M la cual compute a f.

(b) Para cada una de los siguientes pares (x, α) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B \shortmid^x B\alpha \rfloor$.

(i) $(x, \alpha) = (0, \varepsilon)$

(ii) $(x, \alpha) = (100, @)$

(iii) $(x, \alpha) = (3, @@\%)$

(iv) $(x, \alpha) = (100, @\%)$

(Note que dicha sucesion para ciertos casos debe ser infinita)

Ejercicio 16: Sea $\Sigma = \{@, \%\}$. Sea

$$\begin{array}{rcl} R : \{\beta\beta : \beta \in \{@, \%\}^+\} & \to & \Sigma^* \\ \alpha & \to & \text{único } \beta \text{ tal que } \alpha = \beta\beta \end{array}$$

(a) De el diagrama de una maquina de Turing M la cual compute a R.

(b) Para cada una de las siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.

(i) $\alpha = \varepsilon$

(ii) $\alpha = @@$

(iii) $\alpha = @@\%\%$
(iv) $\alpha = @\%@\%\%$
(v) $\alpha = @@\%@$
(Note que dicha sucesion para ciertos casos debe ser infinita)

Ejercicio 17: Sea $\Sigma = \{@, \%\}$ y sea

$$\begin{array}{rcl} f : \{(\alpha, \beta) \in \Sigma^* \times \Sigma^* : \alpha = \beta\} & \to & \Sigma^* \\ (\alpha, \beta) & \to & \alpha \end{array}$$

(a) De el diagrama de una maquina de Turing M la cual compute a f.
(b) Para cada una de los siguientes pares (α, β) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B \alpha B \beta \rfloor$.
(i) $(\alpha, \beta) = (\%@, \%)$
(ii) $(\alpha, \beta) = (@\%, @\%)$
(iii) $(\alpha, \beta) = (\varepsilon, @@)$
(Note que dicha sucesion para ciertos casos debe ser infinita)

Ejercicio 18: V o F o I, justifique.
(a) Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, \shortmid, F)$ una maquina de Turing con unit y supongamos que M computa a f. Entonces $D_f = \{d \in Des : M \text{ se detiene partiendo desde } d\}$
(b) Si f y g son dos funciones y M es es una máquina de Turing que computa a f y a g entonces $f = g$.
(c) Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, \shortmid, F)$ una maquina de Turing con unit y supongamos que M computa a f y que f es Σ-total. Entonces M se detiene partiendo desde d, cualesquiera sea $d \in Des$

Ejercicio 19: Como se vio anteriormente el modelo de Turing del concepto de funcion Σ-efectivamente computable es el concepto matematico de funcion Σ-Turing computable.
(a) Cual seria el modelo de Turing del concepto de conjunto Σ-efectivamente computable? Mas concretamente ud debe continuar esta definicion: " Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-*Turing computable* cuando ....
(b) Cual seria el modelo de Turing del concepto de conjunto Σ-efectivamente enumerable? Mas concretamente ud debe continuar esta definicion: " Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-*Turing enumerable* cuando ....

Ejercicios de examen

(1) Sea $\Sigma = \{@, \%\}$.
(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que

$$L(M) = \left\{\alpha \in \Sigma^* : \alpha = \alpha^R\right\} \text{ (palabras capicuas)}$$

(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B \alpha \rfloor$.
(i) $\alpha = @@\%$
(ii) $\alpha = \%@\%$
(iii) $\alpha = \varepsilon$
(iv) $\alpha = \%\%$
(Note que dicha sucesion puede ser finita o infinita)

(2) Sea $\Sigma = \{@, \%\}$.

(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que
$$L(M) = \{@^x\%^y : x < y \text{ y } x, y \in \omega\}$$
(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.
(i) $\alpha = @@\%$
(ii) $\alpha = \%@\%$
(iii) $\alpha = @\%\%$
(iv) $\alpha = \%$
(Note que dicha sucesion puede ser finita o infinita)

(3) Sea $\Sigma = \{@, \%\}$. Hacer
(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que
$$L(M) = \{@^{2x}\%^x : x \in \mathbf{N}\}$$
(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.
(i) $\alpha = @\%$
(ii) $\alpha = \%\%\%$
(iii) $\alpha = \varepsilon$
(iv) $\alpha = @@@@\%\%$
(Note que dicha sucesion puede ser finita o infinita)

(4) Sea $\Sigma = \{@, \%\}$. Hacer
(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que
$$L(M) = \{@^x\%^{2x} : x \in \mathbf{N}\}$$
(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.
(i) $\alpha = @\%$
(ii) $\alpha = \%\%\%$
(iii) $\alpha = \varepsilon$
(iv) $\alpha = @@\%\%\%\%$
(Note que dicha sucesion puede ser finita o infinita)

(5) Sea $\Sigma = \{@, \%\}$. Hacer
(a) Dar el grafico de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ tal que
$$L(M) = \{@^x(@\%)^y : x \neq y \text{ y } x, y \in \mathbf{N}\}$$
(b) Para cada una de los siguientes palabras α dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha \rfloor$.
(i) $\alpha = @@\%$
(ii) $\alpha = \%\%\%$
(iii) $\alpha = \varepsilon$
(iv) $\alpha = @@\%@\%$
(Note que dicha sucesion puede ser finita o infinita)

(6) Sea $\Sigma = \{@, \%\}$ y sea
$$\begin{aligned} f : \{(\alpha, \beta) \in \Sigma^* \times \Sigma^* : \alpha \neq \beta\} &\to \Sigma^* \\ (\alpha, \beta) &\to \alpha \end{aligned}$$
(a) De el diagrama de una maquina de Turing M la cual compute a f.

(b) Para cada una de los siguientes pares (α, β) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha B\beta \rfloor$.
   (i) $(\alpha, \beta) = (\%@, \%)$
   (ii) $(\alpha, \beta) = (@\%, @\%)$
   (iii) $(\alpha, \beta) = (\varepsilon, @@)$

   (Note que dicha sucesion para ciertos casos debe ser infinita)

(7) Sea $\Sigma = \{@, \%\}$ y sea

$$\begin{array}{rcl} f : \{(x, \alpha) \in \omega \times \Sigma^* : |\alpha| \neq x\} & \to & \Sigma^* \\ (x, \alpha) & \to & \alpha \end{array}$$

(a) De el diagrama de una maquina de Turing M la cual compute a f.
(b) Para cada una de los siguientes pares (x, α) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B \, \imath^x \, B\alpha \rfloor$.
   (i) $(x, \alpha) = (2, \%)$
   (ii) $(x, \alpha) = (2, @\%)$
   (iii) $(x, \alpha) = (0, \varepsilon)$

   (Note que dicha sucesion para ciertos casos debe ser infinita)

(8) Sea $\Sigma = \{@, \%\}$ y sea

$$\begin{array}{rcl} f : \{(\alpha, \beta) \in \Sigma^+ \times \Sigma^* : |\alpha| < |\beta|\} & \to & \Sigma^* \\ (\alpha, \beta) & \to & [\beta]_{|\alpha|} \end{array}$$

(a) De el diagrama de una maquina de Turing M la cual compute a f.
(b) Para cada una de los siguientes pares (α, β) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha B\beta \rfloor$.
   (i) $(\alpha, \beta) = (\%, \%)$
   (ii) $(\alpha, \beta) = (@, \%@\%)$
   (iii) $(\alpha, \beta) = (\varepsilon, @@)$

   (Note que dicha sucesion para ciertos casos debe ser infinita)

(9) Sea $\Sigma = \{@, \%\}$ y sea $\leq$ el orden total sobre Σ dado por $@ < \%$. De el diagrama de una maquina de Turing M la cual compute a $\#^{\leq}$.

(10) Sea $\Sigma = \{@, \%\}$ y sea $\leq$ el orden total sobre Σ dado por $@ < \%$. De el diagrama de una maquina de Turing M la cual compute a $*^{\leq}$.

(11) Sea $\Sigma = \{@, \%\}$.
(a) De el diagrama de una maquina de Turing M la cual compute a $\lambda\alpha\beta[\alpha = \beta]$.
(b) Para cada una de los siguientes pares (α, β) dar la sucecion de descripciones instantaneas que parte de $\lfloor q_0 B\alpha B\beta \rfloor$.
   (i) $(\alpha, \beta) = (\%, \%)$
   (ii) $(\alpha, \beta) = (@\%, @)$
   (iii) $(\alpha, \beta) = (\varepsilon, @@)$

(12) Sea $\Sigma = \{@, \%\}$. De el diagrama de una maquina de Turing M la cual compute a $\lambda\alpha\beta[\beta\alpha]$.

(13) Sea $\Sigma = \{@, \%\}$ y sea $\leq$ el orden total sobre Σ dado por $@ < \%$. Sea $<$ el orden "menor" asociado al orden total de Σ^* inducido por $\leq$. De el diagrama de una maquina de Turing M la cual compute a $\lambda\alpha\beta[\alpha < \beta]$. (Hint: use la caracterizacion lexicografica de $<$ dada en la Guia 2).

GUIA 5 DE LENGUAJES: FUNCIONES Σ-RECURSIVAS PRIMITIVAS

El paradigma de Godel: Funciones Σ-recursivas

En esta guia y la siguiente desarrollaremos el modelo matematico del concepto de funcion Σ-efectivamente computable, dado por Godel. Dichas funciones seran llamadas Σ-recursivas. La idea es partir de un conjunto inicial de funciones muy simples y obviamente Σ-efectivamente computables y luego obtener nuevas funciones Σ-efectivamente computables usando constructores que preservan la computabilidad efectiva. Las funciones Σ-recursivas seran las que se obtienen iterando el uso de estos constructores, partiendo del conjunto inicial de funciones antes mencionado. Nos referiremos a este paradigma como el paradigma Godeliano o recursivo. A veces tambien lo llamaremos el paradigma funcional.

La familia de funciones simples y obviamente Σ-efectivamente computables de la que partiremos es la siguiente

$$\left\{Suc, Pred, C_0^{0,0}, C_\varepsilon^{0,0}\right\} \cup \{d_a : a \in \Sigma\} \cup \left\{p_j^{n,m} : 1 \leq j \leq n+m\right\}$$

Los constructores que usaremos son:

- Composicion
- Recursion primitiva
- Minimizacion de predicados totales

Estos constructores nos permiten dadas ciertas funciones construir o definir una nueva funcion y tienen la propiedad de preservar la computabilidad efectiva en el sentido que si las funciones iniciales son Σ-efectivamente computables, entonces la funcion obtenida tambien lo es. Un concepto fundamental es el de funcion Σ-recursiva primitiva. Estas funciones seran aquellas que se obtienen a partir de las del conjunto inicial usando solo los dos primeros constructores: composicion y recursion primitiva. Nuestro primer objetivo es definir el concepto de funcion Σ-recursiva primitiva para lo cual en las proximas dos secciones definiremos y estudiaremos los constructores de composicion y recursion primitiva. Luego definiremos el concepto de funcion Σ-recursiva primitiva y nos abocaremos a desarrollar este concepto fundamental. Recien despues ya en la Guia 6 estudiaremos el constructor de minimizacion y definiremos el concepto de funcion Σ-recursiva.

Composicion. Dadas funciones Σ-mixtas $f, f_1, ..., f_r$, con $r \geq 1$, diremos que la funcion $f \circ [f_1, ..., f_r]$ es *obtenida por composicion a partir de las funciones* $f, f_1, ..., f_r$. Un hecho que a priori no es obvio es que si $f, f_1, ..., f_r$ son Σ-mixtas, entonces $f \circ [f_1, ..., f_r]$ lo es. Esto es consecuencia del siguiente lema.

Lema 1. *Supongamos que $f, f_1, ..., f_r$ son funciones Σ-mixtas, con $r \geq 1$. Supongamos ademas que $f \circ [f_1, ..., f_r] \neq \emptyset$. Entonces hay $n, m, k, l \in \omega$ y $s \in \{\#, *\}$ tales que*

- $r = n + m$

- f *es de tipo* (n, m, s)
- f_i *es de tipo* $(k, l, \#)$*, para cada* $i = 1, ..., n$
- f_i *es de tipo* $(k, l, *)$*, para cada* $i = n + 1, ..., n + m$

Mas aun, en tal caso la funcion $f \circ [f_1, ..., f_{n+m}]$ *es de tipo* (k, l, s) *y:*

$$D_{f\circ[f_1,...,f_{n+m}]} = \left\{ (\vec{x}, \vec{\alpha}) \in \bigcap_{i=1}^{n+m} D_{f_i} : (f_1(\vec{x}, \vec{\alpha}), ..., f_{n+m}(\vec{x}, \vec{\alpha})) \in D_f \right\}$$

$$f \circ [f_1, ..., f_{n+m}](\vec{x}, \vec{\alpha}) = f(f_1(\vec{x}, \vec{\alpha}), ..., f_{n+m}(\vec{x}, \vec{\alpha})).$$

Ejercicio 1: Justifique con palabras la veracidad del lema anterior (a lo mariposa)

Ahora si es facil probar que la composicion preserva la computabilidad efectiva. Mas formalmente:

Lema 2. *Si* $f, f_1, ..., f_r$*, con* $r \geq 1$*, son* Σ*-efectivamente computables, entonces* $f \circ [f_1, ..., f_r]$ *lo es.*

Proof. Si $f \circ [f_1, ..., f_r] = \emptyset$, entonces claramente es Σ-efectivamente computable. Supongamos entonces que $f \circ [f_1, ..., f_r] \neq \emptyset$. Por el lema anterior hay $n, m, k, l \in \omega$ y $s \in \{\#, *\}$ tales que

- $r = n + m$
- f es de tipo (n, m, s)
- f_i es de tipo $(k, l, \#)$, para cada $i = 1, ..., n$
- f_i es de tipo $(k, l, *)$, para cada $i = n + 1, ..., n + m$

Sean $\mathbb{P}, \mathbb{P}_1, ..., \mathbb{P}_{n+m}$ procedimientos efectivos los cuales computen las funciones $f, f_1, ..., f_{n+m}$, respectivamente. Usando estos procedimientos es facil definir un procedimiento efectivo el cual compute a $f \circ [f_1, ..., f_{n+m}]$ ■

Ejercicio 2: Complete la prueba anterior

Recursion primitiva. La recursion primitiva es un tipo muy particular de recursion. Mas adelante lo definiremos matematicamente pero antes daremos varios ejemplos para aproximarnos gradualmente a la definicion. Consideremos por ejemplo las siguientes ecuaciones:

(1) $R(0) = 1$
(2) $R(t + 1) = 1 + R(t) + R(t)^2$

Notese que hay una unica funcion $R : \omega \to \omega$ la cual cumple (1) y (2). Esto es ya que el valor de R en t esta determinado por sucesivas aplicaciones de las ecuaciones (1) y (2). Por ejemplo la ecuacion (1) nos dice que $R(0) = 1$ pero entonces la ecuacion (2) nos dice que $R(1) = 1 + 1 + 1^2 = 3$ por lo cual nuevamente la ecuacion (2) nos dice que $R(2) = 1 + 3 + 3^2 = 13$ y asi podemos notar facilmente que R esta determinada por dichas ecuaciones.

Se suele decir que las ecuaciones (1) y (2) definen recursivamente a la funcion R pero hay que tener cuidado porque esto es una manera de hablar ya que la funcion R podria en nuestro discurso ya haber sido definida de otra manera. Mas propio es pensar que dichas ecuaciones determinan a R en el sentido que R es la unica que las cumple. Por ejemplo las ecuaciones:

(a) $R(0) = 50$

(b) $R(t+1) = R(t)$

"definen recursivamente" a la funcion $C_{50}^{1,0}$ pero esta claro que la definicion de $C_{50}^{1,0}$ en esta materia no fue dada de esta forma.

Ejercicio 3: Encuentre ecuaciones que "definan recursivamente" a la funcion $R = \lambda t[2^t]$

Hay casos de recursiones en las cuales el valor de $R(t+1)$ no solo depende de $R(t)$ sino que tambien depende de t. Por ejemplo

(i) $R(0) = 1$
(ii) $R(t+1) = t.R(t) + 1$

De todas maneras deberia quedar claro que las ecuaciones (i) y (ii) determinan una unica funcion $R : \omega \to \omega$ que las satisface.

Ejercicio 4: Encuentre ecuaciones que "definan recursivamente" a la funcion $R = \lambda t[t!]$

Tambien podemos generalizar pensando que la funcion R depende no solo de un parametro t sino que tiene otras variables. Por ejemplo

(p) $R(0, x_1, x_2, x_3) = x_1 + 2x_3$
(q) $R(t+1, x_1, x_2, x_3) = t + x_1 + x_2 + x_3 + R(t, x_1, x_2, x_3)$

Ejercicio 5: Explique con palabras por que las ecuaciones (p) y (q) determinan una unica funcion $R : \omega^4 \to \omega$. Cuanto vale $R(3, 1, 2, 3)$?

Por supuesto la cantidad de variables extra puede ser cualquiera y no justo 3.

Ejercicio 6: Encuentre ecuaciones que "definan recursivamente" a la funcion $R = \lambda t x_1[t + x_1]$, usando la funcion Suc.

Tambien podriamos tener variables alfabeticas. Por ejemplo consideremos

(r) $R(0, x_1, x_2, \alpha_1, \alpha_2) = x_1 + |\alpha_1|^{x_2}$
(s) $R(t+1, x_1, x_2, \alpha_1, \alpha_2) = t + x_1 + x_2 + |\alpha_1| + |\alpha_2| + R(t, x_1, x_2, \alpha_1, \alpha_2)$

Es claro aqui que las ecuaciones (r) y (s) determinan una unica funcion $R : \omega^3 \times \Sigma^{*2} \to \omega$ que las cumple. Esto se puede explicar de la siguiente manera:

- La ecuacion (r) determina los valores de R sobre el conjunto $\{0\} \times \omega \times \omega \times \Sigma^* \times \Sigma^*$. Pero una ves determinados estos valores, la ecuacion (s) tomada con $t = 0$, determina los valores de R sobre el conjunto $\{1\} \times \omega \times \omega \times \Sigma^* \times \Sigma^*$. Pero una ves determinados estos valores, la ecuacion (s) tomada con $t = 1$, determina los valores de R sobre el conjunto $\{2\} \times \omega \times \omega \times \Sigma^* \times \Sigma^*$, etc

El caso anterior podria generalizarse de la siguiente manera: Si tenemos dadas dos funciones

$$f : \omega^n \times \Sigma^{*m} \to \omega$$
$$g : \omega^{n+2} \times \Sigma^{*m} \to \omega$$

entonces las ecuaciones:

(a) $R(0, \vec{x}, \vec{\alpha}) = f(\vec{x}, \vec{\alpha})$
(b) $R(t+1, \vec{x}, \vec{\alpha}) = g(R(t, \vec{x}, \vec{\alpha}), t, \vec{x}, \vec{\alpha})$

determinan una unica funcion $R : \omega^{n+1} \times \Sigma^{*m} \to \omega$ que las cumple. Notese que para el caso

$$n = m = 2$$
$$f = \lambda x_1 x_2 \alpha_1 \alpha_2 [x_1 + |\alpha_1|^{x_2}]$$
$$g = \lambda x t x_1 x_2 \alpha_1 \alpha_2 [t + x_1 + x_2 + |\alpha_1| + |\alpha_2| + x]$$

las ecuaciones (a) y (b) se transforman en las ecuaciones (r) y (s).

Conjuntos rectangulares. El primer caso de recursion primitiva que definiremos a continuacion engloba todos los ejemplos vistos recien dentro de un marco general. Para enunciarlo necesitaremos una definicion muy importante en la materia. Sea Σ un alfabeto finito. Un conjunto Σ-mixto S es llamado *rectangular* si es de la forma

$$S_1 \times ... \times S_n \times L_1 \times ... \times L_m$$

con cada $S_i \subseteq \omega$ y cada $L_i \subseteq \Sigma^*$. Notar que todo subconjunto de ω es rectangular (es el caso $n = 1$ y $m = 0$). Analogamente, todo subconjunto de Σ^* es rectangular (es el caso $n = 0$ y $m = 1$). Tambien $\{\Diamond\}$ es rectangular (es el caso $n = m = 0$). Otros ejemplos:

- $\mathbf{N} \times \{1, 2\} \times \{@@, \varepsilon\}$ es rectangular (aqui $n = 2$ y $m = 1$)
- $\{!!!, !!\} \times \{@@, \varepsilon\}$ es rectangular (aqui $n = 0$ y $m = 2$)

Tambien notese que $\emptyset = \emptyset \times \emptyset$ por lo cual $\emptyset$ es un conjunto rectangular.

El concepto de conjunto rectangular es muy importante en nuestro enfoque. Aunque en general no habra restricciones acerca del dominio de las funciones y predicados, nuestra filosofia sera tratar en lo posible que los dominios de las funciones que utilicemos para hacer nuestro analisis de recursividad de los distintos paradigmas, sean rectangulares.

Aunque en principio puede parecer que todos los conjuntos son rectangulares, el siguiente lema mostrara cuan ingenua es esta vision. Lo aceptaremos sin demostracion aunque es facil de probar.

Lema 3. *Sea $S \subseteq \omega \times \Sigma^*$. Entonces S es rectangular si y solo si se cumple la siguiente propiedad:*

(R) *Si $(x, \alpha), (y, \beta) \in S$, entonces $(x, \beta) \in S$*

Ejercicio 7: Supongamos $\Sigma = \{\#, \blacktriangle, \%\}$. Use el lema anterior para probar que

$$\{(0, \#\#), (1, \%\%\%)\} \text{ y } \{(x, \alpha) \in \omega \times \Sigma^* : |\alpha| = x\}$$

no son rectangulares

Recursion primitiva sobre variable numerica con valores numericos. Ahora si daremos el primer caso del constructor de recursion primitiva. Supongamos tenemos dadas funciones

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$
$$g : \omega \times \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$

con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ conjuntos no vacios. Usando el razonamiento inductivo usado en los ejemplos anteriores, se puede probar que hay una unica funcion

$$R : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$

la cual cumple las ecuaciones

- $R(0, \vec{x}, \vec{\alpha}) = f(\vec{x}, \vec{\alpha})$
- $R(t+1, \vec{x}, \vec{\alpha}) = g(R(t, \vec{x}, \vec{\alpha}), t, \vec{x}, \vec{\alpha})$

LLamaremos $R(f, g)$ a esta unica funcion que cumple las ecuaciones anteriores. Resumiendo este hecho, diremos que las ecuaciones

(1) $R(f, g)(0, \vec{x}, \vec{\alpha}) = f(\vec{x}, \vec{\alpha})$
(2) $R(f, g)(t+1, \vec{x}, \vec{\alpha}) = g(R(f, g)(t, \vec{x}, \vec{\alpha}), t, \vec{x}, \vec{\alpha})$

definen recursivamente a la funcion $R(f, g)$. Tambien diremos que $R(f, g)$ es obtenida por *recursion primitiva* a partir de f y g.

NOTA IMPOTANTE: No confundirse y pensar que $R(f, g)$ es el resultado de aplicar una funcion R al par (f, g), de hecho hasta el momento no hemos definido ninguna funcion R cuyo dominio sea cierto conjunto de pares ordenados de funciones!

Ejercicio 8: Justifique con palabras (a lo mariposa sabia) que la funcion $R(f, g)$ esta bien definida, es decir que dada una $(1+n+m)$-upla $(t, \vec{x}, \vec{\alpha})$ perteneciente a $\omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m$, las ecuaciones de (1) y (2) determinan el valor $R(f, g)(t, \vec{x}, \vec{\alpha})$

Notese que cuando $n = m = 0$, se tiene que $D_f = \{\Diamond\}$ y (1) y (2) se transforman en

(1) $R(f, g)(0) = f(\Diamond)$
(2) $R(f, g)(t+1) = g(R(f, g)(t), t)$

Veamos algunos ejemplos

(E1) Tomemos $f = p_1^{1,0}$ y $g = Suc \circ p_1^{3,0}$. De la definicion de $R(f, g)$, obtenemos que su dominio es ω^2 y

$$R(f, g)(0, x_1) = p_1^{1,0}(x_1) = x_1$$
$$R(f, g)(t+1, x_1) = \left(Suc \circ p_1^{3,0}\right)(R(f, g)(t, x_1), t, x_1) = R(f, g)(t, x_1) + 1$$

Es facil notar que la unica funcion que cumple estas dos ecuaciones es $\lambda t x_1 [t + x_1]$, lo cual implica que $\lambda t x_1 [t + x_1] = R\left(p_1^{1,0}, Suc \circ p_1^{3,0}\right)$

(E2) Sean $f = C_0^{0,0}$ y $g = p_1^{2,0}$. De la definicion de $R(f, g)$, obtenemos que su dominio es ω y

$$R(f, g)(0) = C_0^{0,0}(\Diamond) = 0$$
$$R(f, g)(t+1) = p_1^{2,0}(R(f, g)(t), t) = R(f, g)(t)$$

Es facil notar que la unica funcion que cumple estas dos ecuaciones es $C_0^{1,0}$ lo cual implica que $C_0^{1,0} = R\left(C_0^{0,0}, p_1^{2,0}\right)$

Nota importante: En los dos ejemplos anteriores y en todos los casos que manejaremos en la Guia 5, en las aplicaciones del constructor de recursion primitiva (en sus cuatro formas) las funciones iniciales seran Σ-totales (es decir $S_1 = ... = S_n = \omega$ y $L_1 = ... = L_m = \Sigma^*$). Solo a partir de la Guia 6 veremos aplicaciones con funciones no Σ-totales

Recordemos que por definicion teniamos que $0^0 = 1$. Esto nos dice que $D_{\lambda xy[x^y]} = \omega \times \omega$.

Ejercicio 9: Encuentre f y g tales que:
(a) $R(f,g) = C_k^{n,m}$ (con $n, m, k \in \omega$ y $n \geq 1$)
(b) $R(f,g) = \lambda t x_1[t.x_1]$
(c) $R(f,g) = \lambda t x_1 \alpha_1 \alpha_2[t.x_1]$
(d) $R(f,g) = \lambda t x_1[(x_1)^t]$
(e) $R(f,g) = \lambda t[t!]$

Ejercicio 10: Explique la forma en la que aplicando los constructores de composicion y recursion primitiva a las funciones del conjunto inicial se puede obtener la funcion $\lambda x_1 x_2 \alpha_1[x_1!]$

Ejercicio 11: V o F o I, justifique.
(a) $\lambda t x_1 [t + x_1] = R\left(p_1^{1,0}, Suc \circ p_1^{2,0}\right)$
(b) $R(\lambda xy[0], p_2^{4,0}) = p_1^{3,0}$
(c) Si $f : \omega^2 \to \omega$ y $g : \omega^4 \to \omega$, entonces para cada $(x,y) \in \omega^2$, se tiene que $R(f,g)(2,x,y) = g \circ (g \circ [f \circ [p_2^{3,0}, p_3^{3,0}], p_1^{3,0}, p_2^{3,0}, p_3^{3,0}])(0,x,y)$

Como era de esperar, este caso del constructor de recursion primitiva preserva la computabilidad efectiva

Lema 4. *Sean*

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$
$$g : \omega \times \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$

con $S_1, ..., S_n \subseteq \omega$ *y* $L_1, ..., L_m \subseteq \Sigma^*$ *conjuntos no vacios. Si* f *y* g *son* Σ*-efectivamente computables, entonces* $R(f,g)$ *lo es.*

Proof. Sean $\mathbb{P}_f$ y $\mathbb{P}_g$ procedimientos efectivos que computan a f y g, respectivamente. Es facil construir entonces un procedimiento efectivo que compute a $R(f,g)$. ■

Ejercicio 12: Construya un procedimiento efectivo que compute a $R(f,g)$ en terminos de los procedimientos $\mathbb{P}_f$ y $\mathbb{P}_g$.

Recursion primitiva sobre variable numerica con valores alfabeticos. Ahora haremos el caso en el que la funcion definida recursivamente tiene imagen contenida en Σ^*. Es claro que entonces f y g tambien deberan tener imagen contenida en Σ^*. El unico detalle a tener en cuenta en la definicion de este caso es que si solo hicieramos estos cambios y pusieramos las mismas ecuaciones la funcion g no resultaria Σ-mixta en general (por que?). Para que la g de la recursion siga siendo Σ-mixta deberemos modificar levemente su dominio en relacion al caso ya hecho

Supongamos Σ es un alfabeto finito. Sean

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$
$$g : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \Sigma^*$$

con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ conjuntos no vacios. Definamos

$$R(f, g) : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$

de la siguiente manera

(1) $R(f, g)(0, \vec{x}, \vec{\alpha}) = f(\vec{x}, \vec{\alpha})$
(2) $R(f, g)(t + 1, \vec{x}, \vec{\alpha}) = g(t, \vec{x}, \vec{\alpha}, R(f, g)(t, \vec{x}, \vec{\alpha}))$

Diremos que $R(f, g)$ es obtenida por *recursion primitiva* a partir de f y g. Notese que cuando $m = n = 0$, se tiene que $D_f = \{\Diamond\}$ y (1) y (2) se transforman en

(1) $R(f, g)(0) = f(\Diamond)$
(2) $R(f, g)(t + 1) = g(t, R(f, g)(t))$

Veamos algunos ejemplos

(E1) Tomemos $f = C_\varepsilon^{0,1}$ y $g = \lambda\alpha\beta\,[\alpha\beta] \circ \left[p_3^{1,2}, p_2^{1,2}\right]$. De la definicion de $R(f, g)$, obtenemos que

$$R(f, g)(0, \alpha_1) = C_\varepsilon^{0,1}(\alpha_1) = \varepsilon$$
$$R(f, g)(t + 1, \alpha_1) = \lambda\alpha\beta\,[\alpha\beta] \circ \left[p_3^{1,2}, p_2^{1,2}\right](t, \alpha_1, R(f, g)(t, \alpha_1)) = R(f, g)(t, \alpha_1)\alpha_1$$

Es facil notar que la unica funcion que cumple estas dos ecuaciones es $\lambda t\alpha_1\,[{\alpha_1}^t]$, lo cual implica que $\lambda t\alpha_1\,[{\alpha_1}^t] = R\left(C_\varepsilon^{0,1}, \lambda\alpha\beta\,[\alpha\beta] \circ \left[p_3^{1,2}, p_2^{1,2}\right]\right)$

(E2) Sean $f = C_\varepsilon^{0,0}$ y $g = p_2^{1,1}$. De la definicion de $R(f, g)$, obtenemos que

$$R(f, g)(0) = C_\varepsilon^{0,0}(\Diamond) = \varepsilon$$
$$R(f, g)(t + 1) = p_2^{1,1}(t, R(f, g)(t)) = R(f, g)(t)$$

Es facil notar que la unica funcion que cumple estas dos ecuaciones es $C_\varepsilon^{1,0}$ lo cual implica que $C_\varepsilon^{1,0} = R\left(C_\varepsilon^{0,0}, p_2^{1,1}\right)$

Ejercicio 13: Sea $\Sigma = \{\%, @, ?\}$. Encuentre f y g tales que $\lambda t x_1[\%@\%\%\%?^t] = R(f, g)$

Ejercicio 14: V o F o I, justifique.

(a) $C_\varepsilon^{2,,2} = R\left(C_\varepsilon^{1,2}, C_\varepsilon^{2,3}\right)$
(b) $R\left(C_\varepsilon^{1,1}, C_\varepsilon^{1,1}\right) = C_\varepsilon^{1,1}$
(c) Si f, g son funciones Σ-mixtas tales que $R(f, g)$ esta definida y es de tipo $(1 + n, m, *)$, entonces f es de tipo $(n, m, *)$ y g es de tipo $(n, m + 1, *)$

La prueba del siguiente lema es completamente analoga a la del lema anterior que fue dejada como ejercicio.

Lema 5. *Sean*

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$
$$g : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \Sigma^*$$

con $S_1, ..., S_n \subseteq \omega$ *y* $L_1, ..., L_m \subseteq \Sigma^*$ *conjuntos no vacios. Si* f *y* g *son* Σ*-efectivamente computables, entonces* $R(f, g)$ *lo es.*

Recursion primitiva sobre variable alfabetica con valores numericos. Ya vimos dos casos de recursion donde el parametro o variable que comanda la recursion es numerico. Daremos a continuacion un ejemplo de recursion en el cual el parametro principal es alfabetico. Sea $\Sigma = \{\%, @, ?\}$ y consideremos las siguientes ecuaciones:

(1) $R(\varepsilon) = 15$
(2) $R(\alpha\%) = R(\alpha) + 1$
(3) $R(\alpha@) = R(\alpha).5$
(4) $R(\alpha?) = R(\alpha)^{20}$

Notese que las ecuaciones anteriores determinan una funcion $R : \Sigma^* \to \omega$. Esto es ya que R en ε debe valer 15 y sabiendo esto las ecuaciones (2), (3) y (4) (con $\alpha = \varepsilon$) nos dicen que

$$R(\%) = 16$$
$$R(@) = 75$$
$$R(?) = 15^{20}$$

por lo cual podemos aplicarlas nuevamente a dichas ecuaciones (con $\alpha \in \{\%, @, ?\}$) para calcular R en todas las palabras de longitud 2; y asi sucesivamente.

Daremos otro ejemplo un poco mas complicado para seguir aproximandonos al caso general. Nuevamente supongamos que $\Sigma = \{\%, @, ?\}$ y supongamos tenemos una funcion

$$f : \omega \times \Sigma^* \to \omega$$

y tres funciones

$$\mathcal{G}_\% : \omega \times \omega \times \Sigma^* \times \Sigma^* \to \omega$$
$$\mathcal{G}_@ : \omega \times \omega \times \Sigma^* \times \Sigma^* \to \omega$$
$$\mathcal{G}_? : \omega \times \omega \times \Sigma^* \times \Sigma^* \to \omega$$

Entonces hay una unica funcion $R : \omega \times \Sigma^* \times \Sigma^* \to \omega$ la cual cumple las siguientes ecuaciones

(1) $R(x_1, \alpha_1, \varepsilon) = f(x_1, \alpha_1)$
(2) $R(x_1, \alpha_1, \alpha\%) = \mathcal{G}_\%(R(x_1, \alpha_1, \alpha), x_1, \alpha_1, \alpha)$
(3) $R(x_1, \alpha_1, \alpha@) = \mathcal{G}_@(R(x_1, \alpha_1, \alpha), x_1, \alpha_1, \alpha)$
(4) $R(x_1, \alpha_1, \alpha?) = \mathcal{G}_?(R(x_1, \alpha_1, \alpha), x_1, \alpha_1, \alpha)$

Ejercicio 15: (a) Justifique que las ecuaciones anteriores determinan a la funcion R
(b) Por que el parametro α de la recursion es la ultima coordenada de R?

El ejemplo anterior nos muestra que para hacer recursion sobre parametro alfabetico nos hace falta "una funcion g por cada simbolo de Σ". Esto motiva la siguiente definicion. Dado un alfabeto Σ, una *familia Σ-indexada de funciones* sera una funcion $\mathcal{G}$ tal que $D_{\mathcal{G}} = \Sigma$ y para cada $a \in D_{\mathcal{G}}$ se tiene que $\mathcal{G}(a)$ es una funcion. Algunos ejemplos:

(E1) Sea $\mathcal{G}$ dada por

$$\begin{array}{rcl} \mathcal{G} : \{\square, \%, \blacktriangle\} & \to & \{Suc, Pred\} \\ \square & \to & Suc \\ \% & \to & Suc \\ \blacktriangle & \to & Pred \end{array}$$

Claramente $\mathcal{G}$ es una familia $\{\square, \%, \blacktriangle\}$-indexada de funciones. Notar que

$$\mathcal{G} = \{(\square, Suc), (\%, Suc), (\blacktriangle, Pred)\}$$

Se tiene tambien por ejemplo que $\mathcal{G}(\%) = Suc$ por lo cual tambien es cierto que $\mathcal{G}(\%)(22) = 23$, etc.

(E2) Si Σ es un alfabeto no vacio, la funcion

$$\begin{array}{rcl} \mathcal{G} : \Sigma & \to & \{f : f \text{ es una funcion de } \Sigma^* \text{ en } \Sigma^*\} \\ a & \to & d_a \end{array}$$

es una familia Σ-indexada de funciones. Notar que

$$\mathcal{G} = \{(a, d_a) : a \in \Sigma\}$$

(E3) Sea $\Sigma = \{\square, \%, \blacktriangle\}$. Entonces $\{(\square, Suc), (\%, p_3^{2,4}), (\blacktriangle, \emptyset)\}$ es una familia $\{\square, \%, \blacktriangle\}$-indexada de funciones.

NOTACION: Si $\mathcal{G}$ es una familia Σ-indexada de funciones, entonces para $a \in \Sigma$, escribiremos $\mathcal{G}_a$ en lugar de $\mathcal{G}(a)$.

Ahora si podemos dar la definicion matematica precisa del primero de los dos casos de recursion primitiva sobre parametro alfabetico. Sea

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$

con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ conjuntos no vacios y sea $\mathcal{G}$ una familia Σ-indexada de funciones tal que

$$\mathcal{G}_a : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \omega$$

para cada $a \in \Sigma$. Definamos

$$R(f, \mathcal{G}) : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \omega$$

de la siguiente manera

(1) $R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \varepsilon) = f(\vec{x}, \vec{\alpha})$
(2) $R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha a) = \mathcal{G}_a(R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha), \vec{x}, \vec{\alpha}, \alpha)$

Diremos que $R(f, \mathcal{G})$ es obtenida por *recursion primitiva* a partir de f y $\mathcal{G}$.

Notese que cuando $n = m = 0$, se tiene que $D_f = \{\Diamond\}$ y (1) y (2) se transforman en

(1) $R(f, \mathcal{G})(\varepsilon) = f(\Diamond)$
(2) $R(f, \mathcal{G})(\alpha a) = \mathcal{G}_a(R(f, \mathcal{G})(\alpha), \alpha)$

Ejercicio 16: Sea $\Sigma = \{\%, @, ?\}$. Encuentre f y $\mathcal{G}$ tales que
(a) $\lambda\alpha[|\alpha|] = R(f, \mathcal{G})$
(b) $\lambda\alpha_1\alpha[|\alpha_1| + |\alpha|_{@}] = R(f, \mathcal{G})$

Ejercicio 17: V o F o I, justifique.
(a) Sea $\Sigma = \{@, \&\}$. Se tiene que $\lambda\alpha\,[|\alpha|] = R\left(C_0^{0,0}, \{Suc \circ p_1^{1,1}, Suc \circ p_1^{1,1}\}\right)$
(b) $R\left(p_1^{2,0}, \{(@, p_1^{3,1}), (\&, p_2^{3,1})\}\right) = p_1^{2,1}$

Tenemos que

Lema 6. *Sea*

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$$

con $S_1, ..., S_n \subseteq \omega$ *y* $L_1, ..., L_m \subseteq \Sigma^*$ *conjuntos no vacios y sea* $\mathcal{G}$ *una familia* Σ*-indexada de funciones tal que*

$$\mathcal{G}_a : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \omega$$

para cada $a \in \Sigma$. *Si* f *y cada* $\mathcal{G}_a$ *son* Σ*-efectivamente computables, entonces* $R(f, \mathcal{G})$ *lo es.*

Ejercicio 18: Haga la prueba del lema anterior para el caso $\Sigma = \{@, \blacktriangle\}$

Recursion primitiva sobre variable alfabetica con valores alfabeticos. Supongamos Σ es un alfabeto finito. Sea

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$

con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ conjuntos no vacios y sea $\mathcal{G}$ una familia Σ-indexada de funciones tal que

$$\mathcal{G}_a : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^*$$

para cada $a \in \Sigma$. Definamos

$$R(f, \mathcal{G}) : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \Sigma^*$$

de la siguiente manera

(1) $R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \varepsilon) = f(\vec{x}, \vec{\alpha})$
(2) $R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha a) = \mathcal{G}_a(\vec{x}, \vec{\alpha}, \alpha, R(f, \mathcal{G})(\vec{x}, \vec{\alpha}, \alpha))$.

Diremos que $R(f, \mathcal{G})$ es obtenida por *recursion primitiva* a partir de f y $\mathcal{G}$. Notese que cuando $n = m = 0$, se tiene que $D_f = \{\Diamond\}$ y (1) y (2) se transforman en

(1) $R(f, \mathcal{G})(\varepsilon) = f(\Diamond)$
(2) $R(f, \mathcal{G})(\alpha a) = \mathcal{G}_a(\alpha, R(f, \mathcal{G})(\alpha))$

Ejercicio 19: Encuentre f y $\mathcal{G}$ tales que $\lambda\alpha_1\alpha[\alpha_1\alpha] = R(f, \mathcal{G})$

Ejercicio 20: Sea $\Sigma = \{\triangle, \blacktriangle\}$. Diga que funcion conocida es $R(C_\varepsilon^{0,1}, \mathcal{G})$, donde $\mathcal{G}$ es dada por $\mathcal{G}_\triangle = d_\triangle \circ p_3^{0,3}$ y $\mathcal{G}_\blacktriangle = d_\blacktriangle \circ p_3^{0,3}$

Sea Σ un alfabeto finito. Dada $\gamma \in \Sigma^*$, definamos

$$\gamma^R = \begin{cases} [\gamma]_{|\gamma|}[\gamma]_{|\gamma|-1}...[\gamma]_1 & \text{si } |\gamma| \geq 1 \\ \varepsilon & \text{caso contrario} \end{cases}$$

La palabra γ^R es llamada la *resiproca* de γ. Para $a \in \Sigma$, definamos la funcion

$$\begin{array}{rcl} I_a : \Sigma^* & \to & \Sigma^* \\ \alpha & \to & a\alpha \end{array}$$

Recordemos que $\alpha^0 = \varepsilon$, para cada $\alpha \in \Sigma^*$, por lo cual tenemos que $D_{\lambda x \alpha[\alpha^x]} = \omega \times \Sigma^*$.

Ejercicio 21: Sea $\Sigma = \{\triangle, \blacktriangle\}$. Explique la forma en la que aplicando los constructores de composicion y recursion primitiva a las funciones del conjunto inicial se pueden obtener las siguientes funciones
(a) I_a
(b) $\lambda\alpha[\alpha^R]$
(c) $\lambda t\alpha[\alpha^t]$

Ejercicio 22: V o F o I, justifique.
(a) Sea $\Sigma = \{\triangle, \blacktriangle\}$. Entonces $R(p_1^{0,1}, \{(\triangle, p_3^{0,3}), (\blacktriangle, d_\blacktriangle \circ p_3^{0,3})\})(\triangle\blacktriangle, \triangle\blacktriangle) = \triangle\blacktriangle\blacktriangle\blacktriangle$
(b) $R(p_1^{0,1}, d_\alpha \circ p_3^{0,3}) = \lambda\alpha_1\alpha[\alpha_1\alpha]$

Tenemos que

Lema 7. *Sea*

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$

con $S_1, ..., S_n \subseteq \omega$ *y* $L_1, ..., L_m \subseteq \Sigma^*$ *conjuntos no vacios y sea* $\mathcal{G}$ *una familia* Σ*-indexada de funciones tal que*

$$\mathcal{G}_a : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^*$$

para cada $a \in \Sigma$. *Si* f *y cada* $\mathcal{G}_a$ *son* Σ*-efectivamente computables, entonces* $R(f, \mathcal{G})$ *lo es.*

Cosmetica. Como hemos visto muchas veces para poder aplicar mas naturalmente

algun lema, nos es util cambiar las variables que estan siendo usadas en la notacion lambda que describe alguna funcion. Por ejemplo si queremos ver que la funcion $\lambda xy\alpha[x + y]$ es de la forma $R(f, g)$, con f de tipo $(1, 1, \#)$ y g de tipo $(3, 1, \#)$, nos conviene escribir a la funcion $\lambda xy\alpha[x + y]$ en la forma $\lambda tx_1\alpha_1[t + x_1]$, donde se ve mejor cual es el parametro de la recursion primitiva y cual es el bloque fijo. Obviamente esto es posible ya que $\lambda xy\alpha[x + y] = \lambda tx_1\alpha_1[t + x_1]$. Esto da lugar a nuestra regla de cosmetica:

REGLA DE COSMETICA: Si ud tiene una expresion lambda $\lambda x_1...x_n\alpha_1...\alpha_m\,[E]$ que denota una funcion f, entonces puede reemplazar en dicha expresion cada ocurrencia de una de las variables de la lista $x_1...x_n\alpha_1...\alpha_m$ por una nueva variable (del mismo tipo, i.e. numerica o alfabetica) la cual no figure en la lista y el resultado sera una expresion lambda que tambien denota a f.

Por supuesto que dicha regla la puede aplicar varias veces para modificar su notacion lambda. Por ejemplo en el caso de $\lambda xy\alpha[x+y]$ aplicamos tres veces la regla y obtenemos $\lambda t x_1 \alpha_1 [t + x_1]$.

Funciones Σ-recursivas primitivas. Intuitivamente hablando ya sabemos que una funcion es Σ-recursiva primitiva si se puede obtener de las iniciales usando los constructores de composicion y recursion primitiva. Daremos ahora una definicion matematica de este concepto. Definamos los conjuntos $\mathrm{PR}_0^\Sigma \subseteq \mathrm{PR}_1^\Sigma \subseteq \mathrm{PR}_2^\Sigma \subseteq ... \subseteq \mathrm{PR}^\Sigma$ de la siguiente manera

$$\begin{aligned}
\mathrm{PR}_0^\Sigma &= \left\{Suc, Pred, C_0^{0,0}, C_\varepsilon^{0,0}\right\} \cup \{d_a : a \in \Sigma\} \cup \left\{p_j^{n,m} : 1 \le j \le n+m\right\} \\
\mathrm{PR}_{k+1}^\Sigma &= \mathrm{PR}_k^\Sigma \cup \left\{f \circ [f_1, ..., f_r] : f, f_1, ..., f_r \in \mathrm{PR}_k^\Sigma,\ r \ge 1\right\} \cup \\
&\qquad \left\{R(f, \mathcal{G}) : R(f, \mathcal{G}) \text{ esta definida y } \{f\} \cup \{\mathcal{G}_a : a \in \Sigma\} \subseteq \mathrm{PR}_k^\Sigma\right\} \cup \\
&\qquad \left\{R(f, g) : R(f, g) \text{ esta definida y } f, g \in \mathrm{PR}_k^\Sigma\right\} \\
\mathrm{PR}^\Sigma &= \textstyle\bigcup_{k \ge 0} \mathrm{PR}_k^\Sigma
\end{aligned}$$

Una funcion es llamada Σ-*recursiva primitiva* (Σ-*p.r.*) si pertenece a PR^Σ.

Proposición 8. *Si $f \in \mathrm{PR}^\Sigma$, entonces f es Σ-efectivamente computable.*

Ejercicio 23: Explique con palabras (a lo mariposa) por que es cierta la proposicion anterior

Algunas funciones Σ-recursivas primitivas. En los siguientes lemas se prueba bien

formalmente que varias funciones bien conocidas son Σ-primitivas recursivas. La mayoria de estas funciones ya fueron obtenidas usando los constructores de composicion y recursion primitiva, en los desarrollos anteriores o en los ejercicios.

Lema 9. *Sea Σ un alfabeto finito.*

(1) $\emptyset \in \mathrm{PR}^\Sigma$.
(2) $\lambda xy\,[x+y] \in \mathrm{PR}^\Sigma$.
(3) $\lambda xy\,[x.y] \in \mathrm{PR}^\Sigma$.
(4) $\lambda x\,[x!] \in \mathrm{PR}^\Sigma$.

Proof. (1) Notese que $\emptyset = Pred \circ C_0^{0,0} \in \mathrm{PR}_1^\Sigma$

(2) Notar que

$$\begin{aligned}
\lambda xy\,[x+y]\,(0, x_1) &= x_1 = p_1^{1,0}(x_1) \\
\lambda xy\,[x+y]\,(t+1, x_1) &= \lambda xy\,[x+y]\,(t, x_1) + 1 \\
&= \left(Suc \circ p_1^{3,0}\right)(\lambda xy\,[x+y]\,(t, x_1), t, x_1)
\end{aligned}$$

lo cual implica que $\lambda xy\,[x+y] = R\left(p_1^{1,0}, Suc \circ p_1^{3,0}\right) \in \mathrm{PR}_2^\Sigma$.

(3) Primero note que

$$\begin{aligned}
C_0^{1,0}(0) &= C_0^{0,0}(\Diamond) \\
C_0^{1,0}(t+1) &= C_0^{1,0}(t)
\end{aligned}$$

lo cual implica que $C_0^{1,0} = R\left(C_0^{0,0}, p_1^{2,0}\right) \in \mathrm{PR}_1^{\Sigma}$. Tambien note que

$$\lambda tx\,[t.x] = R\left(C_0^{1,0}, \lambda xy\,[x+y] \circ \left[p_1^{3,0}, p_3^{3,0}\right]\right),$$

lo cual por (2) implica que $\lambda tx\,[t.x] \in \mathrm{PR}_4^{\Sigma}$.

(4) Note que

$$\lambda x\,[x!]\,(0) = 1 = C_1^{0,0}(\Diamond)$$
$$\lambda x\,[x!]\,(t+1) = \lambda x\,[x!]\,(t).(t+1),$$

lo cual implica que

$$\lambda x\,[x!] = R\left(C_1^{0,0}, \lambda xy\,[x.y] \circ \left[p_1^{2,0}, Suc \circ p_2^{2,0}\right]\right).$$

Ya que $C_1^{0,0} = Suc \circ C_0^{0,0}$, tenemos que $C_1^{0,0} \in \mathrm{PR}_1^{\Sigma}$. Por (3), tenemos que

$$\lambda xy\,[x.y] \circ \left[p_1^{2,0}, Suc \circ p_2^{2,0}\right] \in \mathrm{PR}_5^{\Sigma},$$

obteniendo que $\lambda x\,[x!] \in \mathrm{PR}_6^{\Sigma}$. ■

Ahora consideraremos dos funciones las cuales son obtenidas naturalmente por recursion primitiva sobre variable alfabetica.

Lema 10. *Supongamos Σ es un alfabeto finito.*

(a) $\lambda\alpha\beta\,[\alpha\beta] \in \mathrm{PR}^{\Sigma}$
(b) $\lambda\alpha\,[|\alpha|] \in \mathrm{PR}^{\Sigma}$

Proof. (a) Ya que

$$\lambda\alpha\beta\,[\alpha\beta]\,(\alpha_1, \varepsilon) = \alpha_1 = p_1^{0,1}(\alpha_1)$$
$$\lambda\alpha\beta\,[\alpha\beta]\,(\alpha_1, \alpha a) = d_a(\lambda\alpha\beta\,[\alpha\beta]\,(\alpha_1, \alpha)),\ a \in \Sigma$$

tenemos que $\lambda\alpha\beta\,[\alpha\beta] = R\left(p_1^{0,1}, \mathcal{G}\right)$, donde $\mathcal{G}_a = d_a \circ p_3^{0,3}$, para cada $a \in \Sigma$.

(b) Ya que

$$\lambda\alpha\,[|\alpha|]\,(\varepsilon) = 0 = C_0^{0,0}(\Diamond)$$
$$\lambda\alpha\,[|\alpha|]\,(\alpha a) = \lambda\alpha\,[|\alpha|]\,(\alpha) + 1$$

tenemos que $\lambda\alpha\,[|\alpha|] = R\left(C_0^{0,0}, \mathcal{G}\right)$, donde $\mathcal{G}_a = Suc \circ p_1^{1,1}$, para cada $a \in \Sigma$. ■

Lema 11. *Sea Σ un alfabeto finito. Entonces $C_k^{n,m}, C_\alpha^{n,m} \in \mathrm{PR}^{\Sigma}$, para cada $n, m, k \geq 0$ y $\alpha \in \Sigma^*$.*

Proof. (a) Note que $C_{k+1}^{0,0} = Suc \circ C_k^{0,0}$, lo cual implica $C_k^{0,0} \in \mathrm{PR}_k^{\Sigma}$, para $k \geq 0$. Tambien note que $C_{\alpha a}^{0,0} = d_a \circ C_\alpha^{0,0}$, lo cual dice que $C_\alpha^{0,0} \in \mathrm{PR}^{\Sigma}$, para $\alpha \in \Sigma^*$. Para ver que $C_k^{0,1} \in \mathrm{PR}^{\Sigma}$ notar que

$$C_k^{0,1}(\varepsilon) = k = C_k^{0,0}(\Diamond)$$
$$C_k^{0,1}(\alpha a) = C_k^{0,1}(\alpha) = p_1^{1,1}\left(C_k^{0,1}(\alpha), \alpha\right)$$

lo cual implica que $C_k^{0,1} = R\left(C_k^{0,0}, \mathcal{G}\right)$, con $\mathcal{G}_a = p_1^{1,1}$, $a \in \Sigma$. En forma similar podemos ver que $C_k^{1,0}, C_\alpha^{1,0}, C_\alpha^{0,1} \in \mathrm{PR}^\Sigma$. Supongamos ahora que $m > 0$. Entonces

$$C_k^{n,m} = C_k^{0,1} \circ p_{n+1}^{n,m}$$
$$C_\alpha^{n,m} = C_\alpha^{0,1} \circ p_{n+1}^{n,m}$$

de lo cual obtenemos que $C_k^{n,m}, C_\alpha^{n,m} \in \mathrm{PR}^\Sigma$. El caso $n > 0$ es similar. ■

Lema 12. *Sea Σ un alfabeto finito.*

(a) $\lambda xy\,[x^y] \in \mathrm{PR}^\Sigma$.
(b) $\lambda t\alpha\,[\alpha^t] \in \mathrm{PR}^\Sigma$.

Proof. (a) Note que

$$\lambda tx\left[x^t\right] = R\left(C_1^{1,0}, \lambda xy\,[x.y] \circ \left[p_1^{3,0}, p_3^{3,0}\right]\right) \in \mathrm{PR}^\Sigma.$$

O sea que $\lambda xy\,[x^y] = \lambda tx\,[x^t] \circ \left[p_2^{2,0}, p_1^{2,0}\right] \in \mathrm{PR}^\Sigma$.

(b) Note que

$$\lambda t\alpha\left[\alpha^t\right] = R\left(C_\varepsilon^{0,1}, \lambda\alpha\beta\,[\alpha\beta] \circ \left[p_3^{1,2}, p_2^{1,2}\right]\right) \in \mathrm{PR}^\Sigma.$$

■

Ejercicio 24: Sea $\Sigma = \{@, \%, \$\}$ y sea $\leq$ el orden total sobre Σ dado por $@ < \% < \$$. Pruebe que $s^{\leq}$, $\#^{\leq}$ y $*^{\leq}$ pertenecen a PR^Σ (esta probado en el apunte).

Ejercicio 25: Sea $\Sigma = \{\$, ?, @, \forall, \rightarrow, (\}$ y sea $S = \{\$, ?\}^*$. Pruebe que

$$\begin{array}{rcl} \chi_S^{\Sigma^*} : \Sigma^* & \rightarrow & \omega \\ \alpha & \rightarrow & \begin{cases} 1 & \text{si } \alpha \in S \\ 0 & \text{si } \alpha \notin S \end{cases} \end{array}$$

es Σ-p.r.

Recordemos que llamabamos *numerales* a los siguientes simbolos

0 1 2 3 4 5 6 7 8 9

Tambien recordemos que Num denotaba el conjunto de los numerales. Sea $Sig : Num^* \to Num^*$ definida de la siguiente manera

$$\begin{aligned} Sig(\varepsilon) &= 1 \\ Sig(\alpha 0) &= \alpha 1 \\ Sig(\alpha 1) &= \alpha 2 \\ Sig(\alpha 2) &= \alpha 3 \\ Sig(\alpha 3) &= \alpha 4 \\ Sig(\alpha 4) &= \alpha 5 \\ Sig(\alpha 5) &= \alpha 6 \\ Sig(\alpha 6) &= \alpha 7 \\ Sig(\alpha 7) &= \alpha 8 \\ Sig(\alpha 8) &= \alpha 9 \\ Sig(\alpha 9) &= Sig(\alpha)0 \end{aligned}$$

Definamos $Dec : \omega \to Num^*$ de la siguiente manera

$$\begin{aligned} Dec(0) &= \varepsilon \\ Dec(n+1) &= Sig(Dec(n)) \end{aligned}$$

Notese que para $n \in \mathbf{N}$, la palabra $Dec(n)$ es la notacion usual decimal de n. Notese que $Sig = R(C_1^{0,0}, \mathcal{G})$, donde $\mathcal{G}$ es la familia Num-indexada de funciones dada por:

$$\begin{aligned} \mathcal{G}_0 &= d_1 \circ p_2^{0,2} \\ \mathcal{G}_1 &= d_2 \circ p_2^{0,2} \\ \mathcal{G}_2 &= d_3 \circ p_2^{0,2} \\ \mathcal{G}_3 &= d_4 \circ p_2^{0,2} \\ \mathcal{G}_4 &= d_5 \circ p_2^{0,2} \\ \mathcal{G}_5 &= d_6 \circ p_2^{0,2} \\ \mathcal{G}_6 &= d_7 \circ p_2^{0,2} \\ \mathcal{G}_7 &= d_8 \circ p_2^{0,2} \\ \mathcal{G}_8 &= d_9 \circ p_2^{0,2} \\ \mathcal{G}_9 &= d_0 \circ p_1^{0,2} \end{aligned}$$

por lo cual Sig es Num-p.r.. Tambien tenemos que $Dec = R(C_\varepsilon^{0,0}, Sig \circ p_2^{1,1})$ por lo cual Dec es Num-p.r.. En la batalla Godel vence a Neumman (Guia 8) utilizaremos el siguiente resultado.

Lema 13. *Sea Γ un alfabeto que contiene a Num. Entonces Dec es Γ-p.r..*

Proof. Sea $\widetilde{Sig} : \Gamma^* \to \Gamma^*$ definida de la siguiente manera:

$$\begin{aligned}
\widetilde{Sig}(\varepsilon) &= 1\\
\widetilde{Sig}(\alpha 0) &= \alpha 1\\
\widetilde{Sig}(\alpha 1) &= \alpha 2\\
\widetilde{Sig}(\alpha 2) &= \alpha 3\\
\widetilde{Sig}(\alpha 3) &= \alpha 4\\
\widetilde{Sig}(\alpha 4) &= \alpha 5\\
\widetilde{Sig}(\alpha 5) &= \alpha 6\\
\widetilde{Sig}(\alpha 6) &= \alpha 7\\
\widetilde{Sig}(\alpha 7) &= \alpha 8\\
\widetilde{Sig}(\alpha 8) &= \alpha 9\\
\widetilde{Sig}(\alpha 9) &= Sig(\alpha)0\\
\widetilde{Sig}(\alpha a) &= \varepsilon, \text{ palabra } a \in \Gamma - Num
\end{aligned}$$

Notese que $\widetilde{Sig} = R(C_1^{0,0}, \widetilde{\mathcal{G}})$, donde $\widetilde{\mathcal{G}}$ es la familia Γ-indexada de funciones dada por:

$$\begin{aligned}
\widetilde{\mathcal{G}}_0 &= d_1 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_1 &= d_2 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_2 &= d_3 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_3 &= d_4 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_4 &= d_5 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_5 &= d_6 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_6 &= d_7 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_7 &= d_8 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_8 &= d_9 \circ p_2^{0,2}\\
\widetilde{\mathcal{G}}_9 &= d_0 \circ p_1^{0,2}\\
\widetilde{\mathcal{G}}_a &= C_\varepsilon^{0,2}, \text{ para } a \in \Gamma - Num
\end{aligned}$$

por lo cual $\widetilde{Sig}$ es Γ-p.r. (ojo que aqui las funciones $p_2^{0,2}$, $C_\varepsilon^{0,2}$, $C_1^{0,0}$ y $d_0, d_1, d_2, ..., d_9$ son relativas al alfabeto Γ). Pero notese que

$$\begin{aligned}
Dec(0) &= \varepsilon\\
Dec(n+1) &= \widetilde{Sig}(Dec(n)), \text{ para cada } n \in \omega
\end{aligned}$$

lo cual nos dice que $Dec = R(C_\varepsilon^{0,0}, \widetilde{Sig} \circ p_2^{1,1})$ por lo cual Dec es Γ-p.r. (de nuevo, aqui las funciones $C_\varepsilon^{0,0}$ y $p_2^{1,1}$ son relativas al alfabeto Γ). ■ Dados $x, y \in \omega$, definamos

$$x \dot{-} y = \max(x - y, 0).$$

Lema 14. *Se tiene*

(a) $\lambda xy\left[x\dot{-}y\right] \in \mathrm{PR}^{\Sigma}$
(b) $\lambda xy\left[\max(x,y)\right] \in \mathrm{PR}^{\Sigma}$
(c) $\lambda xy\left[x=y\right] \in \mathrm{PR}^{\Sigma}$
(d) $\lambda xy\left[x\leq y\right] \in \mathrm{PR}^{\Sigma}$
(e) $\lambda\alpha\beta\left[\alpha=\beta\right] \in \mathrm{PR}^{\Sigma}$

Proof. (a) Primero notar que $\lambda x\left[x\dot{-}1\right] = R\left(C_0^{0,0}, p_2^{2,0}\right) \in \mathrm{PR}^{\Sigma}$. Tambien note que

$$\lambda tx\left[x\dot{-}t\right] = R\left(p_1^{1,0}, \lambda x\left[x\dot{-}1\right]\circ p_1^{3,0}\right) \in \mathrm{PR}^{\Sigma}.$$

O sea que $\lambda xy\left[x\dot{-}y\right] = \lambda tx\left[x\dot{-}t\right]\circ\left[p_2^{2,0}, p_1^{2,0}\right] \in \mathrm{PR}^{\Sigma}$.

(b) Note que $\lambda xy\left[\max(x,y)\right] = \lambda xy\left[x+(y\dot{-}x)\right]$.
(c) Note que $\lambda xy\left[x=y\right] = \lambda xy\left[1\dot{-}((x\dot{-}y)+(y\dot{-}x))\right]$.
(d) Note que $\lambda xy\left[x\leq y\right] = \lambda xy\left[1\dot{-}(x\dot{-}y)\right]$.
(e) Sea $\leq$ un orden total sobre Σ. Ya que

$$\alpha=\beta \text{ sii } \#^{\leq}(\alpha) = \#^{\leq}(\beta)$$

tenemos que

$$\lambda\alpha\beta\left[\alpha=\beta\right] = \lambda xy\left[x=y\right]\circ\left[\#^{\leq}\circ p_1^{0,2}, \#^{\leq}\circ p_2^{0,2}\right]$$

lo cual nos dice que $\lambda\alpha\beta\left[\alpha=\beta\right]$ es Σ-p.r. ∎

Ejercicio 26: Complete las pruebas de (b), (c), (d) y (e) del lema anterior
Ejercicio 27: Sea Σ un alfabeto finito.

(a) $\lambda x[x$ es par$]$ es Σ-p.r..
(b) $\lambda xyz\alpha\beta\gamma[x.y+\max(x,|\alpha|)^{|\beta|}]$ es Σ-p.r.
(c) $\lambda x\alpha\left[x=|\alpha|\right]$ es Σ-p.r..
(d) $\lambda xy\alpha\beta\left[\alpha^x=\beta\right]$ es Σ-p.r..

Operaciones logicas entre predicados. Dados predicados $P : S\subseteq\omega^n\times\Sigma^{*m}\to\omega$ y $Q : S\subseteq\omega^n\times\Sigma^{*m}\to\omega$, con el mismo dominio, definamos nuevos predicados $(P\vee Q)$, $(P\wedge Q)$ y $\neg P$ de la siguiente manera

$$\begin{array}{rcl}(P\vee Q):S & \to & \omega\\ (\vec{x},\vec{\alpha}) & \to & \begin{cases}1 & \text{si } P(\vec{x},\vec{\alpha})=1 \text{ o } Q(\vec{x},\vec{\alpha})=1\\ 0 & \text{caso contrario}\end{cases}\\ (P\wedge Q):S & \to & \omega\\ (\vec{x},\vec{\alpha}) & \to & \begin{cases}1 & \text{si } P(\vec{x},\vec{\alpha})=1 \text{ y } Q(\vec{x},\vec{\alpha})=1\\ 0 & \text{caso contrario}\end{cases}\\ \neg P:S & \to & \omega\\ (\vec{x},\vec{\alpha}) & \to & \begin{cases}1 & \text{si } P(\vec{x},\vec{\alpha})=0\\ 0 & \text{si } P(\vec{x},\vec{\alpha})=1\end{cases}\end{array}$$

Lema 15. *Si $P : D_P\subseteq\omega^n\times\Sigma^{*m}\to\omega$ y $Q : D_Q\subseteq\omega^n\times\Sigma^{*m}\to\omega$ son predicados Σ-p.r. tales que $D_P = D_Q$, entonces $(P\vee Q)$, $(P\wedge Q)$ y $\neg P$ son Σ-p.r.*

Proof. Note que

$$\begin{aligned}\neg P &= \lambda xy\left[x\dot{-}y\right]\circ[C_1^{n,m},P]\\ (P\wedge Q) &= \lambda xy\,[x.y]\circ[P,Q]\\ (P\vee Q) &= \neg(\neg P\wedge\neg Q)\end{aligned}$$

■

Ejercicio 28: V o F o I, justifique.

(a) Si P_1,P_2,P_3 son predicados Σ-p.r. y $D_{P_1}=D_{P_2}=D_{P_3}$, entonces el predicado $(P_1\vee P_2\wedge P_3)$ es Σ-p.r.
(b) Sea Σ un alfabeto finito. Entonces $\lambda x\alpha\beta\,[x=|\alpha|\wedge\alpha=\beta]=(\lambda x\alpha\,[x=|\alpha|]\wedge\lambda\alpha\beta\,[\alpha=\beta])$
(c) Sea Σ un alfabeto finito. Entonces $(\lambda x\,[x=1]\wedge\lambda\alpha\,[\alpha=\varepsilon])(2,\varepsilon)=0$
(d) Si $S,T\subseteq\omega$, entonces $\chi_{S\times T}^{\omega\times\omega}=(\chi_S^{\omega}\wedge\chi_T^{\omega})$

Conjuntos Σ-recursivos primitivos. Un conjunto Σ-mixto $S\subseteq\omega^n\times\Sigma^{*m}$ es llamado Σ-*recursivo primitivo* si su funcion caracteristica $\chi_S^{\omega^n\times\Sigma^{*m}}$ es Σ-p.r.. Notese que $\chi_S^{\omega^n\times\Sigma^{*m}}=\lambda\vec{x}\vec{\alpha}[(\vec{x},\vec{\alpha})\in S]$.

Ejercicio 29: Sea $\Sigma=\{@,!\}$. Pruebe que los siguientes conjuntos son Σ-p.r.

(a) ω
(b) Σ^*
(c) $\{(x,y)\in\omega^2 : x=y\}$
(d) $\{(x,\alpha)\in\omega\times\Sigma^* : x=|\alpha|\}$
(e) $\{x\in\omega : x \text{ es par}\}$
(f) $\{(x,y,\alpha,\beta,\gamma)\in\omega^2\times\Sigma^{*3} : x\leq|\gamma|\}$

Lema 16. *Si $S_1,S_2\subseteq\omega^n\times\Sigma^{*m}$ son Σ-p.r., entonces $S_1\cup S_2$, $S_1\cap S_2$ y S_1-S_2 lo son.*

Proof. Note que

$$\begin{aligned}\chi_{S_1\cup S_2}^{\omega^n\times\Sigma^{*m}} &= (\chi_{S_1}^{\omega^n\times\Sigma^{*m}}\vee\chi_{S_2}^{\omega^n\times\Sigma^{*m}})\\ \chi_{S_1\cap S_2}^{\omega^n\times\Sigma^{*m}} &= (\chi_{S_1}^{\omega^n\times\Sigma^{*m}}\wedge\chi_{S_2}^{\omega^n\times\Sigma^{*m}})\\ \chi_{S_1-S_2}^{\omega^n\times\Sigma^{*m}} &= \lambda xy\left[x\dot{-}y\right]\circ\left[\chi_{S_1}^{\omega^n\times\Sigma^{*m}},\chi_{S_2}^{\omega^n\times\Sigma^{*m}}\right]\end{aligned}$$

■

Ejercicio 30: Si $S\subseteq\omega^n\times\Sigma^{*m}$ es finito, entonces S es Σ-p.r. Haga el caso $n=m=1$. (Hint: haga el caso en que S tiene un solo elemento y luego aplique el lema anterior).

Ejercicio 31: Sea Σ un alfabeto finito.

(a) Pruebe que Σ es Σ-p.r.
(b) Pruebe que $\Sigma^*-\{\varepsilon\}$ es Σ-p.r.
(c) Pruebe que $\Sigma^*-(\{\varepsilon\}\cup\Sigma)$ es Σ-p.r.
(d) Pruebe que $\omega-\{0,1\}$ es Σ-p.r.
(e) Pruebe que $\{(x,y,\alpha,\beta,\gamma)\in\omega^2\times\Sigma^{*3} : \alpha\neq\varepsilon\vee x\leq|\gamma|\}$ es Σ-p.r.
(f) Pruebe que $\{(x,\alpha,\beta) : |\alpha|>6\}$ es Σ-p.r.

El siguiente lema caracteriza cuando un conjunto rectangular es Σ-p.r..

Lema 17. *Supongamos $S_1, ..., S_n \subseteq \omega$, $L_1, ..., L_m \subseteq \Sigma^*$ son conjuntos no vacios. Entonces $S_1 \times ... \times S_n \times L_1 \times ... \times L_m$ es Σ-p.r. sii $S_1, ..., S_n, L_1, ..., L_m$ son Σ-p.r.*

Ejercicio 32: Haga la prueba del lema anterior para el caso de $n = m = 1$ (en el apunte esta la prueba general)

Dada una funcion f y un conjunto $S \subseteq D_f$, usaremos $f|_S$ para denotar la *restriccion* de f al conjunto S, i.e. $f|_S = f \cap (S \times I_f)$. Notese que $f|_S$ es la funcion dada por

$$D_{f|_S} = S$$
$$f|_S(e) = f(e), \text{ para cada } e \in S$$

Notese que cualesquiera sea la funcion f tenemos que $f|_\emptyset = \emptyset$ y $f|_{D_f} = f$.

Lema 18. *Supongamos $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-p.r., donde $O \in \{\omega, \Sigma^*\}$. Si $S \subseteq D_f$ es Σ-p.r., entonces $f|_S$ es Σ-p.r..*

Proof. Supongamos $O = \Sigma^*$. Entonces

$$f|_S = \lambda x \alpha \left[\alpha^x\right] \circ \left[Suc \circ Pred \circ \chi_S^{\omega^n \times \Sigma^{*m}}, f\right]$$

lo cual nos dice que $f|_S$ es Σ-p.r.. El caso $O = \omega$ es similar usando $\lambda xy\left[x^y\right]$ en lugar de $\lambda x\alpha\left[\alpha^x\right]$. ■

Usando el lema anterior en combinacion con el Lema 15 podemos ver que muchos predicados usuales son Σ-p.r.. Por ejemplo sea

$$P = \lambda x \alpha \beta \gamma \left[x = |\gamma| \wedge \alpha = \gamma^{Pred(|\beta|)}\right].$$

Notese que

$$D_P = \omega \times \Sigma^* \times (\Sigma^* - \{\varepsilon\}) \times \Sigma^*$$

Ademas D_P es Σ-p.r. ya que

$$\chi_{D_P}^{\omega \times \Sigma^{*3}} = \neg \lambda \alpha \beta \left[\alpha = \beta\right] \circ \left[p_3^{1,3}, C_\varepsilon^{1,3}\right]$$

Tambien note que los predicados

$$P_1 = \lambda x \alpha \beta \gamma \left[x = |\gamma|\right]$$
$$P_2 = \lambda x \alpha \beta \gamma \left[\alpha = \gamma^{Pred(|\beta|)}\right]$$

son Σ-p.r. ya que pueden obtenerse componiendo funciones Σ-p.r.. Un error seria pensar que $P = (P_1 \wedge P_2)$ ya que P_1 y P_2 tienen dominios distintos por lo cual no esta definido $(P_1 \wedge P_2)$. Sin envargo tenemos que $P = (P_1|_{D_P} \wedge P_2)$, lo cual nos dice que P es Σ-p.r. ya que $P_1|_{D_P}$ es Σ-p.r. por el Lema 18 y por lo tanto podemos aplicar el Lema 15

Ejercicio 33: Sea $\Sigma = \{@, !\}$. Sea $P = \lambda xy\alpha\beta\gamma\left[Pred(Pred(|\beta|)) \neq 6 \wedge \alpha^x = \beta\right]$
(a) Encuentre por definicion de notacion lambda el dominio de P
(b) Pruebe que P es Σ-p.r.

Aceptaremos sin prueba el siguiente resultado (ver el apunte por una prueba)

Lema 19. *Sean $O \in \{\omega, \Sigma^*\}$ y $n, m \in \omega$. Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-p.r., entonces existe una funcion Σ-p.r. $\bar{f} : \omega^n \times \Sigma^{*m} \to O$, tal que $f = \bar{f}|_{D_f}$.*

Ahora podemos probar una proposicion muy importante.

Proposición 20. *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Se tiene que S es Σ-p.r. sii S es el dominio de alguna funcion Σ-p.r..*

Proof. ($\Rightarrow$) Note que $S = D_{Pred \circ \chi_S^{\omega^n \times \Sigma^{*m}}}$.

($\Leftarrow$) Probaremos por induccion en k que D_F es Σ-p.r., para cada $F \in \mathrm{PR}_k^\Sigma$. El caso $k = 0$ es facil. Supongamos el resultado vale para un k fijo y supongamos $F \in \mathrm{PR}_{k+1}^\Sigma$. Veremos entonces que D_F es Σ-p.r.. Hay varios casos. Consideremos primero el caso en que $F = R(f, g)$, donde

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$
$$g : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \to \Sigma^*,$$

con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ conjuntos no vacios y $f, g \in \mathrm{PR}_k^\Sigma$. Notese que por definicion de $R(f, g)$, tenemos que

$$D_F = \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m.$$

Por hipotesis inductiva tenemos que $D_f = S_1 \times ... \times S_n \times L_1 \times ... \times L_m$ es Σ-p.r., lo cual por el Lema 17 nos dice que los conjuntos $S_1, ..., S_n$, $L_1, ..., L_m$ son Σ-p.r.. Ya que ω es Σ-p.r., el Lema 17 nos dice que D_F es Σ-p.r..

Los otros casos de recursion primitiva son dejados al lector.

Supongamos ahora que $F = g \circ [g_1, ..., g_r]$ con $g, g_1, ..., g_r \in \mathrm{PR}_k^\Sigma$. Si $F = \emptyset$, entonces es claro que $D_F = \emptyset$ es Σ-p.r.. Supongamos entonces que F no es la funcion $\emptyset$. Tenemos entonces que r es de la forma $n + m$ y

$$g : D_g \subseteq \omega^n \times \Sigma^{*m} \to O$$
$$g_i : D_{g_i} \subseteq \omega^k \times \Sigma^{*l} \to \omega,\ i = 1, ..., n$$
$$g_i : D_{g_i} \subseteq \omega^k \times \Sigma^{*l} \to \Sigma^*, i = n + 1, ..., n + m$$

con $O \in \{\omega, \Sigma^*\}$ y $k, l \in \omega$. Por Lema 19, hay funciones Σ-p.r. $\bar{g}_1, ..., \bar{g}_{n+m}$ las cuales son Σ-totales y cumplen

$$g_i = \bar{g}_i|_{D_{g_i}}, \text{ para } i = 1, ..., n + m.$$

Por hipotesis inductiva los conjuntos D_g, D_{g_i}, $i = 1, ..., n + m$, son Σ-p.r. y por lo tanto

$$S = \bigcap_{i=1}^{n+m} D_{g_i}$$

lo es. Notese que

$$\chi_{D_F}^{\omega^k \times \Sigma^{*l}} = (\chi_{D_g}^{\omega^n \times \Sigma^{*m}} \circ [\bar{g}_1, ..., \bar{g}_{n+m}] \wedge \chi_S^{\omega^k \times \Sigma^{*l}})$$

lo cual nos dice que D_F es Σ-p.r.. ■

Lema de division por casos para funciones Σ*-p.r.* Una observacion interesante es que si $f_i : D_{f_i} \to O$, $i = 1, ..., k$, son funciones tales que $D_{f_i} \cap D_{f_j} = \emptyset$ para $i \neq j$, entonces $f_1 \cup ... \cup f_k$ es la funcion

$$\begin{array}{rcl} D_{f_1} \cup ... \cup D_{f_k} & \to & O \\ e & \to & \begin{cases} f_1(e) & \text{si } e \in D_{f_1} \\ \vdots & \vdots \\ f_k(e) & \text{si } e \in D_{f_k} \end{cases} \end{array}$$

Lema 21. *Sean* $O \in \{\omega, \Sigma^*\}$ *y* $n, m \in \omega$*. Supongamos* $f_i : D_{f_i} \subseteq \omega^n \times \Sigma^{*m} \to O$*,* $i = 1, ..., k$*, son funciones* Σ*-p.r. tales que* $D_{f_i} \cap D_{f_j} = \emptyset$ *para* $i \neq j$*. Entonces* $f_1 \cup ... \cup f_k$ *es* Σ*-p.r..*

Proof. Supongamos $O = \Sigma^*$ y $k = 2$. Sean

$$\bar{f}_i : \omega^n \times \Sigma^{*m} \to \Sigma^*, i = 1, 2,$$

funciones Σ-p.r. tales que $\bar{f}_i|_{D_{f_i}} = f_i$, $i = 1, 2$ (Lema 19). Por la Proposicion 20 los conjuntos D_{f_1} y D_{f_2} son Σ-p.r. y por lo tanto lo es $D_{f_1} \cup D_{f_2}$. Ya que

$$f_1 \cup f_2 = \left(\lambda\alpha\beta\left[\alpha\beta\right] \circ \left[\lambda x\alpha\left[\alpha^x\right] \circ \left[\chi_{D_{f_1}}^{\omega^n \times \Sigma^{*m}}, \bar{f}_1\right], \lambda x\alpha\left[\alpha^x\right] \circ \left[\chi_{D_{f_2}}^{\omega^n \times \Sigma^{*m}}, \bar{f}_2\right]\right]\right)|_{D_{f_1} \cup D_{f_2}}$$

tenemos que $f_1 \cup f_2$ es Σ-p.r..

El caso $k > 2$ puede probarse por induccion ya que

$$f_1 \cup ... \cup f_k = (f_1 \cup ... \cup f_{k-1}) \cup f_k.$$

■

CONSEJO IMPORTANTE: Si uno quiere usar el lema de division por casos para probar que una funcion f es Σ-p.r., entonces lo primero que hay que hacer, antes de ver que algo sea Σ-p.r. o no, es (a lo mariposa) definir correctamente funciones $f_1, ..., f_k$ tales que $D_{f_i} \cap D_{f_j} = \emptyset$ para $i \neq j$ y ademas $f_1 \cup ... \cup f_k = f$. Consejos para encontrar dichas funciones:

(1) Determinar el k, es decir, k es justamente la cantidad de "casos" en la descripcion de f
(2) Para cada "caso" de la descripcion de f, asociar un subconjunto del dominio de f el cual sea justamente definido por la propiedad correspondiente a ese caso. Ojo que dijimos subconjunto de D_f, no confundir los tipos!! (a veces los casos se describen usando no todas las variables de las cuales depende la funcion)
(3) Notar que los subconjuntos $S_1, ..., S_k$ asi definidos deben ser disjuntos de a pares y unidos deben dar el dominio de f
(4) Para cada i defina f_i de la siguiente manera:
    (a) dominio de $f_i = S_i$
    (b) regla de f_i dada por la regla que describe f para el caso i-esimo
(5) En general suele suceder que f_i es la restriccion a S_i de una funcion con dominio mas amplio y se prueba entonces que tanto dicha funcion como S_i son Σ-p.r., resultando asi que f_i es Σ-p.r.

Ejercicio 34: Sea $\Sigma = \{@, !, \%\}$. Pruebe que f es Σ-p.r..

(a) $$f : \omega \times \Sigma^* \rightarrow \omega$$
$$(x, \alpha) \rightarrow \begin{cases} |\alpha| . x^2 & \text{si } x + |\alpha| \text{ es impar} \\ 0 & \text{si } x + |\alpha| \text{ es par} \end{cases}$$

(b) $$f : \{10, 11, 17\} \times \Sigma^+ \rightarrow \omega$$
$$(x, \alpha) \rightarrow \begin{cases} Pred(x) & \text{si } x \text{ es impar} \\ |\alpha| & \text{si } x \text{ es par} \end{cases}$$

(c) $$f : \mathbf{N} \times \Sigma^+ \rightarrow \omega$$
$$(x, \alpha) \rightarrow \begin{cases} x^2 & \text{si } x + |\alpha| \text{ es par} \\ 0 & \text{si } x + |\alpha| \text{ es impar} \end{cases}$$

(d) $$f : \{(x, y, \alpha) : x \leq y\} \rightarrow \omega$$
$$(x, y, \alpha) \rightarrow \begin{cases} x^2 & \text{si } |\alpha| \leq y \\ 0 & \text{si } |\alpha| > y \end{cases}$$

(Explicado en video colgado en granlogico.com)

(e) $$f : \{1, 2, 3, 4, 5\} \times \mathbf{N} \times \{@, \%\}^* \rightarrow \Sigma^*$$
$$(x, y, \alpha) \rightarrow \begin{cases} \alpha^2 & \text{si } \alpha = @@ \\ !!! & \text{si } \alpha \neq @@ \wedge |\alpha| > y \\ \alpha^{x+y} & \text{si } \alpha \neq @@ \wedge |\alpha| \leq y \end{cases}$$

Ejercicio 35: Sea $F : \omega \rightarrow \omega$ dada por

$$F(0) = 2$$
$$F(1) = 2^2$$
$$F(2) = (2^2)^3$$
$$F(3) = ((2^2)^3)^2$$
$$F(4) = (((2^2)^3)^2)^3$$
$$\vdots$$

Pruebe que F es Σ-p.r..

Ejercicio 36: Sean $f_1, f_2 : \omega \rightarrow \omega$ funciones Σ-p.r.. Sea $F : \omega \rightarrow \omega$ dada por

$$F(0) = 0$$
$$F(1) = f_1(0)$$
$$F(2) = f_2(f_1(0))$$
$$F(3) = f_1(f_2(f_1(0)))$$
$$F(4) = f_2(f_1(f_2(f_1(0))))$$
$$\vdots$$

Pruebe que F es Σ-p.r.. Obtenga el ejercicio anterior a partir de este resultado.

Usaremos el lema de division por casos para probar que la funcion $\lambda i \alpha \, [[\alpha]_i]$ es Σ-p.r.. Recordemos que dados $i \in \omega$ y $\alpha \in \Sigma^*$, definimos

$$[\alpha]_i = \begin{cases} i\text{-esimo elemento de } \alpha & \text{si } 1 \leq i \leq |\alpha| \\ \varepsilon & \text{caso contrario} \end{cases}$$

Notese que $D_{\lambda i \alpha [[\alpha]_i]} = \omega \times \Sigma^*$.

Lema 22. *$\lambda i\alpha\,[[\alpha]_i]$ es Σ-p.r..*

Proof. Supongamos $\Sigma = \{@, !\}$. Note que

$$[\varepsilon]_i = \varepsilon$$
$$[\alpha@]_i = \begin{cases} [\alpha]_i & \text{si } i \neq |\alpha|+1 \\ @ & \text{si } i = |\alpha|+1 \end{cases}$$
$$[\alpha!]_i = \begin{cases} [\alpha]_i & \text{si } i \neq |\alpha|+1 \\ ! & \text{si } i = |\alpha|+1 \end{cases}$$

lo cual dice que $\lambda i\alpha\,[[\alpha]_i] = R\left(C_\varepsilon^{1,0}, \mathcal{G}\right)$, donde $\mathcal{G}_a : \omega \times \Sigma^* \times \Sigma^* \to \Sigma^*$ es dada por

$$\mathcal{G}_a(i,\alpha,\zeta) = \begin{cases} \zeta & \text{si } i \neq |\alpha|+1 \\ a & \text{si } i = |\alpha|+1 \end{cases}$$

para cada $a \in \Sigma$. O sea que solo resta probar que cada $\mathcal{G}_a$ es Σ-p.r.. Veamos que $\mathcal{G}_@$ es Σ-p.r.. Primero note que los conjuntos

$$S_1 = \{(i,\alpha,\zeta) \in \omega \times \Sigma^* \times \Sigma^* : i \neq |\alpha|+1\}$$
$$S_2 = \{(i,\alpha,\zeta) \in \omega \times \Sigma^* \times \Sigma^* : i = |\alpha|+1\}$$

son Σ-p.r. ya que

$$\chi_{S_1}^{\omega\times\Sigma^*\times\Sigma^*} = \lambda xy\,[x \neq y] \circ \left[p_1^{1,2}, Suc \circ \lambda\alpha\,[|\alpha|] \circ p_2^{1,2}\right]$$
$$\chi_{S_2}^{\omega\times\Sigma^*\times\Sigma^*} = \lambda xy\,[x = y] \circ \left[p_1^{1,2}, Suc \circ \lambda\alpha\,[|\alpha|] \circ p_2^{1,2}\right]$$

Notese que por el Lema 18 tenemos que $p_3^{1,2}|_{S_1}$ y $C_@^{1,2}|_{S_2}$ son Σ-p.r.. Ademas

$$\mathcal{G}_@ = p_3^{1,2}|_{S_1} \cup C_@^{1,2}|_{S_2}$$

por lo cual el Lema 21 nos dice que $\mathcal{G}_@$ es Σ-p.r.. Analogamente se prueba que $\mathcal{G}_!$ es Σ-p.r.. ■

Sumatoria, productoria y concatenatoria de funciones Σ-p.r. Sea Σ un alfabeto finito. Sea $f : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$, con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ no vacios. Para $x, y \in \omega$ y $(\vec{x}, \vec{\alpha}) \in S_1 \times ... \times S_n \times L_1 \times ... \times L_m$, definamos

$$\sum_{t=x}^{t=y} f(t,\vec{x},\vec{\alpha}) = \begin{cases} 0 & \text{si } x > y \\ f(x,\vec{x},\vec{\alpha}) + f(x+1,\vec{x},\vec{\alpha}) + ... + f(y,\vec{x},\vec{\alpha}) & \text{si } x \leq y \end{cases}$$
$$\prod_{t=x}^{t=y} f(t,\vec{x},\vec{\alpha}) = \begin{cases} 1 & \text{si } x > y \\ f(x,\vec{x},\vec{\alpha}).f(x+1,\vec{x},\vec{\alpha})....f(y,\vec{x},\vec{\alpha}) & \text{si } x \leq y \end{cases}$$

En forma similar, cuando $I_f \subseteq \Sigma^*$, definamos

$$\overset{t=y}{\underset{t=x}{\subset}} f(t,\vec{x},\vec{\alpha}) = \begin{cases} \varepsilon & \text{si } x > y \\ f(x,\vec{x},\vec{\alpha})f(x+1,\vec{x},\vec{\alpha})....f(y,\vec{x},\vec{\alpha}) & \text{si } x \leq y \end{cases}$$

Note que, en virtud de la definicion anterior, el dominio de las funciones

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

es $\omega \times \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m$.

Dejamos al lector que analice en las consideraciones de recien el caso $n = m = 0$ (es decir cuando $D_f = \omega$).

Lema 23 (Lema de la sumatoria)**.** *Sea Σ un alfabeto finito.*

(a) *Si $f : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \omega$ es Σ-p.r., con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ no vacios, entonces las funciones $\lambda xy\vec{x}\vec{\alpha}\left[\sum_{t=x}^{t=y} f(t, \vec{x}, \vec{\alpha})\right]$ y $\lambda xy\vec{x}\vec{\alpha}\left[\prod_{t=x}^{t=y} f(t, \vec{x}, \vec{\alpha})\right]$ son Σ-p.r.*

(b) *Si $f : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$ es Σ-p.r., con $S_1, ..., S_n \subseteq \omega$ y $L_1, ..., L_m \subseteq \Sigma^*$ no vacios, entonces la funcion $\lambda xy\vec{x}\vec{\alpha}\left[\subset_{t=x}^{t=y} f(t, \vec{x}, \vec{\alpha})\right]$ es Σ-p.r.*

Proof. (a) Sea $G = \lambda tx\vec{x}\vec{\alpha}\left[\sum_{i=x}^{i=t} f(i, \vec{x}, \vec{\alpha})\right]$. Ya que

$$\lambda xy\vec{x}\vec{\alpha}\left[\sum_{i=x}^{i=y} f(i, \vec{x}, \vec{\alpha})\right] = G \circ \left[p_2^{n+2,m}, p_1^{n+2,m}, p_3^{n+2,m}, ..., p_{n+m+2}^{n+2,m}\right]$$

solo tenemos que probar que G es Σ-p.r.. Primero note que

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

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

O sea que si definimos

$$\begin{aligned} h : \omega \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m &\to \omega \\ (x, \vec{x}, \vec{\alpha}) &\to \begin{cases} 0 & \text{si } x > 0 \\ f(0, \vec{x}, \vec{\alpha}) & \text{si } x = 0 \end{cases} \end{aligned}$$

$$\begin{aligned} g : \omega^3 \times S_1 \times ... \times S_n \times L_1 \times ... \times L_m &\to \omega \\ (A, t, x, \vec{x}, \vec{\alpha}) &\to \begin{cases} 0 & \text{si } x > t+1 \\ A + f(t+1, \vec{x}, \vec{\alpha}) & \text{si } x \leq t+1 \end{cases} \end{aligned}$$

tenemos que $G = R(h, g)$. Es decir que solo nos falta probar que h y g son Σ-p.r.. Notese que

$$\begin{aligned} h &= C_0^{n+1,m}|_{D_1} \cup \lambda x\vec{x}\vec{\alpha}\left[f(0, \vec{x}, \vec{\alpha})\right]|_{D_2} \\ g &= C_0^{n+3,m}|_{H_1} \cup \lambda Atx\vec{x}\vec{\alpha}\left[A + f(t+1, \vec{x}, \vec{\alpha})\right])|_{H_2} \end{aligned}$$

asique para ver que h y g son Σ-p.r. podemos aplicar el Lema 21 una ves que hayamos visto que las funciones

$$\begin{aligned} &C_0^{n+1,m}|_{D_1} \quad \lambda x\vec{x}\vec{\alpha}\left[f(0, \vec{x}, \vec{\alpha})\right]|_{D_2} \\ &C_0^{n+3,m}|_{H_1} \quad \lambda Atx\vec{x}\vec{\alpha}\left[A + f(t+1, \vec{x}, \vec{\alpha})\right])|_{H_2} \end{aligned}$$

son Σ-p.r.. Es claro que $C_0^{n+1,m}$ y $C_0^{n+3,m}$ son Σ-p.r.. Tambien notese que

$$\lambda x\vec{x}\vec{\alpha}\left[f(0, \vec{x}, \vec{\alpha})\right] = f \circ \left[C_0^{n+1,m}, p_2^{n+1,m}, p_3^{n+1,m}, ..., p_{n+1+m}^{n+1,m}\right]$$

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

lo cual, ya que f es Σ-p.r., nos dice que $\lambda x\vec{x}\vec{\alpha}\,[f(0,\vec{x},\vec{\alpha})]$ y $\lambda Atx\vec{x}\vec{\alpha}\,[A+f(t+1,\vec{x},\vec{\alpha})]$) son Σ-p.r.. O sea que el Lema 18 nos dice que solo nos resta probar que

$$D_1=\{(x,\vec{x},\vec{\alpha})\in\omega\times S_1\times...\times S_n\times L_1\times...\times L_m : x>0\}$$
$$D_2=\{(x,\vec{x},\vec{\alpha})\in\omega\times S_1\times...\times S_n\times L_1\times...\times L_m : x=0\}$$
$$H_1=\left\{(z,t,x,\vec{x},\vec{\alpha})\in\omega^3\times S_1\times...\times S_n\times L_1\times...\times L_m : x>t+1\right\}$$
$$H_2=\left\{(z,t,x,\vec{x},\vec{\alpha})\in\omega^3\times S_1\times...\times S_n\times L_1\times...\times L_m : x\leq t+1\right\}.$$

son Σ-p.r.. Veamos por ejemplo que H_1 lo es. Es decir debemos ver que $\chi_{H_1}^{\omega^{3+n}\times\Sigma^{*m}}$ es Σ-p.r.. Ya que f es Σ-p.r. tenemos que $D_f=\omega\times S_1\times...\times S_n\times L_1\times...\times L_m$ es Σ-p.r., lo cual por el Lema 17 nos dice que los conjuntos $S_1,...,S_n,\ L_1,...,L_m$ son Σ-p.r.. Ya que ω es Σ-p.r., el Lema 17 nos dice que

$$R=\omega^3\times S_1\times...\times S_n\times L_1\times...\times L_m$$

es Σ-p.r.. Notese que

$$\chi_{H_1}^{\omega^{3+n}\times\Sigma^{*m}}=(\chi_R^{\omega^{3+n}\times\Sigma^{*m}}\wedge\lambda ztx\vec{x}\vec{\alpha}\,[x>t+1])$$

por lo cual $\chi_{H_1}^{\omega^{3+n}\times\Sigma^{*m}}$ es Σ-p.r. ya que es la conjuncion de dos predicados Σ-p.r. La prueba de que D_1, D_2 y H_2 son Σ-p.r. es similar. ■

Nota: Aceptaremos sin prueba (b) y el caso de la productoria en (a). Las pruebas son muy similares a la dada para la sumatoria.

Veamos un ejemplo de como se puede aplicar el lema anterior. Sea $F=\lambda yx_1\left[\sum_{t=0}^{t=y}(x_1)^t\right]$. Es claro que $D_F=\omega^2$. Para ver que F es Σ-p.r. aplicaremos el lema anterior por lo cual es importante encontrar la f adecuada a la cual se le aplicara el lema. Tomemos $f=\lambda tx_1[(x_1)^t]$. Claramente f es Σ-p.r. por lo cual el lema anterior nos dice que

$$G=\lambda xyx_1\left[\sum_{t=x}^{t=y}f(t,x_1)\right]=\lambda xyx_1\left[\sum_{t=x}^{t=y}(x_1)^t\right]$$

es Σ-p.r.. Notar que G no es la funcion F pero es en algun sentido "mas amplia" que F ya que tiene una variable mas y se tiene que $F(y,x_1)=G(0,y,x_1)$, para cada $y,x_1\in\omega$. Es facil ver que

$$F=G\circ\left[C_0^{2,0},p_1^{2,0},p_2^{2,0}\right]$$

por lo cual F es Σ-p.r..

Haga los siguientes ejercicios aplicando el lema anterior. No caiga en la tentacion de hacerlos aplicando recursion primitiva ya que no se ejercitara en la habilidad de aplicar el lema en forma madura.

Ejercicio 37: Pruebe que las siguientes funciones son Σ-p.r..

(a) $\lambda x\left[\prod_{t=10}^{t=x}t^t\right]$

(b) $\lambda xy\alpha\left[\prod_{t=y+1}^{t=|\alpha|}(t+|\alpha|)\right]$

(c) $\lambda xyz\alpha\beta[\subset_{t=3}^{t=z+5}\alpha^t]$

(d) $\lambda xyx_1\beta\left[\sum_{t=0}^{t=y}|\beta|.(x_1+t)\right]$

(e) $\lambda x_2xyz\alpha\beta[\subset_{t=x_2}^{t=z+5}\alpha^{y.x.t}\beta^z]$

Ejercicio 38: Describa el dominio de G y pruebe que G es Σ-p.r.

(a) $G=\lambda xx_1\left[\sum_{t=1}^{t=x}Pred(x_1)^t\right]$

(b) $G=\lambda xyz\alpha\beta\left[\subset_{t=3}^{t=z+5}\alpha^{Pred(z).t^x}\beta^{Pred(Pred(|\alpha|^y))}\right]$

(Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

Cuantificacion acotada de predicados Σ-p.r. con dominio rectangular. Sea $P:S\times S_1\times...\times S_n\times L_1\times...\times L_m\to\omega$ un predicado, con $n,m\in\omega$. Supongamos ademas que $S,S_1,...,S_n\subseteq\omega$ y $L_1,...,L_m\subseteq\Sigma^*$ son no vacios. Sea $\bar{S}\subseteq S$. Entonces la expresion Booleana

$$(\forall t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})$$

depende de las variables $x,\vec{x},\vec{\alpha}$ y valdra 1 en una $(1+n+m)$-upla $(x,\vec{x},\vec{\alpha})$ cuando $P(t,\vec{x},\vec{\alpha})$ sea igual a 1 para cada $t\in\{u\in\bar{S}:u\leq x\}$; y 0 en caso contrario. Notese que aqui es crucial que $\bar{S}\subseteq S$ ya que si no sucediera esto no tendria sentido preguntarnos si $P(t,\vec{x},\vec{\alpha})$ es 1 para cada $t\in\{u\in\bar{S}:u\leq x\}$. Ademas tambien para que la expresion $(\forall t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})$ este definida es preciso que $(\vec{x},\vec{\alpha})\in S_1\times...\times S_n\times L_1\times...\times L_m$. Sobre la variable x no hay ninguna restriccion o sea que puede ser cualquier elemento de ω. Tenemos entonces que el dominio del predicado

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

es $\omega\times S_1\times...\times S_n\times L_1\times...\times L_m$. En forma analoga se define la forma de interpretar la expresion Booleana

$$(\exists t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})$$

Cabe destacar que

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

Analicemos un poco por separado el caso $\bar{S}=\emptyset$. Nos queda la expresion

$$(\forall t\in\emptyset)_{t\leq x}\ P(t,\vec{x},\vec{\alpha})$$

la cual para que este definida requiere que $(x,\vec{x},\vec{\alpha})$ este en $\omega\times S_1\times...\times S_n\times L_1\times...\times L_m$ (ver (12) (d) de la notacion lambda dada en la Guia 1); y siempre es verdadera. O sea que

$$\lambda x\vec{x}\vec{\alpha}\left[(\forall t\in\emptyset)_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]=C_1^{1+n,m}|_{\omega\times S_1\times...\times S_n\times L_1\times...\times L_m}$$

Analogamente

$$\lambda x\vec{x}\vec{\alpha}\left[(\exists t\in\emptyset)_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]=C_0^{1+n,m}|_{\omega\times S_1\times...\times S_n\times L_1\times...\times L_m}$$

Tambien podemos cuantificar sobre variable alfabetica. Sea $P:S_1\times...\times S_n\times L_1\times...\times L_m\times L\to\omega$ un predicado, con $S_1,...,S_n\subseteq\omega$ y $L,L_1,...,L_m\subseteq\Sigma^*$ no vacios. Supongamos $\bar{L}\subseteq L$. Entonces la expresion Booleana

$$(\forall\alpha\in\bar{L})_{|\alpha|\leq x}\ P(\vec{x},\vec{\alpha},\alpha)$$

depende de las variables $x,\vec{x},\vec{\alpha}$ y esta definida cuando $(x,\vec{x},\vec{\alpha})$ pertenece a $\omega\times S_1\times...\times S_n\times L_1\times...\times L_m$. Dejamos al lector analizar cuando vale 1 el predicado

$$\lambda x\vec{x}\vec{\alpha}\left[(\forall\alpha\in\bar{L})_{|\alpha|\leq x}\ P(\vec{x},\vec{\alpha},\alpha)\right]$$

Tambien dejamos al lector estudiar el comportamiento de la funcion

$$\lambda x\vec{x}\vec{\alpha}\left[(\exists\alpha\in\bar{L})_{|\alpha|\leq x}P(\vec{x},\vec{\alpha},\alpha)\right]$$

Cabe destacar que tambien

$$\lambda x\vec{x}\vec{\alpha}\left[(\exists\alpha\in\bar{L})_{|\alpha|\leq x}P(\vec{x},\vec{\alpha},\alpha)\right]=\neg\lambda x\vec{x}\vec{\alpha}\left[(\forall\alpha\in\bar{L})_{|\alpha|\leq x}\neg P(\vec{x},\vec{\alpha},\alpha)\right]$$

Dejamos al lector que analice el caso $\bar{L}=\emptyset$ y tambien el caso $n=m=0$ para ambos tipos de cuantificacion (es decir cuando $P:S\subseteq\omega\to\omega$ o $P:L\subseteq\Sigma^*\to\omega$).

Lema 24 (Lema de cuantificacion acotada)**.** *Sea Σ un alfabeto finito.*

(a) *Sea $P:S\times S_1\times...\times S_n\times L_1\times...\times L_m\to\omega$ un predicado Σ-p.r., con $S,S_1,...,S_n\subseteq\omega$ y $L_1,...,L_m\subseteq\Sigma^*$ no vacios. Supongamos $\bar{S}\subseteq S$ es Σ-p.r.. Entonces $\lambda x\vec{x}\vec{\alpha}\left[(\forall t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]$ y $\lambda x\vec{x}\vec{\alpha}\left[(\exists t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]$ son predicados Σ-p.r..*

(b) *Sea $P:S_1\times...\times S_n\times L_1\times...\times L_m\times L\to\omega$ un predicado Σ-p.r., con $S_1,...,S_n\subseteq\omega$ y $L,L_1,...,L_m\subseteq\Sigma^*$ no vacios. Supongamos $\bar{L}\subseteq L$ es Σ-p.r.. Entonces $\lambda x\vec{x}\vec{\alpha}\left[(\forall\alpha\in\bar{L})_{|\alpha|\leq x}\ P(\vec{x},\vec{\alpha},\alpha)\right]$ y $\lambda x\vec{x}\vec{\alpha}\left[(\exists\alpha\in\bar{L})_{|\alpha|\leq x}\ P(\vec{x},\vec{\alpha},\alpha)\right]$ son predicados Σ-p.r..*

Proof. (a) Sea

$$\bar{P}=P|_{\bar{S}\times S_1\times...\times S_n\times L_1\times...\times L_m}\cup C_1^{1+n,m}|_{(\omega-\bar{S})\times S_1\times...\times S_n\times L_1\times...\times L_m}$$

Notese que $\bar{P}$ tiene dominio $\omega\times S_1\times...\times S_n\times L_1\times...\times L_m$. Veamos que $\bar{P}$ es Σ-p.r.. Ya que P es Σ-p.r. tenemos que $D_P=S\times S_1\times...\times S_n\times L_1\times...\times L_m$ es Σ-p.r., lo cual por el Lema 17 nos dice que los conjuntos $S_1,...,S_n,\ L_1,...,L_m$ son Σ-p.r.. Si $\bar{S}\neq\emptyset$, ya que es Σ-p.r. por hipotesis, el Lema 17 nos dice que

$$\bar{S}\times S_1\times...\times S_n\times L_1\times...\times L_m$$

es Σ-p.r.. Si $\bar{S}=\emptyset$ entonces

$$\bar{S}\times S_1\times...\times S_n\times L_1\times...\times L_m$$

es tambien Σ-p.r. ya que es igual al conjunto vacio. O sea que el Lema 18 nos dice que

$$P|_{\bar{S}\times S_1\times...\times S_n\times L_1\times...\times L_m}$$

es Σ-p.r.. En forma similar, ya que $\omega-\bar{S}$ es Σ-p.r. (Lema 16), obtenemos que

$$C_1^{1+n,m}|_{(\omega-\bar{S})\times S_1\times...\times S_n\times L_1\times...\times L_m}$$

es Σ-p.r.. Pero entonces el Lema 21 nos dice que $\bar{P}$ es Σ-p.r.. El Lema 23 nos dice que $\lambda xy\vec{x}\vec{\alpha}\left[\prod_{t=x}^{t=y}\bar{P}(t,\vec{x},\vec{\alpha})\right]$ es Σ-p.r.. Ya que

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

tenemos que $\lambda x\vec{x}\vec{\alpha}\left[(\forall t\in\bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]$ es Σ-p.r..

Ya que

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

tenemos que $\lambda x\vec{x}\vec{\alpha}\left[(\exists t \in \bar{S})_{t\leq x}\ P(t,\vec{x},\vec{\alpha})\right]$ es Σ-p.r. ■

Nota: Aceptaremos (b) sin prueba. Su prueba se basa en (a) y el lector puede verla en el apunte.

OBSERVACION: La cuantificacion no acotada no preserva la propiedad de ser Σ-p.r.. Como veremos mas adelante hay un predicado Σ-p.r., $P : \omega \times L_1 \to \omega$, tal que el predicado $\lambda\alpha\,[(\exists t \in \omega)\ P(t,\alpha)]$ no es Σ-efectivamente computable, por lo cual tampoco es Σ-p.r. (ni siquiera podra ser Σ-recursivo).

Veamos por ejemplo que el predicado $\lambda xy\,[x \text{ divide } y]$ es Σ-p.r.. Sea $P = \lambda t x_1 x_2\,[x_2 = t.x_1]$. Es claro que P es Σ-p.r.. El lema anterior nos dice que $\lambda x x_1 x_2\,[(\exists t \in \omega)_{t\leq x}\ P(t,x_1,x_2)]$ es Σ-p.r.. Notese que x_1 divide x_2 si y solo si hay un $t \leq x_2$ tal que $x_2 = t.x_1$. Esto nos dice que

$$\lambda x_1 x_2\,[x_1 \text{ divide } x_2] = \lambda x_1 x_2\,[(\exists t \in \omega)_{t\leq x_2}\ P(t,x_1,x_2)]$$

Pero

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

por lo cual $\lambda x_1 x_2\,[x_1 \text{ divide } x_2]$ es Σ-p.r.

La idea fundamental subyacente en la aplicacion anterior es que en muchos casos de predicados obtenidos por cuantificacion a partir de otros predicados, la variable cuantificada tiene una cota natural en terminos de las otras variables y entonces componiendo adecuadamente se lo puede presentar como un caso de cuantificacion acotada

Ejercicio 39: Use que

$$x \text{ es primo sii } x > 1 \wedge ((\forall t \in \omega)_{t\leq x}\ t = 1 \vee t = x \vee \neg(t \text{ divide } x))$$

para probar que $\lambda x\,[x \text{ es primo}]$ es Σ-p.r.

A continuacion enunciamos una regla que es util cuando queremos probar que cierto conjunto es Σ-p.r..

Regla Caracterizar Pertenencia: Si Ud esta intentando probar que cierto conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-p.r., entonces puede ser util primero caracterizar la pertenecia a S, es decir escribir algo del tipo:

- Para cada $(\vec{x},\vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ se tiene que: $(\vec{x},\vec{\alpha}) \in S$ si y solo si ...

Por ejemplo si queremos probar que $S = \{\beta^2 : \beta \in \Sigma^*\}$ es un conjunto Σ-p.r. podemos primero notar que

- Para cada $\alpha \in \Sigma^*$ se tiene que: $\alpha \in S$ si y solo si $(\exists\beta \in \Sigma^*)$ tal que $\alpha = \beta^2$

lo cual ya nos deja en evidencia que podemos aplicar el Lema de Cuantificacion Acotada para probar que $\chi_S^{\Sigma^*} = \lambda\alpha[\alpha \in S]$ es Σ-p.r.. A esta regla la llamaremos *Regla CP* por brevedad.

Ejercicio 40: Pruebe que

(a) $\lambda\alpha\beta\,[\alpha$ inicial $\beta]$ es un predicado Σ-p.r..
(b) $\{\beta^2 : \beta \in \Sigma^*\}$ es un conjunto Σ-p.r..
(c) $\{2^x : x \in \omega$ y x es impar$\}$ es un conjunto Σ-p.r.. (Aplique Regla CP).
(d) $\{(x, \alpha, \beta) \in \omega \times \Sigma^* \times \Sigma^* : (\exists t \in \omega)\ \alpha^x = \beta^t\}$ es un conjunto Σ-p.r..
(e) $\{(2^x, @^x, \$) : x \in \omega$ y x es impar$\}$ es un conjunto Σ-p.r. (aqui asuma $\Sigma = \{@, \$\}$). (Aplique Regla CP).

Ejercicio 41: Dos numeros $p, q \in \mathbf{N}$ son llamados *primos consecutivos* si ambos son primos, $p < q$ y no hay ningun primo r tal que $p < r < q$. Pruebe que el conjunto

$$\{(p, q) : p \text{ y } q \text{ son primos consecutivos}\}$$

es Σ-p.r

Ejercicio 42: Dados $x, y \in \omega$, diremos que x e y son *coprimos* cuando 1 sea el unico elemento de ω que divide a ambos. Sea $P = \lambda xy[x$ e y son coprimos$]$. Pruebe que P es Σ-p.r.

Ejercicio 43: Pruebe que los siguientes conjuntos son Σ-p.r..
(a) $\{(x, \alpha, \beta) \in \omega \times \Sigma^* \times \Sigma^* : (\exists t \in \mathrm{Im}(pr))\ \alpha^{Pred(Pred(x)).Pred(|\alpha|)} = \beta^t\}$
(b) $\{(x, \alpha) \in \omega \times \Sigma^* : (\exists \beta \in \{@, \%\}^+)\ \alpha\beta^{Pred(x)} = \beta^{Pred(x)}\alpha\}$ (aqui suponga que $\{@, \%\} \subseteq \Sigma$)
(c) $\{(x, \alpha) \in \{1, 9\} \times \Sigma^+ : (\exists \beta \in \Sigma^+)\ |\beta| > x$ y $\beta^{Pred(x)} = \alpha\}$

(Ojo que el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total)

Ejercicio 44: Sea $\Sigma = \{@, \%\}$. Sea $S \subseteq \omega$ un conjunto Σ-p.r. Pruebe que

$$\{(x, y, \alpha) \in S \times S \times \Sigma^+ : (\exists t \in S)\ |\alpha|^t = Pred(Pred(x))\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

Como puede notarse, en los ejercicios anteriores se aplica una sola ves el lema de cuantificacion acotada. En los ejercicios que siguen veremos algunos casos en los cuales es necesario anidar cuantificaciones acotadas.

Ejercicio 45: Pruebe que
(a) $\lambda\alpha\beta\,[\alpha$ ocurre en $\beta]$ es un predicado Σ-p.r..
(b) $\{@^t!^l : t \in \mathbf{N}$ y l es impar$\}$ es un conjunto Σ-p.r.. (Aplique Regla CP).
(c) $\{@^t!^l : t \in \mathbf{N}$ y l es impar$\}$ es un conjunto Σ-p.r. (aqui suponga que $@, ! \in \Sigma$). (Aplique Regla CP).
(d) $\{(2^t, \gamma^5) : \gamma \in \{@, \%\}^+$ y $t > |\gamma|\}$ es un conjunto Σ-p.r. (aqui suponga que $\{@, \%\} \subseteq \Sigma$). (Aplique Regla CP).
(e) $\{(2^{t+l}, 10^{t.l}, \gamma) : \gamma \in \{@, \%\}^+$ y $t > l\}$ es un conjunto Σ-p.r. (aqui suponga que $\{@, \%\} \subseteq \Sigma$). (Aplique Regla CP).
(f) $\{x \in \mathbf{N} : \exists p, q$ tales que $x = p.q$ y p, q son primos$\}$ es un conjunto Σ-p.r.

Ejercicio 46: Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$ y sean $L_1, L_2 \subseteq \{\triangle, \heartsuit, \bigcirc\}^*$ conjuntos Σ-p.r.. Pruebe que el conjunto

$$\{\alpha@\beta : \alpha \in L_1 \text{ y } \beta \in L_2\}$$

es Σ-p.r.. (Aplique Regla CP).

Ejercicio 47: Sea $M = (Q, \{@, \%\}, \Gamma, \delta, q_0, B, F)$ una maquina de Turing y supongamos $Q = \{q_0, q_1\}$ es un alfabeto disjunto con Γ. Pruebe que el conjunto Des formado por todas las descripciones instantaneas de M es $(\Gamma \cup Q)$-p.r.. (Hint: note que $Des = Des_0 \cup Des_1$, donde $Des_i = \{\alpha \in Des : St(\alpha) = q_i\}$, por lo cual basta con probar que cada Des_i es $(\Gamma \cup Q)$-p.r.). Puede usar que la funcion $\lambda i \alpha\, [[\alpha]_i]$ es $(\Gamma \cup Q)$-p.r. (aqui la notacion lambda es relativa al alfabeto $\Gamma \cup Q$).

Ejercicios de examen

Cada resultado teorico que aplique en la resolucion debera enunciarlo por separado detalladamente (en la forma en la que esta en las guias). Para los ejercicios listados a continuacion puede usar sin demostracion que las siguientes funciones son Σ-p.r.: Suc, $Pred$, d_a (con $a \in \Sigma$), $p_j^{n,m}$ (con $n, m, j \in \omega$ y $1 \leq j \leq n + m$), $\lambda xy[x \leq y]$, $\lambda\alpha[|\alpha|]$, $\lambda xy[x = y]$, $\lambda\alpha\beta\,[\alpha = \beta]$, $\lambda xy[x + y]$, $\lambda xy[x.y]$, $\lambda xy\left[x \dot{-} y\right]$, $\lambda x\alpha\,[\alpha^x]$, $\lambda xy\,[x^y]$, $C_k^{n,m}$, $C_\alpha^{n,m}$ (con $n, m, k \in \omega$ y $\alpha \in \Sigma^*$).

(1) Sea $\Sigma = \{@, \%, \$\}$ y sea $\leq$ el orden total sobre Σ dado por $@ < \% < \$$. Pruebe que $s^{\leq}$, $\#^{\leq}$ y $*^{\leq}$ son Σ-p.r.

(2) Defina las funciones Sig y Dec.
   (a) Pruebe que Sig y Des son Num-p.r.
   (b) Sea $\Gamma = \{@, \%\} \cup Num$. Pruebe que Dec es Γ-p.r..

(3) Sea $\Sigma = \{@, !, \%\}$. Sea

$$\begin{array}{rcl} f : \mathbf{N} \times \{@, \%\}^* \times \{@, !, \%\}^+ & \to & \omega \\ (x, \alpha, \beta) & \to & \begin{cases} Pred(|\alpha|) & \text{si } |\alpha| > 2 \text{ y } x \geq 1 \\ |\beta| & \text{si } |\alpha| \leq 2 \text{ o } x = 0 \end{cases} \end{array}$$

Pruebe que f es Σ-p.r..

(4) Sea $\Sigma = \{@, !, \%\}$. Sea

$$\begin{array}{rcl} f : \{(x, \alpha, \beta) \in \omega \times \Sigma^* \times \Sigma^* : x \geq |\beta|\} & \to & \omega \\ (x, \alpha, \beta) & \to & \begin{cases} Pred(|\alpha|) & \text{si } |\alpha| > 2 \text{ y } x \geq 1 \\ x & \text{si } |\alpha| \leq 2 \text{ o } x = 0 \end{cases} \end{array}$$

Pruebe que f es Σ-p.r..

(5) Sea $\Sigma = \{@, \%, !\}$. Sea

$$\begin{array}{rcl} f : \{1, 4\} \times \mathbf{N} \times \{@, \%\}^* & \to & \omega \\ (x, y, \alpha) & \to & \begin{cases} x & \text{si } \alpha = @@ \\ Pred(|\alpha|) & \text{si } \alpha \neq @@ \wedge |\alpha| > y \\ y & \text{si } \alpha \neq @@ \wedge |\alpha| \leq y \end{cases} \end{array}$$

Pruebe que f es Σ-p.r..

(6) Sea $\Sigma = \{@, \$, \%, !\}$. Sea

$$\begin{array}{rcl} f : \{(x, y, \alpha, \beta) \in \omega^2 \times \Sigma^{*2} : \beta \in \{@, \$\}^* \text{ o } x \text{ es par}\} & \to & \omega \\ (x, y, \alpha, \beta) & \to & \begin{cases} x^2 & \text{si } |\alpha| \leq y \\ Pred(|\alpha|) & \text{si } |\alpha| > y \end{cases} \end{array}$$

Pruebe que f es Σ-p.r..

(7) Sea $\Sigma = \{@, \$, \%, !\}$. Sea

$$\begin{aligned} f : \{(\alpha, \beta) \in \Sigma^{*2} : \beta \in \{@, \$\}^* \text{ o } \alpha \in \{\$, !\}^+\} &\rightarrow \Sigma^* \\ (\alpha, \beta) &\rightarrow \begin{cases} \beta & \text{si } |\alpha| \leq 5 \\ \alpha^{Pred(|\alpha|)} & \text{si } |\alpha| > 5 \end{cases} \end{aligned}$$

Pruebe que f es Σ-p.r..

(8) Sean $f_1, f_2 : \omega \rightarrow \omega$ funciones Σ-p.r.. Sea $F : \omega \rightarrow \omega$ dada por

$$\begin{aligned} F(0) &= 0 \\ F(1) &= f_1(0) \\ F(2) &= f_2(f_1(0)) \\ F(3) &= f_1(f_2(f_1(0))) \\ F(4) &= f_2(f_1(f_2(f_1(0)))) \\ &\vdots \end{aligned}$$

Pruebe que F es Σ-p.r..

(9) Sea $\Sigma = \{@, !, \%\}$.
   (a) Describa el dominio de $\lambda i \alpha\, [[\alpha]_i]$ (segun manda la notacion lambda)
   (b) Pruebe que $\lambda i \alpha\, [[\alpha]_i]$ es Σ-p.r..

(10) Sea

$$G = \lambda z x x_1 \alpha \left[\prod_{t=1}^{t=|\alpha|} z^t . Pred(x. \, |\alpha|) \right]$$

   (a) Describa el dominio de G (segun manda la notacion lambda)
   (b) Pruebe que G es Σ-p.r.. (Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

(11) Sea $g : \omega \rightarrow \omega$ una funcion Σ-p.r.. Sea

$$G = \lambda z x_1 \alpha \left[\prod_{t=1}^{t=|\alpha|} g(Pred((t+1).x_1. \, |\alpha|)) \right]$$

   (a) Describa el dominio de G (segun manda la notacion lambda)
   (b) Pruebe que G es Σ-p.r.. (Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

(12) Sea $g : \omega \rightarrow \Sigma^*$ una funcion Σ-p.r.. Sea

$$G = \lambda z x_1 x_2 \alpha \left[\subset_{t=x_2}^{t=z+5} g(t.Pred(x_1. \, |\alpha|)) \right]$$

   (a) Describa el dominio de G (segun manda la notacion lambda)
   (b) Pruebe que G es Σ-p.r.. (Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

(13) Sea

$$G = \lambda x z y x_2 \left[\sum_{t=Pred(y)}^{t=x} Pred(z)^t \right]$$

   (a) Describa el dominio de G (segun manda la notacion lambda)
   (b) Pruebe que G es Σ-p.r.. (Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

(14) Sea

$$G = \lambda xzyx_2 \left[\sum_{t=Pred(x_2)}^{t=x_2} x.y.z.t \right]$$

   (a) Describa el dominio de G (segun manda la notacion lambda)
   (b) Pruebe que G es Σ-p.r.. (Ojo que la la f a la cual le debe aplicar el lema de la sumatoria no es Σ-total)

(15) Sea $\Sigma = \{@, !, \%\}$.
   (a) Describa el dominio de $\lambda x\,[x \text{ es primo}]$ (segun manda la notacion lambda)
   (b) Pruebe que $\lambda x\,[x \text{ es primo}]$ es Σ-p.r..

(16) Sea $\Sigma = \{@, \%, !\}$. Sea $S = \{x \in \omega : x \text{ es par}\}$. Pruebe que

$$L = \{\alpha \in \Sigma^* : \text{ si } [\alpha]_i = @, \text{ entonces } [\alpha]_{i+1} = \%, \text{ para cada } i \in S\}$$

es Σ-p.r.. Puede usar que la funcion $\lambda i\alpha\,[[\alpha]_i]$ es Σ-p.r..

(17) Sea $\Sigma = \{@, \%, !\}$. Sea $L = \{\alpha \in \Sigma^* : |\alpha| \text{ es impar}\}$. Pruebe que

$$H = \{(\alpha, \beta, \rho) \in L \times \Sigma^* \times L : (\exists \gamma \in L)\ \beta\rho = \alpha \text{ y } \beta^{Pred(|\gamma|)} = \alpha\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(18) Sea $\Sigma = \{\triangle, \bigcirc\}$. Sea $S = \{x \in \omega : x \text{ es par}\}$. Pruebe que

$$H = \{(x, \alpha) \in \omega \times \{\triangle\}^* : (\exists t \in S)\ t^2 = |\alpha|\,.Pred(x)\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(19) Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$. Sea $S = \{x \in \omega : 10 \leq x\}$. Pruebe que

$$H = \{(x, \alpha, \beta) \in \omega \times \{@, \triangle\}^* \times \Sigma^+ : (\exists t \in S)\ \alpha^{Pred(Pred(x)).Pred(|\alpha|)} = \beta^t\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(20) Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$. Sea $S = \{x \in \omega : 100 \leq x\}$. Pruebe que

$$H = \{(x, \alpha, \rho) \in S \times \Sigma^* \times \Sigma^+ : (\exists \beta \in \{@, \triangle\}^*)\ \alpha\beta^{Pred(x)} = \rho\alpha\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(21) Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$. Sea $S = \{x \in \omega : 3 \text{ divide a } x\}$. Pruebe que

$$H = \{(x, \alpha) \in S \times \Sigma^* : (\exists \beta \in \{\triangle, \heartsuit, \bigcirc\}^+)\ |\beta| > x \text{ y } \beta^{Pred(x)} = \alpha\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(22) Sea $\Sigma = \{@, \%\}$. Sea $L \subseteq \Sigma^*$ un conjunto Σ-p.r. Pruebe que

$$H = \{(x, y, \alpha) \in \omega \times \omega \times L : (\exists t \in \mathbf{N})\ t^2 = |\alpha|\,.y^{Pred(Pred(x))}\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(23) Sea $\Sigma = \{@, \%\}$. Sea $S \subseteq \omega$ un conjunto Σ-p.r. Pruebe que

$$H = \{(x, y, \alpha) \in S \times S \times \Sigma^+ : (\exists t \in S)\ |\alpha|^t = Pred(Pred(x))\}$$

es Σ-p.r.. (Ojo que aqui el predicado al cual debera aplicarle el lema de cuantificacion acotada no es Σ-total).

(24) Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$ y sean $L_1, L_2 \subseteq \{\triangle, \heartsuit, \bigcirc\}^*$ conjuntos Σ-p.r.. Pruebe que el conjunto

$$H = \{\alpha @ \beta : \alpha \in L_1 \text{ y } \beta \in L_2\}$$

es Σ-p.r..

(25) Sea $\Sigma = \{@, \triangle, \heartsuit, \bigcirc\}$ y sean $L \subseteq \{\triangle, \heartsuit, \bigcirc\}^*$ y $S \subseteq \omega$ conjuntos Σ-p.r.. Pruebe que el conjunto

$$H = \{[\alpha]_i @ \alpha : \alpha \in L \text{ y } i \in S\}$$

es Σ-p.r.. Puede usar que la funcion $\lambda i \alpha\, [[\alpha]_i]$ es Σ-p.r..

(26) Sea $\Sigma = \{\triangle, \heartsuit, \bigcirc\}$ y sean $L \subseteq \{\triangle, \heartsuit, \bigcirc\}^*$ y $S \subseteq \omega$ conjuntos Σ-p.r.. Pruebe que el conjunto

$$H = \{\alpha^x : \alpha \in L \text{ y } x \in S\}$$

es Σ-p.r

(27) Sea $\Sigma = \{\triangle, \heartsuit, \bigcirc\}$. Sean $S_1, S_2 \subseteq \omega$ conjuntos Σ-p.r.. Pruebe que el conjunto

$$H = \{(x.y, \triangle\triangle\bigcirc) : x \in S_1 \text{ y } y \in S_2\}$$

es Σ-p.r

(28) Sea $\Sigma = \{@, !\}$. Pruebe que

$$H = \{2^t 5^l : t, l \in \mathbf{N} \text{ y } t > l\}$$

es Σ-p.r..

(29) Sea $\Sigma = \{@, !\}$. Pruebe que

$$H = \{(2^t, (@@!!)^l) : t, l \in \mathbf{N} \text{ y } t > l\}$$

es Σ-p.r..

(30) Sea $\Sigma = \{@, \%, !\}$. Pruebe que

$$H = \{(2^t, \gamma^5) : \gamma \in \{@, \%\}^+ \text{ y } t > |\gamma|\}$$

es Σ-p.r..

(31) Sea $\Sigma = \{@, \%, !\}$. Pruebe que

$$H = \{(2^{t+l}, 10^{t.l}, \gamma) : \gamma \in \{@, \%\}^+ \text{ y } t > l\}$$

es Σ-p.r..

(32) Sea $M = (Q, \{@, \%\}, \Gamma, \delta, q_0, B, F)$ una maquina de Turing y supongamos $Q = \{q_0, q_1\}$ es un alfabeto disjunto con Γ. Pruebe que el conjunto Des formado por todas las descripciones instantaneas de M es $(\Gamma \cup Q)$-p.r.. (Hint: note que $Des = Des_0 \cup Des_1$, donde $Des_i = \{\alpha \in Des : St(\alpha) = q_i\}$, por lo cual basta con probar que cada Des_i es $(\Gamma \cup Q)$-p.r.). Puede usar que la funcion $\lambda i \alpha\, [[\alpha]_i]$ es $(\Gamma \cup Q)$-p.r. (aqui la notacion lambda es relativa al alfabeto $\Gamma \cup Q$).

(33) Sea $\Gamma = \{\mathrm{N}, \leftarrow\} \cup Num$. Para hacer mas agil la notacion escribiremos $\bar{n}$ en lugar de $Dec(n)$. Sea

$$L = \{\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n} : k, n \in \mathbf{N}\}$$

Pruebe que L es Γ-p.r.. (Puede usar que Dec es Γ-p.r.).

GUIA 6 DE LENGUAJES: MINIMIZACION Y FUNCIONES Σ-RECURSIVAS

Tal como fue explicado en el comienzo de la Guia 5, para obtener la clase de las funciones Σ-recursivas debemos agregar un nuevo constructor a los ya definidos de composicion y recursion primitiva, a saber, el constructor de *minimizacion*. Tiene dos casos aunque solo usaremos el primero para la definicion de funcion Σ-recursiva.

Minimizacion de variable numerica

Sea Σ un alfabeto finito y sea $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$ un predicado. Dado $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$, cuando exista al menos un $t \in \omega$ tal que $P(t, \vec{x}, \vec{\alpha}) = 1$, usaremos $\min_t P(t, \vec{x}, \vec{\alpha})$ para denotar al menor de tales $t's$. Notese que la expresion $\min_t P(t, \vec{x}, \vec{\alpha})$ esta definida solo para aquellas $(n+m)$-uplas $(\vec{x}, \vec{\alpha})$ para las cuales hay al menos un t tal que se da $P(t, \vec{x}, \vec{\alpha}) = 1$. Dicho de otra forma, $\min_t P(t, \vec{x}, \vec{\alpha})$ no estara definida cuando para cada $t \in \omega$ se de que $(t, \vec{x}, \vec{\alpha})$ no pertenece a D_P o $P(t, \vec{x}, \vec{\alpha}) = 0$. Otro detalle importante a tener en cuenta es que la expresion $\min_t P(t, \vec{x}, \vec{\alpha})$ no depende de la variable t. Por ejemplo, las expresiones $\min_t P(t, \vec{x}, \vec{\alpha})$ y $\min_i P(i, \vec{x}, \vec{\alpha})$ son equivalentes en el sentido que estan definidas en las mismas $(n+m)$-uplas y cuando estan definidas asumen el mismo valor.

Definamos

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

Notese que

$$D_{M(P)} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists t \in \omega)\ P(t, \vec{x}, \vec{\alpha})\}$$
$$M(P)(\vec{x}, \vec{\alpha}) = \min_t P(t, \vec{x}, \vec{\alpha}), \text{ para cada } (\vec{x}, \vec{\alpha}) \in D_{M(P)}$$

Diremos que $M(P)$ se obtiene por *minimizacion de variable numerica* a partir de P.

Veamos algunos ejemplos:

(E1) Tomemos $P = \lambda t x_1 [t^2 = x_1]$. Tenemos que:

$$\begin{aligned} D_{M(P)} &= \{x_1 \in \omega : (\exists t \in \omega)\ P(t, x_1)\} \\ &= \left\{x_1 \in \omega : (\exists t \in \omega)\ t^2 = x_1\right\} \end{aligned}$$

Es decir el dominio de $M(P)$ es el conjunto de los cuadrados. Ademas para cada $x_1 \in D_{M(P)}$ tenemos que

$$M(P)(x_1) = \min_t P(t, x_1) = \min_t (t^2 = x_1)$$

por lo cual $M(P)(x) = \sqrt{x}$, para cada $x \in D_{M(P)}$.

(E2) Cuando $n = m = 0$, tenemos que $P : D_P \subseteq \omega \to \omega$. O sea que

$$M(P) = \lambda \left[\min_t P(t)\right]$$

Esto nos dice que $D_{M(P)} \subseteq \{\Diamond\}$. Ademas $D_{M(P)} = \{\Diamond\}$ si y solo si $P(t) = 1$, para algun $t \in D_P$, y en tal caso $M(P)(\Diamond) = \min_t P(t)$.

(E3) Recordemos que dados $x_1, x_2 \in \omega$, con x_2 no nulo, el *cociente de dividir* x_1 *por* x_2 se define como el maximo elemento del conjunto $\{t \in \omega : t.x_2 \leq x_1\}$. Sea

$$\begin{array}{rcl} Q : \omega \times \mathbf{N} & \to & \omega \\ (x_1, x_2) & \to & \text{cociente de dividir } x_1 \text{ por } x_2 \end{array}$$

Sea $P = \lambda t x_1 x_2 \, [x_1 < t.x_2]$. Notar que

$$\begin{aligned} D_{M(P)} &= \{(x_1, x_2) \in \omega^2 : (\exists t \in \omega) \; P(t, x_1, x_2) = 1\} \\ &= \{(x_1, x_2) : (\exists t \in \omega) \; x_1 < t.x_2\} \\ &= \omega \times \mathbf{N} \end{aligned}$$

Ademas si $(x_1, x_2) \in \omega \times \mathbf{N}$, es facil de probar que

$$\min_t \; x_1 < t.x_2 = Q(x_1, x_2) + 1$$

por lo que $M(P) = Suc \circ Q$. Si quisieramos encontrar un predicado P' tal que $M(P') = Q$, entonces podemos tomar $P' = \lambda t x_1 x_2 \, [x_1 < (t+1).x_2]$ y con un poco de concentracion nos daremos cuenta que $M(P') = Q$. Por supuesto esto tiene una cuota de artificialidad ya que no es tan obvio que

$$Q(x_1, x_2) = \text{ menor } t \in \omega \text{ tal que } x_1 < (t+1).x_2$$

Hay una forma mas intuitiva de hacerlo y es diseñando P' con la consigna de que para cada $(x_1, x_2) \in D_Q$ se de que

$$Q(x_1, x_2) = \text{ unico } t \in \omega \text{ tal que } P'(t, x_1, x_2)$$

Es decir diseñamos P' de manera que

$$P'(t, x_1, x_2) = 1 \text{ si y solo si } t = Q(x_1, x_2)$$

Por ejemplo se puede tomar $P' = \lambda t x_1 x_2 \, [x_1 \geq t.x_2 \text{ y } x_1 < (t+1).x_2]$ que dicho sea de paso es justo la definicion de cociente dada en la escuela primaria. Dejamos al lector corroborar que $M(P') = Q$, para este ultimo P', pero notese que lo unico que queda por chequear es que $M(P')$ y Q tienen el mismo dominio ya que las reglas de asignacion obviamente producen lo mismo.

Tal como lo vimos recien muchas veces que querramos diseñar un predicado P tal que $M(P)$ sea igual a una funcion dada f, sera mas facil diseñar P de manera que cumpla

$$f(\vec{x}, \vec{\alpha}) = \text{ unico } t \in \omega \text{ tal que } P(t, \vec{x}, \vec{\alpha})$$

cada vez que $(\vec{x}, \vec{\alpha}) \in D_f$. Es decir un predicado P que caracterice al valor que toma f, es decir el cual cumpla

$$P(t, \vec{x}, \vec{\alpha}) = 1 \text{ si y solo si } t = f(\vec{x}, \vec{\alpha})$$

Luego por supuesto deberemos prestar atencion a que $D_{M(P)} = D_f$, es decir deberemos tambien lograr que para cada $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ se de que

$$(\vec{x}, \vec{\alpha}) \in D_f \text{ si y solo si } \exists t \in \omega \; P(t, \vec{x}, \vec{\alpha})$$

Enunciamos esto en forma de regla.

REGLA U: Si tiene una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ y busca un predicado P tal que $f = M(P)$, intente diseñar P de manera que para cada $(\vec{x}, \vec{\alpha}) \in D_f$ se de que

$$f(\vec{x}, \vec{\alpha}) = \text{ unico } t \in \omega \text{ tal que } P(t, \vec{x}, \vec{\alpha})$$

Luego chequee que ademas para cada $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ se de que

$$(\vec{x}, \vec{\alpha}) \in D_f \text{ si y solo si } \exists t \in \omega \; P(t, \vec{x}, \vec{\alpha})$$

Ejercicio 1: Sea $\Sigma = \{@, !, \%\}$. Para cada caso describa la funcion $M(P)$:

(a) $P = \lambda t x_1[x_1 < t]$
(b) $P = \lambda t x_1[t < x_1]$
(c) $P = \lambda t \alpha \beta[\alpha^t = \beta]$
(d) $P = \lambda t \alpha \beta[\alpha = \beta]$
(e) $P = \lambda t x_1 x_2[x_2 + t = x_1]$
(f) $P = \lambda t \alpha_1[[\alpha_1]_t = \%]$
(g) $P = \lambda x y[y < x]$
(h) $P = \lambda x_1 t[x_1 < t]$
(i) $P = \lambda y x \alpha[[\alpha]_x = \%]$

Ejercicio 2: Sea $E = \lambda x_1[\text{parte entera de } \sqrt{x_1}]$. Note que por notacion lambda $D_E = \omega$. Aplique la REGLA U para encontrar un predicado P tal que $M(P) = E$.

Ejercicio 3: Encuentre un predicado P tal que $M(P) = \lambda x_1 x_2[x_1 \dot{-} x_2]$. (Aqui es natural hacerlo sin la idea de la REGLA U.)

Ejercicio 4: Sea $F : \{z^2 : z \in \omega\} \to \omega$ dada por $F(x) = \sqrt{x}$. Encuentre un predicado P tal que $M(P) = F$.

Ejercicio 5: Sea $F : \{(x, y) \in \omega \times \mathbf{N} : y \text{ divide a } x\} \to \omega$ dada por $F(x, y) = x/y$. Encuentre un predicado P tal que $M(P) = F$.

El siguiente lema nos muestra que en algunos casos el constructor de minimizacion preserva la computabilidad efectiva.

Lema 1. *Si $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$ es un predicado Σ-efectivamente computable y D_P es Σ-efectivamente computable, entonces la funcion $M(P)$ es Σ-efectivamente computable.*

Proof. Sea $\mathbb{P}_P$ un procedimiento efectivo que compute a P y sea $\mathbb{P}_{D_P}$ un procedimiento efectivo que compute a $\chi_{D_P}^{\omega \times \omega^n \times \Sigma^{*m}}$. Notese que el siguiente procedimiento efectivo (con dato de entrada $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$) computa la funcion $M(P)$.

Etapa 1: Hacer $T = 0$ e ir a Etapa 2

Etapa 2: Correr $\mathbb{P}_{D_P}$ con dato de entrada $(T, \vec{x}, \vec{\alpha})$ y guardar la salida en A. Ir a Etapa 3.

Etapa 3: Si $A = 1$ ir a Etapa 4, caso contrario ir a etapa 6.

Etapa 4: Correr $\mathbb{P}_P$ con dato de entrada $(T, \vec{x}, \vec{\alpha})$ y guardar la salida en B. Ir a Etapa 5.

Etapa 5: Si $B = 1$, dar como salida T y terminar. Caso contrario ir a Etapa 6.

Etapa 6: Hacer $T = T + 1$ e ir a Etapa 2. ■ Como corolario obtenemos que la

minimizacion de predicados Σ-totales preserva la computabilidad efectiva:

Corolario 2. *Si $P : \omega \times \omega^n \times \Sigma^{*m} \to \omega$ es un predicado Σ-efectivamente computable, entonces la funcion $M(P)$ es Σ-efectivamente computable.*

Lamentablemente si quitamos la hipotesis en el lema anterior de que D_P sea Σ-efectivamente computable, el lema resulta falso. Mas adelante veremos un ejemplo de esto, basado en la Tesis de Church (esta tesis basicamente dice que el modelo Godeliano de la computabilidad efectiva es correcto (Guia 8)).

Ejercicio 6: Intente construir un procedimiento efectivo que que compute a $M(P)$ teniendo un procedimiento efectivo que compute a un predicado $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$. Seguro fallara en sus intentos pero entendera cual es la dificultad subyacente.

Ejercicio 7: (Responder V o F) Si $P : D_P \subseteq \omega \times \omega^2 \to \omega$ es un predicado Σ-efectivamente computable, entonces el siguiente procedimiento computa a $M(P)$:

Etapa 1: Hacer $T = 0$ e ir a Etapa 2

Etapa 2: Si $(T, x, y) \in D_P$ y $P(T, x, y) = 1$, entonces ir a Etapa 4, en caso contrario ir a Etapa 3.

Etapa 3: Hacer $T = T + 1$ e ir a Etapa 2.

Etapa 4: Dar T como salida y terminar

Ejercicio 8: V o F o I. Justifique.

(a) Sea $P : D_P \subseteq \omega \times \omega^n \to \omega$ un predicado. Si $\vec{x} \in \omega^n$ es tal que existe t en ω que cumple $(t, \vec{x}) \in D_P$, entonces $\vec{x} \in D_{M(P)}$.

(b) Sea $P : D_P \subseteq \omega \times \omega^n \to \omega$ un predicado. Entonces $M(P)(\vec{x}) \leq t$

(c) Sea $P : \omega^n \to \omega$ un predicado, con $n \geq 1$. Entonces $D_{M(P)} \subseteq \omega^{n-1}$

(d) Sea Σ un alfabeto finito. Entonces $M(p_1^{1,2}) = C_0^{0,2}$

Definicion de funcion Σ-recursiva

Con este nuevo constructor de funciones estamos en condiciones de definir la clase de las funciones Σ-recursivas. Definamos los conjuntos $\mathrm{R}_0^\Sigma \subseteq \mathrm{R}_1^\Sigma \subseteq \mathrm{R}_2^\Sigma \subseteq ... \subseteq \mathrm{R}^\Sigma$ de la siguiente manera

$$\begin{array}{lcl}
\mathrm{R}_0^\Sigma & = & \mathrm{PR}_0^\Sigma \\
\mathrm{R}_{k+1}^\Sigma & = & \mathrm{R}_k^\Sigma \cup \big\{ f \circ [f_1, ..., f_n] : f, f_1, ..., f_r \in \mathrm{R}_k^\Sigma,\ r \geq 1 \big\} \cup \\
& & \quad \big\{ R(f, \mathcal{G}) : R(f, \mathcal{G}) \text{ esta definida y } \{f\} \cup \{\mathcal{G}_a : a \in \Sigma\} \subseteq \mathrm{R}_k^\Sigma \big\} \cup \\
& & \quad\quad \big\{ R(f, g) : R(f, g) \text{ esta definida y } f, g \in \mathrm{R}_k^\Sigma \big\} \cup \\
& & \quad\quad\quad \big\{ M(P) : P \text{ es un predicado } \Sigma\text{-total y } P \in \mathrm{R}_k^\Sigma \big\} \\
\mathrm{R}^\Sigma & = & \bigcup_{k \geq 0} \mathrm{R}_k^\Sigma
\end{array}$$

Una funcion f es llamada Σ-*recursiva* si pertenece a R^Σ. Cabe destacar que aunque $M(P)$ fue definido para predicados no necesariamente Σ-totales, en la definicion de los conjuntos R_k^Σ, nos restringimos al caso en que P es Σ-total. Obviamente esto lo hacemos ya que como se explico antes el constructor de minimizacion no siempre preserva la computabilidad efectiva.

Notese que $\mathrm{PR}_k^\Sigma \subseteq \mathrm{R}_k^\Sigma$, para cada $k \in \omega$, por lo cual $\mathrm{PR}^\Sigma \subseteq \mathrm{R}^\Sigma$. Por supuesto el modelo de Godel seria incorrecto si no fuera cierto el siguiente resultado.

Proposición 3 (Leibniz vence a Godel)**.** *Si $F \in \mathrm{R}^\Sigma$, entonces F es Σ-efectivamente computable.*

Proof. Por induccion en k probaremos que

Teo$_k$: Si $F \in \mathrm{R}_k^\Sigma$, entonces F es Σ-efectivamente computable.

Teo_0 es facil y dejado al lector.

$\text{Teo}_k \Rightarrow \text{Teo}_{k+1}$. Supongamos entonces que $F \in \text{R}^\Sigma_{k+1}$ y que vale Teo_k. Veamos que entonces F es Σ-efectivamente computable. Ya que vale Teo_k, podemos suponer que $F \in \text{R}^\Sigma_{k+1} - \text{R}^\Sigma_k$. Por la definicion de R^Σ_{k+1} surgen varios casos:

Caso $f \circ [f_1, ..., f_n]$, con $f, f_1, ..., f_r \in \text{R}^\Sigma_k$, $r \geq 1$. Entonces por un lema del comienzo de la Guia 5 tenemos que F es Σ-efectivamente computable ya que $f, f_1, ..., f_r$ lo son.

Los distintos casos en los que F se obtiene por recursion primitiva a partir de funciones de R^Σ_k se siguen de Teo_k y los respectivos lemas dados en la Guia 5 los cuales prueban que cada uno de los cuatro casos de recursion primitiva preserva la computabilidad efectiva.

Caso $F = M(P)$, con $P : \omega \times \omega^n \times \Sigma^{*m} \to \omega$ un predicado perteneciente a R^Σ_k. Sigue directamente del corolario anterior y Teo_k. ■

Daremos sin prueba el siguiente conceptualmente importante resultado.

Proposición 4. *Sea Σ un alfabeto finito. Entonces no toda funcion Σ-recursiva es Σ-p.r.. Es decir que $\text{PR}^\Sigma \subseteq \text{R}^\Sigma$ y $\text{PR}^\Sigma \neq \text{R}^\Sigma$.*

Este resultado no es facil de probar. Mas adelante veremos ejemplos naturales de funciones Σ-recursivas que no son Σ-p.r.. Otro ejemplo natural es la famosa funcion de Ackermann.

Lema de minimizacion acotada de variable numerica de predicados Σ-p.r. Como veremos mas adelante, no siempre que $P \in \text{R}^\Sigma$, tendremos que $M(P) \in \text{R}^\Sigma$. Sin envargo, el siguiente lema nos garantiza que cuando $P \in \text{PR}^\Sigma$, se da que $M(P) \in \text{R}^\Sigma$ y ademas da condiciones para que $M(P)$ sea Σ-p.r..

Lema 5 (Lema de minimizacion acotada)**.** *Sean $n, m \geq 0$. Sea $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$ un predicado Σ-p.r.. Entonces*

(a) *$M(P)$ es Σ-recursiva.*

(b) *Si hay una funcion Σ-p.r. $f : \omega^n \times \Sigma^{*m} \to \omega$ tal que*

$$M(P)(\vec{x}, \vec{\alpha}) = \min_t P(t, \vec{x}, \vec{\alpha}) \leq f(\vec{x}, \vec{\alpha}), \text{ para cada } (\vec{x}, \vec{\alpha}) \in D_{M(P)},$$

entonces $M(P)$ es Σ-p.r..

Proof. (a) Sea $\bar{P} = P \cup C_0^{n+1,m}|_{(\omega^{n+1} \times \Sigma^{*m}) - D_P}$. Note que $\bar{P}$ es Σ-p.r. (por que?). Veremos a continuacion que $M(P) = M(\bar{P})$. Notese que

$$\{t \in \omega : P(t, \vec{x}, \vec{\alpha}) = 1\} = \{t \in \omega : \bar{P}(t, \vec{x}, \vec{\alpha}) = 1\}$$

Esto claramente dice que $D_{M(P)} = D_{M(\bar{P})}$ y que $M(P)(\vec{x}, \vec{\alpha}) = M(\bar{P})(\vec{x}, \vec{\alpha})$, para cada $(\vec{x}, \vec{\alpha}) \in D_{M(P)}$., por lo cual $M(P) = M(\bar{P})$.

Veremos entonces que $M(\bar{P})$ es Σ-recursiva. Sea k tal que $\bar{P} \in \text{PR}^\Sigma_k$. Ya que $\bar{P}$ es Σ-total y $\bar{P} \in \text{PR}^\Sigma_k \subseteq \text{R}^\Sigma_k$, tenemos que $M(\bar{P}) \in \text{R}^\Sigma_{k+1}$ y por lo tanto $M(\bar{P}) \in \text{R}^\Sigma$.

(b) Ya que $M(P) = M(\bar{P})$, basta con probar que $M(\bar{P})$ es Σ-p.r. Primero veremos que $D_{M(\bar{P})}$ es un conjunto Σ-p.r.. Notese que

$$\chi^{\omega^n \times \Sigma^{*m}}_{D_{M(\bar{P})}} = \lambda \vec{x} \vec{\alpha} \left[(\exists t \in \omega)_{t \leq f(\vec{x}, \vec{\alpha})} \, \bar{P}(t, \vec{x}, \vec{\alpha}) \right]$$

lo cual nos dice que

$$\chi^{\omega^n\times\Sigma^{*m}}_{D_{M(\bar{P})}} = \lambda x\vec{x}\vec{\alpha}\left[(\exists t\in\omega)_{t\leq x}\ \bar{P}(t,\vec{x},\vec{\alpha})\right]\circ\left[f,p_1^{n,m},...,p_{n+m}^{n,m}\right]$$

Pero el Lema de Cuantificacion acotada probado en la Guia 5 nos dice que el predicado $\lambda x\vec{x}\vec{\alpha}\left[(\exists t\in\omega)_{t\leq x}\ \bar{P}(t,\vec{x},\vec{\alpha})\right]$ es Σ-p.r. por lo cual tenemos que $\chi^{\omega^n\times\Sigma^{*m}}_{D_{M(\bar{P})}}$ lo es. Sea

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

Note que P_1 es Σ-total. Dejamos al lector usando lemas anteriores probar que P_1 es Σ-p.r. Ademas notese que para $(\vec{x},\vec{\alpha})\in\omega^n\times\Sigma^{*m}$ se tiene que

$$P_1(t,\vec{x},\vec{\alpha}) = 1 \text{ si y solo si } (\vec{x},\vec{\alpha})\in D_{M(\bar{P})} \text{ y } t = M(\bar{P})(\vec{x},\vec{\alpha})$$

Esto nos dice que

$$M(\bar{P}) = \left(\lambda\vec{x}\vec{\alpha}\left[\prod_{t=0}^{f(\vec{x},\vec{\alpha})} t^{P_1(t,\vec{x},\vec{\alpha})}\right]\right)|_{D_{M(\bar{P})}}$$

por lo cual para probar que $M(\bar{P})$ es Σ-p.r. solo nos resta probar que

$$F = \lambda\vec{x}\vec{\alpha}\left[\prod_{t=0}^{f(\vec{x},\vec{\alpha})} t^{P_1(t,\vec{x},\vec{\alpha})}\right]$$

lo es. Pero

$$F = \lambda xy\vec{x}\vec{\alpha}\left[\prod_{t=x}^{y} t^{P_1(t,\vec{x},\vec{\alpha})}\right]\circ\left[C_0^{n,m},f,p_1^{n,m},...,p_{n+m}^{n,m}\right]$$

y por lo tanto el Lema de la Sumatoria probado en la Guia 5 nos dice que F es Σ-p.r.. ■

OBSERVACION: No siempre que P sea Σ-p.r. tendremos que $M(P)$ lo sera. Notese que si $M(P)$ fuera Σ-p.r., cada ves que P lo sea, entonces tendriamos que $\mathrm{PR}^\Sigma = \mathrm{R}^\Sigma$ (justifique) lo cual contradiria la Proposicion 4. Mas adelante veremos un ejemplo natural de un predicado P el cual es Σ-p.r. pero $M(P)$ no es Σ-p.r.

El lema de minimizacion recien probado es muy util como lo veremos a continuacion.

Lema 6. *Sea Σ un alfabeto finito. Las siguientes funciones son Σ-p.r.:*

(a) $Q:\omega\times\mathbf{N} \to \omega$
$(x,y) \to$ *cociente de la division de x por y*

(b) $R:\omega\times\mathbf{N} \to \omega$
$(x,y) \to$ *resto de la division de x por y*

Proof. (a) Ya vimos anteriormente que $Q = M(P')$, donde $P' = \lambda t x_1 x_2\,[x_1\geq t.x_2 \text{ y } x_1<(t+1).x_2]$. Ya que P' es Σ-p.r. y

$$Q(x_1,x_2)\leq p_1^{2,0}(x_1,x_2),\ \text{para cada } (x_1,x_2)\in\omega\times\mathbf{N}$$

(b) del Lema 5 implica que $Q\in\mathrm{PR}^\Sigma$.

(b) Notese que

$$R = \lambda xy\left[x\dot{-}Q(x,y).y\right]$$

y por lo tanto $R\in\mathrm{PR}^\Sigma$. ■

Ejercicio 9: Dados $x, y \in \omega$ tales que $x \neq 0$ o $y \neq 0$, usaremos $mcd(x, y)$ para denotar el maximo comun divisor de x e y, es decir el mayor numero que divide a x y divide a y. Note que la funcion $M = \lambda xy[mcd(x, y)]$ tiene dominio igual a $\omega^2 - \{(0, 0)\}$. Pruebe que M es Σ-p.r.. (Hint: use la REGLA U).

Ejercicio 10: Dados $x, y \in \mathbf{N}$, usaremos $mcm(x, y)$ para denotar el minimo comun multiplo de x e y, es decir el menor numero no nulo que es multiplo de x y de y. Note que la funcion $G = \lambda xy[mcm(x, y)]$ tiene dominio igual a $\mathbf{N}^2$. Pruebe que G es Σ-p.r..

Ejercicio 11: Sea $F : \{z^2 : z \in \omega\} \to \omega$ dada por $F(x) = \sqrt{x}$. Pruebe que F es Σ-p.r..

Ejercicio 12: Sea $F : \Sigma^* \times \Sigma^* \to \omega$ dada por $F(\alpha, \beta) = \max\{|\rho| : \rho$ es tramo inicial de α y $\beta\}$. Pruebe que F es Σ-p.r..

Lema 7. *Sea Σ un alfabeto finito. Entonces la funcion*

$$\begin{array}{rcl} pr : \mathbf{N} & \to & \omega \\ n & \to & n\text{-esimo numero primo} \end{array}$$

es Σ-p.r.

Proof. Para ver que pr es Σ-p.r., veremos que la extension $h : \omega \to \omega$, dada por $h(0) = 0$ y $h(n) = pr(n)$, $n \geq 1$, es Σ-p.r.. Luego $pr = h|_{\mathbf{N}}$ resultara Σ-p.r. por ser la restriccion de una funcion Σ-p.r. a un conjunto Σ-p.r.. Primero note que

$$\begin{aligned} h(0) &= 0 \\ h(t+1) &= \min_i (i \text{ es primo} \wedge i > h(t)) \end{aligned}$$

O sea que $h = R\left(C_0^{0,0}, g\right)$, donde

$$\begin{array}{rcl} g : \omega \times \omega & \to & \omega \\ (A, t) & \to & \min_i (i \text{ es primo} \wedge i > A) \end{array}$$

Es decir que solo nos resta ver que g es Σ-p.r.. Pero notese que $g = M(P)$, donde $P = \lambda iAt\,[i \text{ es primo} \wedge i > A]$. Claramente P es Σ-p.r. por lo cual para poder aplicar (b) del lema anterior debemos encontrar una funcion $f : \omega \times \omega \to \omega$ tal que

$$M(P)(A, t) \leq f(A, t), \text{ para cada } (A, t) \in \omega^2$$

Aceptaremos sin prueba que

$$\min_i (i \text{ es primo} \wedge i > A) \leq A! + 1, \text{ para cada } A \in \omega$$

Es decir que $f = \lambda At[A! + 1]$ cumple lo deseado, lo cual implica que $g = M(P)$ es Σ-p.r. ■

Ejercicio 13: (Opcional) Si tiene ganas y recuerda las propiedades basicas de divisibilidad, intente un rato probar que

$$\min_i (i \text{ es primo} \wedge i > A) \leq A! + 1, \text{ para cada } A \in \omega$$

(Hint: factorice $A! + 1$ en producto de primos y vea que alguno debe ser mayor que A.)

Ejercicio 14: Sea Σ un alfabeto finito.

(a) Pruebe que $\lambda xi\,[(x)_i]$ es Σ-p.r. (Hint: repase el significado de la expresion $(x)_i$ y encuentre entonces el dominio de $\lambda xi\,[(x)_i]$ antes de hacer el ejercicio)

(b) Pruebe que la funcion Lt es Σ-p.r.

Ejercicio 15: Sea $F : \mathbf{N} \to \omega$ dada por $F(x) = \max\{t \in \omega : t! \leq x\}$. Pruebe que F es Σ-p.r..

Ejercicio 16: Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda\alpha\beta[\max\{t \in \mathbf{N} : \alpha^t \text{ es tramo inicial de } \beta\}]$.

(a) Encuentre (segun manda notacion lambda) el dominio de F

(b) Pruebe que F es Σ-p.r. (Hint: use la Regla U).

Ejercicio 17: Sea $F : \Sigma^+ \to \omega$ dada por $F(\alpha) = \max\{t \in \mathbf{N} : t \leq |\alpha| \text{ y } [\alpha]_1[\alpha]_2...[\alpha]_t \text{ es capicua}\}$. Pruebe que F es Σ-p.r..

Ejercicio 18: Sea $\Sigma = \{@, !\}$. Sea $f : \Sigma^* \to \omega$ dada por:

$$f(\alpha) = \max\{|\beta| : \beta \text{ ocurre en } \alpha \text{ y } \beta \text{ es capicua}\}$$

Pruebe que f es Σ-p.r. (Hint: use la Regla U). Ojo, puede haber un β capicua que ocurra en α de manera "inextensible" pero que $f(\alpha)$ no sea igual a $|\beta|$.

Minimizacion de variable alfabetica

Supongamos que $\Sigma \neq \emptyset$. Sea $\leq$ un orden total sobre Σ. Recordemos que $\leq$ puede ser naturalmente extendido a un orden total sobre Σ^*. Sea $P : D_P \subseteq \omega^n \times \Sigma^{*m} \times \Sigma^* \to \omega$ un predicado. Cuando $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ es tal que existe al menos un $\alpha \in \Sigma^*$ tal que $P(\vec{x}, \vec{\alpha}, \alpha) = 1$, usaremos $\min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha)$ para denotar al menor $\alpha \in \Sigma^*$ tal que $P(\vec{x}, \vec{\alpha}, \alpha) = 1$. Notese que la expresion $\min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha)$ esta definida solo para aquellas $(n+m)$-uplas $(\vec{x}, \vec{\alpha})$ para las cuales hay al menos un α tal que se da $P(\vec{x}, \vec{\alpha}, \alpha) = 1$. Dicho de otra forma, $\min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha)$ no estara definida cuando para cada $\alpha \in \Sigma^*$ se de que $(\vec{x}, \vec{\alpha}, \alpha)$ no pertenece a D_P o $P(\vec{x}, \vec{\alpha}, \alpha) = 0$. Otro detalle importante a tener en cuenta es que la expresion $\min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha)$ no depende de la variable α. Por ejemplo, las expresiones $\min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha)$ y $\min_\beta^\leq P(\vec{x}, \vec{\alpha}, \beta)$ son equivalentes en el sentido que estan definidas en las mismas $(n+m)$-uplas y cuando estan definidas asumen el mismo valor.

Definamos

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

Notese que

$$D_{M^\leq(P)} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists\alpha \in \Sigma^*)\ P(\vec{x}, \vec{\alpha}, \alpha)\}$$
$$M^\leq(P)(\vec{x}, \vec{\alpha}) = \min_\alpha^\leq P(\vec{x}, \vec{\alpha}, \alpha), \text{ para cada } (\vec{x}, \vec{\alpha}) \in D_{M^\leq(P)}$$

Diremos que $M^\leq(P)$ es obtenida por *minimizacion de variable alfabetica* a partir de P.

Algunos ejemplos:

(E1) Sea $\Sigma = \{@, a, b, c, d, e\}$ y sea $\leq$ un orden total sobre Σ. Sea $Dir = \{\alpha_1 \in \Sigma^* : |\alpha_1|_@ = 1\}$ y definamos $U : Dir \to \Sigma^*$ de la siguiente manera

$$U(\alpha_1) = \text{unico } \alpha \text{ tal que } \alpha@ \text{ es tramo inicial de } \alpha_1$$

Sea

$$P = \lambda\alpha_1\alpha[\alpha_1 \in Dir \text{ y } \alpha@ \text{ es tramo inicial de } \alpha_1]$$

Tenemos que

$$\begin{aligned} D_{M^{\leq}(P)} &= \{\alpha_1 \in \Sigma^* : (\exists \alpha \in \Sigma^*)\ P(\alpha_1, \alpha)\} \\ &= \{\alpha_1 \in \Sigma^* : \alpha_1 \in Dir \text{ y } (\exists \alpha \in \Sigma^*)\ \alpha@ \text{ es tramo inicial de } \alpha_1\} \\ &= Dir \end{aligned}$$

y ademas es claro que $M^{\leq}(P)(\alpha_1) = U(\alpha_1)$, para cada $\alpha_1 \in Dir$, por lo cual $M^{\leq}(P) = U$.

(E2) Cuando $n = m = 0$, tenemos que $P : D_P \subseteq \Sigma^* \to \omega$. O sea que

$$M^{\leq}(P) = \lambda[\min_{\alpha}^{\leq} P(\alpha)]$$

Esto nos dice que $D_{M^{\leq}(P)} \subseteq \{\Diamond\}$. Ademas $D_{M^{\leq}(P)} = \{\Diamond\}$ si y solo si $P(\alpha) = 1$, para algun $\alpha \in D_P$, y en tal caso $M^{\leq}(P)(\Diamond) = \min_{\alpha}^{\leq} P(\alpha)$.

Como puede notarse para el caso alfabetico tambien tenemos:

REGLA U: Si tiene una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$ y busca un predicado P tal que $f = M^{\leq}(P)$, intente diseñar P de manera que para cada $(\vec{x}, \vec{\alpha}) \in D_f$ se de que

$$f(\vec{x}, \vec{\alpha}) = \text{ unico } \alpha \in \Sigma^* \text{ tal que } P(\vec{x}, \vec{\alpha}, \alpha)$$

Luego chequee que ademas para cada $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ se de que

$$(\vec{x}, \vec{\alpha}) \in D_f \text{ si y solo si } \exists \alpha \in \Sigma^*\ P(\vec{x}, \vec{\alpha}, \alpha)$$

Ejercicio 19: V o F o I, justifique.

(a) Sea Σ un alfabeto no vacio y sea $\leq$ un orden total sobre Σ. Entonces $p_1^{0,2} = M^{\leq}(\lambda\alpha_1\alpha\,[\alpha = \alpha_1])$

(b) Sea $\leq$ un orden total sobre Σ y sea $P : D_P \subseteq \omega^n \times \Sigma^{*m} \times \Sigma^* \to \omega$ un predicado, entonces

$$D_{M^{\leq}(P)} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : (\exists \alpha \in \Sigma^*)\ (\vec{x}, \vec{\alpha}, \alpha) \in D_P\}$$

(c) Sea $\leq$ un orden total sobre Σ y sea $P : D_P \subseteq \omega^n \times \Sigma^{*m} \times \Sigma^* \to \omega$ un predicado, entonces

$$D_{M^{\leq}(P)} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : P(\vec{x}, \vec{\alpha}, \alpha) \wedge (\forall \beta \in \Sigma^*)_{\beta < \alpha} \neg P(\vec{x}, \vec{\alpha}, \beta)\}$$

$$M^{\leq}(P)(\vec{x}, \vec{\alpha}) = \alpha$$

Ejercicio 20: Sea Σ un alfabeto no vacio y sea $\leq$ un orden total sobre Σ.

(a) Diga que funcion es $M^{\leq}(\lambda\alpha_1\alpha_2\alpha\,[\alpha_1 = \varepsilon])$

(b) Diga que funcion es $M^{\leq}(\lambda\alpha_1\alpha\,[\alpha^2 = \alpha_1 \vee \alpha = \alpha_1])$

(c) Diga que funcion es $M^{\leq}(\lambda x_1\alpha_1\alpha\,[\alpha = \alpha_1])$

(d) Diga que funcion es $M^{\leq}(\lambda x_1\alpha_1\alpha\,[\alpha = \alpha_1 \text{ y } x_1 < 20])$

(e) Diga que funcion es $M^{\leq}(\lambda xy\alpha\beta\,[y \leq |\beta|])$

(f) Diga que funcion es $M^{\leq}(\lambda\alpha_1\alpha_2\alpha\,[\alpha_1 = \alpha\alpha_2])$

Lema de minimizacion acotada de variable alfabetica de predicados Σ-p.r. Aceptaremos sin prueba el siguiente resultado. Su prueba es rutinaria y se basa en el Lema 5 (ver el apunte).

Lema 8 (Lema de minimizacion acotada de variable alfabetica)**.** *Supongamos que* $\Sigma \neq \emptyset$*. Sea* $\leq$ *un orden total sobre* Σ*, sean* $n, m \geq 0$ *y sea* $P : D_P \subseteq \omega^n \times \Sigma^{*m} \times \Sigma^* \to \omega$ *un predicado* Σ*-p.r.. Entonces*

(a) $M^{\leq}(P)$ *es* Σ*-recursiva.*
(b) *Si existe una funcion* Σ*-p.r.* $f : \omega^n \times \Sigma^{*m} \to \omega$ *tal que*

$$\left|M^{\leq}(P)(\vec{x}, \vec{\alpha})\right| = \left|\min_{\alpha}^{\leq} P(\vec{x}, \vec{\alpha}, \alpha)\right| \leq f(\vec{x}, \vec{\alpha}), \text{ para cada } (\vec{x}, \vec{\alpha}) \in D_{M^{\leq}(P)},$$

entonces $M^{\leq}(P)$ *es* Σ*-p.r..*

Ejercicio 21: Pruebe que la funcion U del ejemplo anterior es Σ-p.r.. Por que se elijieron los nombres Dir y U?

Ejercicio 22: Dada una palabra $\alpha \in \Sigma^*$, si hay una palabra ρ tal que $\rho^2 = \alpha$, usaremos $\sqrt{\alpha}$ para denotar a ρ. Notese que la expresion $\sqrt{\alpha}$ tiene sentido o esta definida solo para ciertas palabras. Pruebe que $\lambda\alpha[\sqrt{\alpha}]$ es Σ-p.r..

Ejercicio 23: Sea $\Sigma = \{@, \%, !, \$\}$. Sea $\leq$ un orden total sobre Σ. Sea

$$F = \lambda\alpha_1[\max\{\alpha \in \Sigma^* : \alpha \text{ ocurre en } \alpha_1 \text{ y } \alpha \text{ es capicua}\}]$$

(aqui el maximo es tomado respecto del orden total de Σ^* inducido por $\leq$). Pruebe que F es Σ-p.r..

Ejercicio 24: Sea $F : \Sigma^* \to \Sigma^*$ dada por $F(\alpha) =$ tramo inicial capicua mas largo de α. Pruebe que F es Σ-p.r.. (Hint: use la REGLA U).

Ejercicio 25: Sea $\Sigma = \{@, \bigcirc, \%, \square\}$. Sea $\leq$ un orden total sobre Σ. Sea $L = \{\alpha\beta : \alpha \in \{@, \bigcirc\}^* \text{ y } \beta \in \{\%, \square\}^*\}$. Sea $F : L \to \{\%, \square\}^*$ dada por $F(\alpha\beta) = \beta$, cada vez que $\alpha \in \{@, \bigcirc\}^*$ y $\beta \in \{\%, \square\}^*$.
(a) Explique por que la definicion de F es inambigua
(b) Encuentre un predicado P el cual sea Σ-p.r., cumpla que $D_P = \Sigma^* \times \Sigma^*$ y ademas $F = M^{\leq}(P)$.
(c) Pruebe que F es Σ-p.r..

Ejercicio 26: Sea $\Sigma = \{@, \%, !, \$\}$. Sea $\leq$ un orden total sobre Σ. Sea

$$F = \lambda\beta[\max\{\gamma \in \Sigma^* : \gamma \text{ ocurre en } \beta \text{ y } \gamma \in \{@, \%\}^*]$$

(aqui el maximo es tomado respecto del orden total de Σ^* inducido por $\leq$). Pruebe que F es Σ-p.r..

Ejercicio 27: Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una maquina de Turing y supongamos Q es un alfabeto disjunto con Γ. Usaremos la notacion lambda respecto del alfabeto $\Gamma \cup Q$. Sea $\leq$ un orden total sobre $\Gamma \cup Q$.
(a) Sea $P = \lambda\alpha_1\alpha[\alpha \in Q \text{ y } \alpha \text{ ocurre en } \alpha_1]$. Encuentre $D_{M^{\leq}(P)}$. Que relacion hay entre la funcion $St : Des \to Q$ y $M^{\leq}(P)$
(b) Encuentre un predicado R (modificando P) tal que $M^{\leq}(R) = St$.
(c) Pruebe que St es $(\Gamma \cup Q)$-p.r.
(Puede usar que Des es un conjunto $(\Gamma \cup Q)$-p.r.).

Conjuntos Σ-recursivamente enumerables

Ya que la nocion de funcion Σ-recursiva es el modelo matematico Godeliano del concepto de funcion Σ-efectivamente computable, nos podriamos preguntar entonces cual es el modelo matematico Godeliano del concepto de conjunto Σ-efectivamente enumerable. Si prestamos atencion a la definicion de conjunto Σ-efectivamente enumerable, notaremos que depende de la existencia de ciertas funciones Σ-efectivamente computables por lo cual la siguiente definicion cae de maduro:

Diremos que un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*recursivamente enumerable* cuando sea vacio o haya una funcion $F : \omega \to \omega^n \times \Sigma^{*m}$ tal que $I_F = S$ y $F_{(i)}$ sea Σ-recursiva, para cada $i \in \{1, ..., n + m\}$.

Deberia entonces quedar claro que si el concepto de funcion Σ-recursiva modeliza correctamente al concepto de funcion Σ-efectivamente computable, entonces el concepto de conjunto Σ-recursivamente enumerable recien definido modeliza correctamente al concepto de conjunto Σ-efectivamente enumerable. Sin envargo para probar algunos de los resultados basicos acerca de los conjuntos Σ-recursivamente enumerables, deberemos esperar a tener probada la equivalencia del paradigma Godeliano con el imperativo.

Conjuntos Σ-recursivos

La version Godeliana del concepto de conjunto Σ-efectivamente computable es facil de dar: un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*recursivo* cuando la funcion $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ-recursiva. Todo conjunto Σ-recursivo es Σ-recursivamente enumerable pero esto lo probaremos mas adelante junto con otros resultados basicos sobre conjuntos Σ-r.e., los cuales se prueban usando el mparadigma imperativo. Mas adelante daremos un ejemplo natural de un conjunto que es Σ-r.e. pero el cual no es Σ-recursivo.

Ejercicio 28: Sea Σ un alfabeto finito.

(a) Si $P : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ y $Q : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ son predicados Σ-r., entonces $(P \vee Q)$, $(P \wedge Q)$ y $\neg P$ lo son tambien.

(b) Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-recursivos. Entonces $S_1 \cup S_2$, $S_1 \cap S_2$ y $S_1 - S_2$ son Σ-recursivos

Independencia del alfabeto

El siguiente resultado es conceptualmente muy importante. Su prueba tiene cierta dificultad tecnica por lo cual la omitiremos. Se la puede ver en el apunte.

Teorema 9. *Sean* Σ *y* Γ *alfabetos finitos cualesquiera.*

(a) *Supongamos una funcion f es* Σ*-mixta y* Γ*-mixta, entonces f es* Σ*-recursiva (resp.* Σ*-p.r.) sii f es* Γ*-recursiva (resp.* Γ*-p.r.)*

(b) *Supongamos un conjunto S es* Σ*-mixto y* Γ*-mixto, entonces S es* Σ*-recursivo (resp.* Σ*-r.e.,* Σ*-p.r.) sii S es* Γ*-recursivo (resp.* Γ*-r.e.,* Γ*-p.r.)*

Ejercicio 29: Explique con palabras por que no es obvio el resultado anterior

Ejercicio 30: Que hubiera implicado acerca de la completitud del modelo Godeliano el hecho de que no fuera cierto (a) del teorema anterior?

Ejercicios de examen

Cada resultado teorico que aplique en la resolucion debera enunciarlo por separado detalladamente (en la forma en la que esta en las guias). Para los ejercicios listados a continuacion puede usar sin demostracion que las siguientes funciones son Σ-p.r.: Suc, $Pred$, d_a (con $a \in \Sigma$), $p_j^{n,m}$ (con $n, m, j \in \omega$ y $1 \leq j \leq n+m$), $\lambda xy[x \leq y]$, $\lambda\alpha[|\alpha|]$, $\lambda xy[x = y]$, $\lambda\alpha\beta\,[\alpha = \beta]$, $\lambda xy[x + y]$, $\lambda xy[x.y]$, $\lambda x[x!]$, $\lambda xy\left[x \dot{-} y\right]$, $\lambda x\alpha\,[\alpha^x]$, $\lambda xy\,[x^y]$, $\lambda\alpha[\alpha^R]$, $\lambda i\alpha\,[[\alpha]_i]$, $C_k^{n,m}$, $C_\alpha^{n,m}$ (con $n, m, k \in \omega$ y $\alpha \in \Sigma^*$), $s^{\leq}$, $\#^{\leq}$ y $*^{\leq}$ ($\leq$ un orden total sobre Σ), $\lambda xy[x$ divide a $y]$, $\lambda\alpha\beta\,[\alpha$ ocurre en $\beta]$, $\lambda\alpha\beta\,[\alpha$ es tramo inicial de $\beta]$, $\lambda x[x$ es impar$]$. Tambien puede usar que si $\Gamma \subseteq \Sigma$ entonces Γ^+ y Γ^* son conjuntos Σ-p.r..

(1) Pruebe que la funcion pr es Σ-p.r.. Puede usar sin demostracion que
$$\min_i (i \text{ es primo} \wedge i > A) \leq A! + 1, \text{ para cada } A \in \omega$$
(2) Dados $x, y \in \omega$ tales que $x \neq 0$ o $y \neq 0$, usaremos $mcd(x, y)$ para denotar el maximo comun divisor de x e y, es decir el mayor numero que divide a x y divide a y. Note que la funcion $M = \lambda xy[mcd(x, y)]$ tiene dominio igual a $\omega^2 - \{(0, 0)\}$. Pruebe que M es Σ-p.r..
(3) Sea Σ un alfabeto finito.
   (a) Pruebe que $\lambda xi\,[(x)_i]$ es Σ-p.r. (Hint: repase el significado de la expresion $(x)_i$ y encuentre entonces el dominio de $\lambda xi\,[(x)_i]$ antes de hacer el ejercicio)
   (b) Pruebe que la funcion Lt es Σ-p.r.

   Puede usar que pr es Σ-p.r..
(4) Sea $\Sigma = \{@, !, \%, ?\}$. Sea $f : \Sigma^* \to \omega$ dada por:
$$f(\alpha) = \max\{|\beta| : \beta \text{ ocurre en } \alpha \text{ y } \beta \in \{\%, ?\}^*\}$$
   Pruebe que f es Σ-p.r.. Ojo, puede haber un $\beta \in \{\%, ?\}^*$ que ocurra en α de manera "inextensible" pero que $f(\alpha)$ no sea igual a $|\beta|$.
(5) Sea $\Sigma = \{@, !, \%\}$. Sea $L = \{@\}^* \cup \{!\}^* \cup \{\%\}^*$. Sea $f : \Sigma^* \to \omega$ dada por:
$$f(\alpha) = \max\{|\beta| : \beta \text{ ocurre en } \alpha \text{ y } \beta \in L\}$$
   Pruebe que f es Σ-p.r.. Ojo, puede haber un $\beta \in L$ que ocurra en α de manera "inextensible" pero que $f(\alpha)$ no sea igual a $|\beta|$.
(6) Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda\alpha\beta[\max\{t \in \mathbf{N} : \alpha^t \text{ es tramo inicial de } \beta\}]$.
   (a) Encuentre (segun manda notacion lambda) el dominio de F
   (b) Pruebe que F es Σ-p.r..
(7) Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda\beta[\max\{t \in \mathbf{N} : @^t \text{ es tramo inicial de } \beta\}]$.
   (a) Encuentre (segun manda notacion lambda) el dominio de F
   (b) Pruebe que F es Σ-p.r..
(8) Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda\beta[\max\{t \in \omega : @^t \text{ ocurre en } \beta \text{ y } t \text{ es impar}\}]$.
   (a) Encuentre (segun manda notacion lambda) el dominio de F
   (b) Pruebe que F es Σ-p.r..
(9) Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda x[\max\{t \in \omega : t^2 + 5 \leq x \text{ y } t \text{ es impar}\}]$.
   (a) Encuentre (segun manda notacion lambda) el dominio de F
   (b) Pruebe que F es Σ-p.r..
(10) Sea $\Sigma = \{@, !, \%\}$. Sea $F = \lambda xy[\max\{t \in \omega : x \leq t^2 \leq y\}]$.
   (a) Encuentre (segun manda notacion lambda) el dominio de F

(b) Pruebe que F es Σ-p.r..

(11) Sea $\Sigma = \{@, \bigcirc, \%, \square\}$. Sea $\leq$ un orden total sobre Σ. Sea $L = \{\alpha\beta : \alpha \in \{@, \bigcirc\}^* \text{ y } \beta \in \{\%, \square\}^*\}$. Sea $F : L \to \{\%, \square\}^*$ dada por $F(\alpha\beta) = \beta$, cada vez que $\alpha \in \{@, \bigcirc\}^*$ y $\beta \in \{\%, \square\}^*$.
   (a) Explique por que la definicion de F es inambigua
   (b) Encuentre un predicado P el cual sea Σ-p.r., cumpla que $D_P = \Sigma^* \times \Sigma^*$ y ademas $F = M^{\leq}(P)$.
   (c) Pruebe que F es Σ-p.r..

(12) Sea $\Sigma = \{@, \bigcirc, \%, !, \$\}$. Sea $\leq$ un orden total sobre Σ. Diremos que $\alpha \in \Sigma^*$ es un *auto* si $|\alpha|_{\bigcirc} = 2$. Sea $Au = \{\alpha \in \Sigma^* : \alpha \text{ es un auto}\}$. Obviamente todo auto se escribe univocamente en la forma $\alpha_1 \bigcirc \alpha_2 \bigcirc \alpha_3$, con $\alpha_1, \alpha_2, \alpha_3 \in \{@, \%, !, \$\}^*$.
   (a) Explique el "univocamente"
   (b) Sea $F : Au \to \Sigma^*$ dada por $F(\alpha_1 \bigcirc \alpha_2 \bigcirc \alpha_3) = \alpha_2$, cada vez que $\alpha_1, \alpha_2, \alpha_3 \in \{@, \%, !, \$\}^*$. Pruebe que F es Σ-p.r..

(13) Sea $\Sigma = \{@, \%, !, \$\}$. Sea $\leq$ un orden total sobre Σ. Sea

$$F = \lambda\beta[\max\{\gamma \in \Sigma^* : \gamma \text{ ocurre en } \beta \text{ y } \gamma \in \{@, \%\}^*\}]$$

(aqui el maximo es tomado respecto del orden total de Σ^* inducido por $\leq$). Pruebe que F es Σ-p.r..

(14) Sea $\Sigma = \{@, \%, !, \$\}$. Sea $\leq$ un orden total sobre Σ. Sea

$$F = \lambda\alpha\beta[\max\{\rho \in \{\%, !\}^+ : \rho \text{ es tramo inicial de } \alpha \text{ y } \beta\}]$$

(aqui el maximo es tomado respecto del orden total de Σ^* inducido por $\leq$).
   (a) Encuentre (segun manda notacion lambda) el dominio de F
   (b) Pruebe que F es Σ-p.r..

(15) Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una maquina de Turing y supongamos Q es un alfabeto disjunto con Γ. Usaremos la notacion lambda respecto del alfabeto $\Gamma \cup Q$. Sea $\leq$ un orden total sobre $\Gamma \cup Q$.
   (a) Sea $P = \lambda\alpha_1\alpha[\alpha \in Q \text{ y } \alpha \text{ ocurre en } \alpha_1]$. Encuentre $D_{M^{\leq}(P)}$. Que relacion hay entre la funcion $St : Des \to Q$ y $M^{\leq}(P)$
   (b) Encuentre un predicado R (modificando P) tal que $M^{\leq}(R) = St$.
   (c) Pruebe que St es $(\Gamma \cup Q)$-p.r.

(Puede usar que Des es un conjunto $(\Gamma \cup Q)$-p.r.).

GUIA 7 DE LENGUAJES: EL PARADIGMA IMPERATIVO DE NEUMANN

En esta seccion daremos una modelizacion matematica del concepto de funcion Σ-efectivamente computable utilizando un lenguaje de programacion teorico el cual depende del alfabeto Σ. Lo llamaremos $\mathcal{S}^{\Sigma}$ a dicho lenguaje. Dado que fue el matematico Von Neumann quien contribuyo al desarrollo de la primera computadora de proposito general (es decir a la cual se le pueden hacer correr programas tal como a las computadoras actuales), nos referiremos a este paradigma de computabilidad efectiva como el *paradigma de Von Neumann*. Tambien lo llamaremos algunas veces el *paradigma imperativo*.

Sintaxis de $\mathcal{S}^{\Sigma}$

Recordemos que llamabamos *numerales* a los siguientes simbolos

$$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

Tambien recordemos que Num denotaba el conjunto de los numerales. Sea $Sig : Num^* \to Num^*$ definida de la siguiente manera

$$\begin{aligned} Sig(\varepsilon) &= 1 \\ Sig(\alpha 0) &= \alpha 1 \\ Sig(\alpha 1) &= \alpha 2 \\ Sig(\alpha 2) &= \alpha 3 \\ Sig(\alpha 3) &= \alpha 4 \\ Sig(\alpha 4) &= \alpha 5 \\ Sig(\alpha 5) &= \alpha 6 \\ Sig(\alpha 6) &= \alpha 7 \\ Sig(\alpha 7) &= \alpha 8 \\ Sig(\alpha 8) &= \alpha 9 \\ Sig(\alpha 9) &= Sig(\alpha)0 \end{aligned}$$

Definamos $Dec : \omega \to Num^*$ de la siguiente manera

$$\begin{aligned} Dec(0) &= \varepsilon \\ Dec(n+1) &= Sig(Dec(n)) \end{aligned}$$

Notese que para $n \in \mathbf{N}$, la palabra $Dec(n)$ es la notacion usual decimal de n. Para hacer mas agil la notacion escribiremos $\bar{n}$ en lugar de $Dec(n)$.

La sintaxis de $\mathcal{S}^{\Sigma}$ sera dada utilizando solo simbolos del alfabeto $\Sigma \cup \Sigma_p$, donde

$$\Sigma_p = Num \cup \left\{\leftarrow, +, \dot{-}, ., \neq, ^\frown, \varepsilon, \mathrm{N}, \mathrm{K}, \mathrm{P}, \mathrm{L}, \mathrm{I}, \mathrm{F}, \mathrm{G}, \mathrm{O}, \mathrm{T}, \mathrm{B}, \mathrm{E}, \mathrm{S}\right\}.$$

Cabe aclarar que la palabra de longitud 0 no es un elemento de Σ_p sino que la letra griega ε que usualmente denota esta palabra, lo es. Tambien notese que en Σ_p hay simbolos que a veces representan operaciones como por ejemplo $+$, $^\frown$ y $\dot{-}$, pero

deberia quedar claro que en Σ_p estan los simbolos $+$, $^\frown$ y $\dot{-}$ y no las operaciones que ellos usualmente denotan.

Las palabras de la forma $\mathrm{N}\bar{k}$ con $k \in \mathbf{N}$, son llamadas *variables numericas de* $\mathcal{S}^\Sigma$. Las palabras de la forma $\mathrm{P}\bar{k}$ con $k \in \mathbf{N}$, son llamadas *variables alfabeticas de* $\mathcal{S}^\Sigma$. Las palabras de la forma $\mathrm{L}\bar{k}$ con $k \in \mathbf{N}$, son llamadas *labels de* $\mathcal{S}^\Sigma$. Una *instruccion basica de* $\mathcal{S}^\Sigma$ es una palabra de $(\Sigma \cup \Sigma_p)^*$ la cual es de alguna de las siguientes formas

$\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} + 1$
$\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} \dot{-} 1$
$\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n}$
$\mathrm{N}\bar{k} \leftarrow 0$
$\mathrm{P}\bar{k} \leftarrow \mathrm{P}\bar{k}.a$
$\mathrm{P}\bar{k} \leftarrow {}^\frown\mathrm{P}\bar{k}$
$\mathrm{P}\bar{k} \leftarrow \mathrm{P}\bar{n}$
$\mathrm{P}\bar{k} \leftarrow \varepsilon$
IF $\mathrm{N}\bar{k} \neq 0$ GOTO $\mathrm{L}\bar{n}$
IF $\mathrm{P}\bar{k}$ BEGINS a GOTO $\mathrm{L}\bar{n}$
GOTO $\mathrm{L}\bar{n}$
SKIP

donde $a \in \Sigma$ y $k, n \in \mathbf{N}$. Como puede observarse para que las instrucciones basicas sean mas lejibles usamos espacios entre ciertos simbolos. Por ejemplo, hemos escrito $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} + 1$ pero en realidad nos referimos a la palabra

$$\mathrm{N}\bar{k}{\leftarrow}\mathrm{N}\bar{k}{+}1$$

cuya longitud es $2\left|\bar{k}\right| + 5$. Otro ejemplo, hemos escrito IF $\mathrm{P}\bar{k}$ BEGINS a GOTO $\mathrm{L}\bar{n}$ pero en realidad nos referiamos a la palabra IF$\mathrm{P}\bar{k}$BEGINSaGOTO$\mathrm{L}\bar{n}$ cuya longitud es $\left|\bar{k}\right| + |\bar{n}| + 15$.

Una *instruccion de* $\mathcal{S}^\Sigma$ es ya sea una instruccion basica de $\mathcal{S}^\Sigma$ o una palabra de la forma αI, donde $\alpha \in \{\mathrm{L}\bar{n} : n \in \mathbf{N}\}$ y I es una instruccion basica de $\mathcal{S}^\Sigma$. Usaremos Ins^Σ para denotar el conjunto de todas las instrucciones de $\mathcal{S}^\Sigma$. Cuando la instruccion I es de la forma $\mathrm{L}\bar{n}J$ con J una instruccion basica, diremos que $\mathrm{L}\bar{n}$ es el *label* de I.

Ejercicio 1: V o F o I, justificar

(a) Para cada $n \in \mathbf{N}$, se tiene que $\bar{n} \in \omega$
(b) Si $k \in \mathbf{N}$, entonces $\mathrm{P}\bar{k}\leftarrow$ es una instruccion de $\mathcal{S}^\Sigma$
(c) Sea Σ un alfabeto. Entonces Ins^Σ es un conjunto Σ-mixto
(d) $Ti(\mathrm{Ins}^\Sigma) = \mathrm{PALABRA}$
(e) Si $I \in \mathrm{Ins}^\Sigma$, entonces $Ti(I) = \mathrm{PALABRA}$
(f) Si I es una instruccion de $\mathcal{S}^\Sigma$ y $n \in \mathbf{N}$ es tal que $\mathrm{L}\bar{n}$ es tramo inicial de I, entonces $\mathrm{L}\bar{n}$ es el label de I.

Damos a continuacion, a modo de ejemplo, la interpretacion intuitiva asociada a ciertas instrucciones basicas de $\mathcal{S}^{\Sigma}$:

INSTRUCCION : $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} \dot{-} 1$

INTERPRETACION : Si el contenido de $\mathrm{N}\bar{k}$ es 0 dejarlo sin modificar; en caso contrario disminuya en 1 el contenido de $\mathrm{N}\bar{k}$

INSTRUCCION : $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n}$

INTERPRETACION : Copiar en $\mathrm{N}\bar{k}$ el contenido de $\mathrm{N}\bar{n}$ sin modificar el contenido de $\mathrm{N}\bar{n}$

INSTRUCCION : $\mathrm{P}\bar{k} \leftarrow^{\frown} \mathrm{P}\bar{k}$

INTERPRETACION : Si el contenido de $\mathrm{P}\bar{k}$ es ε dejarlo sin modificar; en caso contrario remueva el 1er simbolo del contenido de $\mathrm{P}\bar{k}$

INSTRUCCION : $\mathrm{P}\bar{k} \leftarrow \mathrm{P}\bar{k}.a$

INTERPRETACION : Modificar el contenido de $\mathrm{P}\bar{k}$ agregandole el simbolo a a la derecha

INSTRUCCION : IF $\mathrm{P}\bar{k}$ BEGINS a GOTO $\mathrm{L}\bar{m}$

INTERPRETACION : Si el contenido de $\mathrm{P}\bar{k}$ comiensa con a, ejecute la primer instruccion con label $\mathrm{L}\bar{m}$; en caso contrario ejecute la siguiente instruccion

Un *programa de* $\mathcal{S}^{\Sigma}$ es una palabra de la forma

$$I_1 I_2 ... I_n$$

donde $n \geq 1$, $I_1, ..., I_n \in \mathrm{Ins}^{\Sigma}$ y ademas se cumple la siguiente propiedad, llamada *la ley de los GOTO*,

(G) Para cada $i \in \{1, ..., n\}$, si GOTOL$\bar{m}$ es un tramo final de I_i, entonces existe $j \in \{1, ..., n\}$ tal que I_j tiene label $\mathrm{L}\bar{m}$

Usaremos Pro^{Σ} para denotar el conjunto de todos los programas de $\mathcal{S}^{\Sigma}$. Como es usual cuando escribamos un programa lo haremos linea por linea, con la finalidad de que sea mas lejible. Por ejemplo, escribiremos

$$\begin{array}{ll} \mathrm{L2} & \mathrm{N12} \leftarrow \mathrm{N12} \dot{-} 1 \\ & \mathrm{P1} \leftarrow {}^{\frown}\mathrm{P1} \\ & \text{IF N12} \neq 0 \text{ GOTO L2} \end{array}$$

en lugar de

$$\mathrm{L2N12} \leftarrow \mathrm{N12} \dot{-} 1\mathrm{P1} \leftarrow {}^{\frown}\mathrm{P1IFN12} \neq 0\mathrm{GOTOL2}$$

Un importante resultado es el siguiente lema que garantiza que los programas pueden ser parseados en forma unica como concatenacion de instrucciones. Lo aceptaremos sin demostracion, en el apunte esta probado.

Lema 1. *Sea Σ un alfabeto finito. Se tiene que:*

(a) *Si $I_1 ... I_n = J_1 ... J_m$, con $I_1, ..., I_n, J_1, ..., J_m \in \mathrm{Ins}^{\Sigma}$, entonces $n = m$ y $I_j = J_j$ para cada $j \geq 1$.*

(b) *Si $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$, entonces existe una unica sucesion de instrucciones $I_1, ..., I_n$ tal que $\mathcal{P} = I_1 ... I_n$*

(b) del lema anterior nos dice que dado un programa $\mathcal{P}$, tenemos univocamente determinados $n(\mathcal{P}) \in \mathbf{N}$ y $I_1^{\mathcal{P}}, ..., I_{n(\mathcal{P})}^{\mathcal{P}} \in \mathrm{Ins}^{\Sigma}$ tales que $\mathcal{P} = I_1^{\mathcal{P}} ... I_{n(\mathcal{P})}^{\mathcal{P}}$. Definamos tambien

$$I_i^{\mathcal{P}} = \varepsilon$$

cuando $i = 0$ o $i > n(\mathcal{P})$. Notese que las expresiones $n(\alpha)$ y I_i^{α} estan definidas solo cuando α es un programa (y i es un elemento de ω), es decir, cierta palabra del alfabeto $\Sigma \cup \Sigma_p$. O sea que cuando usemos notacion lambda que involucre dichas expresiones, el alfabeto respecto del cual usaremos dicha notacion sera $\Sigma \cup \Sigma_p$. Esto nos dice entonces que $\lambda\alpha[n(\alpha)]$ tiene dominio igual a $\mathrm{Pro}^{\Sigma} \subseteq (\Sigma \cup \Sigma_p)^*$ y $\lambda i\alpha[I_i^{\alpha}]$ tiene dominio igual a $\omega \times \mathrm{Pro}^{\Sigma}$. Para hacer mas sugestiva la notacion a veces escribiremos $\lambda\mathcal{P}[n(\mathcal{P})]$ y $\lambda i\mathcal{P}[I_i^{\mathcal{P}}]$ en lugar de $\lambda\alpha[n(\alpha)]$ y $\lambda i\alpha[I_i^{\alpha}]$

Ejercicio 2: V o F o I, justificar

(a) $\mathrm{Ins}^{\Sigma} \subseteq \mathrm{Pro}^{\Sigma}$

(b) $\mathrm{Ins}^{\Sigma} \cap \mathrm{Pro}^{\Sigma} = \emptyset$

(c) $\lambda i\mathcal{P}\left[I_i^{\mathcal{P}}\right]$ tiene dominio igual a $\{(i, \mathcal{P}) \in \mathbf{N} \times \mathrm{Pro}^{\Sigma} : i \leq n(\mathcal{P})\}$

(d) Sea Σ un alfabeto. Si $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$, entonces $\mathcal{P} \in \overbrace{\mathrm{Ins}^{\Sigma} \times \cdots \times \mathrm{Ins}^{\Sigma}}^{n(\mathcal{P}) \text{ veces}}$

Ejercicio 3: Si $\mathcal{P}_1, \mathcal{P}_2 \in \mathrm{Pro}^{\Sigma}$ y $\mathcal{P}_1\mathcal{P}_1 = \mathcal{P}_2\mathcal{P}_2$, entonces $\mathcal{P}_1 = \mathcal{P}_2$

Semantica de $\mathcal{S}^{\Sigma}$

Para definir la semantica nos sera util la funcion $Bas : \mathrm{Ins}^{\Sigma} \to (\Sigma \cup \Sigma_p)^*$, dada por

$$Bas(I) = \begin{cases} J & \text{si } I \text{ es de la forma } \mathrm{L}\bar{k}J \text{ con } J \in \mathrm{Ins}^{\Sigma} \\ I & \text{caso contrario} \end{cases}$$

Recordemos que para una palabra α definiamos

$$\curvearrowright\alpha = \begin{cases} [\alpha]_2 ... [\alpha]_{|\alpha|} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases}$$

Definamos

$$\omega^{[\mathbf{N}]} = \left\{(s_1, s_2, ...) \in \omega^{\mathbf{N}} : \text{ hay } n \in \mathbf{N} \text{ tal que } s_i = 0, \text{ para } i \geq n\right\}$$

$$\Sigma^{*[\mathbf{N}]} = \left\{(\sigma_1, \sigma_2, ...) \in \Sigma^{*\mathbf{N}} : \text{ hay } n \in \mathbf{N} \text{ tal que } \sigma_i = \varepsilon, \text{ para } i \geq n\right\}.$$

Asumiremos siempre que en una computacion via un programa de $\mathcal{S}^{\Sigma}$, todas exepto una cantidad finita de las variables numericas tienen el valor 0 y todas exepto una cantiad finita de las variables alfabeticas tienen el valor ε. Esto no quita generalidad a nuestra modelizacion del funcionamiento de los programas ya que todo programa envuelve una cantidad finita de variables.

Un *estado* es un par

$$(\vec{s}, \vec{\sigma}) = ((s_1, s_2, ...), (\sigma_1, \sigma_2, ...)) \in \omega^{[\mathbf{N}]} \times \Sigma^{*[\mathbf{N}]}.$$

Si $i \geq 1$, entonces diremos que s_i es el *contenido* o *valor* de la variable $\mathrm{N}\bar{\imath}$ en el estado $(\vec{s}, \vec{\sigma})$ y σ_i es el *contenido* o *valor* de la variable $\mathrm{P}\bar{\imath}$ en el estado $(\vec{s}, \vec{\sigma})$. Es decir, intuitivamente hablando, un estado es un par de infinituplas que contiene la informacion de que valores tienen alojados las distintas variables.

Imaginemos que corremos un programa $\mathcal{P}$ partiendo de un estado inicial $(\vec{s}, \vec{\sigma})$. Por supuesto la primera instruccion a realizar sera $I_1^{\mathcal{P}}$ pero, dado que $I_1^{\mathcal{P}}$ puede ser de tipo GOTO, la segunda instruccion que realizaremos puede no ser $I_2^{\mathcal{P}}$. Es decir en cada paso iremos decidiendo en funcion de la instruccion ejecutada cual es la siguiente instruccion a realizar. O sea que mientras corremos $\mathcal{P}$, en cada paso la informacion importante a tener en cuenta es, por una parte, cuales son los valores que tienen cada una de las variables y, por otra parte, cual es la instruccion que nos tocara realizar a continuacion. Esto da lugar al concepto de descripcion instantanea, a saber, un objeto matematico que describe en un instante dado de la computacion cuales son los valores de las variables y cual es la instruccion que se debe realizar en el instante siguiente. Mas formalmente una *descripcion instantanea* es una terna $(i, \vec{s}, \vec{\sigma})$ tal que $(\vec{s}, \vec{\sigma})$ es un estado e $i \in \omega$. Es decir que $\omega \times \omega^{[\mathbf{N}]} \times \Sigma^{*[\mathbf{N}]}$ es el conjunto formado por todas las descripciones instantaneas. Intuitivamente hablando, cuando $i \in \{1, ..., n(\mathcal{P})\}$, la descripcion instantanea $(i, \vec{s}, \vec{\sigma})$ nos dice que las variables estan en el estado $(\vec{s}, \vec{\sigma})$ y que la instruccion que *debemos realizar* es $I_i^{\mathcal{P}}$. Dado que sera conveniente para simplificar el tratamiento formal, nos abstraeremos un poco y cuando $i = 0$ o $i > n(\mathcal{P})$ pensaremos tambien que la descripcion instantanea $(i, \vec{s}, \vec{\sigma})$ nos dice que las variables estan en el estado $(\vec{s}, \vec{\sigma})$ y que debemos realizar $I_i^{\mathcal{P}} = \varepsilon$ (aunque por supuesto no podremos realizarla ya que no es una instruccion).

Dado un programa $\mathcal{P}$ definiremos a continuacion una funcion

$$S_{\mathcal{P}} : \omega \times \omega^{[\mathbf{N}]} \times \Sigma^{*[\mathbf{N}]} \to \omega \times \omega^{[\mathbf{N}]} \times \Sigma^{*[\mathbf{N}]}$$

la cual le asignara a una descripcion instantanea $(i, \vec{s}, \vec{\sigma})$ la *descripcion instantanea sucesora de* $(i, \vec{s}, \vec{\sigma})$ *con respecto a* $\mathcal{P}$. Cuando $i \in \{1, ..., n(\mathcal{P})\}$, intuitivamente hablando, $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma})$ sera la descripcion instantanea que resulta luego de realizar $I_i^{\mathcal{P}}$ estando en el estado $(\vec{s}, \vec{\sigma})$. Cuando $i = 0$ o $i > n(\mathcal{P})$ definiremos $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i, \vec{s}, \vec{\sigma})$, lo cual es bastante intuitivo ya que si estamos en estado $(\vec{s}, \vec{\sigma})$ y debemos realizar $I_i^{\mathcal{P}} = \varepsilon$, dado que ε no es una instruccion y por lo tanto no la podremos realizar, seguiremos en el mismo estado y teniendo que realizar $I_i^{\mathcal{P}}$.

Para darle una semantica mas unificada al concepto de descripcion instantanea sucesora debemos crear un nuevo verbo. El verbo "realizarp". Dada una actividad A, diremos que un individuo P *realizap* la actividad A, si P realiza A, en caso de que pueda hacerlo. O sea realizarp una actividad es realizarla si se puede.

Para dar otro ejemplo de este tipo de verbos, consideremos el verbo "comprarp", es decir "comprar si se puede". Un hijo le pide a su padre que le compre un determinado juguete y el padre le dice "si, hijo mio, te lo voy a comprarp". Luego el padre es despedido de su empleo y su cituacion economica hace que no le sea posible comprar dicho juguete. Sin envargo el padre no mintio ya que si bien no compro dicho juguete, él si lo comprop.

Con este verbo podemos describir intuitivamente $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma})$:

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = \text{descripcion instantanea que resulta}$$
$$\text{luego de realizarp } I_i^{\mathcal{P}}, \text{ estando en estado } (\vec{s}, \vec{\sigma})$$

Ahora si, daremos la definicion matematica de $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma})$, segun se den distintos casos posibles.

Caso $i \notin \{1, ..., n(\mathcal{P})\}$. Entonces $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i, \vec{s}, \vec{\sigma})$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} \dot{-} 1$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i + 1, (s_1, ..., s_{k-1}, s_k \dot{-} 1, s_{k+1}, ...), \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{k} + 1$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, (s_1, ..., s_{k-1}, s_k + 1, s_{k+1}, ...), \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, (s_1, ..., s_{k-1}, s_n, s_{k+1}, ...), \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{N}\bar{k} \leftarrow 0$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, (s_1, ..., s_{k-1}, 0, s_{k+1}, ...), \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \text{IF } \mathrm{N}\bar{k} \neq 0 \text{ GOTO } \mathrm{L}\bar{m}$. Entonces tenemos dos subcasos.
Subcaso a. El valor de $\mathrm{N}\bar{k}$ en $(\vec{s}, \vec{\sigma})$ es 0. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, \vec{\sigma})$$

Subcaso b. El valor de $\mathrm{N}\bar{k}$ en $(\vec{s}, \vec{\sigma})$ es no nulo. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (\min\{l : I_l^{\mathcal{P}} \text{ tiene label } \mathrm{L}\bar{m}\}, \vec{s}, \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{P}\bar{k} \leftarrow {}^{\frown}\mathrm{P}\bar{k}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, (\sigma_1, ..., \sigma_{k-1}, {}^{\frown}\sigma_k, \sigma_{k+1}, ...))$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{P}\bar{k} \leftarrow \mathrm{P}\bar{k}.a$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, (\sigma_1, ..., \sigma_{k-1}, \sigma_k a, \sigma_{k+1}, ...))$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{P}\bar{k} \leftarrow \mathrm{P}\bar{n}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, (\sigma_1, ..., \sigma_{k-1}, \sigma_n, \sigma_{k+1}, ...))$$

Caso $Bas(I_i^{\mathcal{P}}) = \mathrm{P}\bar{k} \leftarrow \varepsilon$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, (\sigma_1, ..., \sigma_{k-1}, \varepsilon, \sigma_{k+1}, ...))$$

Caso $Bas(I_i^{\mathcal{P}}) = \text{IF } \mathrm{P}\bar{k} \text{ BEGINS } a \text{ GOTO } \mathrm{L}\bar{m}$. Entonces tenemos dos subcasos.
Subcaso a. El valor de $\mathrm{P}\bar{k}$ en $(\vec{s}, \vec{\sigma})$ comiensa con a. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (\min\{l : I_l^{\mathcal{P}} \text{ tiene label } \mathrm{L}\bar{m}\}, \vec{s}, \vec{\sigma})$$

Subcaso b. El valor de $\mathrm{P}\bar{k}$ en $(\vec{s}, \vec{\sigma})$ no comiensa con a. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \text{GOTO } \mathrm{L}\bar{m}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (\min\{l : I_l^{\mathcal{P}} \text{ tiene label } \mathrm{L}\bar{m}\}, \vec{s}, \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) = \text{SKIP}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i+1, \vec{s}, \vec{\sigma})$$

La computacion partiendo de un estado. Dado un programa $\mathcal{P}$ y un estado $(\vec{s}, \vec{\sigma})$ a la infinitupla

$$((1, \vec{s}, \vec{\sigma}), S_{\mathcal{P}}(1, \vec{s}, \vec{\sigma}), S_{\mathcal{P}}(S_{\mathcal{P}}(1, \vec{s}, \vec{\sigma})), S_{\mathcal{P}}(S_{\mathcal{P}}(S_{\mathcal{P}}(1, \vec{s}, \vec{\sigma}))), ...)$$

la llamaremos la *computacion de $\mathcal{P}$ partiendo del estado* $(\vec{s}, \vec{\sigma})$. Diremos que

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

es la *descripcion instantanea obtenida luego de t pasos, partiendo del estado* $(\vec{s}, \vec{\sigma})$. Si

$$\overbrace{S_{\mathcal{P}}(...S_{\mathcal{P}}(S_{\mathcal{P}}}^{t \text{ veces}}(1, \vec{s}, \vec{\sigma}))...) = (j, \vec{u}, \vec{\eta})$$

diremos que $(\vec{u}, \vec{\eta})$ es el *estado obtenido luego de t pasos, partiendo del estado* $(\vec{s}, \vec{\sigma})$.

Es claro que en la infinitupla de mas arriba esta toda la informacion de la "corrida" del programa $\mathcal{P}$ cuando partimos del estado $(\vec{s}, \vec{\sigma})$. Veamos un ejemplo. Sea $\Sigma = \{▲, \#\}$ y sea $\mathcal{P}$ el siguiente programa

```
L3  N4 ← N4 + 1
    P1 ←  ↷P1
    IF P1 BEGINS ▲ GOTO L3
    P3 ← P3.#
```

Supongamos que tomamos $(\vec{s}, \vec{\sigma})$ igual al estado

$$((2, 1, 0, 5, 3, 0, 0, 0, ...), (\#▲\#\#, \varepsilon, ▲▲, \#▲, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

Tendremos entonces que la computacion de $\mathcal{P}$ partiendo del estado $(\vec{s}, \vec{\sigma})$ es la infinitupla formada por la siguiente sucesion (de arriba hacia abajo) de descripciones

instantaneas:

$$(1,(2,1,0,5,3,0,0,0,...),(\#\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4}+1 \text{ obtenemos}$$
$$(2,(2,1,0,6,3,0,0,0,...),(\#\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{P1} \leftarrow {}^{\frown}\text{P1} \text{ obtenemos}$$
$$(3,(2,1,0,6,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_3^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(1,(2,1,0,6,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4}+1 \text{ obtenemos}$$
$$(2,(2,1,0,7,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{P1} \leftarrow {}^{\frown}\text{P1} \text{ obtenemos}$$
$$(3,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_3^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(4,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{realizando } I_4^{\mathcal{P}} = \text{P3} \leftarrow \text{P3}.\# \text{ obtenemos}$$
$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{intentando realizar } I_5^{\mathcal{P}} = \varepsilon \text{ obtenemos}$$
$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{intentando realizar } I_5^{\mathcal{P}} = \varepsilon \text{ obtenemos}$$
$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\text{intentando realizar } I_5^{\mathcal{P}} = \varepsilon \text{ obtenemos}$$
$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$
$$\vdots$$

Notese que en este caso es natural decir que el programa $\mathcal{P}$ se detiene, partiendo del estado inicial dado ya que llega a un punto en el que queda intentando realizar $I_{n(\mathcal{P})+1}^{\mathcal{P}}$ lo cual no es una instruccion. Veamos un ejemplo de no detencion. Sea $\mathcal{Q}$ el siguiente programa

$$\begin{array}{ll} \text{L3} & \text{N4} \leftarrow \text{N4}+1 \\ & \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3} \end{array}$$

Supongamos que tomamos $(\vec{s},\vec{\sigma})$ igual al estado

$$((2,1,0,5,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

Tendremos entonces que la computacion de $\mathcal{Q}$ partiendo del estado $(\vec{s}, \vec{\sigma})$ es la siguiente infinitupla (de arriba hacia abajo) de descripciones instantaneas:

$$(1, (2, 1, 0, 5, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$
$$(2, (2, 1, 0, 6, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(1, (2, 1, 0, 6, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$
$$(2, (2, 1, 0, 7, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(1, (2, 1, 0, 7, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$
$$(2, (2, 1, 0, 8, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(1, (2, 1, 0, 8, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$
$$(2, (2, 1, 0, 9, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$
$$(1, (2, 1, 0, 9, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$
$$\vdots$$

Notese que en este caso, es claro que el programa $\mathcal{Q}$ no se detiene partiendo del estado inicial dado ya que sigue indefinidamente realizando instrucciones.

Ejercicio 4: V o F o I, justificar

(a) Si $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i, \vec{s}, \vec{\sigma})$, entonces $i \notin \{1, ..., n(\mathcal{P})\}$

(b) Sea $\mathcal{P} \in \text{Pro}^{\Sigma}$ y sea d una descripcion instantanea cuya primer coordenada es i. Si $I_i^{\mathcal{P}} = \text{N2} \leftarrow \text{N2} + 1$, entonces

$$S_{\mathcal{P}}(d) = (i + 1, (\text{N1}, Suc(\text{N2}), \text{N3}, \text{N4}, ...), (\text{P1}, \text{P2}, \text{P3}, \text{P4}, ...))$$

(c) Sea $\mathcal{P} \in \text{Pro}^{\Sigma}$ y sea d una descripcion instantanea cuya primer coordenada es i. Si $I_i^{\mathcal{P}} = \text{N2} \leftarrow 0$, entonces

$$S_{\mathcal{P}}(d) = (i + 1, (\text{N1}, 0, \text{N3}, \text{N4}, ...), (\text{P1}, \text{P2}, \text{P3}, \text{P4}, ...))$$

(d) Sea $\mathcal{P} \in \text{Pro}^{\Sigma_p}$ y sea $(i, \vec{s}, \vec{\sigma})$ una descripcion instantanea. Supongamos $\sigma_3 = \text{GOTO}$. Si $I_i^{\mathcal{P}} = \text{L6 IF P3 BEGINS G GOTO L6}$, entonces $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (i, \vec{s}, \vec{\sigma})$

(e) Sea $\mathcal{P} \in \text{Pro}^{\Sigma}$, sea $a \in \Sigma$ y sea $(i, \vec{s}, \vec{\sigma})$ una descripcion instantanea. Si $Bas(I_i^{\mathcal{P}}) = \text{IF P3 BEGINS } a \text{ GOTO L6}$ y $[\text{P3}]_1 = a$, entonces $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (j, \vec{s}, \vec{\sigma})$, donde j es el menor numero l tal que $I_l^{\mathcal{P}}$ tiene label L6

Definicion matematica de detencion. Ahora definiremos matematicamente el concepto de detencion. Cuando la primer coordenada de

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

sea igual a $n(\mathcal{P})+1$, diremos que $\mathcal{P}$ *se detiene (luego de t pasos), partiendo desde el estado* $(\vec{s},\vec{\sigma})$. Si ninguna de las primeras coordenadas en la computacion

$$((1,\vec{s},\vec{\sigma}), S_{\mathcal{P}}(1,\vec{s},\vec{\sigma}), S_{\mathcal{P}}(S_{\mathcal{P}}(1,\vec{s},\vec{\sigma})), S_{\mathcal{P}}(S_{\mathcal{P}}(S_{\mathcal{P}}(1,\vec{s},\vec{\sigma}))), ...)$$

es igual a $n(\mathcal{P})+1$, diremos que $\mathcal{P}$ *no se detiene partiendo del estado* $(\vec{s},\vec{\sigma})$. Como puede observarse los programas de $\mathcal{S}^{\Sigma}$ tienen una sola manera de detenerse, i.e. siempre que se detienen lo hacen habiendo realizado la ultima de sus instrucciones e intentando realizar la instruccion siguiente a su ultima instruccion.

Notese que en los conceptos antes definidos por "1 paso" entendemos "aplicar una ves la funcion $S_{\mathcal{P}}$" y esto no siempre involucra "realizar una instruccion" ya que podria pasar que aplicaramos $S_{\mathcal{P}}$ a una descripcion instantanea cuya primer coordenada es $n(\mathcal{P})+1$. Podemos redondear esta idea diciendo que "1 paso" equivale a "realizarp una instrucion", donde tal como se lo explico antes "realizarp" significa "realizar si se puede".

Funciones Σ-computables

Ahora que hemos definido matematicamente la semantica de $\mathcal{S}^{\Sigma}$ estamos en condiciones de definir el concepto de funcion Σ-computable, el cual sera una modelizacion matematica del concepto de funcion Σ-efectivamente computable. Intuitivamente hablando una funcion sera Σ-computable cuando haya un programa que la compute. Para precisar este concepto nos sera util la siguiente notacion. Dados $x_1,...,x_n \in \omega$ y $\alpha_1,...,\alpha_m \in \Sigma^*$, con $n,m \in \omega$, usaremos

$$\|x_1,...,x_n,\alpha_1,...,\alpha_m\|$$

para denotar el estado

$$((x_1,...,x_n,0,...),(\alpha_1,...,\alpha_m,\varepsilon,...))$$

Esta notacion requiere aclarar un poco como debe interpretarse en los casos limite, es decir cuando alguno de los numeros n,m es igual a 0. Notese que por ejemplo

$$\|x\| = ((x,0,...),(\varepsilon,...))$$

(es el caso $n=1$ y $m=0$). Tambien

$$\|\alpha\| = ((0,...),(\alpha,\varepsilon,...))$$

(es el caso $n=0$ y $m=1$). En el caso $n=m=0$ pensaremos que $x_1,...,x_n,\alpha_1,...,\alpha_m$ se transforma en $\Diamond$ por lo que se obtiene

$$\|\Diamond\| = ((0,...),(\varepsilon,...))$$

Ademas es claro que

$$\|x_1,...,x_n,\alpha_1,...,\alpha_m\| = \left\| x_1,...,x_n,\overbrace{0,...,0}^{i},\alpha_1,...,\alpha_m,\overbrace{\varepsilon,...,\varepsilon}^{j} \right\|$$

cualesquiera sean $i,j \in \omega$.

Dado $\mathcal{P} \in \text{Pro}^{\Sigma}$, definamos para cada par $n, m \geq 0$, la funcion $\Psi_{\mathcal{P}}^{n,m,\#}$ de la siguiente manera:

$$D_{\Psi_{\mathcal{P}}^{n,m,\#}} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : \mathcal{P} \text{ termina, partiendo del estado } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|\}$$

$$\Psi_{\mathcal{P}}^{n,m,\#}(\vec{x}, \vec{\alpha}) = \text{valor de N1 en el estado obtenido cuando } \mathcal{P} \text{ termina, partiendo de } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$$

Analogamente definamos la funcion $\Psi_{\mathcal{P}}^{n,m,*}$ de la siguiente manera:

$$D_{\Psi_{\mathcal{P}}^{n,m,*}} = \{(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} : \mathcal{P} \text{ termina, partiendo del estado } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|\}$$

$$\Psi_{\mathcal{P}}^{n,m,*}(\vec{x}, \vec{\alpha}) = \text{valor de P1 en el estado obtenido cuando } \mathcal{P} \text{ termina, partiendo de } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$$

Ahora si, daremos la definicion precisa de funcion Σ-computable. Una funcion Σ-mixta $f : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ sera llamada Σ-*computable* si hay un programa $\mathcal{P}$ de $\mathcal{S}^{\Sigma}$ tal que $f = \Psi_{\mathcal{P}}^{n,m,\#}$. En tal caso diremos que la funcion f es *computada* por $\mathcal{P}$. Analogamente una funcion Σ-mixta $f : S \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$ sera llamada Σ-*computable* si hay un programa $\mathcal{P}$ de $\mathcal{S}^{\Sigma}$ tal que $f = \Psi_{\mathcal{P}}^{n,m,*}$. En tal caso diremos que la funcion f es *computada* por $\mathcal{P}$.

Algunos ejemplos:

(E1) El programa

```
    IF N1 ≠ 0 GOTO L1
    GOTO L2
L1  N1 ← N1 ∸ 1
```

computa la funcion $Pred$. Note que este programa tambien computa las funciones $Pred \circ p_1^{n,m}$, para $n \geq 1$ y $m \geq 0$.

(E2) Sea $\Sigma = \{\clubsuit, \triangle\}$. El programa

```
L3  IF P2 BEGINS ♣ GOTO L1
    IF P2 BEGINS △ GOTO L2
    GOTO L4
L1  P2 ← ⌢P2
    P1 ← P1.♣
    GOTO L3
L2  P2 ← ⌢P2
    P1 ← P1.△
    GOTO L3
L4  SKIP
```

computa la funcion $\lambda\alpha\beta\,[\alpha\beta]$.

(E3) Notese que $D_{\Psi_{\mathcal{P}}^{0,0,\#}} = \{\Diamond\}$ en caso que $\mathcal{P}$ termina partiendo del estado $((0, 0, ...), (\varepsilon, \varepsilon, ...))$ y en tal caso

$$\Psi_{\mathcal{P}}^{0,0,\#}(\Diamond) = \text{valor de N1 en el estado obtenido cuando } \mathcal{P} \text{ termina, partiendo de } ((0, 0, ...), (\varepsilon, \varepsilon, ...))$$

En caso de que $\mathcal{P}$ no termina partiendo del estado $((0, 0, ...), (\varepsilon, \varepsilon, ...))$ tenemos que $D_{\Psi_{\mathcal{P}}^{0,0,\#}} = \emptyset$.

Por supuesto para que el concepto de funcion Σ-computable tenga chance de ser una modelizacion adecuada del concepto de funcion Σ-efectivamente computable, tiene que ser cierto el siguiente resultado.

Proposición 2 (Leibniz vence a Neumann)**.** *Si f es Σ-computable, entonces f es Σ-efectivamente computable.*

Ejercicio 5: Pruebe la proposicion anterior

Sin envargo nuestro modelo imperativo de funcion Σ-efectivamente computable todavia podria no ser correcto ya que podria pasar que haya una funcion Σ-mixta que sea computada por un procedimiento efectivo pero que no exista un programa de $\mathcal{S}^{\Sigma}$ que la compute. En otras palabras el modelo imperativo o Neumanniano podria ser incompleto. Por supuesto este no es el caso y los desarrollos que veremos mas adelante nos convenceran de que el paradigma imperativo es completo, es decir Neumann tambien vence a Leibniz.

Ejercicio 6: Sea $\Sigma = \{\#, @\}$. Para cada una de las siguientes funciones haga un programa que la compute

(a) $f : \{0, 1, 2\} \to \omega$, dada por $f(0) = f(1) = 0$ y $f(2) = 5$
(b) $\lambda xy[x + y]$
(c) $C_0^{1,1}|_{\{0,1\}\times\Sigma^*}$
(d) $p_4^{2,3}$
(e) $\lambda i\alpha\,[[\alpha]_i]$
(f) $\lambda\alpha[\sqrt{\alpha}]$
(g) $f : \omega^2 \times \{1, 2, 3\} \to \omega$, $f(x_1, x_2, x_3) = x_{x_3}$

Ejercicio 7: Sea $\Sigma = \{@, \&\}$. De un programa que compute la funcion $s^{\leq}$, donde $\leq$ esta dado por $@ < \&$

Ejercicio 8: V o F o I, justificar

(a) Dado $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ y $n, m \geq 0$, se tiene que $\Psi_{\mathcal{P}}^{n,m,\#} : \omega^{[\mathbf{N}]} \times \Sigma^{*[\mathbf{N}]} \to \omega$
(b) $\Psi^{1,0,\#}_{\mathrm{L1IFN1\neq0GOTOL1}} = \{(0, 0)\}$
(c) Sea Σ un alfabeto y sean $n, m \in \omega$. Sea $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$. Entonces el dominio de $\Psi_{\mathcal{P}}^{n,m,\#}$ es el conjunto formado por todos los estados a partir de los cuales $\mathcal{P}$ termina
(d) Sea Σ un alfabeto y sean $n, m \in \omega$. Entonces cualesquiera sean $x_1, ..., x_n \in \omega$ y $\alpha_1, ..., \alpha_m \in \Sigma^*$ se tiene que
$$\Psi^{n,m,\#}_{\mathrm{SKIP}}((x_1, ..., x_n, 0, 0, ...), (\alpha_1, ..., \alpha_m, \varepsilon, \varepsilon, ...)) = x_1$$
(e) El programa
$$\mathrm{N1} \leftarrow \mathrm{N1}\dot{-}1$$
computa la funcion $\lambda x_2 x_1[x_2\dot{-}1]$
(f) *Suc* se detiene para todo $x \in \omega$
(g) Si $\mathcal{P}$ computa una función $f : D_f \subseteq \omega^2 \to \omega$, entonces $\mathcal{P}$ computa la función $f \circ \left[p_1^{1,0}, C_0^{1,0}\right]$.

Ejercicio 9: Sea $\Sigma = \Sigma_p \cup \{a, b, c, d, e, f, g, ..., x, y, z\}$. De una funcion $f : \mathrm{Pro}^{\Sigma} \to \mathrm{Pro}^{\Sigma}$ la cual sea Σ-p.r. y tal que $\Psi_{f(\mathcal{P})}^{1,1,\#} = \Psi_{\mathcal{P}}^{1,1,\#} \circ \left[\lambda x\alpha[x + 2], C_{bb}^{1,1}\right]$, cualesquiera sea $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$

Macros

Supongamos que estamos escribiendo un programa $\mathcal{P}$ de $\mathcal{S}^{\Sigma}$ con el objeto de que realice cierta tarea. Supongamos ademas que nos vendria muy bien para nuestros propositos poder usar una instruccion

$$\text{N5} \leftarrow \text{N16} + \text{N3}$$

la cual por supuesto al correr el programa, deberia producir el efecto de dejar en la variable N5 la suma de los contenidos de las variables N16 y N3, sin modificar el contenido de las variables distintas a N5. Lamentablemente no tenemos en $\mathcal{S}^{\Sigma}$ este tipo de instruccion pero podriamos reemplazarla por el siguiente programa

	$\text{N1111} \leftarrow \text{N16}$
	$\text{N2222} \leftarrow \text{N3}$
	$\text{N5} \leftarrow \text{N1111}$
L1000	IF $\text{N2222} \neq 0$ GOTO L2000
	GOTO L3000
L2000	$\text{N2222} \leftarrow \text{N2222} \dot{-} 1$
	$\text{N5} \leftarrow \text{N5} + 1$
	GOTO L1000
L3000	SKIP

donde las variables N1111, N2222 y los labels L1000, L2000, L3000 solo seran usados aqui, es decir no apareceran en el resto de nuestro programa $\mathcal{P}$. Notese que este programa cuando es corrido termina dejando en la variable N5 la suma de los contenidos de las variables N16 y N3 y modifica el contenido de las variables N1111 y N2222, lo cual no traera problemas ya que N1111 y N2222 no se usan en el resto de $\mathcal{P}$. La variables N1111 y N2222 son auxiliares y se usan justamente para preservar el valor de las variables N16 y N3 ya que ellas son variables protagonistas de nuestro programa $\mathcal{P}$ y en esta instancia no queremos alterar su contenido sino solo realizar la asignacion $\text{N5} \leftarrow \text{N16} + \text{N3}$. Dejamos al lector explicar por que es necesario para que la simulacion sea correcta que los labels L1000, L2000 y L3000 no sean usados en el resto de $\mathcal{P}$.

Es decir el programa anterior simula la instruccion $\text{N5} \leftarrow \text{N16} + \text{N3}$ que no podiamos usar por no ser una instruccion de $\mathcal{S}^{\Sigma}$, con un costo bastante bajo, es decir el costo de convenir en no usar en el resto de $\mathcal{P}$ las variables N1111 y N2222 ni los labels L1000, L2000 y L3000.

Ahora supongamos que seguimos escribiendo el programa $\mathcal{P}$ y nos hace falta simular la instruccion $\text{N20} \leftarrow \text{N1} + \text{N14}$. Entonces es claro que podriamos modificar el programa que simulaba $\text{N5} \leftarrow \text{N16} + \text{N3}$ haciendole reemplazos adecuados a sus variables y labels. Por ejemplo podriamos escribir

	$\text{N9999} \leftarrow \text{N1}$
	$\text{N8888} \leftarrow \text{N14}$
	$\text{N20} \leftarrow \text{N9999}$
L1001	IF $\text{N8888} \neq 0$ GOTO L2002
	GOTO L3003
L2002	$\text{N8888} \leftarrow \text{N8888} \dot{-} 1$
	$\text{N20} \leftarrow \text{N20} + 1$
	GOTO L1001
L3003	SKIP

donde N9999, N8888, L1001, L2002 y L3003 solo seran usados aqui, es decir no apareceran en el resto de nuestro programa $\mathcal{P}$.

Consideremos el siguiente "molde" que llamaremos M

$$\begin{array}{ll} & \mathrm{V4} \leftarrow \mathrm{V2} \\ & \mathrm{V5} \leftarrow \mathrm{V3} \\ & \mathrm{V1} \leftarrow \mathrm{V4} \\ \mathrm{A1} & \mathrm{IF\ V5} \neq 0 \mathrm{\ GOTO\ A2} \\ & \mathrm{GOTO\ A3} \\ \mathrm{A2} & \mathrm{V5} \leftarrow \mathrm{V5} \dot{-} 1 \\ & \mathrm{V1} \leftarrow \mathrm{V1} + 1 \\ & \mathrm{GOTO\ A1} \\ \mathrm{A3} & \mathrm{SKIP} \end{array}$$

Como puede notarse, cuando reemplazamos en M

- cada ocurrencia de V1 por N5
- cada ocurrencia de V2 por N16
- cada ocurrencia de V3 por N3
- cada ocurrencia de V4 por N1111
- cada ocurrencia de V5 por N2222
- cada ocurrencia de A1 por L1000
- cada ocurrencia de A2 por L2000
- cada ocurrencia de A3 por L3000

obtenemos el programa que simulaba la instruccion $\mathrm{N5} \leftarrow \mathrm{N16} + \mathrm{N3}$ dentro de $\mathcal{P}$. Similarmente, cuando reemplazamos en M

- cada ocurrencia de V1 por N20
- cada ocurrencia de V2 por N1
- cada ocurrencia de V3 por N14
- cada ocurrencia de V4 por N9999
- cada ocurrencia de V5 por N8888
- cada ocurrencia de A1 por L1001
- cada ocurrencia de A2 por L2002
- cada ocurrencia de A3 por L3003

obtenemos el programa que simulaba la instruccion $\mathrm{N20} \leftarrow \mathrm{N1} + \mathrm{N14}$ dentro de $\mathcal{P}$. La practicidad de tener el molde M cae de maduro. Ahora en caso de necesitar una instruccion del tipo $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n} + \mathrm{N}\bar{m}$ solo tenemos que reemplazar en M

- cada ocurrencia de V1 por $\mathrm{N}\bar{k}$
- cada ocurrencia de V2 por $\mathrm{N}\bar{n}$
- cada ocurrencia de V3 por $\mathrm{N}\bar{m}$

y reemplazar las "variables" V4 y V5 y los "labels" A1, A2 y A3, por dos variables concretas y tres labels concretos que no se usen en el programa que estamos realizando. El programa asi obtenido simulara a la instruccion $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n} + \mathrm{N}\bar{m}$.

En la gerga computacional el molde M suele llamarse *macro* y los programas obtenidos luego de realizar los reemplazos son llamados *expansiones de* M. Notese que $Ti(M) = \mathrm{PALABRA}$ ya que, como en el caso de los programas, escribimos a M linea por linea para facilitar su manejo pero en realidad es una sola palabra, a saber:

V1←V2V4←V3A1IFV4≠0GOTOA2GOTOA3A2V4←V4$\dot{-}$1V1←V1+1GOTOA1A3SKIP

Es decir, como objeto matematico, M es simplemente una palabra. A las palabras de la forma $\mathrm{V}\bar{n}$, con $n \in \mathbf{N}$, las llamaremos *variables numericas de macro*. A las palabras de la forma $\mathrm{W}\bar{n}$, con $n \in \mathbf{N}$, las llamaremos *variables alfabeticas de macro* y a las palabras de la forma $\mathrm{A}\bar{n}$, con $n \in \mathbf{N}$, las llamaremos *labels de macro*. Nuestro macro M no tiene variables alfabeticas de macro pero otros macros por supuesto pueden tener este tipo de variables.

Las variables V1, V2 y V3 son llamadas *variables oficiales* de M ya que son las variables que seran reemplazadas por variables que son protagonistas dentro del programa $\mathcal{P}$ que usara la expansion de M. Las palabras V4 y V5 son llamadas *variables auxiliares* de M ya que seran reemplazadas por variables que se usaran solo dentro de la expansion y no intervienen en la "trama" del programa $\mathcal{P}$. Tambien A1, A2 y A3 son llamados *labels auxiliares* de M ya que son usados solo para su funcionamiento interno y no tienen vinculacion con los labels del programa $\mathcal{P}$.

En el siguiente ejemplo veremos un macro que tiene un label que no es auxiliar sino oficial. Supongamos que estamos escribiendo un programa $\mathcal{P}'$ y nos hace falta simular instrucciones de la forma

$$\text{IF } |\mathrm{P}\bar{n}| \leq \mathrm{N}\bar{m} \text{ GOTO } \mathrm{L}\bar{k}$$

(por supuesto estas instrucciones no pertenecen al lenguaje $\mathcal{S}^{\Sigma}$ pero deberia quedar claro como funcionan). Construiremos un macro que nos permita simular dicho tipo de instrucciones. Aqui la construccion del macro dependera de quien es Σ, es decir el macro nos servira para simular instrucciones de $\mathcal{S}^{\Sigma}$ haciendo programas de $\mathcal{S}^{\Sigma}$ y si lo usamos haciendo un programa de $\mathcal{S}^{\Gamma}$, donde Γ es otro alfabeto que contenga a Σ, el macro no funcionara bien. Haremos entonces el macro para el caso de $\Sigma = \{@, !\}$. Sea M' el siguiente macro:

	W2 $\leftarrow$ W1
	V2 $\leftarrow$ V1
A4	IF W2 BEGINS @ GOTO A2
	IF W2 BEGINS ! GOTO A2
	GOTO A1
A2	IF V2 $\neq 0$ GOTO A3
	GOTO A5
A3	W2 $\leftarrow^{\frown}$ W2
	V2 $\leftarrow$ V2$\dot{-}$1
	GOTO A4
A5	SKIP

el cual tiene

- variables oficiales W1 y V1 (correspondientes a $\mathrm{P}\bar{n}$ y $\mathrm{N}\bar{m}$)
- variables auxiliares W2 y V2
- labels auxiliares A2, A3, A4 y A5
- un label oficial A1 (correspondiente a $\mathrm{L}\bar{k}$)

Una descripcion intuitiva del macro M' seria

$$\text{IF } |\mathrm{W1}| \leq \mathrm{V1} \text{ GOTO A1}$$

Notese que en las primeras dos lineas el macro M' guarda los valores de las variables oficiales W1 y V1 en las variables auxiliares W2 y V2, y sigue trabajando con las auxiliares. Esto es para preservar el valor de las variables oficiales. Dado que

$\Sigma = \{@, !\}$, las dos siguientes lineas sirven para decidir si el contenido de W2 es ε o no ya que si una palabra de Σ^* no comienza con @ ni con !,debera ser ε (aqui es donde notamos que nuestro macro M' funcionara bien solo para hacer programas de $\mathcal{S}^{\{@,!\}}$). Dejamos al lector entender el resto del funcionamiento de M'.

Para dar un ejemplo de como usariamos a M', supongamos que para seguir escribiendo nuestro programa $\mathcal{P}'$ nos hace falta simular la instruccion

$$\text{IF } |\text{P5}| \leq \text{N14 GOTO L1}$$

y supongamos que las variables P1000 y N1000 y los labels L6666, L7777, L8888 y L9999 no se usaron hasta el momento en $\mathcal{P}'$. Entonces podemos reemplazar en M'

- cada ocurrencia de W1 por P5
- cada ocurrencia de V1 por N14
- cada ocurrencia de W2 por P1000
- cada ocurrencia de V2 por N1000
- cada ocurrencia de A1 por L1
- cada ocurrencia de A2 por L6666
- cada ocurrencia de A3 por L7777
- cada ocurrencia de A4 por L8888
- cada ocurrencia de A5 por L9999

y la expansion de M' asi obtenida simulara la instruccion IF $|\text{P5}| \leq \text{N14}$ GOTO L1. Cabe destacar que para asegurarnos que la simulacion funcione, tambien deberemos no usar en el resto de $\mathcal{P}'$ las variables P1000 y N1000 y los labels L6666, L7777, L8888 y L9999.

Es decir M' funciona como un molde con el cual haciendo reemplazos adecuados podemos simular en $\mathcal{S}^{\Sigma}$ cualquier instruccion del tipo IF $|\text{P}\bar{n}| \leq \text{N}\bar{m}$ GOTO L$\bar{k}$, con $n, m, k \in \mathbf{N}$.

Deberia quedar claro el caracter oficial del label A1 en M' ya que el label por el que se lo reemplaza para hacer la expansion es uno de los labels protagonistas del programa que se esta escribiendo.

Cabe destacar que las expansiones de M' no son programas ya que si bien son concatenaciones de instrucciones, no cumplen la ley de los GOTO (llamada (G) en la definicion de programa) respecto del label que reemplazo a A1.

Como se vio en este ultimo ejemplo es importante tener en cuenta que cuando contruimos un macro lo hacemos relativo a un alfabeto Σ previamente fijado, es decir estamos pensando que dicho macro sera usado para hacer programas de $\mathcal{S}^{\Sigma}$ y podria pasar que si lo usaramos para hacer un programa de $\mathcal{S}^{\Gamma}$, donde Γ es otro alfabeto, el macro no cumpla las propiedades esperadas. Para visulizar esto mejor, notese que la palabra

```
    IF W1 BEGINS @ GOTO A2
    IF W1 BEGINS % GOTO A2
    IF W1 BEGINS ! GOTO A2
    IF W1 BEGINS □ GOTO A2
    GOTO A1
A2  SKIP
```

es un macro de $\mathcal{S}^{\{@,\%,!,\square\}}$ que simula instrucciones de la forma

$$\text{IF P}\bar{n} = \varepsilon \text{ GOTO L}\bar{m}$$

(el cual obviamente podriamos denotar con IF W1 $= \varepsilon$ GOTO A1), pero es claro que esta palabra no nos servira para simular este tipo de instrucciones en el lenguaje $\mathcal{S}^{\{@,\%,!,\square,\triangle\}}$.

Nota: Siempre supondremos que la primera instruccion de los macros no es labelada. Esto es porque muchas veces cuando expandamos un macro nos interesara labelar la primera instruccion de dicha expansion. Por supuesto, esto es facil de conseguir ya que si M es un macro, entonces SKIPM es tambien un macro que posee las mismas propiedades.

Como hemos visto recien hay dos tipos de macros:

- los de asignacion que cuando son expandidos nos dan un programa que simula la asignacion a una variable dada del resultado de aplicar una funcion a los contenidos de ciertas otras variables; y
- los de tipo IF que cuando son expandidos nos dan un programa salvo por la ley (G), el cual direcciona al label que fue a reemplazar a A1 cuando se cumple cierta propiedad (predicado) relativa a los contenidos de las variables que fueron a reemplazar a las variables oficiales.

Ejemplo concreto de uso de macros. Ya vimos recien que la palabra

$$\begin{array}{ll} & \mathrm{V4} \leftarrow \mathrm{V2} \\ & \mathrm{V5} \leftarrow \mathrm{V3} \\ & \mathrm{V1} \leftarrow \mathrm{V4} \\ \mathrm{A1} & \text{IF } \mathrm{V5} \neq 0 \text{ GOTO A2} \\ & \text{GOTO A3} \\ \mathrm{A2} & \mathrm{V5} \leftarrow \mathrm{V5} \dot{-} 1 \\ & \mathrm{V1} \leftarrow \mathrm{V1} + 1 \\ & \text{GOTO A1} \\ \mathrm{A3} & \text{SKIP} \end{array}$$

es un macro que sirve para simular instrucciones de la forma $\mathrm{N}\bar{k} \leftarrow \mathrm{N}\bar{n} + \mathrm{N}\bar{m}$. Notemos que este macro es de asignacion ya que cuando es expandido nos da un programa que simula la asignacion a una variable dada del resultado de aplicar una funcion a los contenidos de ciertas otras variables. En este caso la funcion es $SUMA = \lambda xy[x + y]$ por lo cual usaremos $[\mathrm{V1} \leftarrow SUMA(\mathrm{V2}, \mathrm{V3})]$ para denotar a dicho macro. Usaremos este macro para dar un programa $\mathcal{P}$ que compute a la funcion $\lambda xy[x.y]$. Notese que podemos tomar $\mathcal{P}$ igual al siguiente programa

$$\begin{array}{ll} \mathrm{L1} & \text{IF } \mathrm{N2} \neq 0 \text{ GOTO L2} \\ & \text{GOTO L3} \\ \mathrm{L2} & [\mathrm{N3} \leftarrow SUMA(\mathrm{N3}, \mathrm{N1})] \\ & \mathrm{N2} \leftarrow \mathrm{N2} \dot{-} 1 \\ & \text{GOTO L1} \\ \mathrm{L3} & \mathrm{N1} \leftarrow \mathrm{N3} \end{array}$$

donde $[\mathrm{N3} \leftarrow SUMA(\mathrm{N3}, \mathrm{N1})]$ es una expansion del macro $[\mathrm{V1} \leftarrow SUMA(\mathrm{V2}, \mathrm{V3})]$ hecha haciendo el reemplazo de las variables oficiales $\mathrm{V1}, \mathrm{V2}$ y $\mathrm{V3}$ por $\mathrm{N3}, \mathrm{N3}$ y $\mathrm{N1}$, respectivamente, y haciendo reemplazos adecuados de sus variables y labels auxiliares. Hay muchas formas de hacer los reemplazos de variables y labels auxiliares pero en general no lo especificaremos explicitamente cuando expandamos un macro

ya que es facil imaginar como hacerlo en funcion del programa que estemos realizando. Por ejemplo en el caso de $\mathcal{P}$ podriamos hacer los siguientes reemplazos:

- cada ocurrencia de V4 por N1111
- cada ocurrencia de V5 por N2222
- cada ocurrencia de A1 por L1000
- cada ocurrencia de A2 por L2000
- cada ocurrencia de A3 por L3000

y claramente esto no afectara la "logica" o "idea" de nuestro programa $\mathcal{P}$. De esta forma la expansion $[\mathrm{N3} \leftarrow SUMA(\mathrm{N3}, \mathrm{N1})]$ es el siguiente programa:

	$\mathrm{N1111} \leftarrow \mathrm{N3}$
	$\mathrm{N2222} \leftarrow \mathrm{N1}$
	$\mathrm{N3} \leftarrow \mathrm{N1111}$
L1000	IF $\mathrm{N2222} \neq 0$ GOTO L2000
	GOTO L3000
L2000	$\mathrm{N2222} \leftarrow \mathrm{N2222} \dot{-} 1$
	$\mathrm{N3} \leftarrow \mathrm{N3} + 1$
	GOTO L1000
L3000	SKIP

el cual por supuesto esta escrito con espacios y en forma vertical pero es una mera palabra. Tenemos entonces que $\mathcal{P}$ es el programa:

L1	IF $\mathrm{N2} \neq 0$ GOTO L2
	GOTO L3
L2	$\mathrm{N1111} \leftarrow \mathrm{N1}$
	$\mathrm{N2222} \leftarrow \mathrm{N3}$
	$\mathrm{N3} \leftarrow \mathrm{N1111}$
L1000	IF $\mathrm{N2222} \neq 0$ GOTO L2000
	GOTO L3000
L2000	$\mathrm{N2222} \leftarrow \mathrm{N2222} \dot{-} 1$
	$\mathrm{N3} \leftarrow \mathrm{N3} + 1$
	GOTO L1000
L3000	SKIP
	$\mathrm{N2} \leftarrow \mathrm{N2} \dot{-} 1$
	GOTO L1
L3	$\mathrm{N1} \leftarrow \mathrm{N3}$

el cual por supuesto esta escrito con espacios y en forma vertical pero es una mera palabra.

Macros asociados a funciones Σ-computables. Dada una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$, usaremos

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

para denotar un macro de $\mathcal{S}^{\Sigma}$ M el cual cumpla las siguientes propiedades (no siempre existira dicho macro, es decir solo para ciertas funciones $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ habra un tal macro).

(1) Las variables oficiales de M son $\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{V}\overline{n+1}, \mathrm{W}1, ..., \mathrm{W}\bar{m}$
(2) M no tiene labels oficiales
(3) Si reemplazamos:

- las variables oficiales de M (i.e. $\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{V}\overline{n+1}, \mathrm{W}1, ..., \mathrm{W}\bar{m}$) por variables concretas

$$\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{N}\overline{k_{n+1}}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$$

(elejidas libremente, es decir los numeros $k_1, ..., k_{n+1}, j_1, ..., j_m$ son cualesquiera)
- las variables auxiliares de M por variables concretas (distintas de a dos) y NO pertenecientes a la lista $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{N}\overline{k_{n+1}}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$
- los labels auxiliares de M por labels concretos (distintos de a dos)

entonces la palabra asi obtenida es un programa $\mathcal{E}$ de $\mathcal{S}^\Sigma$ que denotaremos en general con

$$\left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$$

el cual debe tener la siguiente propiedad:

(E) Si hacemos correr $\mathcal{E}$ partiendo de un estado e que le asigne a las variables $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$ valores $x_1, ..., x_n, \alpha_1, ..., \alpha_m$, entonces independientemente de los valores que les asigne e al resto de las variables (incluidas las que fueron a reemplazar a las variables auxiliares de M) se dara que
  (i) si $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \notin D_f$, entonces $\mathcal{E}$ no se detiene
  (ii) si $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \in D_f$, entonces $\mathcal{E}$ se detiene (i.e. intenta realizar la siguiente a su ultima instrucion) y llega a un estado e' el cual cumple:
    - e' le asigna a $\mathrm{N}\overline{k_{n+1}}$ el valor $f(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$
    - e' solo puede diferir de e en los valores que le asigna a $\mathrm{N}\overline{k_{n+1}}$ o a las variables que fueron a reemplazar a las variables auxiliares de M. Notese que esto nos dice que los valores de las variables $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$ al correr $\mathcal{E}$ desde el estado e no se modificaran, salvo cuando k_i es igual a k_{n+1}, situacion en la cual el valor final de $\mathrm{N}\overline{k_i}$ sera $f(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$

El programa $\mathcal{E} = \left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$ es comunmente llamado la *expansion* del macro $M = \left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$ con respecto a la eleccion de variables y labels realizada.

Tambien, dada una funcion $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$, con

$$\left[\mathrm{W}\overline{m+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

denotaremos un macro de $\mathcal{S}^\Sigma$ el cual cumpla condiciones analogas a las descriptas recien. Dejamos al lector escribirlas en detalle para este caso.

Es importante notar que nuestra notacion

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

tiene cierta ambiguedad ya que no hace mension al alfabeto Σ el cual esta previamente fijado. Podria pasar que la funcion f sea Γ-mixta para otro alfabeto Γ y que el macro

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

sea distinto segun lo consideremos para $\mathcal{S}^\Sigma$ o para $\mathcal{S}^\Gamma$. Es decir, como ya lo remarcamos anteriormente, cuando hablamos de la existencia de un macro, es siempre relativo a un $\mathcal{S}^\Sigma$. Lo mismo pasa con la notacion

$$\left[\mathrm{W}\overline{m+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

Dejamos al lector que piense este fenomeno para el caso de la funcion $f : \{@, \%\}^* \times \{@, \%\}^* \to \{@, \%\}^*$, dada por $f(\alpha, \beta) = \alpha\beta$. Si tomamos $\Sigma = \{@, \%\}$ y $\Gamma = \{@, \%, !\}$, entonces f resulta una funcion Σ-mixta y tambien Γ-mixta y deberia quedar claro que no es lo mismo hacer el macro

$$[\mathrm{W}3 \leftarrow f(\mathrm{W}1, \mathrm{W}2)]$$

para $\mathcal{S}^\Sigma$ que hacerlo para $\mathcal{S}^\Gamma$.

Otra ambiguedad de esta notacion de macros de asignacion es que en el nombre de los macros

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

$$\left[\mathrm{W}\overline{m+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

no se especifica quienes son las variables y labels auxiliares de dichos macros. Ambas ambiguedades no nos traeran problemas si manejamos las cosas con cierta madurez.

Aceptaremos sin demostracion el siguiente resultado fundamental.

Proposición 3. (a) *Sea $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ una funcion Σ-computable. Entonces en $\mathcal{S}^\Sigma$ hay un macro*

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

(b) *Sea $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$ una funcion Σ-computable. Entonces en $\mathcal{S}^\Sigma$ hay un macro*

$$\left[\mathrm{W}\overline{m+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

Ejercicio 10: Sea $SUMA = \lambda xy[x + y]$. Explique por que la palabra

$$\begin{array}{ll} & \mathrm{V}1 \leftarrow \mathrm{V}2 \\ & \mathrm{V}4 \leftarrow \mathrm{V}3 \\ \mathrm{A}1 & \text{IF V4} \neq 0 \text{ GOTO A2} \\ & \text{GOTO A3} \\ \mathrm{A}2 & \mathrm{V}4 \leftarrow \mathrm{V}4 \dot{-} 1 \\ & \mathrm{V}1 \leftarrow \mathrm{V}1 + 1 \\ & \text{GOTO A1} \\ \mathrm{A}3 & \text{SKIP} \end{array}$$

no puede ser tomada como el macro $[\mathrm{V}3 \leftarrow SUMA(\mathrm{V}1, \mathrm{V}2)]$

Ejercicio 11: Sea $\Sigma = \{\#, \$\}$ y sea $f : D_f \subseteq \Sigma^* \to \omega$ una funcion Σ-computable. Sea $L = \{\alpha \in D_f : f(\alpha) = 1\}$. De (usando el macro $[\mathrm{V}1 \leftarrow f(\mathrm{W}1)]$) un programa $\mathcal{P} \in \mathrm{Pro}^\Sigma$ tal que $\mathrm{Dom}(\Psi_\mathcal{P}^{0,1,\#}) = L$.

Ejercicio 12: Sea $\Sigma = \{\#, \$\}$ y sea $f : \omega \to \Sigma^*$ una funcion Σ-computable. De (usando el macro $[\mathrm{W}1 \leftarrow f(\mathrm{V}1)]$) un programa $\mathcal{P} \in \mathrm{Pro}^\Sigma$ tal que $\mathrm{Dom}(\Psi_\mathcal{P}^{0,1,\#}) = \mathrm{Im}\, f$.

Ejercicio 13: Pruebe la resiproca de la proposicion anterior, es decir pruebe que si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ es tal que en $\mathcal{S}^\Sigma$ hay un macro

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

entonces f es Σ-computable.

Macros asociados a predicados Σ-computables. Dado un predicado $P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \omega$, usaremos

$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

para denotar un macro M de $\mathcal{S}^\Sigma$ el cual cumpla las siguientes propiedades (no siempre existira dicho macro, es decir solo para ciertos predicados $P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \omega$ habra un tal macro).

(1) Las variables oficiales de M son $\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}$
(2) A1 es el unico label oficial de M
(3) Si reemplazamos:
   - las variables oficiales de M (i.e. $\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}$) por variables concretas
   $$\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$$
   (elejidas libremente, es decir los numeros $k_1, ..., k_n, j_1, ..., j_m$ son cualesquiera)
   - el label oficial A1 por un label concreto $\mathrm{L}\bar{k}$ (elejido libremente, es decir k es cualquier elemento de $\mathbf{N}$)
   - las variables auxiliares de M por variables concretas (distintas de a dos) y NO pertenecientes a la lista $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$
   - los labels auxiliares de M por labels concretos (distintos de a dos) y ninguno igual a $\mathrm{L}\bar{k}$

entonces la palabra asi obtenida es un programa $\mathcal{E}$ de $\mathcal{S}^\Sigma$ (salvo por la ley de los GOTO respecto de $\mathrm{L}\bar{k}$) que denotaremos en general con

$$\left[\text{IF } P(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}) \text{ GOTO } \mathrm{L}\bar{k}\right]$$

el cual debe tener la siguiente propiedad:

(E) Si hacemos correr $\mathcal{E}$ partiendo de un estado e que le asigne a las variables $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$ valores $x_1, ..., x_n, \alpha_1, ..., \alpha_m$, entonces independientemente de los valores que les asigne e al resto de las variables (incluidas las que fueron a reemplazar a las variables auxiliares de M) se dara que
   (i) si $(\vec{x}, \vec{\alpha}) \notin D_P$, entonces $\mathcal{E}$ no se detiene
   (ii) si $(\vec{x}, \vec{\alpha}) \in D_P$ y $P(\vec{x}, \vec{\alpha}) = 1$, entonces luego de una cantidad finita de pasos, $\mathcal{E}$ direcciona al label $\mathrm{L}\bar{k}$ quedando en un estado e' el cual solo puede diferir de e en los valores que le asigna a las variables que fueron a reemplazar a las variables auxiliares de M. Al resto de las variables, incluidas $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$ no las modifica
   (iii) si $(\vec{x}, \vec{\alpha}) \in D_P$ y $P(\vec{x}, \vec{\alpha}) = 0$, entonces luego de una cantidad finita de pasos, $\mathcal{E}$ se detiene (i.e. intenta realizar la siguiente a su ultima instruccion) quedando en un estado e' el cual solo puede diferir de e en los valores que le asigna a las variables que

fueron a reemplazar a las variables auxiliares de M. Al resto de las variables, incluidas $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$ no las modifica

La palabra $\mathcal{E} = \left[\text{IF } P(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ es llamada la *expansion* del macro con respecto a la eleccion de variables y labels realizada. Al igual que en el caso de los macros de asignacion, la notacion

$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

tiene cierta ambiguedad ya que no explicita el alfabeto ni los labels y variables auxiliares de tipo macro.

Proposición 4. *Sea $P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \omega$ un predicado Σ-computable. Entonces en $\mathcal{S}^\Sigma$ hay un macro*

$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

Proof. Por (a) de la proposicion anterior tenemos un macro $\left[\mathrm{V}\overline{n+1} \leftarrow P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$. Notese que la palabra

$$\left[\mathrm{V}\overline{n+1} \leftarrow P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right] \text{IFV}\overline{n+1}\neq 0\text{GOTOA1}$$

es el macro buscado. ■

Ejercicio 14: Pruebe la resiproca de la proposicion anterior, es decir pruebe que si $P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \omega$ es tal que en $\mathcal{S}^\Sigma$ hay un macro

$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

entonces P es Σ-computable.

Ejercicio 15: Sea Σ un alfabeto finito. Pruebe usando macros (de tipo IF) que si $P : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ y $Q : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ son predicados Σ-computables, entonces $(P \vee Q)$, $(P \wedge Q)$ y $\neg P$ lo son tambien.

Primer Manantial de Macros. Dejamos en forma de una sola proposicion los tres casos de existencia de macros.

Proposición 5 (Primer Manantial de Macros)**.** *Sea Σ un alfabeto finito. Si*

$$f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$$
$$g : D_g \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$$
$$P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \{0, 1\}$$

son Σ-computables, entonces en $\mathcal{S}^\Sigma$ hay macros

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$\left[\mathrm{W}\overline{m+1} \leftarrow g(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

Basicamente habra dos manatiales de macros de donde obtendremos todos los macros que potenciaran nuestro lenguaje $\mathcal{S}^\Sigma$. El Segundo Manantial de Macros es consecuencia del primero y de la batalla Neumann vence a Godel que veremos en la Guia 8.

Conjuntos Σ-enumerables

Ya que la nocion de funcion Σ-computable es el modelo matematico Neumanniano o imperativo del concepto de funcion Σ-efectivamente computable, nos podriamos preguntar entonces cual es el modelo matematico Neumanniano del concepto de conjunto Σ-efectivamente enumerable. Si prestamos atencion a la definicion de conjunto Σ-efectivamente enumerable, notaremos que depende de la existencia de ciertas funciones Σ-efectivamente computables por lo cual la siguiente definicion cae de maduro:

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*enumerable* cuando sea vacio o haya una funcion $F : \omega \to \omega^n \times \Sigma^{*m}$ tal que $I_F = S$ y $F_{(i)}$ sea Σ-computable, para cada $i \in \{1, ..., n+m\}$.

Deberia entonces quedar claro que si el concepto de funcion Σ-computable modeliza correctamente al concepto de funcion Σ-efectivamente computable, entonces el concepto de conjunto Σ-enumerable recien definido modeliza correctamente al concepto de conjunto Σ-efectivamente enumerable.

Notese que segun la definicion que acabamos de escribir, un conjunto no vacio $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-enumerable si y solo si hay programas $\mathcal{P}_1, ..., \mathcal{P}_{n+m}$ tales que

- $Dom(\Psi^{1,0,\#}_{\mathcal{P}_1}) = ... = Dom(\Psi^{1,0,\#}_{\mathcal{P}_n}) = \omega$
- $Dom(\Psi^{1,0,*}_{\mathcal{P}_{n+1}}) = ... = Dom(\Psi^{1,0,*}_{\mathcal{P}_n+m}) = \omega$
- $S = \mathrm{Im}([\Psi^{1,0,\#}_{\mathcal{P}_1}, ..., \Psi^{1,0,\#}_{\mathcal{P}_n}, \Psi^{1,0,*}_{\mathcal{P}_{n+1}}, ..., \Psi^{1,0,*}_{\mathcal{P}_{n+m}}])$

Como puede notarse los programas $\mathcal{P}_1, ..., \mathcal{P}_{n+m}$ puestos secuencialmente a funcionar desde el estado $\|x\|$ producen en forma natural un procedimiento efectivo (con dato de entrada $x \in \omega$) que enumera a S. Por supuesto podemos decir que en tal caso los programas $\mathcal{P}_1, ..., \mathcal{P}_{n+m}$ enumeran a S. La siguiente proposicion muestra que tambien las cosas se pueden hacer con un solo programa

Proposición 6 (Caracterizacion de Σ-enumerabilidad)**.** *Sea $S \subseteq \omega^n \times \Sigma^{*m}$ un conjunto no vacio. Entonces son equivalentes:*

(1) *S es Σ-enumerable*
(2) *Hay un programa $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ tal que:*
   (a) *Para cada $x \in \omega$, tenemos que $\mathcal{P}$ se detiene partiendo desde el estado $\|x\|$ y llega a un estado de la forma $((x_1, ..., x_n, y_1, ...), (\alpha_1, ..., \alpha_m, \beta_1, ...))$, donde $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \in S$.*
   (b) *Para cada $(x_1, ...x_n, \alpha_1, ..., \alpha_m) \in S$ hay un $x \in \omega$ tal que $\mathcal{P}$ se detiene partiendo desde el estado $\|x\|$ y llega a un estado de la forma $((x_1, ..., x_n, y_1, ...), (\alpha_1, ..., \alpha_m, \beta_1, ...))$*

Proof. (1)⇒(2). Ya que S es no vacio, por definicion tenemos que hay una $F : \omega \to \omega^n \times \Sigma^{*m}$ tal que $I_F = S$ y $F_{(i)}$ es Σ-computable, para cada $i \in \{1, ..., n+m\}$. Por

el Primer Manantial de Macros tenemos que existen macros:

$$\begin{gathered}\big[\mathrm{V2} \leftarrow F_{(1)}(\mathrm{V1})\big] \\ \vdots \\ \big[\mathrm{V2} \leftarrow F_{(n)}(\mathrm{V1})\big] \\ \big[\mathrm{W1} \leftarrow F_{(n+1)}(\mathrm{V1})\big] \\ \vdots \\ \big[\mathrm{W1} \leftarrow F_{(n+m)}(\mathrm{V1})\big]\end{gathered}$$

Sea $\mathcal{P}$ el siguiente programa:

$$\begin{gathered}\big[\mathrm{P}\overline{m} \leftarrow F_{(n+m)}(\mathrm{N1})\big] \\ \vdots \\ \big[\mathrm{P1} \leftarrow F_{(n+1)}(\mathrm{N1})\big] \\ \big[\mathrm{N}\overline{n} \leftarrow F_{(n)}(\mathrm{N1})\big] \\ \vdots \\ \big[\mathrm{N1} \leftarrow F_{(1)}(\mathrm{N1})\big]\end{gathered}$$

donde se supone que las expansiones de los macros usados son hechas usando variables auxiliares no pertenecientes a la lista $\mathrm{N1}, ..., \mathrm{N}\overline{n}, \mathrm{P1}, ..., \mathrm{P}\overline{m}$ (por supuesto, dada la fortaleza de nuestros macros se puede usar una misma variable auxiliar para dos distintas expansiones), y tambien se supone que los labels auxiliares usados en dichas expansiones son todos distintos, es decir no usamos el mismo label auxiliar en dos expansiones distintas (por que?).

Dejamos al lector corroborar que el programa $\mathcal{P}$ cumple las propiedades (a) y (b)

(2)⇒(1). Supongamos $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ cumple (a) y (b) de (2). Sean

$$\begin{aligned}\mathcal{P}_1 &= \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N1} \\ \mathcal{P}_2 &= \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N2} \\ &\vdots \\ \mathcal{P}_n &= \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N}\overline{n} \\ \mathcal{P}_{n+1} &= \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P1} \\ \mathcal{P}_{n+2} &= \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P2} \\ &\vdots \\ \mathcal{P}_{n+m} &= \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P}\overline{m}\end{aligned}$$

Definamos

$$\begin{aligned} F_1 &= \Psi^{1,0,\#}_{\mathcal{P}_1} \\ F_2 &= \Psi^{1,0,\#}_{\mathcal{P}_2} \\ &\vdots \\ F_n &= \Psi^{1,0,\#}_{\mathcal{P}_n} \\ F_{n+1} &= \Psi^{1,0,*}_{\mathcal{P}_{n+1}} \\ F_{n+2} &= \Psi^{1,0,*}_{\mathcal{P}_{n+2}} \\ &\vdots \\ F_{n+m} &= \Psi^{1,0,*}_{\mathcal{P}_{n+m}} \end{aligned}$$

Notese que cada F_i es Σ-computable y tiene dominio igual a ω. Sea $F = [F_1, ..., F_{n+m}]$. Tenemos por definicion que $D_F = \omega$ y ya que $F_{(i)} = F_i$, para cada $i = 1, ..., n+m$ tenemos que cada $F_{(i)}$ es Σ-computable. Dejamos al lector verificar que $I_F = S$. ■

Cuando un programa $\mathcal{P}$ cumpla las propiedades dadas en (2) de la proposicion anterior respecto de un conjunto S, diremos que $\mathcal{P}$ *enumera* a S. Notese que:

- Si $S \subseteq \omega$ es no vacio, entonces $\mathcal{P}$ enumera a S sii $\mathrm{Dom}(\Psi^{1,0,\#}_{\mathcal{P}}) = \omega$ y $\mathrm{Im}(\Psi^{1,0,\#}_{\mathcal{P}}) = S$
- Si $S \subseteq \Sigma^*$ es no vacio, entonces $\mathcal{P}$ enumera a S sii $\mathrm{Dom}(\Psi^{1,0,*}_{\mathcal{P}}) = \omega$ y $\mathrm{Im}(\Psi^{1,0,*}_{\mathcal{P}}) = S$

Cabe destacar que (2)⇒(1) de la proposicion anterior es muy util a la hora de probar que un conjunto dado es Σ-enumerable ya que nos permite trabajar dentro de un solo programa.

Ejercicio 16: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros ni la proposicion anterior que $S = \{(2, \%\%), (3, !!!), (0, \varepsilon)\}$ es Σ-enumerable

Ejercicio 17: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros ni la proposicion anterior que $S = \{(i, 5, \%^i) : i \in \omega\}$ es Σ-enumerable

Ejercicio 18: Sea $\Sigma = \{\%, !\}$. Sea $L \subseteq \Sigma^*$ un conjunto no vacio y Σ-enumerable. De (usando macros) un programa $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ tal que $\Psi^{1,0,*}_{\mathcal{P}}$ enumera al conjunto

$$\{\alpha\%! : \alpha \in L\}$$

es decir $\mathrm{Dom}\Psi^{1,0,*}_{\mathcal{P}} = \omega$ y $\mathrm{Im}(\Psi^{1,0,*}_{\mathcal{P}}) = \{\alpha\%! : \alpha \in L\}$

Ejercicio 19: Sea $\Sigma = \{\%, !\}$. Sea

$$S = \{(x, x+1, x+2, \%\%!!) : x \in \omega\}$$

De sin usar macros un programa $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ el cual enumere a S (i.e. que cumpla (2) de la proposicion anterior).

Conjuntos Σ-computables

La version imperativa o Neumanniana del concepto de conjunto Σ-efectivamente computable es facil de dar: un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ-*computable* cuando la funcion $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ-computable. O sea que $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-computable sii hay un programa $\mathcal{P} \in \text{Pro}^\Sigma$ el cual computa a $\chi_S^{\omega^n \times \Sigma^{*m}}$, es decir:

- Si $(\vec{x}, \vec{\alpha}) \in S$, entonces $\mathcal{P}$ se detiene partiendo desde $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$ y la variable N1 queda con contenido igual a 1
- Si $(\vec{x}, \vec{\alpha}) \in (\omega^n \times \Sigma^{*m}) - S$, entonces $\mathcal{P}$ se detiene partiendo desde $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$ y la variable N1 queda con contenido igual a 0

Si $\mathcal{P}$ es un programa el cual computa a $\chi_S^{\omega^n \times \Sigma^{*m}}$, diremos que $\mathcal{P}$ *decide la pertenecia a* S, con respecto al conjunto $\omega^n \times \Sigma^{*m}$.

Ejercicio 20: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros que $S = \{(2, \%\%), (3, !)\}$ es Σ-computable

Ejercicio 21: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros que $S = \{(i, \%^i) : i \in \omega\}$ es Σ-computable

Ejercicio 22: Sea Σ un alfabeto finito. Pruebe usando macros que

(a) Si $P : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ y $Q : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ son predicados Σ-computables, entonces $(P \vee Q)$, $(P \wedge Q)$ y $\neg P$ lo son tambien.

(b) Si $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-computables, entonces $S_1 \cup S_2$, $S_1 \cap S_2$ y $S_1 - S_2$ son Σ-computables

Macros asociados a conjuntos Σ-computables. El Primer Manatial de Macros nos dice que si $S \subseteq \omega^n \times \Sigma^{*m}$ es un conjunto Σ-computable, entonces, ya que $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-computable, en $\mathcal{S}^\Sigma$ hay un macro

$$\left[\text{IF } \chi_S^{\omega^n \times \Sigma^{*m}}(\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \text{ GOTO A1}\right]$$

Escribiremos el nombre de este macro de la siguiente manera mas intuitiva:

$$[\text{IF } (\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \in S \text{ GOTO A1}]$$

Notese que las expanciones de este macro, dado que $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-total, ya sea terminan por la ultima instruccion de la expansion o direccionan a la primera instruccion que tenga label igual al label que reemplazo a A1 en la expansion. Es importante notar que para asegurar la existencia de este macro utilizamos que S es Σ-computable lo cual no siempre sucedera para un conjunto S. Por ejemplo, puede pasar que S sea el dominio de una funcion Σ-computable pero que S no sea Σ-computable (esto se vera mas adelante) y en tal caso no existira en $\mathcal{S}^\Sigma$ un macro

$$[\text{IF } (\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \in S \text{ GOTO A1}]$$

ya que si tal macro existiera seria facil hacer un programa que compute a $\chi_S^{\omega^n \times \Sigma^{*m}}$ lo cual nos diria que S es Σ-computable (ver el ejercicio posterior a la Proposicion 4). Es muy comun el error de suponer que existe un macro

$$[\text{IF } (\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \in S \text{ GOTO A1}]$$

cuando S es el dominio de una funcion Σ-computable.

GUIA 8 DE LENGUAJES: BATALLAS ENTRE PARADIGMAS, TESIS DE CHURCH

En esta guia compararemos los tres paradigmas de computabilidad efectiva que hemos desarrollado anteriormente. Para esto probaremos que cada uno de dichos paradigmas "vence" al otro en el sentido que incluye por lo menos todas las funciones que incluye el otro en su modelizacion del concepto de funcion Σ-efectivamente computable. Por supuesto, esto dice que los tres son equivalentes.

Neumann vence a Godel

Usando macros podemos ahora probar que el paradigma imperativo de Neumann es por lo menos tan abarcativo como el funcional de Godel. Mas concretamente:

Teorema 1 (Neumann vence a Godel)**.** *Si h es Σ-recursiva, entonces h es Σ-computable.*

Proof. Probaremos por induccion en k que

(*) Si $h \in \mathrm{R}_k^\Sigma$, entonces h es Σ-computable.

El caso $k = 0$ es dejado al lector. Supongamos (*) vale para k, veremos que vale para $k+1$. Sea $h \in \mathrm{R}_{k+1}^\Sigma - \mathrm{R}_k^\Sigma$. Hay varios casos

Caso 1. Supongamos $h = M(P)$, con $P : \omega \times \omega^n \times \Sigma^{*m} \to \omega$, un predicado perteneciente a R_k^Σ. Por hipotesis inductiva, P es Σ-computable y por lo tanto tenemos un macro

$$\left[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\overline{n+1}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}\right]$$

lo cual nos permite realizar el siguiente programa

$$\begin{array}{ll} \mathrm{L2} & \left[\text{IF } P(\mathrm{N}\overline{n+1}, \mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}) \text{ GOTO L1}\right] \\ & \mathrm{N}\overline{n+1} \leftarrow \mathrm{N}\overline{n+1} + 1 \\ & \text{GOTO L2} \\ \mathrm{L1} & \mathrm{N}1 \leftarrow \mathrm{N}\overline{n+1} \end{array}$$

Es facil chequear que este programa computa h.

Caso 2. Supongamos $h = R(f, \mathcal{G})$, con

$$f : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^*$$
$$\mathcal{G}_a : S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^*,\ a \in \Sigma$$

elementos de R_k^Σ. Sea $\Sigma = \{a_1, ..., a_r\}$. Por hipotesis inductiva, las funciones f, $\mathcal{G}_a$, $a \in \Sigma$, son Σ-computables y por lo tanto tenemos macros

$$\left[\mathrm{W}\overline{m+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$

$$\left[\mathrm{W}\overline{m+3} \leftarrow \mathcal{G}_{a_i}(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}, \mathrm{W}\overline{m+1}, \mathrm{W}\overline{m+2})\right],\ i = 1, ..., r$$

Podemos entonces hacer el siguiente programa:

$$
\begin{array}{ll}
 & \left[\mathrm{P}\overline{m+3} \leftarrow f(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m})\right] \\
\mathrm{L}\overline{r+1} & \text{IF } \mathrm{P}\overline{m+1} \text{ BEGINS } a_1 \text{ GOTO L1} \\
 & \qquad \vdots \\
 & \text{IF } \mathrm{P}\overline{m+1} \text{ BEGINS } a_r \text{ GOTO L}\bar{r} \\
 & \text{GOTO L}\overline{r+2} \\
\mathrm{L}1 & \mathrm{P}\overline{m+1} \leftarrow \ \curvearrowright \mathrm{P}\overline{m+1} \\
 & \left[\mathrm{P}\overline{m+3} \leftarrow \mathcal{G}_{a_1}(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}, \mathrm{P}\overline{m+2}, \mathrm{P}\overline{m+3})\right] \\
 & \mathrm{P}\overline{m+2} \leftarrow \mathrm{P}\overline{m+2}.a_1 \\
 & \text{GOTO L}\overline{r+1} \\
 & \qquad \vdots \\
\mathrm{L}\bar{r} & \mathrm{P}\overline{m+1} \leftarrow \ \curvearrowright \mathrm{P}\overline{m+1} \\
 & \left[\mathrm{P}\overline{m+3} \leftarrow \mathcal{G}_{a_r}(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}, \mathrm{P}\overline{m+2}, \mathrm{P}\overline{m+3})\right] \\
 & \mathrm{P}\overline{m+2} \leftarrow \mathrm{P}\overline{m+2}.a_r \\
 & \text{GOTO L}\overline{r+1} \\
\mathrm{L}\overline{r+2} & \mathrm{P}1 \leftarrow \mathrm{P}\overline{m+3}
\end{array}
$$

Es facil chequear que este programa computa h.

El resto de los casos son dejados al lector. ■ Un corolario importante de esta batalla es el:

Proposición 2 (Segundo Manatial de Macros)**.** *Sea Σ un alfabeto finito. Si*

$$f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$$
$$g : D_g \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^*$$
$$P : D_P \subseteq \omega^n \times \Sigma^{*m} \to \{0, 1\}$$

son Σ-recursivas, entonces en $\mathcal{S}^\Sigma$ hay macros

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$\left[\mathrm{W}\overline{m+1} \leftarrow g(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

Recordemos que el Primer Manantial de Macros es dado en la Guia 7 al final de la seccion sobre macros.

Ejercicio 1: Pruebe el Segundo Manantial de Macros.

Se lleno de macros. Cabe destacar que el Segundo Manantial de Macros nos dice que en $\mathcal{S}^\Sigma$ hay macros

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$\left[\mathrm{W}\overline{m+1} \leftarrow g(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$$
$$[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

para todas las funciones Σ-mixtas y predicados Σ-mixtos que hemos trabajado hasta el momento en la materia ya que todas eran Σ-p.r.. Esto fortalece mucho al lenguaje $\mathcal{S}^\Sigma$ ya que ahora tenemos macros para todas las funciones y predicados cotidianos en la matematica, ademas de tener macros para todas las funciones y

predicados Σ-computables, debido al Primer Manantial de Macros. Veamos un ejemplo:

Lema 3. *Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-enumerables. Entonces $S_1 \cup S_2$ es Σ-enumerable.*

Proof. Podemos suponer que ni S_1 ni S_2 son vacios ya que de lo contrario el resultado es trivial. Ademas supondremos que $n = 2$ y $m = 1$.

La idea de la prueba es la misma que la que usamos para probar que la union de conjuntos Σ-efectivamente enumerables es Σ-efectivamente enumerable. Daremos usando macros un programa que enumera a $S_1 \cup S_2$ y luego aplicaremos la Proposicion de Caracterizacion de Σ-enumerabilidad de la Guia 7. Por hipotesis hay funciones $F : \omega \to \omega \times \omega \times \Sigma^*$ y $G : \omega \to \omega \times \omega \times \Sigma^*$ tales que $F_{(1)}$, $F_{(2)}$, $F_{(3)}$, $G_{(1)}$, $G_{(2)}$ y $G_{(3)}$ son Σ-computables, $\mathrm{Im}(F) = S_1$ y $\mathrm{Im}(G) = S_2$. O sea que nuestro Primer Manantial de Macros nos dice que en $\mathcal{S}^\Sigma$ hay macros

$$\begin{gathered}
\big[\mathrm{V2} \leftarrow F_{(1)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow F_{(2)}(\mathrm{V1})\big] \\
\big[\mathrm{W1} \leftarrow F_{(3)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow G_{(1)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow G_{(2)}(\mathrm{V1})\big] \\
\big[\mathrm{W1} \leftarrow G_{(3)}(\mathrm{V1})\big]
\end{gathered}$$

Ya que el predicado $Par = \lambda x[x \text{ es par}]$ es Σ-p.r., el Segundo Manantial de Macros nos dice que en $\mathcal{S}^\Sigma$ hay un macro:

$$[\text{IF } Par(\mathrm{V1}) \text{ GOTO A1}]$$

el cual escribiremos de la siguiente manera mas intuitiva

$$[\text{IF V1 es par GOTO A1}]$$

Ya que la funcion $D = \lambda x[\lfloor x/2 \rfloor]$ es Σ-p.r., el Segundo Manantial de Macros nos dice que hay un macro:

$$[\mathrm{V2} \leftarrow D(\mathrm{V1})]$$

el cual escribiremos de la siguiente manera mas intuitiva

$$[\mathrm{V2} \leftarrow \lfloor \mathrm{V1}/2 \rfloor]$$

Sea $\mathcal{P}$ el siguiente programa:

$$\begin{array}{ll}
 & [\text{IF N1 es par GOTO L1} \\
 & \mathrm{N1} \leftarrow \mathrm{N1} \dot{-} 1 \\
 & [\mathrm{N1111} \leftarrow \lfloor \mathrm{N1}/2 \rfloor] \\
 & \big[\mathrm{N1} \leftarrow G_{(1)}(\mathrm{N1111})\big] \\
 & \big[\mathrm{N2} \leftarrow G_{(2)}(\mathrm{N1111})\big] \\
 & \big[\mathrm{P1} \leftarrow G_{(3)}(\mathrm{N1111})\big] \\
 & \text{GOTO L2} \\
\mathrm{L1} & [\mathrm{N1111} \leftarrow \lfloor \mathrm{N1}/2 \rfloor] \\
 & \big[\mathrm{N1} \leftarrow F_{(1)}(\mathrm{N1111})\big] \\
 & \big[\mathrm{N2} \leftarrow F_{(2)}(\mathrm{N1111})\big] \\
 & \big[\mathrm{P1} \leftarrow F_{(3)}(\mathrm{N1111})\big] \\
\mathrm{L2} & \text{SKIP}
\end{array}$$

Es facil ver que $\mathcal{P}$ cumple (a) y (b) de (2) de la Proposicion de Caracterizacion de Σ-enumerabilidad de la Guia 7 por lo cual $S_1 \cup S_2$ es Σ-enumerable. ■

En forma analoga a lo hecho recien, se pueden probar, copiando la respectiva idea en el paradigma de la computabilidad efectiva, los siguientes resultados, los cuales son dejados como ejercicios.

Ejercicio 2: Use los macros asociados a las "bajadas" y a $*^{\leq}$ para dar un programa que enumere a $\omega \times \omega \times \Sigma^*$, es decir que cumpla (a) y (b) de (2) de la Proposicion de Caracterizacion de Σ-enumerabilidad de la Guia 7.

Ejercicio 3: Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-enumerables. Entonces $S_1 \cap S_2$ es Σ-enumerable. (Hacer el caso $n = 2$, $m = 1$ e inspirese en el paradigma efectivo)

En la Guia 3 probamos que si $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable entonces S es Σ-efectivamente enumerable. Via el uso de macros adecuados podemos copiar la idea de la prueba de dicho resultado para probar el siguiente

Lema 4. *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Si S es Σ-computable, entonces S es Σ-enumerable*

Ejercicio 4: Pruebe el lema anterior (haga el caso $n = 2$, $m = 1$)

Ejercicio 5: Si $S \subseteq \Sigma^*$ es Σ-enumerable, entonces

$$T = \{\alpha \in \Sigma^* : \text{existe } \beta \in S \text{ tal que } \alpha \text{ es subpalabra de } \beta\}$$

también es Σ-enumerable.

Ejercicio 6: Sean $S \subseteq \omega$ y $L \subseteq \{@, \uparrow\}^*$ tales que $(0, \varepsilon) \in S \times L$. Entonces $S \times L$ es $\{@, \uparrow\}$-enumerable si y solo si ambos S y L son $\{@, \uparrow\}$-enumerables

O sea que con el uso de nuestros poderosos macros asociados a funciones y predicados Σ-p.r. (gracias al Segundo Manatial) mas los que provee el Primer Manatial podemos simular los procedimientos efectivos realizados dentro del paradigma filosofico con programas concretos del lenguaje $\mathcal{S}^{\Sigma}$. Pero esto no es del todo asi, ya que los ejemplos vistos recien no hacen uso del concepto de "ejecucion de una cantidad de pasos", idea que en muchos diseños de nuestros procedimentos efectivos es usada. Por ejemplo si queremos "copiar" dentro del paradigama imperativo la prueba del resultado

- Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega$ es una funcion Σ-efectivamente computable, entonces D_f es Σ-efectivamente enumerable

notaremos la dificultad de aun no poder hablar de cantidad de pasos en la ejecucion del programa que compute a f. O sea nuestro Primer Manatial nos da un macro de asignacion para f pero no es claro de como correr una expansion de dicho macro una cierta cantidad de veces. Esto lo lograremos usando macros asociados a las funciones $Halt^{n,m}$, $E^{n,m}_{\#}$ y $E^{n,m}_{*}$ las cuales se usan en la batalla Godel vence a Neumann que viene a continuacion. Estas funciones seran clave a la hora de simular con programas a procedimientos efectivos que en su funcionamiento involucran el funcionamiento de otros procedimientos efectivos.

O sea que luego de ambas batallas entre Godel y Neumann, el paradigma imperativo se vera fortalecido sustancialmente. Tambien mas adelante en la Guia 9

veremos como los desarrollos hechos en ambas batallas entre Godel y Neumann fortalecen al paradigma funcional.

Godel vence a Neumann

Primero definiremos tres funciones las cuales contienen toda la informacion acerca del funcionamiento del lenguaje $\mathcal{S}^{\Sigma}$. Sean $n, m \in \omega$, fijos. Definamos

$$i^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega$$
$$E_{\#}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega^{[\mathbf{N}]}$$
$$E_{*}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \Sigma^{*[\mathbf{N}]}$$

de la siguiente manera

$$(i^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P})) = (1,(x_1,...,x_n,0,...),(\alpha_1,...,\alpha_m,\varepsilon,...))$$
$$(i^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P})) =$$
$$=$$
$$S_{\mathcal{P}}(i^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}))$$

Notese que

$$(i^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}))$$

es la descripcion instantanea que se obtiene luego de correr $\mathcal{P}$ una cantidad t de pasos partiendo del estado

$$((x_1,...,x_n,0,...),(\alpha_1,...,\alpha_m,\varepsilon,...))$$

Es importante notar que si bien $i^{n,m}$ es una funcion $(\Sigma \cup \Sigma_p)$-mixta, ni $E_{\#}^{n,m}$ ni $E_{*}^{n,m}$ lo son.

Definamos para cada $j \in \mathbf{N}$, funciones

$$E_{\#j}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega$$
$$E_{*j}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \Sigma^{*}$$

de la siguiente manera

$$E_{\#j}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) = j\text{-esima coordenada de } E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P})$$
$$E_{*j}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) = j\text{-esima coordenada de } E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P})$$

(es claro que estas funciones son $(\Sigma \cup \Sigma_p)$-mixtas). Notese que

$$E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) = (E_{\#1}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#2}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}),...)$$
$$E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) = (E_{*1}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*2}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}),...)$$

Aceptaremos sin prueba la siguiente proposicion (ver el apunte por una prueba).

Proposición 5. *Sean $n, m \geq 0$. Las funciones $i^{n,m}$, $E_{\#j}^{n,m}$, $E_{*j}^{n,m}$, $j = 1, 2, ...$, son $(\Sigma \cup \Sigma_p)$-p.r.*

Las funciones $Halt^{n,m}$ y $T^{n,m}$. Dados $n, m \in \omega$, definamos:

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

Notese que $D_{Halt^{n,m}} = \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma}$ (ojo que aqui la notacion lambda es respecto del alfabeto $\Sigma \cup \Sigma_p$). Ademas notese que usamos la variable $\mathcal{P}$ en la notacion lambda por un tema de comodidad psicologica dado que $i^{n,m}$ esta definida solo cuando la ultima coordenada es un programa pero podriamos haber escrito $\lambda t \vec{x} \vec{\alpha} \alpha \left[i^{n,m}(t, \vec{x}, \vec{\alpha}, \alpha) = n(\alpha) + 1 \right]$ y sigue siendo la misma funcion.

Cabe destacar que $Halt^{n,m}$ tiene una descripcion muy intuitiva, ya que dado $(t, \vec{x}, \vec{\alpha}, \mathcal{P}) \in \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma}$, tenemos que $Halt^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}) = 1$ si y solo si el programa $\mathcal{P}$ se detiene luego de t pasos partiendo desde el estado $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$. En particular $Halt^{0,0}$ tiene dominio igual a $\omega \times \text{Pro}^{\Sigma}$ y tenemos que $Halt^{0,0}(t, \mathcal{P}) = 1$ si y solo si el programa $\mathcal{P}$ se detiene luego de t pasos partiendo desde el estado $\|\Diamond\|$.

Aceptaremos sin demostracion el siguiente lema (ver el apunte por una prueba).

Lema 6. (a) Pro^{Σ} *es un conjunto* $(\Sigma \cup \Sigma_p)$*-p.r.*
(b) $\lambda \mathcal{P} \left[n(\mathcal{P}) \right]$ *y* $\lambda i \mathcal{P} \left[I_i^{\mathcal{P}} \right]$ *son funciones* $(\Sigma \cup \Sigma_p)$*-p.r..*

Ejercicio 7: De una funcion $f : \omega \times \text{Pro}^{\Sigma} \to \text{Pro}^{\Sigma}$ la cual sea $(\Sigma \cup \Sigma_p)$-p.r. y cumpla que
(a) $D_{\Psi^{0,1,\#}_{f(x,\mathcal{P})}} = D_{\Psi^{0,1,\#}_{\mathcal{P}}}$, para cada $x \in \omega$ y $\mathcal{P} \in \text{Pro}^{\Sigma}$
(b) $\Psi^{0,1,\#}_{f(x,\mathcal{P})}(\alpha) = \Psi^{0,1,\#}_{\mathcal{P}}(\alpha) + x$, para cada $x \in \omega$, $\mathcal{P} \in \text{Pro}^{\Sigma}$ y $\alpha \in D_{\Psi^{0,1,\#}_{\mathcal{P}}}$
Pruebe que f es $(\Sigma \cup \Sigma_p)$-p.r.

Ahora podemos probar el siguiente importante resultado.

Lema 7. $Halt^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-p.r.*

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

Ahora definamos $T^{n,m} = M(Halt^{n,m})$. Notese que

$$D_{T^{n,m}} = \{(\vec{x}, \vec{\alpha}, \mathcal{P}) \in \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} : \mathcal{P} \text{ se detiene partiendo de } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|\}$$

y para $(\vec{x}, \vec{\alpha}, \mathcal{P}) \in D_{T^{n,m}}$ tenemos que $T^{n,m}(\vec{x}, \vec{\alpha}, \mathcal{P}) =$ cantidad de pasos necesarios para que $\mathcal{P}$ se detenga partiendo de $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$. En algun sentido, la funcion $T^{n,m}$ mide el "tiempo" que tarda en detenerse $\mathcal{P}$ desde $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$ y de ahi su nombre. Notese que para el caso $n = m = 0$ tenemos que

$$D_{T^{0,0}} = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \mathcal{P} \text{ se detiene partiendo de } \|\Diamond\|\}$$

y para $\mathcal{P} \in D_{T^{0,0}}$ tenemos que $T^{0,0}(\mathcal{P}) =$ cantidad de pasos necesarios para que $\mathcal{P}$ se detenga partiendo de $\|\Diamond\|$.

Proposición 8. $T^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-recursiva*

Proof. Es directo del lema de minimizacion ya que $Halt^{n,m}$ es $(\Sigma \cup \Sigma_p)$-p.r. ■

Ahora nos sera facil probar que el paradigma de Godel es por lo menos tan abarcativo como el imperativo de Von Neumann. Mas concretamente:

Teorema 9 (Godel vence a Neumann)**.** *Si* $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ *es* Σ*-computable, entonces* f *es* Σ*-recursiva.*

Proof. Haremos el caso $O = \Sigma^*$. Sea $\mathcal{P}_0$ un programa que compute a f. Primero veremos que f es $(\Sigma \cup \Sigma_p)$-recursiva. Note que

$$f = E^{n,m}_{*1} \circ \left[T^{n,m} \circ \left[p^{n,m}_1, ..., p^{n,m}_{n+m}, C^{n,m}_{\mathcal{P}_0}\right], p^{n,m}_1, ..., p^{n,m}_{n+m}, C^{n,m}_{\mathcal{P}_0}\right]$$

donde cabe destacar que $p^{n,m}_1, ..., p^{n,m}_{n+m}$ son las proyecciones respecto del alfabeto $\Sigma \cup \Sigma_p$, es decir que tienen dominio $\omega^n \times (\Sigma \cup \Sigma_p)^{*m}$. Esto nos dice que f es $(\Sigma \cup \Sigma_p)$-recursiva. O sea que el Teorema de Independencia del Alfabeto nos dice que f es Σ-recursiva. ■

Aceptaremos sin prueba la siguiente proposicion.

Proposición 10. *La funcion* $T^{n,m}$ *no es* $(\Sigma \cup \Sigma_p)$*-p.r.*

Corolario 11. *En general la minimizacion de un predicado* Σ*-p.r. no necesariamente es* Σ*-p.r.*

Proof. Si consideramos el alfabeto $(\Sigma \cup \Sigma_p)$ tenemos que $T^{n,m} = M(Halt^{n,m})$ y $T^{n,m}$ no es $(\Sigma \cup \Sigma_p)$-p.r. ■

Uso de macros asociados a las funciones $Halt^{n,m}$**,** $E^{n,m}_{\#}$ **y** $E^{n,m}_{*}$**.** Aqui veremos, con ejemplos, como ciertos macros nos permitiran dentro de un programa hablar acerca del funcionamiento de otro programa. En este sentido los desarrollos de las dos batallas entre Neumann y Godel nos permiten fortalecer notablemente al paradigma imperativo en su roll modelizador (o simulador) de los procedimientos efectivos. Esto es importante ya que el paradigma mas comodo, amplio e intuitivo es sin duda el filosofico de Leibniz.

Veamos el primer ejemplo. Probaremos que:

- Si $f : D_f \subseteq \omega \to \omega$ es Σ-computable, $0 \in D_f$ y $f(0) = 2$, entonces
$$S = \{x \in D_f : f(x) \neq 0\}$$
es Σ-enumerable.

Notese que $0 \in S$ por lo cual S es no vacio asique en virtud de la Caracterizacion de Σ-enumerabilidad dada en la Guia 7, deberemos encontrar un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ que enumere a S, es decir tal que $\text{Dom}\Psi^{1,0,\#}_{\mathcal{P}} = \omega$ y $\text{Im}(\Psi^{1,0,\#}_{\mathcal{P}}) = S$. Dicho en palabras, el programa $\mathcal{P}$ debera cumplir:

- Siempre que lo corramos desde un estado de la forma $\|x\|$, con $x \in \omega$, debe detenerse y el contenido de la variable N1 bajo detencion debera ser un elemento de S
- Para cada $s \in S$ debera haber un $x \in \omega$ tal que s es el valor de la variable N1 bajo detencion cuando corremos $\mathcal{P}$ desde $\|x\|$

Sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$ un programa que compute a f. Usaremos $\mathcal{P}_0$ para diseñar $\mathcal{P}$. A continuacion daremos una descripcion intuitiva del funcionamiento de $\mathcal{P}$ (pseudocodigo) para luego escribirlo correctamente usando macros. El programa $\mathcal{P}$ comenzara del estado $\|x\|$ y hara las siguientes tareas

Etapa 1: si $x = 0$ ir a Etapa 6, en caso contrario ir a Etapa 2.
Etapa 2: calcular $(x)_1$ y $(x)_2$ e ir a Etapa 3.

Etapa 3: si $\mathcal{P}_0$ termina desde $\|(x)_1\|$ en $(x)_2$ pasos ir a Etapa 4, en caso contrario ir a Etapa 6
Etapa 4: si el valor que queda en N1 luego de correr $\mathcal{P}_0$ una cantidad $(x)_2$ de pasos, partiendo de $\|(x)_1\|$, es distinto de 0, entonces ir a Etapa 5. En caso contrario ir a Etapa 6.
Etapa 5: asignar a N1 el valor $(x)_1$ y terminar
Etapa 6: asignar a N1 el valor 0 y terminar

Notese que la descripcion anterior no es ni mas ni menos que un procedimiento efectivo (efectivisable) que enumera a S, y nuestra mision es simularlo dentro del lenguaje $\mathcal{S}^{\Sigma}$. Para esto usaremos varios macros. Ya que la funcion $f = \lambda x[(x)_1]$ es Σ-p.r., el Segundo Manantial de Macros nos dice que en $\mathcal{S}^{\Sigma}$ hay un macro:

$$[\mathrm{V2} \leftarrow f(\mathrm{V1})]$$

el cual escribiremos de la siguiente manera mas intuitiva:

$$[\mathrm{V2} \leftarrow (\mathrm{V1})_1]$$

Similarmente hay un macro:

$$[\mathrm{V2} \leftarrow (\mathrm{V1})_2]$$

Tambien, ya que el predicado $P = \lambda x[x = 0]$ es Σ-recursivo, hay un macro:

$$[\mathrm{IF}\ P(\mathrm{V1})\ \mathrm{GOTO\ A1}]$$

el cual escribiremos de la siguiente manera:

$$[\mathrm{IF\ V1} = 0\ \mathrm{GOTO\ A1}]$$

Definamos

$$H = \lambda tx\left[Halt^{1,0}(t, x, \mathcal{P}_0)\right]$$

Notar que $D_H = \omega^2$ y que H es Σ-mixta. Ademas sabemos que la funcion $Halt^{1,0}$ es $(\Sigma \cup \Sigma_p)$-p.r. por lo cual resulta facilmente que H es $(\Sigma \cup \Sigma_p)$-p.r.. Por la Proposicion de Independencia del Alfabeto tenemos que H es Σ-p.r.. O sea que el Segundo Manantial de Macros nos dice que en $\mathcal{S}^{\Sigma}$ hay un macro:

$$[\mathrm{IF}\ H(\mathrm{V1}, \mathrm{V2})\ \mathrm{GOTO\ A1}]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\mathrm{IF}\ Halt^{1,0}(\mathrm{V1}, \mathrm{V2}, \mathcal{P}_0)\ \mathrm{GOTO\ A1}\right]$$

Sea

$$g = \lambda tx\left[E^{1,0}_{\#1}(t, x, \mathcal{P}_0)\right]$$

Ya que g es Σ-recursiva (por que?), hay en $\mathcal{S}^{\Sigma}$ un macro:

$$[\mathrm{V3} \leftarrow g(\mathrm{V1}, \mathrm{V2})]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\mathrm{V3} \leftarrow E^{1,0}_{\#1}(\mathrm{V1}, \mathrm{V2}, \mathcal{P}_0)\right]$$

Ahora si podemos dar nuestro progama $\mathcal{P}$ que enumera a S:

$$
\begin{array}{ll}
 & \text{IF N1} \neq 0 \text{ GOTO L1} \\
 & \text{GOTO L2} \\
\text{L1} & [\text{N3} \leftarrow (\text{N1})_1] \\
 & [\text{N4} \leftarrow (\text{N1})_2] \\
 & \left[\text{IF } Halt^{1,0}(\text{N4}, \text{N3}, \mathcal{P}_0) \text{ GOTO L3}\right] \\
 & \text{GOTO L2} \\
\text{L3} & \left[\text{N5} \leftarrow E^{1,0}_{\#1}(\text{N4}, \text{N3}, \mathcal{P}_0)\right] \\
 & [\text{IF N5} = 0 \text{ GOTO L2}] \\
 & \text{N1} \leftarrow \text{N3} \\
 & \text{GOTO L4} \\
\text{L2} & \text{N1} \leftarrow 0 \\
\text{L4} & \text{SKIP}
\end{array}
$$

Para hacer la proxima tanda de ejercicios recomendamos al lector que repase la definicion de cuando "un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ enumera a un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$" (ver despues de la Caracterizacion de Σ-enumerabilidad dada en la Guia 7).

Ejercicio 8: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-computable y $(0, 0, \varepsilon) \in D_f$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto D_f.

Ejercicio 9: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : D_f \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-computable y $(0, 0, \varepsilon) \in D_f$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto $\text{Im}(f)$.

Ejercicio 10: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : L_1 \to L_2$, con $L_1, L_2 \subseteq \Sigma^*$, es Σ-computable y biyectiva. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que compute a f^{-1}.

Ejercicio 11: Sea $\Sigma = \{\#, \$\}$ y sea $f : D_f \subseteq \Sigma^* \to \omega$ una funcion Σ-computable tal que $f(\varepsilon) = 1$. Sea $L = \{\alpha \in D_f : f(\alpha) = 1\}$. De un programa $\mathcal{Q} \in \text{Pro}^{\Sigma}$ (usando macros) el cual enumere a L (explicado en video en granlogico.com).

Ejercicio 12: Sea $\Sigma = \{@, \%\}$ y sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$. Sea

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

Note que $(0, \varepsilon) \in S$. De un programa $\mathcal{Q} \in \text{Pro}^{\Sigma}$ (usando macros) que enumere al conjunto S.

Ejercicio 13: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ es Σ-computable, $0 \in S$ y $f(0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

Ejercicio 14: Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ es Σ-computable, $0 \in S$ y $f(0) = 0$. De un programa $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$ (usando macros) tal que

$$\text{Dom}(\Psi^{1,0,\#}_{\mathcal{P}_0}) = \{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

Ejercicio 15: Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{(x, y) \in S : (f(x, y), y) \in S \text{ y } f(f(x, y), y) = 0\}$$

Ejercicio 16: Sean $f : D_f \subseteq \omega \times \Sigma^* \to \omega$ y $g : D_g \subseteq \omega \times \Sigma^* \to \omega$ funciones Σ-computables tales que $D_f \cap D_g = \emptyset$. Sea $F : D_f \cup D_g \to \omega$ dada por

$$e \quad \to \quad \begin{cases} f(e) & \text{si } e \in D_f \\ g(e) & \text{si } e \in D_g \end{cases}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que compute a F.

Enumeracion de conjuntos de programas. Ya que los programas de $\mathcal{S}^{\Sigma}$ son palabras del alfabeto $\Sigma \cup \Sigma_p$, nos podemos preguntar cuando un conjunto L de programas es $(\Sigma \cup \Sigma_p)$-enumerable. Daremos un ejemplo. Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \Psi_{\mathcal{P}}^{1,0,\#}(10) = 10\}$$

Veremos que L es $(\Sigma \cup \Sigma_p)$-enumerable, dando un programa $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ que enumere a L, es decir tal que $\text{Dom}(\Psi_{\mathcal{Q}}^{1,0,*}) = \omega$ y $\text{Im}(\Psi_{\mathcal{Q}}^{1,0,*}) = L$. Cabe destacar que aqui hay en juego dos versiones de nuestro lenguaje imperativo, es decir enumeraremos un conjunto de programas de $\mathcal{S}^{\Sigma}$ usando un programa de $\mathcal{S}^{\Sigma \cup \Sigma_p}$. Sea $\leq$ un orden total sobre el conjunto $\Sigma \cup \Sigma_p$.

A continuacion daremos una descripcion intuitiva del funcionamiento de $\mathcal{Q}$ (pseudocodigo) para luego escribirlo correctamente usando macros. Notese que SKIP $\in L$. El programa $\mathcal{Q}$ comenzara del estado $\|x\|$ y hara las siguientes tareas

Etapa 1: si $x = 0$ ir a Etapa 6, en caso contrario ir a Etapa 2.
Etapa 2: calcular $(x)_1$, $(x)_2$ y $*^{\leq}((x)_1)$ e ir a Etapa 3.
Etapa 3: si $*^{\leq}((x)_1) \in \text{Pro}^{\Sigma}$ y termina partiendo desde $\|10\|$ en $(x)_2$ pasos ir a Etapa 4, en caso contrario ir a Etapa 6
Etapa 4: si el valor que queda en N1 luego de correr $*^{\leq}((x)_1)$ una cantidad $(x)_2$ de pasos, partiendo de $\|10\|$ es igual a 10, entonces ir a Etapa 5. En caso contrario ir a Etapa 6.
Etapa 5: asignar a P1 la palabra $*^{\leq}((x)_1)$ y terminar
Etapa 6: asignar a P1 la palabra SKIP y terminar

Notese que la descripcion anterior no es ni mas ni menos que un procedimiento efectivo que enumera a L, y nuestra mision es simularlo dentro del lenguaje $\mathcal{S}^{\Sigma \cup \Sigma_p}$. Para esto usaremos varios macros. Es importante notar que los macros que usaremos corresponden al lenguaje $\mathcal{S}^{\Sigma \cup \Sigma_p}$ ya que los usaremos en $\mathcal{Q}$ el cual sera un programa de $\mathcal{S}^{\Sigma \cup \Sigma_p}$.

Ya que las funciones $\lambda x[(x)_1]$ y $\lambda x[(x)_2]$ son $(\Sigma \cup \Sigma_p)$-recursivas el Segundo Manantial nos dice que hay macros en $\mathcal{S}^{\Sigma \cup \Sigma_p}$ asociados a estas funciones los cuales escribiremos de la siguiente manera mas intuitiva:

$$[\text{V2} \leftarrow (\text{V1})_1]$$
$$[\text{V2} \leftarrow (\text{V1})_2]$$

Ya que el predicado $P = \lambda x[x = 10]$ es $(\Sigma \cup \Sigma_p)$-recursivo tenemos en $\mathcal{S}^{\Sigma \cup \Sigma_p}$ su macro asociado el cual escribiremos de la siguiente manera:

$$[\text{IF V1} = 10 \text{ GOTO A1}]$$

Por un lema anterior sabemos que Pro^{Σ} es un conjunto $(\Sigma \cup \Sigma_p)$-p.r., por lo cual $\chi_{\text{Pro}^{\Sigma}}^{(\Sigma\cup\Sigma_p)^*}$ es $(\Sigma \cup \Sigma_p)$-p.r., y por lo tanto hay un macro

$$\left[\text{IF } \chi_{\text{Pro}^{\Sigma}}^{(\Sigma\cup\Sigma_p)^*}(\text{W1}) \text{ GOTO A1}\right]$$

el cual escribiremos de la siguiente manera

$$\left[\text{IF W1} \in \text{Pro}^{\Sigma} \text{ GOTO A1}\right]$$

Ya que el predicado $Halt^{1,0}$ es $(\Sigma \cup \Sigma_p)$-recursivo tenemos un macro asociado a el, el cual escribiremos de la siguiente forma

$$\left[\text{IF } Halt^{1,0}(\text{V1}, \text{V2}, \text{W1}) \text{ GOTO A1}\right]$$

Ya que $E_{\#1}^{1,0}$ es $(\Sigma \cup \Sigma_p)$-recursivo tenemos un macro asociado a ella, el cual escribiremos de la siguiente forma

$$\left[\text{V3} \leftarrow E_{\#1}^{1,0}(\text{V1}, \text{V2}, \text{W1})\right]$$

Tambien usaremos macros

$$[\text{V1} \leftarrow 10]$$
$$[\text{W1} \leftarrow \text{SKIP}]$$

(dejamos al lector hacerlos a mano o tambien se puede justificar su existencia via el Segundo Manantial aplicado a las funciones $C_{10}^{0,0}$ y $C_{\text{SKIP}}^{0,0}$).

Ahora si podemos hacer el programa $\mathcal{Q}$ de $\mathcal{S}^{\Sigma\cup\Sigma_p}$ que enumera a L:

```
    IF N1 ≠ 0 GOTO L1
    GOTO L2
L1  [N2 ← (N1)_1]
    [N3 ← (N1)_2]
    [P1 ← *^≤(N2)]
    [IF P1 ∈ Pro^Σ GOTO L3]
    GOTO L2
L3  [N4 ← 10]
    [IF Halt^{1,0}(N3, N4, P1) GOTO L4]
    GOTO L2
L4  [N5 ← E^{1,0}_{#1}(N3, N4, P1)]
    [IF N5 = 10 GOTO L5]
L2  [P1 ← SKIP]
L5  SKIP
```

Ejercicio 17: (Explicado en video en granlogico.com.) Sea $\Sigma = \{\#, \$\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists n, \alpha \text{ tales que } \Psi_{\mathcal{P}}^{2,1,*}(|\mathcal{P}|, n, \alpha) = \$\}$$

(a) Dar un programa $\mathcal{Q} \in \text{Pro}^{\Sigma\cup\Sigma_p}$ tal que $\text{Dom}(\Psi_{\mathcal{Q}}^{0,1,\#}) = L$

(b) Dar un programa $\mathcal{Q}' \in \text{Pro}^{\Sigma\cup\Sigma_p}$ tal que $\text{Dom}(\Psi_{\mathcal{Q}'}^{1,0,*}) = \omega$ y $\text{Im}(\Psi_{\mathcal{Q}'}^{1,0,*}) = L$

Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro.

Ejercicio 18: Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists \alpha \text{ tal que } \Psi_{\mathcal{P}}^{0,2,*}(\alpha, \alpha\alpha) = \alpha\alpha\alpha\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a L. Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro.

Ejercicio 19: Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(x, \mathcal{P}) \in \omega \times \text{Pro}^{\Sigma} : x \in \text{Dom}(\Psi_{\mathcal{P}}^{1,0,\#})\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a S. Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro.

Ejercicio 20: Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(x, \mathcal{P}) \in \omega \times \text{Pro}^{\Sigma} : \Psi_{\mathcal{P}}^{1,0,\#}(x) = x^2\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a S. Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro.

Ejercicio 21: Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{\mathcal{P}_1 \alpha \mathcal{P}_2 : \mathcal{P}_1, \mathcal{P}_2 \in \text{Pro}^{\Sigma}, \alpha \in \Sigma^* \text{ y } \Psi_{\mathcal{P}_1}^{0,1,*}(\alpha) = \Psi_{\mathcal{P}_2}^{0,1,*}(\alpha)\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a S. Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro.

Cuando $\Sigma \supseteq \Sigma_p$ la cosa se pone mas loca...

Cuando $\Sigma \supseteq \Sigma_p$ podemos correr un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ partiendo de un estado que asigne a sus variables alfabeticas programas (ya que los programas son meras palabras de Σ^*). En particular podriamos correr un programa $\mathcal{P}$ desde el estado $\|\mathcal{P}\|$. Llamaremos A al conjunto formado por aquellos programas $\mathcal{P}$ tales que $\mathcal{P}$ se detiene partiendo del estado $\|\mathcal{P}\|$. Es decir

$$A = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists t \in \omega \text{ tal que } Halt^{0,1}(t, \mathcal{P}, \mathcal{P}) = 1\}$$

Por ejemplo SKIP $\in A$. Dicho rapida y sugestivamente A es el conjunto formado por aquellos programas que se detienen partiendo de si mismos. Dejamos como ejercicio la tarea de hacer un programa que enumere a A. Como veremos en la Guia 9, el conjunto A, si bien es Σ-enumerable, no es Σ-computable.

Ahora consiremos el conjunto

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \Psi_{\mathcal{P}}^{0,0,*}(\Diamond) = \mathcal{P}\}$$

En palabras, un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ estara en L si y solo si $\mathcal{P}$ termina desde $((0, 0, ...), (\varepsilon, \varepsilon, ...))$ dejando en P1 la palabra $\mathcal{P}$. En algun sentido los elementos de L son programas que se autopropagandean ya que "de la nada ellos terminan dandose a si mismos como salida". Dejamos al lector la tarea de reflexionar hasta entender que resulta dificil encontrar "a mano" un elemento de L. En algun sentido, para lograr hacer un programa que este en L debemos ir escribiendo instrucciones que a su ves vayan garantizando que ellas mismas vayan quedando en el contenido de la variable P1. Sin embargo el Teorema de la Recursion (de Kleene) nos garantiza que L es no vacio! Dejamos como ejercicio la tarea de hacer un programa que enumere a L (obviamente utilizando un elemento fijo de L).

Consideremos ahora el conjunto

$$S = \{(\mathcal{P}_1, \mathcal{P}_2) \in \text{Pro}^{\Sigma} \times \text{Pro}^{\Sigma} : \Psi_{\mathcal{P}_1}^{0,0,*}(\Diamond) = \mathcal{P}_1\mathcal{P}_2 = \Psi_{\mathcal{P}_2}^{0,0,*}(\Diamond)\}$$

Note que nuevamente es dificil imaginar como uno programaria un par de programas $\mathcal{P}_1$ y $\mathcal{P}_2$ de manera que $(\mathcal{P}_1, \mathcal{P}_2)$ este en S. El Teorema de la Recursion Doble (de Smullyan) nos garantiza que S es no vacio! Dejamos al lector la tarea de hacer un programa que enumere a L (obviamente utilizando un elemento fijo de S).

Ejercicio 22: Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^\Sigma$ que enumere a A.

Ejercicio 23: Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$L = \{\mathcal{P} \in \text{Pro}^\Sigma : \Psi_{\mathcal{P}}^{0,0,*}(\Diamond) = \mathcal{P}\}$$

De un programa de $\mathcal{S}^\Sigma$ que enumere a L.

Ejercicio 24: Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$L = \{\mathcal{P} \in \text{Pro}^\Sigma : \Psi_{\mathcal{P}}^{2,1,*}(1, 1, \mathcal{P}) = \mathcal{P}\mathcal{P}\}$$

Encuentre un elemento concreto de L y de un programa de $\mathcal{S}^\Sigma$ que enumere a L

Ejercicio 25: Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^\Sigma$ que enumere al conjunto

$$L = \{\mathcal{P} \in \text{Pro}^\Sigma : \exists x \text{ tal que } \Psi_{\mathcal{P}}^{1,1,*}(x, \mathcal{P}) = I_1^{\mathcal{P}}\}$$

Ejercicio 26: Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^\Sigma$ que enumere al conjunto

$$S = \{(x, \mathcal{P}, \alpha) \in \omega \times \text{Pro}^\Sigma \times \Sigma^* : \Psi_{\mathcal{P}}^{1,2,\#}(x, \alpha, \mathcal{P}^x) = 0\}$$

Ejercicio 27: Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$S = \{(\mathcal{P}_1, \mathcal{P}_2) \in \text{Pro}^\Sigma \times \text{Pro}^\Sigma : \Psi_{\mathcal{P}_1}^{0,0,*}(\Diamond) = \mathcal{P}_1\mathcal{P}_2 = \Psi_{\mathcal{P}_2}^{0,0,*}(\Diamond)\}$$

De un programa de $\mathcal{S}^\Sigma$ que enumere al conjunto S

Dos batallas mas

Aceptaremos sin prueba el siguiente teorema.

Teorema 12 (Godel vence a Turing)**.** *Supongamos $f : S \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-Turing computable. Entonces f es Σ-recursiva.*

Ejercicio 28: Haga un esquema a nivel de ideas (sin demostraciones y sin demaciada precision) de la prueba del teorema anterior (la prueba completa esta en el apunte).

Aceptaremos sin prueba el siguiente teorema.

Teorema 13 (Turing vence a Neumann)**.** *Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-computable, entonces f es Σ-Turing computable.*

Ejercicio 29: Haga un esquema a nivel de ideas (sin demostraciones y sin demaciada precision) de la prueba del teorema anterior (la prueba completa esta en el apunte y en granlogico.com hay un video con la prueba del teorema)

La tesis de Church

En virtud de los teoremas ya expuestos tenemos el siguiente teorema que asegura que los tres paradigmas son equivalentes.

Teorema 14. *Sea Σ un alfabeto finito. Dada una funcion f, las siguientes son equivalentes:*

(1) *f es Σ-Turing computable*
(2) *f es Σ-recursiva*
(3) *f es Σ-computable.*

Tambien los tres paradigmas son equivalentes con respecto a los dos tipos de conjuntos estudiados, es decir:

Teorema 15. *Sea Σ un alfabeto finito y sea $S \subseteq \omega^n \times \Sigma^{*m}$. Las siguientes son equivalentes:*

(1) *S es Σ-Turing enumerable*
(2) *S es Σ-recursivamente enumerable*
(3) *S es Σ-enumerable*

Teorema 16. *Sea Σ un alfabeto finito y sea $S \subseteq \omega^n \times \Sigma^{*m}$. Las siguientes son equivalentes:*

(1) *S es Σ-Turing computable*
(2) *S es Σ-recursivo*
(3) *S es Σ-computable*

Ejercicio 30: Pruebe los dos teoremas anteriores

Otro modelo matematico de computabilidad efectiva es el llamado lambda calculus, introducido por Church, el cual tambien resulta equivalente a los estudiados por nosotros. El hecho de que tan distintos paradigmas computacionales hayan resultado equivalentes hace pensar que en realidad los mismos han tenido exito en capturar la totalidad de las funciones Σ-efectivamente computables. Esta aceveracion es conocida como la

Tesis de Church: *Toda funcion Σ-efectivamente computable es Σ-recursiva.*

Y por supuesto puede ser sintetizada diciendo que Godel vence a Leibniz (y por lo tanto los cuatro proceres empatan!). Si bien no se ha podido dar una prueba estrictamente matematica de la Tesis de Church, es un sentimiento comun de los investigadores del area que la misma es verdadera.

Ejercicios de examen

Para cada macro que use dar la funcion o predicado asociado y justificar por que existe tal macro usando alguno de los dos Manantiales de Macros (el primer Manantial esta en la Guia 7 y el segundo en esta guia). Note que si ud esta haciendo un programa de $\mathcal{S}^{\Sigma}$ y quiere usar los Manantiales de Macros para producir un macro, la funcion a la cual le aplicara el respectivo manatial debe ser Σ-mixta. O sea que cuando se dirija a alguno de los manatiales de macros para obtener un macro, antes piense bien en que "$\mathcal{S}^{\Sigma}$" esta ud programando.

(1) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : L_1 \to L_2$, con $L_1, L_2 \subseteq \Sigma^*$, es Σ-computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que compute a f^{-1}.

(2) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S_1 \subseteq \omega \times \Sigma^*$ y $S_2 \subseteq \omega$. Supongamos ademas que $f : S_1 \to S_2$ es Σ-computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) De un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ (usando macros) tal que $f^{-1} = [\Psi_{\mathcal{P}}^{1,0,\#}, \Psi_{\mathcal{P}}^{1,0,*}]$.

(3) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S \subseteq \omega \times \Sigma^*$ y $L \subseteq \Sigma^*$. Supongamos ademas que $f : S \to L$ es Σ-computable y biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion f^{-1}.
   (b) De un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ (usando macros) tal que $f^{-1} = [\Psi_{\mathcal{P}}^{0,1,\#}, \Psi_{\mathcal{P}}^{0,1,*}]$.

(4) Sea $\Sigma = \{@, !, \%\}$. Supongamos $S_1 \subseteq \omega$ y $S_2 \subseteq \omega \times \Sigma^*$. Supongamos ademas que $F : S_1 \to S_2$ es biyectiva.
   (a) Defina (dando dominio, imagen y regla) la funcion F^{-1}.
   (b) Suponga que $F_{(1)}$ y $F_{(2)}$ son Σ-computables. De un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ (usando macros) que compute a F^{-1}.

(5) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-computable. Suponga ademas que $(0, 0, \varepsilon) \in S$ y $f(0, 0, \varepsilon) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{(x, y, \alpha) \in S : f(x, y, \alpha) = 0\}$$

(6) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{(x, y) \in S : f(x, y) = x\}$$

(7) Sea $\Sigma = \{@, \&\}$. Supongamos $f : L \subseteq \Sigma^* \times \Sigma^* \to \Sigma^*$ es Σ-computable. Suponga ademas que $(@, \&) \in L$ y $f(@, \&) = @@@$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{(\alpha, \beta) \in L : f(\alpha, \beta) = \alpha\alpha\alpha\}$$

(8) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{(x, y) \in S : (y, x) \in S \text{ y } f(x, y) = f(y, x)\}$$

(9) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$

(usando macros) que enumere al conjunto

$$H = \{(x, y) \in S : (f(x, y), y) \in S \text{ y } f(f(x, y), y) = 0\}$$

(10) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

$$H = \{x \in \omega : \text{ hay un } y\text{tal que } (x, y) \in S \text{ y } f(x, y) = 0\}$$

(11) Sea $\Sigma = \{@, \&\}$. Supongamos $f : S \subseteq \omega \times \omega \to \omega$ es Σ-computable. Suponga ademas que $(0, 0) \in S$ y $f(0, 0) = 0$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

$$H = \{(x, x) : (x, x) \in S \text{ y } f(x, x) = x\}$$

(12) Sea $\Sigma = \{@, \&\}$. Supongamos $f : L \subseteq \Sigma^* \times \Sigma^* \to \Sigma^*$ es Σ-computable. Suponga ademas que $(@, @) \in L$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

$$H = \{(\alpha, \beta) \in L : (\beta, \alpha) \in L \text{ y } f(\alpha, \beta) = f(\beta, \alpha)\}$$

(13) Sea $\Sigma = \{@, \&\}$. Sea $f : S \subseteq \Sigma^* \to \omega$ una funcion Σ-computable. Supongamos que $\varepsilon \in S$ y que $f(\varepsilon) = 0$. Notar que $\varepsilon \in L$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

$$L = \{\alpha \in S : @^{f(\alpha)} \in S \text{ y } f(@^{f(\alpha)}) = 0\}$$

(14) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \times \omega \times \Sigma^* \to \omega$ es Σ-computable. Suponga ademas que $(0, 0, \varepsilon) \in S$ y $f(0, 0, \varepsilon) = 0$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

$$H = \{(x, y, \alpha) \in S : (y, x, \alpha) \in S \text{ y } f(x, y, \alpha) = f(y, x, \alpha)\}$$

(15) Sea $\Sigma = \{@, \%\}$ y sea $\mathcal{P}_0 \in \text{Pro}^\Sigma$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto

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

(note que $\varepsilon \in L$).

(16) Sea $\Sigma = \{@, \%\}$ y sea $\mathcal{P}_0 \in \text{Pro}^\Sigma$. Sea

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

Note que $(0, \varepsilon) \in S$. De un programa $\mathcal{Q} \in \text{Pro}^\Sigma$ (usando macros) que enumere al conjunto S.

(17) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f_1 : S_1 \subseteq \omega \times \omega \times \Sigma^* \to \omega$ y $f_2 : S_2 \subseteq \omega \times \omega \times \Sigma^* \to \Sigma^*$ son Σ-computables y $(0, 0, \varepsilon) \in S_1 \cap S_2$. De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto $\text{Im}([f_1, f_2])$.

(18) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ y $g : L \subseteq \Sigma^* \to \omega$ son Σ-computables. Supongamos ademas que 0 pertenece al conjunto

$$H = \{x : x \in S,\ @^x \in L \text{ y } f(x) = g(@^x)\}$$

De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto H.

(19) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ y $g : L \subseteq \Sigma^* \to \omega$ son Σ-computables. Supongamos ademas que ε pertenece al conjunto

$$H = \{\alpha \in L : g(\alpha) \in S \text{ y } f(g(\alpha)) = 0\}$$

De un programa de $\mathcal{S}^\Sigma$ (usando macros) que enumere al conjunto H.

(20) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ es Σ-computable, $0 \in S$ y $f(0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{x \in S : x^2 \in S \text{ y } f(x) = f(x^2)\}$$

(21) Sea $\Sigma = \{@, !, \%\}$. Supongamos $f : S \subseteq \omega \to \omega$ es Σ-computable, $0 \in S$ y $f(0) = 0$. De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto

$$H = \{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

(22) Sea $\Sigma = \{@, !, \%\}$. Supongamos $L_1, L_2 \subseteq \Sigma^*$ son Σ-enumerables. Supongamos ademas que ε pertenece al conjunto

$$H = \{\alpha \in L_1 : \text{hay un } \beta \in L_2 \text{ tal que } |\beta| = |\alpha|\}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto H.

(23) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \omega \times \omega \times \Sigma^*$ y $L \subseteq \Sigma^*$ son Σ-enumerables. Supongamos ademas que $(0, 0, \varepsilon)$ pertenece al conjunto

$$H = \{(x, y, \alpha) \in S : \text{hay un } \beta \in L \text{ tal que } |\beta| = |\alpha|\}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto H.

(24) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \omega \times \omega$ y $S' \subseteq \omega$ son Σ-enumerables. Supongamos ademas que $(0, 0)$ pertenece al conjunto

$$H = \{(x, y) : (x, y) \in S \text{ y } x + y \in S'\}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto H.

(25) Sea $\Sigma = \{!, \%\}$. Supongamos $S \subseteq \Sigma^* \times \Sigma^*$ y $L \subseteq \Sigma^*$ son Σ-enumerables. Supongamos ademas que $(\varepsilon, \varepsilon)$ pertenece al conjunto

$$H = \{(\alpha, \beta) \in S : \text{hay un } \gamma \in L \text{ tal que } \alpha = \beta\gamma\}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto H.

(26) Sean $f : D_f \subseteq \omega \times \Sigma^* \to \omega$ y $g : D_g \subseteq \omega \times \Sigma^* \to \omega$ funciones Σ-computables tales que $D_f \cap D_g = \emptyset$. Sea $F : D_f \cup D_g \to \omega$ dada por

$$e \quad \to \quad \begin{cases} f(e) & \text{si } e \in D_f \\ g(e) & \text{si } e \in D_g \end{cases}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que compute a F.

(27) Sea $\Sigma = \{@, !, \%\}$. Supongamos $L_1, L_2 \subseteq \Sigma^*$ son Σ-enumerables. Supongamos ademas que ε pertenece al conjunto

$$H = \{\alpha \in L_1 : \text{hay un } \beta \in L_2 \text{ tal que } \beta^3 = \alpha\}$$

De un programa de $\mathcal{S}^{\Sigma}$ (usando macros) que enumere al conjunto H.

(28) Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(x, \mathcal{P}, \alpha) \in \omega \times \text{Pro}^{\Sigma} \times \Sigma^* : \Psi_{\mathcal{P}}^{1,1,\#}(x, \alpha) = 0\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a S.

(29) Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(\mathcal{P}_1, \mathcal{P}_2) \in \text{Pro}^{\Sigma} \times \text{Pro}^{\Sigma} : \exists \alpha \text{ tal que } \Psi_{\mathcal{P}_1}^{0,1,*}(\alpha) = \Psi_{\mathcal{P}_2}^{0,1,*}(\alpha)\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ el cual enumere a S.

(30) Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists \alpha \text{ tal que } \Psi_{\mathcal{P}}^{1,1,*}(11, \alpha) = \alpha!@\}$$

(a) Dar un programa $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ tal que $\text{Dom}(\Psi_{\mathcal{Q}}^{0,1,\#}) = L$

(b) Dar un programa $\mathcal{Q}' \in \text{Pro}^{\Sigma\cup\Sigma_p}$ tal que $\text{Dom}(\Psi^{1,0,*}_{\mathcal{Q}'}) = \omega$ y $\text{Im}(\Psi^{1,0,*}_{\mathcal{Q}'}) = L$

(31) Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists \alpha \text{ tal que } \Psi^{0,2,*}_{\mathcal{P}}(\alpha, \alpha\alpha) = \alpha\alpha\alpha\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma\cup\Sigma_p}$ el cual enumere a L.

(32) Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(x, \mathcal{P}) \in \omega \times \text{Pro}^{\Sigma} : x \in \text{Dom}(\Psi^{1,0,\#}_{\mathcal{P}})\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma\cup\Sigma_p}$ el cual enumere a S.

(33) Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{(x, \mathcal{P}) \in \omega \times \text{Pro}^{\Sigma} : \Psi^{1,0,\#}_{\mathcal{P}}(x) = x^2\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma\cup\Sigma_p}$ el cual enumere a S.

(34) Sea $\Sigma = \{@, \%\}$ y sea

$$S = \{\mathcal{P}_1\alpha\mathcal{P}_2 : \mathcal{P}_1, \mathcal{P}_2 \in \text{Pro}^{\Sigma}, \alpha \in \Sigma^* \text{ y } \Psi^{0,1,*}_{\mathcal{P}_1}(\alpha) = \Psi^{0,1,*}_{\mathcal{P}_2}(\alpha)\}$$

Dar $\mathcal{Q} \in \text{Pro}^{\Sigma\cup\Sigma_p}$ el cual enumere a S.

(35) Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \Psi^{2,1,*}_{\mathcal{P}}(1, 1, \mathcal{P}) = \mathcal{P}\mathcal{P}\}$$

Encuentre un elemento concreto de L y de un programa de $\mathcal{S}^{\Sigma}$ que enumere a L

(36) Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^{\Sigma}$ que enumere al conjunto

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists x \text{ tal que } \Psi^{1,1,*}_{\mathcal{P}}(x, \mathcal{P}) = I^{\mathcal{P}}_1\}$$

(37) Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^{\Sigma}$ que enumere al conjunto

$$S = \{(x, \mathcal{P}, \alpha) \in \omega \times \text{Pro}^{\Sigma} \times \Sigma^* : \Psi^{1,2,\#}_{\mathcal{P}}(x, \alpha, \mathcal{P}^x) = 0\}$$

(38) Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$S = \{(\mathcal{P}_1, \mathcal{P}_2) \in \text{Pro}^{\Sigma} \times \text{Pro}^{\Sigma} : \mathcal{P}_1 \neq \mathcal{P}_2 \text{ y } \Psi^{0,1,*}_{\mathcal{P}_1}(\mathcal{P}_2) = \Psi^{0,1,*}_{\mathcal{P}_2}(\mathcal{P}_1)\}$$

Note que $(\text{P1} \leftarrow \varepsilon\text{SKIP}, \text{P1} \leftarrow \varepsilon) \in S$. De un programa de $\mathcal{S}^{\Sigma}$ que enumere al conjunto S

GUIA 9 DE LENGUAJES: RESULTADOS BASICOS PRESENTADOS EN PARADIGMA RECURSIVO

En esta guia presentaremos varios resultados basicos de computabilidad, expresados en el paradigma recursivo, ya que es el mas habitual y comodo. Varios de estos resultados pueden ser establecidos y probados en forma natural dentro del paradigma de la computabilidad efectiva (ver apunte). A ellos los enunciaremos dentro del paradigma de Godel y los probaremos rigurosamente usando la teoria desarrollada hasta ahora. Sin envargo, veremos que hay otros resultados que son dependientes del desarrollo matematico hecho y aportan nueva informacion al paradigma filosofico (la indecidibilidad del halting problem, por ejemplo).

Como veremos muchas de las pruebas seran de naturaleza imperativa basadas en la equivalencia del paradigma de Godel con el imperativo de Neumann.

Lema de division por casos para funciones Σ-recursivas

Usando los resultados probados en las dos anteriores batallas entre los paradigmas de Godel y Neumann podemos probar el siguiente resultado el cual a priori no parese facil de probar si nos quedamos solo en el contexto del paradigma Godeliano.

Lema 1. *Supongamos $f_i : D_{f_i} \subseteq \omega^n \times \Sigma^{*m} \to O$, $i = 1, ..., k$, son funciones Σ-recursivas tales que $D_{f_i} \cap D_{f_j} = \emptyset$ para $i \neq j$. Entonces la funcion $f_1 \cup ... \cup f_k$ es Σ-recursiva.*

Proof. Probaremos el caso $k = 2$ y $O = \Sigma^*$. Ademas supondremos que $n = m = 1$. Sean $\mathcal{P}_1$ y $\mathcal{P}_2$ programas que computen las funciones f_1 y f_2, respectivamente. Para $i = 1, 2$, definamos

$$H_i = \lambda t x_1 \alpha_1 \left[Halt^{1,1}(t, x_1, \alpha_1, \mathcal{P}_i) \right]$$

Notar que $D_{H_i} = \omega^2 \times \Sigma^*$ y que H_i es Σ-mixta. Ademas sabemos que la funcion $Halt^{1,1}$ es $(\Sigma \cup \Sigma_p)$-p.r. por lo cual resulta facilmente que H_i es $(\Sigma \cup \Sigma_p)$-p.r.. Por el Teorema de Independencia del Alfabeto tenemos que H_i es Σ-p.r.. y por lo tanto el Segundo Manantial de Macros nos dice que en $\mathcal{S}^\Sigma$ hay un macro:

$$[\text{IF } H_i(\text{V1}, \text{V2}, \text{W1}) \text{ GOTO A1}]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\text{IF } Halt^{1,1}(\text{V1}, \text{V2}, \text{W1}, \mathcal{P}_i) \text{ GOTO A1}\right]$$

Ya que cada f_i es Σ-recursiva, el Segundo Manantial nos dice que en $\mathcal{S}^\Sigma$ hay macros

$$[\text{W2} \leftarrow f_1(\text{V1}, \text{W1})]$$
$$[\text{W2} \leftarrow f_2(\text{V1}, \text{W1})]$$

Sea $\mathcal{P}$ el siguiente programa:

$$\begin{array}{ll}
\text{L1} & \text{N20} \leftarrow \text{N20} + 1 \\
& \left[\text{IF } Halt^{1,1}(\text{N20}, \text{N1}, \text{P1}, \mathcal{P}_1) \text{ GOTO L2}\right] \\
& \left[\text{IF } Halt^{1,1}(\text{N20}, \text{N1}, \text{P1}, \mathcal{P}_2) \text{ GOTO L3}\right] \\
& \text{GOTO L1} \\
\text{L2} & [\text{P1} \leftarrow f_1(\text{N1}, \text{P1})] \\
& \text{GOTO L4} \\
\text{L3} & [\text{P1} \leftarrow f_2(\text{N1}, \text{P1})] \\
\text{L4} & \text{SKIP}
\end{array}$$

Notese que $\mathcal{P}$ computa la funcion $f_1 \cup f_2$. ■

Ejercicio 1: Si $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$ es un predicado Σ-recursivo y D_P es Σ-recursivo, entonces la funcion $M(P)$ es Σ-recursiva.

Lema de restriccion de funciones Σ-recursivas. Nos sera util tambien el siguiente resultado.

Lema 2. *Supongamos $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-recursiva y $S \subseteq D_f$ es Σ-r.e., entonces $f|_S$ es Σ-recursiva.*

Proof. Si $S = \emptyset$, entonces $f|_S = \emptyset$ y por lo tanto $f|_S$ es Σ-recursiva. Supongamos $S \neq \emptyset$. Haremos el caso $n = m = 1$ y $O = \Sigma^*$. Tenemos que hay una $F : \omega \to \omega \times \Sigma^*$ tal que $\text{Im}(F) = S$ y $F_{(1)}$, $F_{(2)}$ son Σ-recursivas. Por el Segundo Manantial en $\mathcal{S}^\Sigma$ hay macros

$$\begin{array}{l}
[\text{W2} \leftarrow f(\text{V1}, \text{W1})] \\
\left[\text{V2} \leftarrow F_{(1)}(\text{V1})\right] \\
\left[\text{W1} \leftarrow F_{(2)}(\text{V1})\right]
\end{array}$$

Ya que los predicados $D = \lambda xy[x \neq y]$ y $D' = \lambda\alpha\beta[\alpha \neq \beta]$ son Σ-p.r., en $\mathcal{S}^\Sigma$ hay macros

$$\begin{array}{l}
[\text{IF } D(\text{V1}, \text{V2}) \text{ GOTO A1}] \\
[\text{IF } D'(\text{W1}, \text{W2}) \text{ GOTO A1}]
\end{array}$$

Para hacer mas amigable la lectura los escribieremos de la siguiente manera

$$\begin{array}{l}
[\text{IF V1} \neq \text{V2 GOTO A1}] \\
[\text{IF W1} \neq \text{W2 GOTO A1}]
\end{array}$$

Sea $\mathcal{P}$ el siguiente programa

$$\begin{array}{ll}
\text{L2} & [\text{N2} \leftarrow F_{(1)}(\text{N20})] \\
& [\text{P2} \leftarrow F_{(2)}(\text{N20})] \\
& [\text{IF N1} \neq \text{N2 GOTO L1}] \\
& [\text{IF P1} \neq \text{P2 GOTO L1}] \\
& [\text{P1} \leftarrow f(\text{N1}, \text{P1})] \\
& \text{GOTO L3} \\
\text{L1} & \text{N20} \leftarrow \text{N20} + 1 \\
& \text{GOTO L2} \\
\text{L3} & \text{SKIP}
\end{array}$$

Es facil ver que $\mathcal{P}$ computa a $f|_S$. ■

Conjuntos Σ-r.e. y Σ-r. Daremos primero algunas propiedades basicas de los conjuntos Σ-r.e. y Σ-r. El siguiente resultado puede probarse facilmente dentro del paradigma Godeliano y lo dejamos como ejercicio.

Ejercicio 2: Sea Σ un alfabeto finito. Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-recursivos. Entonces $S_1 \cup S_2$, $S_1 \cap S_2$ y $S_1 - S_2$ son Σ-recursivos

Lema 3. *Sea Σ un alfabeto finito. Se tiene que*

(1) *Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-r.e.. Entonces $S_1 \cup S_2$ es Σ-r.e.*
(2) *Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-r.e.. Entonces $S_1 \cap S_2$ es Σ-r.e.*
(3) *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Si S es Σ-recursivo, entonces S es Σ-recursivamente enumerable*

Proof. Los tres resultados en su version imperativa fueron probados o dejados como ejercicio en la Guia 8, por lo que podemos aplicar los teoremas que nos dicen que los paradigmas recursivo e imperativo son equivalentes. ■

Tal como veremos mas adelante hay conjuntos Σ-recursivamente enumerables los cuales no son Σ-recursivos. Sin envargo tenemos el siguiente interesante resultado.

Lema 4. *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Si S y $(\omega^n \times \Sigma^{*m}) - S$ son Σ-recursivamente enumerables, entonces S es Σ-recursivo*

Proof. (Prueba cheta) Por definicion, para probar que S es Σ-recursivo deberemos probar que $\chi_S^{\omega^n \times \Sigma^{*m}}$ es Σ-recursiva. Notese que $\chi_S^{\omega^n \times \Sigma^{*m}} = C_1^{n,m}|_S \cup C_0^{n,m}|_{(\omega^n \times \Sigma^{*m}) - S}$. O sea que por el Lema de Division por Casos, solo nos resta probar que $C_1^{n,m}|_S$ y $C_0^{n,m}|_{(\omega^n \times \Sigma^{*m}) - S}$ son Σ-recursivas. Pero esto se desprende directamente del Lema 2 ya que $C_1^{n,m}$ y $C_0^{n,m}$ son Σ-recursivas y por hipotesis S y $(\omega^n \times \Sigma^{*m}) - S$ son Σ-recursivamente enumerables. ■

Ejercicio 3: (Prueba intuitiva) Dar una prueba imperativa del lema anterior. Hint: inspirese en su analogo dentro del paradigma de la computabilidad efectiva, dado al final de la Guia 3.

El siguiente teorema es muy importante ya que caracteriza a los conjuntos Σ-r.e.

Teorema 5. *Dado $S \subseteq \omega^n \times \Sigma^{*m}$, son equivalentes*

(1) *S es Σ-recursivamente enumerable*
(2) *$S = I_F$, para alguna $F : D_F \subseteq \omega^k \times \Sigma^{*l} \to \omega^n \times \Sigma^{*m}$ tal que cada $F_{(i)}$ es Σ-recursiva.*
(3) *$S = D_f$, para alguna funcion Σ-recursiva f*
(4) *$S = \emptyset$ o $S = I_F$, para alguna $F : \omega \to \omega^n \times \Sigma^{*m}$ tal que cada $F_{(i)}$ es Σ-p.r.*

Proof. El caso $n = m = 0$ es facil y es dejado al lector. Supongamos entonces que $n + m \geq 1$.

(2)⇒(3). Haremos el caso $k = l = 1$ y $n = m = 2$. El caso general es completamente analogo. Notese que entonces tenemos que $S \subseteq \omega^2 \times \Sigma^{*2}$ y $F : D_F \subseteq \omega \times \Sigma^* \to \omega^2 \times \Sigma^{*2}$ es tal que $\operatorname{Im} F = S$ y $F_{(1)}$, $F_{(2)}$, $F_{(3)}$, $F_{(4)}$ son Σ-recursivas. Para cada $i \in \{1, 2, 3, 4\}$, sea $\mathcal{P}_i$ un programa el cual computa a $F_{(i)}$. Sea $\leq$ un orden total sobre Σ. Definamos

$$H_i = \lambda t x_1 \alpha_1 \left[\neg Halt^{1,1}(t, x_1, \alpha_1, \mathcal{P}_i)\right]$$

Notar que $D_{H_i} = \omega^2 \times \Sigma^*$ y que H_i es Σ-mixta. Ademas sabemos que la funcion $Halt^{1,1}$ es $(\Sigma \cup \Sigma_p)$-p.r. por lo cual resulta facilmente que H_i es $(\Sigma \cup \Sigma_p)$-p.r.. Por la Proposicion de Independencia del Alfabeto tenemos que H_i es Σ-p.r., lo cual por el Segundo Manantial nos dice que hay un macro:

$$[\text{IF } H_i(\text{V2}, \text{V1}, \text{W1}) \text{ GOTO A1}]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\text{IF } \neg Halt^{1,1}(\text{V2}, \text{V1}, \text{W1}, \mathcal{P}_i) \text{ GOTO A1}\right]$$

Para $i = 1, 2$, definamos

$$E_i = \lambda x t x_1 \alpha_1 \left[x \neq E^{1,1}_{\#1}(t, x_1, \alpha_1, \mathcal{P}_i)\right]$$

Para $i = 3, 4$, definamos

$$E_i = \lambda t x_1 \alpha_1 \alpha \left[\alpha \neq E^{1,1}_{*1}(t, x_1, \alpha_1, \mathcal{P}_i)\right]$$

Dejamos al lector probar que las funciones E_i son Σ-p.r.. O sea que para cada $i \in \{1, 2\}$ hay un macro

$$[\text{IF } E_i(\text{V2}, \text{V3}, \text{V1}, \text{W1}) \text{ GOTO A1}]$$

y para cada $i \in \{3, 4\}$ hay un macro

$$[\text{IF } E_i(\text{V2}, \text{V1}, \text{W1}, \text{W2}) \text{ GOTO A1}]$$

Haremos mas intuitiva la forma de escribir estos macros, por ejemplo para $i = 1$, lo escribiremos de la siguiente manera

$$\left[\text{IF V2} \neq E^{1,1}_{\#1}(\text{V3}, \text{V1}, \text{W1}, \mathcal{P}_1) \text{ GOTO A1}\right]$$

Ya que la funcion $f = \lambda x[(x)_1]$ es Σ-p.r. hay un macro

$$[\text{V2} \leftarrow f(\text{V1})]$$

el cual escribiremos de la siguiente manera:

$$[\text{V2} \leftarrow (\text{V1})_1]$$

Similarmente hay macros:

$$[\text{W1} \leftarrow *^{\leq}(\text{V1})_3]$$

$$[\text{V2} \leftarrow (\text{V1})_2]$$

(dejamos al lector entender bien el funcionamiento de estos macros). Sea $\mathcal{P}$ el siguiente programa de $\mathcal{S}^{\Sigma}$:

$$\begin{array}{l}
\text{L1 N20} \leftarrow \text{N20} + 1 \\
[\text{N10} \leftarrow (\text{N20})_1] \\
[\text{N3} \leftarrow (\text{N20})_2] \\
[\text{P3} \leftarrow *^{\leq}(\text{N20})_3] \\
\left[\text{IF } \neg Halt^{1,1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_1) \text{ GOTO L1}\right] \\
\left[\text{IF } \neg Halt^{1,1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_2) \text{ GOTO L1}\right] \\
\left[\text{IF } \neg Halt^{1,1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_3) \text{ GOTO L1}\right] \\
\left[\text{IF } \neg Halt^{1,1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_4) \text{ GOTO L1}\right] \\
\left[\text{IF N1} \neq E^{1,1}_{\#1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_1) \text{ GOTO L1}\right] \\
\left[\text{IF N2} \neq E^{1,1}_{\#1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_2) \text{ GOTO L1}\right] \\
\left[\text{IF P1} \neq E^{1,1}_{*1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_3) \text{ GOTO L1}\right] \\
\left[\text{IF P2} \neq E^{1,1}_{*1}(\text{N10}, \text{N3}, \text{P3}, \mathcal{P}_4) \text{ GOTO L1}\right]
\end{array}$$

Dejamos al lector la tarea de comprender el funcionamiento de este programa y convenserse de que computa la funcion $p^{2,2}_1|_S$. Pero entonces $p^{2,2}_1|_S$ es Σ-computable por lo cual es Σ-recursiva, lo cual prueba (3) ya que $Dom(p^{2,2}_1|_S) = S$.

(3)$\Rightarrow$(4). Supongamos $S \neq \emptyset$. Sea $(z_1, ..., z_n, \gamma_1, ..., \gamma_m) \in S$ fijo. Sea $\mathcal{P}$ un programa el cual compute a f y Sea $\leq$ un orden total sobre Σ. Sea $P : \mathbf{N} \to \omega$ dado por $P(x) = 1$ sii

$$Halt^{n,m}\left((x)_{n+m+1}, (x)_1, ..., (x)_n, *^{\leq}((x)_{n+1}), ..., *^{\leq}((x)_{n+m})), \mathcal{P}\right) = 1$$

Es facil ver que P es $(\Sigma \cup \Sigma_p)$-p.r. por lo cual es Σ-p.r.. Sea $\bar{P} = P \cup C^{1,0}_0|_{\{0\}}$. Para $i = 1, ..., n$, definamos $F_i : \omega \to \omega$ de la siguiente manera

$$F_i(x) = \begin{cases} (x)_i & \text{si} \quad \bar{P}(x) = 1 \\ z_i & \text{si} \quad \bar{P}(x) \neq 1 \end{cases}$$

Para $i = n+1, ..., n+m$, definamos $F_i : \omega \to \Sigma^*$ de la siguiente manera

$$F_i(x) = \begin{cases} *^{\leq}((x)_i) & \text{si} \quad \bar{P}(x) = 1 \\ \gamma_{i-n} & \text{si} \quad \bar{P}(x) \neq 1 \end{cases}$$

Por el lema de division por casos (para funciones Σ-p.r.), cada F_i es Σ-p.r.. Es facil ver que $F = [F_1, ..., F_{n+m}]$ cumple (4). ■

El halting problem y los conjuntos A y N. Cuando $\Sigma \supseteq \Sigma_p$, podemos definir

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

Notar que el dominio de $AutoHalt^{\Sigma}$ es Pro^{Σ} y que para cada $\mathcal{P} \in \text{Pro}^{\Sigma}$ tenemos que

(*) $AutoHalt^{\Sigma}(\mathcal{P}) = 1$ sii $\mathcal{P}$ se detiene partiendo del estado $\|\mathcal{P}\|$.

Lema 6. *Supongamos $\Sigma \supseteq \Sigma_p$. Entonces $AutoHalt^{\Sigma}$ no es Σ-recursivo.*

Proof. Supongamos $AutoHalt^{\Sigma}$ es Σ-recursivo. Por el Segundo Manantial tenemos que hay un macro

$$\big[\text{IF } AutoHalt^{\Sigma}(\text{W1}) \text{ GOTO A1}\big]$$

Sea $\mathcal{P}_0$ el siguiente programa de $\mathcal{S}^{\Sigma}$

$$\text{L1 } \big[\text{IF } AutoHalt^{\Sigma}(\text{P1}) \text{ GOTO L1}\big]$$

Note que

- $\mathcal{P}_0$ termina partiendo desde $\|\mathcal{P}_0\|$ sii $AutoHalt^{\Sigma}(\mathcal{P}_0) = 0$,

lo cual produce una contradiccion si tomamos en (*) $\mathcal{P} = \mathcal{P}_0$. ■

Usando el lema anterior y la Tesis de Church podemos probar el siguiente impactante resultado.

Teorema 7. *Supongamos $\Sigma \supseteq \Sigma_p$. Entonces $AutoHalt^{\Sigma}$ no es Σ-efectivamente computable. Es decir no hay ningun procedimiento efectivo que decida si un programa de $\mathcal{S}^{\Sigma}$ termina partiendo de si mismo.*

Proof. Si $AutoHalt^{\Sigma}$ fuera Σ-efectivamente computable, la Tesis de Church nos diria que es Σ-recursivo, contradiciendo el lema anterior. ■

Notese que $AutoHalt^{\Sigma}$ provee de un ejemplo natural en el cual la cuantificacion (no acotada) de un predicado Σ-p.r. con dominio rectangular no es Σ-efectivamente computable

Ahora estamos en condiciones de dar un ejemplo natural de un conjunto A que es Σ-recursivamente enumerable pero el cual no es Σ-recursivo.

Lema 8. *Supongamos que $\Sigma \supseteq \Sigma_p$. Entonces*

$$A = \big\{\mathcal{P} \in \text{Pro}^{\Sigma} : AutoHalt^{\Sigma}(\mathcal{P}) = 1\big\}$$

es Σ-r.e. y no es Σ-recursivo. Mas aun el conjunto

$$N = \big\{\mathcal{P} \in \text{Pro}^{\Sigma} : AutoHalt^{\Sigma}(\mathcal{P}) = 0\big\}$$

no es Σ-r.e.

Proof. Para ver que A es Σ-r.e. se lo puede hacer imperativamente dando un programa (usando macros) que enumere a A. De esta forma probariamos que A es Σ-enumerable y por lo tanto es Σ-r.e.. Daremos ahora una prueba no imperativa de este hecho, es decir mas propia del paradigma funcional. Sea $P = \lambda t\mathcal{P}\big[Halt^{0,1}(t, \mathcal{P}, \mathcal{P})\big]$. Note que P es Σ-p.r. por lo que $M(P)$ es Σ-r.. Ademas note que $D_{M(P)} = A$, lo cual implica que A es Σ-r.e..

Supongamos ahora que N es Σ-r.e.. Entonces la funcion $C_0^{0,1}|_N$ es Σ-recursiva ya que $C_0^{0,1}$ lo es. Ademas ya que A es Σ-r.e. tenemos que $C_1^{0,1}|_A$ es Σ-recursiva. Ya que

$$AutoHalt^{\Sigma} = C_1^{0,1}|_A \cup C_0^{0,1}|_N$$

el lema de division por casos nos dice que $AutoHalt^{\Sigma}$ es Σ-recursivo, contradiciendo el Lema 6. Esto prueba que N no es Σ-r.e..

Finalmente supongamos A es Σ-recursivo. Entonces el conjunto

$$N = (\Sigma^* - A) \cap \text{Pro}^{\Sigma}$$

deberia serlo, lo cual es absurdo. Hemos probado entonces que A no es Σ-recursivo.
■

Usando la Tesis de Church obtenemos el siguiente resultado.

Proposición 9. *Supongamos que $\Sigma \supseteq \Sigma_p$. Entonces A es Σ-efectivamente enumerable y no es Σ-efectivamente computable. El conjunto N no es Σ-efectivamente enumerable. Es decir, A puede ser enumerado por un procedimiento efectivo pero no hay ningun procedimiento efectivo que decida la pertenencia a A y no hay ningun procedimiento efectivo que enumere a N. Mas aun, si un procedimiento efectivo da como salida siempre elementos de N, entonces hay una cantidad infinita de elementos de N los cuales nunca da como salida.*

Ejercicio 4: Justifique la ultima aseveracion en la proposicion anterior

Cabe destacar aqui que el dominio de una funcion Σ-recursiva no siempre sera un conjunto Σ-recursivo. Por ejemplo si tomamos Σ tal que $\Sigma \supseteq \Sigma_p$, entonces $C_1^{0,1}|_A$ es una funcion Σ-recursiva ya que es la restriccion de la funcion Σ-recursiva $C_1^{0,1}$ al conjunto Σ-r.e. A, pero $\mathrm{Dom}(C_1^{0,1}|_A) = A$ no es Σ-recursivo.

Ejercicio 5: Pruebe que no es cierta la siguiente (hermosa y tentadora) propiedad
- Si $f : S \subseteq \omega^n \times \Sigma^{*m} \to \omega$ es una funcion Σ-computable, entonces hay un macro

$$\left[\text{IF } \chi_S^{\omega^n \times \Sigma^{*m}}(\text{V1}, ..., \text{V}\bar{n}, \text{W1}, ..., \text{W}\bar{m}) \text{ GOTO A1}\right]$$

Ejercicio 6: V o F o I. justifique.
(a) Sea Σ un alfabeto y sean $n, m \in \omega$. Entonces el dominio de $T^{n,m}$ es rectangular
(b) Para cada $\mathcal{P} \in \mathrm{Pro}^\Sigma$ hay un $\mathcal{P}' \in \mathrm{Pro}^\Sigma$ tal que $\mathrm{Dom}(\Psi_{\mathcal{P}'}^{1,0,\#}) = \omega - \mathrm{Dom}(\Psi_{\mathcal{P}}^{1,0,\#})$.
(c) Sea $\mathcal{P} \in \mathrm{Pro}^\Sigma$ y supongamos que para cada $x \in \omega$, $\mathcal{P}$ termina partiendo de $((x, 0, 0...), (\varepsilon, \varepsilon, ...))$ en a lo sumo $n(\mathcal{P})^2 + x$ pasos. Entonces $\Psi_{\mathcal{P}}^{1,0,*}$ es Σ-p.r.
(d) Hay $\mathcal{P} \in \mathrm{Pro}^\Sigma$ tal que $0 \in D_{\Psi_{\mathcal{P}}^{1,0,\#}}$ y

$$\Psi_{\mathcal{P}}^{1,0,\#}(0) = 1 + \min_t(i^{1,0}(t, 0, \mathcal{P}) = n(\mathcal{P}) + 1)$$

Ejercicio 7: (Opcional) Pruebe que la reciproca del Ejercicio 1 no es cierta, es decir de un predicado Σ-recursivo $P : D_P \subseteq \omega \times \omega^n \times \Sigma^{*m} \to \omega$ el cual cumpla que $M(P)$ es Σ-recursiva y que D_P no es Σ-recursivo (Hint: tome $P = C_1^{1,1}|_{\omega \times A}$)

Con los resultados anteriores estamos en condiciones de dar un ejemplo de un predicado Σ-recursivo, cuya minimizacion no es Σ-efectivamente computable (y por lo tanto no es Σ-recursiva). Aceptaremos el resultado sin demostracion.

Proposición 10. *Supongamos que $\Sigma \supseteq \Sigma_p$. Sea $P = C_1^{0,1}|_A \circ \lambda t \alpha \left[\alpha^{1 \dot{-} t} \mathrm{SKIP}^t\right]|_{\omega \times \mathrm{Pro}^\Sigma}$. El predicado P es Σ-recursivo pero la funcion $M(P)$ no es Σ-efectivamente computable (y por lo tanto no es Σ-recursiva).*

Ejercicio 8: (Opcional) Daremos aqui una guia para probar la proposicion anterior.
(a) Pruebe que P es Σ-recursivo

(b) Pruebe que $D_{M(P)} = \text{Pro}^{\Sigma}$

(c) Para cada $\mathcal{P} \in \text{Pro}^{\Sigma}$ se tiene que

$$M(P)(\mathcal{P}) = 0 \text{ sii } \mathcal{P} \in A$$

(d) Pruebe que $AutoHalt^{\Sigma} = \lambda x[x = 0] \circ M(P)$ por lo cual $M(P)$ no es Σ-recursiva

(e) $M(P)$ no es Σ-efectivamente computable