Nota: Los ejercicios que tienen (S) son para una "Segunda vuelta" es decir conviene hacerlos una vez que ya se completó la guía haciendo los otros y ya se tiene mas madurez e intuición basica sobre los conceptos. Los que tienen (O) son opcionales por lo cual no se toman en los examenes.

El paradigma imperativo de Neumann: El lenguaje $\mathcal{S}^{\Sigma}$

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.

Sintaxis de $\mathcal{S}^{\Sigma}$

Necesitaremos algunas funciones basicas para poder describir la sintaxis de $\mathcal{S}^{\Sigma}$ en forma precisa. 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

$$\begin{aligned} Des(0) &= \varepsilon \\ Des(n+1) &= Sig(Des(n)) \end{aligned}$$

Notese que para $n \in \mathbf{N}$, la palabra $Des(n)$ es la notacion usual decimal de n. Para hacer mas agil la notacion escribiremos $\bar{n}$ en lugar de $Dec(n)$. Notese que, en virtud de esta convencion notacional se tiene que $Dec = \lambda n[\bar{n}]$.

Ejercicio 1: Hacer

(a) Pruebe que Sig y Dec son Num-p.r.

(b) Sea Γ un alfabeto que contiene a Num. Pruebe sin usar el teorema de independencia del alfabeto que Sig y Dec son Γ-p.r.. Hint: extienda Sig a una $\widetilde{Sig}$ que sea Γ-total. Use $\widetilde{Sig}$ para definir una g que sea Γ-p.r. y tal que $Dec = R(C_\varepsilon^{0,0}, g)$ (respecto del alfabeto Γ)

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 $+$ y $\dot{-}$, pero deberia quedar claro que en Σ_p estan los simbolos $+$ y $\dot{-}$ y no las operaciones que ellos 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 1,3: 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

Ejercicio 1,6: Sea $\Sigma = \{@, \uparrow\}$. Sea $L = \{\mathrm{IFP}\bar{k}\mathrm{BEGINS}a\mathrm{GOTOL}\bar{n} : a \in \Sigma \text{ y } k, n \in \mathbf{N}\}$. Note que $L \subseteq \mathrm{Ins}^{\Sigma}$. Pruebe que L es $(\Sigma \cup \Sigma_p)$-p.r..

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 $\mathrm{GOTOL}\bar{m}$ 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

$$\begin{array}{ll} \mathrm{L2} & \mathrm{N12} \leftarrow \mathrm{N12} \dot{-} 1 \\ & \mathrm{P1} \leftarrow {}^{\frown}\mathrm{P1} \\ & \mathrm{IF\ N12} \neq 0 \ \mathrm{GOTO\ L2} \end{array}$$

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.

Lemma 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: (S) 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

$$\frown\alpha = \begin{cases} [\alpha]_2 ... [\alpha]_{|\alpha|} & \text{si} \quad |\alpha| \geq 2 \\ \varepsilon & \text{si} \quad |\alpha| \leq 1 \end{cases}$$

Definamos

$$\begin{aligned} \omega^{[\mathbf{N}]} &= \left\{(s_1, s_2, ...) \in \omega^{\mathbf{N}} : \text{ hay } n \in \mathbf{N} \text{ tal que } s_i = 0, \text{ para } i \geq n\right\} \\ \Sigma^{*[\mathbf{N}]} &= \left\{(\sigma_1, \sigma_2, ...) \in \Sigma^{*\mathbf{N}} : \text{ hay } n \in \mathbf{N} \text{ tal que } \sigma_i = \varepsilon, \text{ para } i \geq n\right\}. \end{aligned}$$

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 N$\overline{\imath}$ en el estado $(\vec{s}, \vec{\sigma})$ y σ_i es el *contenido* o *valor* de la variable P$\overline{\imath}$ 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 $S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma})$:

$$\begin{aligned} S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) \quad = \quad & \text{descripcion instantanea que resulta} \\ & \text{luego de realizarp } I_i^{\mathcal{P}}, \text{ estando en estado } (\vec{s}, \vec{\sigma}) \end{aligned}$$

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 $Bas(I_i^{\mathcal{P}}) =$ IF P$\bar{k}$ BEGINS a GOTO L$\bar{m}$. Entonces tenemos dos subcasos.

Subcaso a. El valor de P$\bar{k}$ en $(\vec{s}, \vec{\sigma})$ comiensa con a. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (\min\{l : I_l^{\mathcal{P}} \text{ tiene label L}\bar{m}\}, \vec{s}, \vec{\sigma})$$

Subcaso b. El valor de 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 $Bas(I_i^{\mathcal{P}}) =$ GOTO L$\bar{m}$. Entonces

$$S_{\mathcal{P}}(i, \vec{s}, \vec{\sigma}) = (\min\{l : I_l^{\mathcal{P}} \text{ tiene label L}\bar{m}\}, \vec{s}, \vec{\sigma})$$

Caso $Bas(I_i^{\mathcal{P}}) =$ 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 $\Sigma = \{\blacktriangle, \#\}$ y sea $\mathcal{P}$ el siguiente programa

L3 N4 $\leftarrow$ N4 + 1
P1 $\leftarrow$ $\curvearrowright$P1
IF P1 BEGINS $\blacktriangle$ GOTO L3
P3 $\leftarrow$ P3.#

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 $\mathcal{P}$ partiendo del estado $(\vec{s},\vec{\sigma})$ es la siguiente sucesion (de arriba hacia abajo) de descripciones instantaneas:

$$(1,(2,1,0,5,3,0,0,0,...),(\#\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_1^{\mathcal{P}} = \mathrm{N4} \leftarrow \mathrm{N4}+1$ obtenemos

$$(2,(2,1,0,6,3,0,0,0,...),(\#\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_2^{\mathcal{P}} = \mathrm{P1} \leftarrow {}^{\frown}\mathrm{P1}$ obtenemos

$$(3,(2,1,0,6,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_3^{\mathcal{P}} =$ IF P1 BEGINS ▲ GOTO L3 obtenemos

$$(1,(2,1,0,6,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_1^{\mathcal{P}} = \mathrm{N4} \leftarrow \mathrm{N4}+1$ obtenemos

$$(2,(2,1,0,7,3,0,0,0,...),(\blacktriangle\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_2^{\mathcal{P}} = \mathrm{P1} \leftarrow {}^{\frown}\mathrm{P1}$ obtenemos

$$(3,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_3^{\mathcal{P}} =$ IF P1 BEGINS ▲ GOTO L3 obtenemos

$$(4,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

realizando $I_4^{\mathcal{P}} = \mathrm{P3} \leftarrow \mathrm{P3}.\#$ obtenemos

$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

intentando realizar $I_5^{\mathcal{P}} = \varepsilon$ obtenemos

$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

intentando realizar $I_5^{\mathcal{P}} = \varepsilon$ obtenemos

$$(5,(2,1,0,7,3,0,0,0,...),(\#\#,\varepsilon,\blacktriangle\blacktriangle\#,\#\blacktriangle,\#,\varepsilon,\varepsilon,\varepsilon,...))$$

intentando realizar $I_5^{\mathcal{P}} = \varepsilon$ 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

```
L3  N4 ← N4 + 1
    IF P1 BEGINS ▲ GOTO L3
```

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 sucesion (de arriba hacia abajo) de descripciones instantaneas:

$$(1, (2, 1, 0, 5, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$

$$(2, (2, 1, 0, 6, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$

$$(1, (2, 1, 0, 6, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$

$$(2, (2, 1, 0, 7, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$

$$(1, (2, 1, 0, 7, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$

$$(2, (2, 1, 0, 8, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$

$$(1, (2, 1, 0, 8, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_1^{\mathcal{P}} = \text{N4} \leftarrow \text{N4} + 1 \text{ obtenemos}$$

$$(2, (2, 1, 0, 9, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\text{realizando } I_2^{\mathcal{P}} = \text{IF P1 BEGINS } \blacktriangle \text{ GOTO L3 obtenemos}$$

$$(1, (2, 1, 0, 9, 3, 0, 0, 0, ...), (\blacktriangle\#\#, \varepsilon, \blacktriangle\blacktriangle, \#\blacktriangle, \#, \varepsilon, \varepsilon, \varepsilon, ...))$$

$$\vdots$$

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* $(\vec{s}, \vec{\sigma})$.

Cabe destacar que en los conceptos antes definidos por "1 paso" entendemos "realizarp una instrucion", donde tal como se lo explico antes "realizarp" significa "realizar si se puede". Otra observacion importante es que 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

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:

E_1 El programa

L2 IF N1 $\neq 0$ GOTO L1
GOTO L2
L1 N1 $\leftarrow$ N1$\dot{-}$1

computa la funcion $Pred$. Note que este programa tambien computa las funciones $Pred \circ p_1^{n,m}$, para $n \geq 1$ y $m \geq 0$.

E_2 Sea $\Sigma = \{\clubsuit, \triangle\}$. El programa

L3 IF P2 BEGINS $\clubsuit$ GOTO L1
IF P2 BEGINS $\triangle$ GOTO L2
GOTO L4
L1 P2 $\leftarrow$ $^\frown$P2
P1 $\leftarrow$ P1$\clubsuit$
GOTO L3
L2 P2 $\leftarrow$ $^\frown$P2
P1 $\leftarrow$ P1$\triangle$
GOTO L3
L4 SKIP

computa la funcion $\lambda\alpha\beta\,[\alpha\beta]$.

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.

Proposition 2 *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.

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) $\Psi^{1,0,\#}_{\mathrm{L1IFN1}\neq 0\mathrm{GOTOL1}} = \{(0,0)\}$

(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

$$\Psi_{\text{SKIP}}^{n,m,\#}((x_1, ..., x_n, 0, 0, ...), (\alpha_1, ..., \alpha_m, \varepsilon, \varepsilon, ...)) = x_1$$

(e) El programa

$$\text{N1} \leftarrow \text{N1}\dot{-}1$$

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 $f : \Sigma^* \to \Sigma^*$ 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 $\mathcal{P} \in \text{Pro}^{\Sigma}$

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

```
        N1111 ← N16
        N2222 ← N3
        N5 ← N1111
L1000   IF N2222 ≠ 0 GOTO L2000
        GOTO L3000
L2000   N2222 ← N2222 ∸ 1
        N5 ← 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 N5 $\leftarrow$ N16 + 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 N5 $\leftarrow$ N16 + 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 N20 $\leftarrow$ N1 + N14. Entonces es claro que podriamos modificar el programa que simulaba N5 $\leftarrow$ N16 + N3 haciendole reemplazos adecuados a sus variables y labels. Por ejemplo podriamos escribir

$$
\begin{array}{ll}
 & \text{N9999} \leftarrow \text{N1} \\
 & \text{N8888} \leftarrow \text{N14} \\
 & \text{N20} \leftarrow \text{N9999} \\
\text{L1001} & \text{IF N8888} \neq 0 \text{ GOTO L2002} \\
 & \text{GOTO L3003} \\
\text{L2002} & \text{N8888} \leftarrow \text{N8888} \dot{-} 1 \\
 & \text{N20} \leftarrow \text{N20} + 1 \\
 & \text{GOTO L1001} \\
\text{L3003} & \text{SKIP}
\end{array}
$$

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

$$
\begin{array}{ll}
 & \text{V4} \leftarrow \text{V2} \\
 & \text{V5} \leftarrow \text{V3} \\
 & \text{V1} \leftarrow \text{V4} \\
\text{A1} & \text{IF V5} \neq 0 \text{ GOTO A2} \\
 & \text{GOTO A3} \\
\text{A2} & \text{V5} \leftarrow \text{V5} \dot{-} 1 \\
 & \text{V1} \leftarrow \text{V1} + 1 \\
 & \text{GOTO A1} \\
\text{A3} & \text{SKIP}
\end{array}
$$

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 N5 $\leftarrow$ N16 + 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 N20 $\leftarrow$ N1 + N14 dentro de $\mathcal{P}$. La practicidad de tener el molde M cae de maduro. Ahora en caso de necesitar una instruccion del tipo N$\bar{k} \leftarrow$ N$\bar{n}$ + N$\bar{m}$ solo tenemos que reemplazar en M

- cada ocurrencia de V1 por N$\bar{k}$

- cada ocurrencia de V2 por N$\bar{n}$

- cada ocurrencia de V3 por N$\bar{m}$

y reemplazar la variable auxiliar de M y los labels auxiliares de M por una variable concreta y tres labels concretos que no se usen en el programa que estamos realizando. El programa asi obtenido simulara a la instruccion N$\bar{k} \leftarrow$ N$\bar{n}$ + 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 $Ti(M)$ = 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$\leftarrow$V2V4$\leftarrow$V3A1IFV4$\neq$0GOTOA2GOTOA3A2V4$\leftarrow$V4$\dot{-}$1V1$\leftarrow$V1+1GOTOA1A3SKIP

Es decir, como objeto matematico, M es simplemente una palabra. A las palabras de la forma V$\bar{n}$, con $n \in \mathbf{N}$, las llamaremos *variables numericas de macro.*

A las palabras de la forma W$\bar{n}$, con $n \in \mathbf{N}$, las llamaremos *variables alfabeticas de macro* y a las palabras de la forma 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 y V4 y V5 son llamadas *variables auxiliares* de M. 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 en el cual se realizara la expansion de M.

En el siguiente ejemplo veremos un macro que tiene un label que no es auxiliar sino oficial. Sea $\Sigma = \{@, !\}$. Supongamos que estamos escribiendo un programa $\mathcal{P}'$ y nos hace falta simular instrucciones de la forma

$$\text{IF } |\text{P}\bar{n}| \leq \text{N}\bar{m} \text{ GOTO L}\bar{k}$$

(por supuesto estas instrucciones no pertenecen al lenguaje $\mathcal{S}^{\Sigma}$ pero deberia quedar claro como funcionan). Entonces podemos tomar el macro M':

	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 P$\bar{n}$ y N$\bar{m}$)
- variable auxiliares W2 y V2
- labels auxiliares A2, A3, A4 y A5
- un label oficial A1 (correspondiente a L$\bar{k}$)

Una descripcion intuitiva del macro M' seria

$$\text{IF } |\text{W1}| \leq \text{V1 GOTO A1}$$

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. 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 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.

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.

Ejercicio 10: Sea $\Sigma = \{\#, \$\}$. De explicitamente macros que simulen las instrucciones de cada uno de los siguientes formatos

(a) $\mathrm{N}\bar{n} \leftarrow \mathrm{N}\bar{n} + 20$, con $n \in \mathbf{N}$

(b) $\mathrm{P}\bar{n} \leftarrow \#\#\$\#\$$, con $n \in \mathbf{N}$

(c) IF $\mathrm{P}\bar{n} \neq \varepsilon$ GOTO $\mathrm{L}\bar{k}$, con $n, k \in \mathbf{N}$

(d) $\mathrm{N}\bar{n} \leftarrow |\mathrm{P}\bar{m}|$, con $n, m \in \mathbf{N}$

(e) IF $\mathrm{P}\bar{n} = \mathrm{P}\bar{m}$ GOTO $\mathrm{L}\bar{k}$, con $n, m, k \in \mathbf{N}$

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 $[\mathrm{V3} \leftarrow SUMA(\mathrm{V1}, \mathrm{V2})]$ 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

$$
\begin{array}{ll}
\mathrm{L1} & \text{IF N2} \neq 0 \text{ GOTO L2} \\
 & \text{GOTO L3} \\
\mathrm{L2} & [\mathrm{N3} \leftarrow SUMA(\mathrm{N3}, \mathrm{N1})] \\
 & \mathrm{N2} \leftarrow \mathrm{N2} \dot{-} 1 \\
 & \text{GOTO L1} \\
\mathrm{L3} & \mathrm{N1} \leftarrow \mathrm{N3}
\end{array}
$$

donde [N3 $\leftarrow SUMA$(N3, N1)] es una expansion del macro [V3 $\leftarrow SUMA$(V1, V2)] hecha haciendo el reemplazo de las variables oficiales V3, V1 y V2 por N3, N3 y 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 dependiendo 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 [N3 $\leftarrow SUMA$(N3, N1)] es el siguiente programa:

	N1111 $\leftarrow$ N1
	N2222 $\leftarrow$ N3
	N3 $\leftarrow$ N1111
L1000	IF N2222 $\neq 0$ GOTO L2000
	GOTO L3000
L2000	N2222 $\leftarrow$ N2222$\dot{-}$1
	N3 $\leftarrow$ 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 N2 $\neq 0$ GOTO L2
	GOTO L3
L2	N1111 $\leftarrow$ N1
	N2222 $\leftarrow$ N3
	N3 $\leftarrow$ N1111
L1000	IF N2222 $\neq 0$ GOTO L2000
	GOTO L3000
L2000	N2222 $\leftarrow$ N2222$\dot{-}$1
	N3 $\leftarrow$ N3 + 1
	GOTO L1000
L3000	SKIP
	N2 $\leftarrow$ N2$\dot{-}$1
	GOTO L1
L3	N1 $\leftarrow$ 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 M el cual cumpla las siguientes propiedades. Cabe destacar que 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:

   (a) 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)

   (b) las variables auxiliares de M por variables concretas (distintas de a dos) y NO pertenecientes a la lista $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{N}\overline{k_{n+1}}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$

   (c) los labels auxiliares de M por labels concretos (distintos de a dos)

   Entonces la palabra asi obtenida es un programa de $\mathcal{S}^\Sigma$ que denotaremos 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:

   - Si hacemos correr $\left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$ 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 $\left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$ no se detiene

     ii. si $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \in D_f$, entonces $\left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$ se detiene (i.e. intenta realizar la siguiente a su ultima instrucion) y llega a un estado e' el cual cumple:

A. e' le asigna a $\mathrm{N}\overline{k_{n+1}}$ el valor $f(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$

B. 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. 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 (salvo en el caso de que alguna $\mathrm{N}\overline{k_i}$ sea la variable $\mathrm{N}\overline{k_{n+1}}$, situacion en la cual el valor final de la variable $\mathrm{N}\overline{k_i}$ sera $f(x_1, ..., x_n, \alpha_1, ..., \alpha_m)$)

El programa $\left[\mathrm{N}\overline{k_{n+1}} \leftarrow f(\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m})\right]$ es comunmente llamado la expansion del macro $\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right]$ 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 el cual cumpla condiciones analogas a las descriptas recien. Dejamos al lector escribirlas en detalle para este caso.

Aceptaremos sin demostracion el siguiente resultado fundamental.

Proposition 3 *Sea Σ un alfabeto finito.*

(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 11: Sea $SUMA = \lambda xy[x + y]$. Explique por que la palabra

$$\begin{array}{ll} & \mathrm{V}1 \leftarrow \mathrm{V}2 \\ & \mathrm{V}4 \leftarrow \mathrm{V}3 \\ \mathrm{A}1 & \text{IF } \mathrm{V}4 \neq 0 \text{ GOTO A2} \\ & \text{GOTO A3} \\ \mathrm{A}2 & \mathrm{V}4 \leftarrow \mathrm{V}4 \dot{-} 1 \\ & \mathrm{V}1 \leftarrow \mathrm{V}1 + 1 \\ & \text{GOTO A1} \\ \mathrm{A}3 & \text{SKIP} \end{array}$$

no puede ser tomada como el macro $[\mathrm{V}3 \leftarrow SUMA(\mathrm{V}1, \mathrm{V}2)]$

Ejercicio 12: 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 $[\mathrm{V1} \leftarrow f(\mathrm{W1})]$) un programa $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ tal que $\mathrm{Dom}(\Psi_{\mathcal{P}}^{0,1,\#}) = L$.

Ejercicio 13: Sea $\Sigma = \{\#, \$\}$ y sea $f : \omega \to \Sigma^*$ una funcion Σ-computable. De (usando el macro $[\mathrm{W1} \leftarrow f(\mathrm{V1})]$) un programa $\mathcal{P} \in \mathrm{Pro}^{\Sigma}$ tal que $\mathrm{Dom}(\Psi_{\mathcal{P}}^{0,1,\#}) = \mathrm{Im}\, f$.

Ejercicio 14: 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

$$\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V1}, ..., \mathrm{V}\bar{n}, \mathrm{W1}, ..., \mathrm{W}\bar{m})\right]$$

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

$$[\text{IF } P(\mathrm{V1}, ..., \mathrm{V}\bar{n}, \mathrm{W1}, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]$$

para denotar un macro M el cual cumpla las siguientes propiedades. Cabe destacar que 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 $\mathrm{V1}, ..., \mathrm{V}\bar{n}, \mathrm{W1}, ..., \mathrm{W}\bar{m}$

(2) A1 es el unico label oficial de M

(3) Si reemplazamos:

   (a) las variables oficiales de M (i.e. $\mathrm{V1}, ..., \mathrm{V}\bar{n}, \mathrm{W1}, ..., \mathrm{W}\bar{m}$) 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)

   (b) el label oficial A1 por el label concreto $\mathrm{L}\bar{k}$ (elejido libremente, es decir k es cualquier elemento de $\mathbf{N}$)

   (c) las variables auxiliares de M por variables concretas (distintas de a dos) y NO pertenecientes a la lista $\mathrm{N}\overline{k_1}, ..., \mathrm{N}\overline{k_n}, \mathrm{P}\overline{j_1}, ..., \mathrm{P}\overline{j_m}$

   (d) 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 de $\mathcal{S}^{\Sigma}$, salvo por la ley de los GOTO respecto de $\mathrm{L}\bar{k}$, que denotaremos 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:

- Si hacemos correr $\left[\text{IF } P(\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ partiendo de un estado e que le asigne a las variables $\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}$ 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 $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \notin D_P$, entonces $\left[\text{IF } P(\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ no se detiene
  ii. si $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \in D_P$ y $P(x_1, ..., x_n, \alpha_1, ..., \alpha_m) = 1$, entonces luego de una cantidad finita de pasos, $\left[\text{IF } P(\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ direcciona al label $\text{L}\bar{k}$ 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 $\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}$ no las modifica
  iii. si $(x_1, ..., x_n, \alpha_1, ..., \alpha_m) \in D_P$ y $P(x_1, ..., x_n, \alpha_1, ..., \alpha_m) = 0$, entonces luego de una cantidad finita de pasos, $\left[\text{IF } P(\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ 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 $\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}$ no las modifica

La palabra $\left[\text{IF } P(\text{N}\overline{k_1}, ..., \text{N}\overline{k_n}, \text{P}\overline{j_1}, ..., \text{P}\overline{j_m}) \text{ GOTO L}\bar{k}\right]$ es llamada la expansion del macro con respecto a la eleccion de variables y labels realizada

Proposition 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*

$$[\text{IF } P(\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \text{ GOTO A1}]$$

Proof. Por (a) de la proposicion anterior tenemos un macro $\left[\text{V}\overline{n+1} \leftarrow P(\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m})\right]$. Notese que la palabra

$$\left[\text{V}\overline{n+1} \leftarrow P(\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m})\right] \text{IFV}\overline{n+1}\neq 0\text{GOTOA1}$$

es el macro buscado. ■

Ejercicio 15: 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

$$[\text{IF } P(\text{V}1, ..., \text{V}\bar{n}, \text{W}1, ..., \text{W}\bar{m}) \text{ GOTO A1}]$$

entonces P es Σ-computable.

Ejercicio 16: 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.

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 en paralelo 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

Proposition 5 *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 la Proposicion 3 tenemos que existen macros:

$$\begin{array}{c} \big[\mathrm{V2} \leftarrow F_{(1)}(\mathrm{V1})\big] \\ \vdots \\ \big[\mathrm{V2} \leftarrow F_{(n)}(\mathrm{V1})\big] \\ \big[\mathrm{W1} \leftarrow F_{(n+1)}(\mathrm{V1})\big] \\ \vdots \\ \big[\mathrm{W1} \leftarrow F_{(n+m)}(\mathrm{V1})\big] \end{array}$$

Sea $\mathcal{P}$ el siguiente programa:

$$\begin{array}{c} \big[\mathrm{P}\overline{m} \leftarrow F_{(n+m)}(\mathrm{N1})\big] \\ \vdots \\ \big[\mathrm{P1} \leftarrow F_{(n+1)}(\mathrm{N1})\big] \\ \big[\mathrm{N}\overline{n} \leftarrow F_{(n)}(\mathrm{N1})\big] \\ \vdots \\ \big[\mathrm{N1} \leftarrow F_{(1)}(\mathrm{N1})\big] \end{array}$$

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 usa 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

$$\begin{array}{rcl} \mathcal{P}_1 & = & \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N1} \\ \mathcal{P}_2 & = & \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N2} \\ & \vdots & \\ \mathcal{P}_n & = & \mathcal{P}\mathrm{N1} \leftarrow \mathrm{N}\overline{n} \\ \mathcal{P}_{n+1} & = & \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P1} \\ \mathcal{P}_{n+2} & = & \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P2} \\ & \vdots & \\ \mathcal{P}_{n+m} & = & \mathcal{P}\mathrm{P1} \leftarrow \mathrm{P}\overline{m} \end{array}$$

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.

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 17: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros ni la proposicion anterior que $S = \{(2, \%\%), (3, !!!), (0, \varepsilon)\}$ es Σ-enumerable

Ejercicio 18: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros ni la proposicion anterior que $S = \{(i, 5, \%^i) : i \in \omega\}$ es Σ-enumerable

Ejercicio 19: Sea $\Sigma = \{\%, !\}$. Sea $L \subseteq \Sigma^*$ un conjunto no vacio y Σ-enumerable. De (usando macros) un programa $\mathcal{P} \in \text{Pro}^\Sigma$ tal que $\Psi^{1,0,*}_{\mathcal{P}}$ enumera al conjunto

$$\{\alpha\%! : \alpha \in L\}$$

es decir $\text{Dom}\Psi^{1,0,*}_{\mathcal{P}} = \omega$ y $\text{Im}\,\Psi^{1,0,*}_{\mathcal{P}} = \{\alpha\%! : \alpha \in L\}$

Ejercicio 20: Sea $\Sigma = \{\%, !\}$. Sea

$$S = \{(x, x+1, x+2, \%\%!!) : x \in \omega\}$$

De sin usar macros un programa $\mathcal{P} \in \text{Pro}^\Sigma$ 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 21: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros que $S = \{(2, \%\%), (3, !)\}$ es Σ-computable

Ejercicio 22: Sea $\Sigma = \{\%, !\}$. Pruebe sin usar macros que $S = \{(i, \%^i) : i \in \omega\}$ es Σ-computable

Ejercicio 23: 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 La Proposicion 4 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, 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 un macro

$$[\text{IF } (\text{V1}, ..., \text{V}\bar{n}, \text{W1}, ..., \text{W}\bar{m}) \in S \text{ GOTO A1}]$$

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 $[\text{IF } (\text{V1}, ..., \text{V}\bar{n}, \text{W1}, ..., \text{W}\bar{m}) \in S \text{ GOTO A1}]$ cuando S es el dominio de una funcion Σ-computable.