

Lenguajes Formales y Computabilidad | FAMAF - UNC

Combos de definiciones y convenciones notacionales y los Combos de teoremas

24.06.2025

Matias Viola

Contenido

1. Combos de definiciones y convenciones notacionales	1
1.1. Combo 1: Defina:	1
1.1.1. Cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -recursivo	1
1.1.2. $\langle s_1, s_2, \dots \rangle$	1
1.1.3. « f es una función Σ -mixta»	1
1.1.4. «familia Σ -indexada de funciones»	1
1.1.5. $R(f, G)$: Recursion primitiva sobre variable alfabética con valores numéricos..	1
1.2. Combo 2: Defina:	2
1.2.1. $d \vdash^n d'$ y $d \vdash^* d'$	2
1.2.2. $L(M)$	2
1.2.3. « f es una función de tipo (n, m, s) »	2
1.2.4. (x)	2
1.2.5. $(x)_i$	2
1.3. Combo 3: Defina:	2
1.3.1. Cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -recursivamente enumerable	2
1.3.2. s^{\leq}	3
1.3.3. $*^{\leq}$	3
1.3.4. $\#^{\leq}$	3
1.4. Combo 4: Defina cuando una función $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ es llamada Σ -efectivamente computable y defina «el procedimiento P computa a la función f »	3
1.5. Combo 5: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -efectivamente computable y defina: «el procedimiento efectivo P decide la pertenencia a S »	4
1.6. Combo 6: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -efectivamente enumerable y defina: «el procedimiento efectivo P enumera a S »	4
1.7. Combo 7: Defina cuando una función $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ es llamada Σ -Turing computable y defina «la máquina de Turing M computa a la función f »	4
1.8. Combo 8: Defina:	4
1.8.1. $M(P)$ Minimización de variable numérica	4
1.8.2. Lt	5
1.8.3. Conjunto rectangular	5
1.8.4. « S es un conjunto de tipo (n, m) »	5
1.9. Combo 9	5
1.9.1. Conjunto rectangular	5
1.9.2. « I es una instrucción de S^Σ »	5
1.9.3. « P es un programa de S^Σ »	5
1.9.4. I_i^P	6
1.9.5. $n(P)$	6
1.9.6. Bas	6

1.10. Combo 10: Defina relativo al lenguaje S^Σ :	6
1.10.1. «estado»	6
1.10.2. «descripción instantánea»	6
1.10.3. S_P	6
1.10.4. «estado obtenido luego de t pasos, partiendo del estado $(\vec{x}, \vec{\alpha})$ »	7
1.10.5. « P se detiene (luego de t pasos), partiendo desde el estado $(\vec{x}, \vec{\alpha})$ »	7
1.11. Combo 11: Defina:	7
1.11.1. $\Psi_P^{n,m,\#}$	7
1.11.2. « f es Σ -computable» y « P computa a f »	7
1.11.3. $M^{\leq}(P)$ Minimización de variable alfabética	8
1.12. Combo 12: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -computable, cuando es llamado Σ -enumerable y defina «el programa P enumera a S »	8
1.13. Combo 13	8
1.13.1. $i^{n,m}$	8
1.13.2. $E_{\#}^{n,m}$	9
1.13.3. $E_{\#}^{n,m} + E_{*}^{n,m}$	9
1.13.4. $E_{\#j}^{n,m}$	9
1.13.5. $E_{*j}^{n,m}$	9
1.13.6. $\text{Halt}^{n,m}$	9
1.13.7. $T^{n,m}$	9
1.13.8. AutoHalt^Σ	9
1.13.9. Los conjuntos A y N	9
1.14. Combo 14: Explique en forma detallada la notación lambda	10
1.15. Combo 15: Dada una función $f : \text{Dom}_f \subseteq \omega \times \Sigma^* \rightarrow \omega$, describa qué tipo de objeto es y qué propiedades debe tener el macro: $[V2 \leftarrow f(V1, W1)]$	10
1.16. Combo 16: Dado un predicado $p : D_f \subseteq \omega \times \Sigma^* \rightarrow \omega$, describa qué tipo de objeto es y qué propiedades debe tener el macro: $[\text{IF } P(V1, W1) \text{ GOTO } A1]$	11
1.17. Combo 17: Defina el concepto de función y desarrolle las tres Convenciones Notacionales asociadas a dicho concepto	12
2. Combos de teoremas	12
2.1. Combo 1	12
2.2. Combo 2	12
2.3. Combo 3	13
2.4. Combo 4	13
2.5. Combo 5	13
2.6. Combo 6	13
2.7. Combo 7	14
2.8. Combo 8	14
2.9. Combo 9	14

1. Combos de definiciones y convenciones notacionales

1.1. Combo 1: Defina:

1.1.1. Cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -recursivo

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -recursivo cuando la función $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ -recursiva.

1.1.2. $\langle s_1, s_2, \dots \rangle$

Dada una infinitupla $(s_1, s_2, \dots) \in \omega^{[\mathbb{N}]}$ usaremos $\langle s_1, s_2, \dots \rangle$ para denotar al numero $\prod_{i=1}^{\infty} \text{pr}(i)^{s_i}$

1.1.3. « f es una función Σ -mixta»

Sea Σ un alfabeto finito. Una función f es Σ -mixta si:

1. $(\exists n, m \in \omega) \text{Dom}_f \subseteq \omega^n \times \Sigma^{*m}$
2. $\text{Im}_f \subseteq O \in \{\omega, \Sigma^*\}$

1.1.4. «familia Σ -indexada de funciones»

Dado un alfabeto Σ , una familia Σ -indexada de funciones sera una función $G : \Sigma \rightarrow \text{Im}_G$ donde Im_G es el conjunto de funciones $G(a)$ asociadas a cada $a \in \Sigma$.

NOTACIÓN: Si G es una familia Σ -indexada de funciones, entonces para $a \in \Sigma$, escribiremos G_a en lugar de $G(a)$.

1.1.5. $R(f, G)$: Recursion primitiva sobre variable alfabética con valores numéricos.

Sea una función $f : S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m \rightarrow \omega$ con $S_1, \dots, S_n \subseteq \omega$ y $L_1, \dots, L_m \subseteq \Sigma^*$ conjuntos no vacíos.

Sea una familia Σ -indexada de funciones $G_a : \omega \times S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m \times \Sigma^* \rightarrow \omega$ para cada $a \in \Sigma$.

Definimos recursivamente la función $R(f, G) : S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m \times \Sigma^* \rightarrow \omega$ de la siguiente manera:

1. $R(f, G)(\vec{x}, \vec{\alpha}, \varepsilon) = f(\vec{x}, \vec{\alpha})$
2. $R(f, G)(\vec{x}, \vec{\alpha}, \alpha a) = G_a(R(f, G)(\vec{x}, \vec{\alpha}, \alpha), \vec{x}, \vec{\alpha}, \alpha)$

También diremos que $R(f, g)$ es obtenida por recursion primitiva a partir de f y G .

1.2. Combo 2: Defina:

1.2.1. $d \vdash^n d'$ y $d \vdash^* d'$

(no hace falta que defina \vdash)

- $d \vdash^n d'$ si $(\exists d_2, \dots, d_n \in \text{Des}) d \vdash d_2 \vdash \dots \vdash d_n \vdash d'$.
- $d \vdash^* d'$ sii $(\exists n \in \omega) d \vdash^n d'$

1.2.2. $L(M)$

Llamamos $L(M)$ al conjunto formado por todas las palabras que son aceptadas por alcance de estado final.

Una palabra $\alpha_1 \dots \alpha_n \in \Sigma^*$ es aceptada por M por alcance de estado final si partiendo de $Bq_0 \alpha_1 \dots \alpha_n B \dots$ en algún momento de la computación M esta en un estado de F .

1.2.3. «f es una función de tipo (n, m, s) »

Dada una función Σ -mixta f ,

- Si $f = \emptyset$, entonces es una función de tipo (n, m, s) cualquiera sean $n, m \in \omega$ y $s \in \{\#, *\}$.
- Si $f \neq \emptyset$, entonces hay únicos $n, m \in \omega$ tales que $D_f \subseteq \omega^n \times \Sigma^{*m}$.
 - Si $I_f \subseteq \omega$, entonces es una función de tipo $(n, m, \#)$.
 - Si $I_f \subseteq \Sigma^*$, entonces es una función de tipo $(n, m, *)$.

De esta forma, cuando $f \neq \emptyset$, hablaremos de «el tipo de f » para referirnos a esta única terna (n, m, s) .

1.2.4. (x)

Dado $x \in \mathbb{N}$, usaremos (x) para denotar a la única infinitupla $(s_1, s_2, \dots) \in \omega^{[\mathbb{N}]}$ tq $x = \langle s_1, s_2, \dots \rangle = \prod_{i=1}^{\infty} \text{pr}(i)^{s_i}$

1.2.5. $(x)_i$

Dados $x, i \in \mathbb{N}$, usaremos $(x)_i$ para denotar a s_i de $(s_1, s_2, \dots) = (x)$.

Se le suele llamar la «i-esima bajada de x» al numero $(x)_i$ (al «bajar» el i-esimo exponente de la única posible factorización de x como producto de primos).

1.3. Combo 3: Defina:

1.3.1. Cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -recursivamente enumerable

(no hace falta que defina «función Σ -recursiva»)

Diremos que un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -recursivamente enumerable cuando sea vacío o haya una función sobreyectiva $F : \omega \rightarrow S$ tq $F_{(i)} = p_i^{n,m} \circ F$ sea Σ -recursiva para cada $i \in \{1, \dots, n+m\}$.

1.3.2. s^{\leq}

Sea \leq un orden sobre Σ^* .

$$S^{\leq} : \Sigma^* \rightarrow \Sigma^*$$

$$(a_n)^m \rightarrow (a_1)^{m+1}$$

$$\alpha a_i (a_n)^m \rightarrow \alpha a_{i+1} (a_1)^m \text{ con } 1 \leq i < n$$

1.3.3. $*^{\leq}$

Sea \leq un orden sobre Σ^* .

$$*^{\leq} : \omega \rightarrow \Sigma^*$$

$$0 \rightarrow \varepsilon$$

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

1.3.4. $\#^{\leq}$

Sea \leq un orden sobre Σ^* .

$$\#^{\leq} : \Sigma^* \rightarrow \omega$$

$$\varepsilon \rightarrow 0$$

$$a_{i_k} \dots a_{i_0} \rightarrow i_k n^k + \dots + i_0 n^0$$

1.4. Combo 4: Defina cuando una función $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ es llamada Σ -efectivamente computable y defina «el procedimiento P computa a la función f »

Sea $O \in \{\omega, \Sigma^*\}$. Una función Σ -mixta $f : \text{Dom}_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow O$ sera llamada Σ -efectivamente computable si hay un procedimiento efectivo P tq

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

En ambos casos diremos que P computa a la función f .

Obs: $f = \emptyset$ es un procedimiento que nunca se detiene cualesquiera sea su dato de entrada. Por lo tanto es Σ -efectivamente computable, cualesquiera sean n, m, O y Σ .

1.5. Combo 5: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -efectivamente computable y defina: «el procedimiento efectivo P decide la pertenencia a S »

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -efectivamente computable cuando la función $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ -efectivamente computable.

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

Obs: $f = \emptyset$ es un procedimiento que siempre da 0 cualesquiera sea su dato de entrada. Por lo tanto es Σ -efectivamente computable, cualesquiera sean n, m, O y Σ .

1.6. Combo 6: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -efectivamente enumerable y defina: «el procedimiento efectivo P enumera a S »

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -efectivamente enumerable cuando sea vacío o haya una función sobreyectiva $F: \omega \rightarrow S$ tq $F_{(i)}$ sea Σ -efectivamente computable, para cada $i \in \{1, \dots, n+m\}$.

1.7. Combo 7: Defina cuando una función $f: D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ es llamada Σ -Turing computable y defina «la máquina de Turing M computa a la función f »

Diremos que una función $f: \text{Dom}_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \Sigma^*$ es Σ -Turing computable si existe una máquina de Turing con unit, $M = (Q, \Sigma^*, \Gamma, \delta, q_0, B, \nu, F)$ tq:

1. Si $(\vec{x}, \vec{\alpha}) \in \text{Dom}_f$, entonces hay un $p \in Q$ tq $[q_0 B \nu^{x_1} B \dots B \nu^{x_n} B \alpha_1 B \dots B \alpha_m]^* \vdash [p B f(\vec{x}, \vec{\alpha})]$ y $[p B f(\vec{x}, \vec{\alpha})] \not\vdash d$ para cada $d \in \text{Des}$
2. Si $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m} - \text{Dom}_f$, entonces M no se detiene partiendo de $[q_0 B \nu^{x_1} B \dots B \nu^{x_n} B \alpha_1 B \dots B \alpha_m]$.

Cuando una maquina de Turing con unit M cumpla ambos items, diremos que M computa a la función f o que f es computada por M .

Cabe destacar que la condición $[p B f(\vec{x}, \vec{\alpha})] \not\vdash d$ para cada $d \in \text{Des}$ 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 .

1.8. Combo 8: Defina:

1.8.1. $M(P)$ Minimización de variable numérica

Sea Σ un alfabeto finito y sea $P: \text{Dom}_P \subseteq \omega^n \times \Sigma^{*m}$. Dado $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$, cuando exista al menos un $t \in \omega$ tq $P(t, \vec{x}, \vec{\alpha}) = 1$, usaremos $\min_t P(t, \vec{x}, \vec{\alpha})$ para denotar al menor de tales t 's.

Definimos $M(P) = \lambda \vec{x} \vec{\alpha} [\min_t P(t, \vec{x}, \vec{\alpha})]$

Diremos que $M(P)$ es obtenida por minimización de variable numérica a partir de P .

Obs: $M(P)$ esta definida solo para aquellas $(n + m)$ -uplas $(\vec{x}, \vec{\alpha})$ para las cuales hay al menos un t tq se da $P(t, \vec{x}, \vec{\alpha}) = 1$

1.8.2. Lt

$Lt : \mathbb{N} \rightarrow \omega$

$1 \rightarrow 0$

$x \rightarrow \max_i (x)_i \neq 0$

1.8.3. Conjunto rectangular

Sea Σ un alfabeto finito. Un conjunto Σ -mixto es llamado rectangular si es de la forma $S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m$ con cada $S_i \subseteq \omega$ y cada $L_i \subseteq \Sigma^*$.

1.8.4. « S es un conjunto de tipo (n, m) »

Dado un conjunto Σ -mixto $S \neq \emptyset$, decimos que S es un conjunto de tipo (n, m) para referirnos a los únicos $n, m \in \omega$ tq $S \subseteq \omega^n \times \Sigma^{*m}$

\emptyset es un conjunto de tipo (n, m) cualesquiera sean $n, m \in \omega$ por lo cual cuando hablemos de el tipo de un conjunto deberemos estar seguros de que dicho conjunto es no vacío.

1.9. Combo 9

1.9.1. Conjunto rectangular

Sea Σ un alfabeto finito. Un conjunto Σ -mixto es llamado rectangular si es de la forma $S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m$ con cada $S_i \subseteq \omega$ y cada $L_i \subseteq \Sigma^*$.

1.9.2. « I es una instrucción de S^Σ »

Una instrucción de S^Σ es ya sea una instrucción básica de S^Σ o una palabra de la forma αI , donde $\alpha \in \{L\bar{n} : n \in \mathbb{N}\}$ y I es una instrucción básica de S^Σ . Llamamos Ins^Σ al conjunto de todas las instrucciones de S^Σ .

1.9.3. « P es un programa de S^Σ »

Un programa de S^Σ es una palabra de la forma $I_1 I_2 \dots I_n$ donde $n \geq 1$, $I_1, \dots, I_n \in \text{Ins}^\Sigma$ y se cumple la ley de los GOTO.

Ley de los GOTO: Para cada $i \in \{1, \dots, n\}$, si GOTO $L\bar{m}$ es un tramo final de I_i , entonces existe $j \in \{1, \dots, n\}$ tq I_j tiene label $L\bar{m}$.

1.9.4. I_i^P

$\lambda i P[I_i^P] : \omega \times \text{Pro}^\Sigma \rightarrow \Sigma^*$

$$(i, P) \rightarrow \begin{cases} \text{i-esima instrucción de P} & \text{si } i \in \{1, \dots, n(P)\} \\ \varepsilon & \text{si } i \notin \{1, \dots, n(P)\} \end{cases}$$

1.9.5. $n(P)$

$\lambda P[n(P)] : \text{Pro}^\Sigma \rightarrow \omega$

$$P \rightarrow m \text{ tq } P = I_1 I_2 \dots I_m$$

1.9.6. Bas

$\text{Bas} : \text{Ins}^\Sigma \rightarrow (\Sigma \cup \Sigma_p)^*$

$$I \rightarrow \begin{cases} J & \text{si } I \text{ es de la forma } L\bar{k}J \text{ con } J \in \text{Ins}^\Sigma \\ I & \text{c.c.} \end{cases}$$

1.10. Combo 10: Defina relativo al lenguaje S^Σ :

1.10.1. «estado»

Es un par $(\vec{x}, \vec{\sigma}) = ((s_1, s_2, \dots), (\sigma_1, \sigma_2, \dots)) \in \omega^{[\mathbb{N}]} \times \Sigma^{*[\mathbb{N}]}$

Si $i \geq 1$, entonces diremos que s_i es el valor de la variable $N\vec{i}$ y α_i es el valor de la variable $P\vec{i}$ en el estado $(\vec{x}, \vec{\sigma})$.

1.10.2. «descripción instantánea»

Es una terna $(i, \vec{x}, \vec{\sigma}) \in \text{Des}^\Sigma = \omega \times \omega^{[\mathbb{N}]} \times \Sigma^{*[\mathbb{N}]}$ tq $(\vec{x}, \vec{\sigma})$ es un estado.

Si $i \in \{1, \dots, n(P)\}$, $(i, \vec{x}, \vec{\sigma})$ nos dice que las variables están en el estado $(\vec{x}, \vec{\sigma})$ y que la instrucción que debemos realizar es I_i^P

1.10.3. S_P

Dado un programa P .

$$S_P : \text{Des}^\Sigma \rightarrow \text{Des}^\Sigma$$

$$(i, \vec{x}, \vec{\sigma}) \rightarrow \begin{cases} (i, \vec{x}, \vec{\sigma}) & \text{si } i \notin \{1, \dots, n(P)\} \\ (i+1, (s_1, \dots, s_k-1, \dots), \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = N\bar{k} \leftarrow N\bar{k}-1 \\ (i+1, (s_1, \dots, s_k+1, \dots), \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = N\bar{k} \leftarrow N\bar{k}+1 \\ (i+1, (s_1, \dots, s_n, \dots), \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = N\bar{k} \leftarrow N\bar{n} \\ (i+1, (s_1, \dots, 0, \dots), \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = N\bar{k} \leftarrow 0 \\ (i+1, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{IF } N\bar{k} \neq 0 \text{ GOTO } L\bar{m} \wedge s_k = 0 \\ (\min\{l: I_l^P \text{ tiene label } L\bar{m}\}, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{IF } N\bar{k} \neq 0 \text{ GOTO } L\bar{m} \wedge s_k \neq 0 \\ (i+1, \vec{s}, (\sigma_1, \dots, \sim \sigma_k, \dots)) & \text{si } \text{Bas}(I_i^P) = P\bar{k} \leftarrow \sim P\bar{k} \\ (i+1, \vec{s}, (\sigma_1, \dots, \sigma_k a, \dots)) & \text{si } \text{Bas}(I_i^P) = P\bar{k} \leftarrow P\bar{k}.a \\ (i+1, \vec{s}, (\sigma_1, \dots, \sigma_{\bar{n}}, \dots)) & \text{si } \text{Bas}(I_i^P) = P\bar{k} \leftarrow P\bar{n} \\ (i+1, \vec{s}, (\sigma_1, \dots, \varepsilon, \dots)) & \text{si } \text{Bas}(I_i^P) = P\bar{k} \leftarrow \varepsilon \\ (\min\{l: I_l^P \text{ tiene label } L\bar{m}\}, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{IF } P\bar{k} \text{ BEGINS } a \text{ GOTO } L\bar{m} \wedge [\sigma_k]_1 = a \\ (i+1, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{IF } P\bar{k} \text{ BEGINS } a \text{ GOTO } L\bar{m} \wedge [\sigma_k]_1 \neq a \\ (\min\{l: I_l^P \text{ tiene label } L\bar{m}\}, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{GOTO } L\bar{m} \\ (i+1, \vec{s}, \vec{\sigma}) & \text{si } \text{Bas}(I_i^P) = \text{SKIP} \end{cases}$$

1.10.4. «estado obtenido luego de t pasos, partiendo del estado $(\vec{x}, \vec{\alpha})$ »

Dado un programa P y la descripción instantánea obtenida luego de t pasos desde el estado $(\vec{x}, \vec{\sigma})$

$$\overbrace{S_P(\dots S_P(S_P(1, \vec{x}, \vec{\sigma}))\dots)}^{t \text{ veces}} = (j, \vec{u}, \vec{\eta})$$

diremos que $(\vec{u}, \vec{\eta})$ es el estado obtenido luego de t pasos, partiendo del estado $(\vec{x}, \vec{\sigma})$.

1.10.5. « P se detiene (luego de t pasos), partiendo desde el estado

$$\overbrace{(\vec{x}, \vec{\alpha})}^{t \text{ veces}}$$

Dado $\overbrace{S_P(\dots S_P(S_P(1, \vec{x}, \vec{\sigma}))\dots)}^{t \text{ veces}} = (j, \vec{u}, \vec{\eta})$, si su primer coordenada j es igual a $n(P) + 1$, diremos que P se detiene (luego de t pasos), partiendo desde el estado $(\vec{x}, \vec{\sigma})$.

1.11. Combo 11: Defina:

1.11.1. $\Psi_P^{n,m,\#}$

Dado $P \in \text{Pro}^\Sigma$.

$$D_{\Psi_P^{n,m,\#}} = \{(\vec{x}, \vec{\sigma}) \in \omega^n \times \Sigma^{*m} : P \text{ termina partiendo de } \|x_1, \dots, x_n, \alpha_1, \dots, \alpha_m\|\}$$

$$\Psi_P^{n,m,\#} : D_{\Psi_P^{n,m,\#}} \rightarrow \omega$$

$$(\vec{x}, \vec{\sigma}) \rightarrow \text{valor de } N_1 \text{ cuando } P \text{ termina partiendo de } \|x_1, \dots, x_n, \alpha_1, \dots, \alpha_m\|$$

1.11.2. « f es Σ -computable» y « P computa a f »

Dado $s, O \in \{(\#, \omega), (*, \Sigma^*)\}$. Una función Σ -mixta $f : S \subseteq \omega^n \times \Sigma^{*m} \rightarrow O$ sera llamada Σ -computable si hay un programa P de S^Σ tq $f = \Psi_P^{n,m,s}$.

En tal caso diremos que la función f es computada por P .

1.11.3. $M^{\leq}(P)$ Minimización de variable alfabética

Sea que $\Sigma \neq \emptyset$. Sea \leq un orden total sobre Σ , \leq puede ser naturalmente extendido a un orden total sobre Σ^* . Sea $P : \text{Dom}_P \subseteq \omega^n \times \Sigma^{*m} \times \Sigma^*$ un predicado. Cuando $(\vec{x}, \vec{\alpha}) \in \omega^n \times \Sigma^{*m}$ es tq existe al menos un $\alpha \in \Sigma^*$ tq $P(\vec{x}, \vec{\alpha}, \alpha) = 1$, usaremos $\min_{\{\alpha\}}^{\leq} P(\vec{x}, \vec{\alpha}, \alpha)$ para denotar al menor $\alpha \in \Sigma^*$ tq $P(\vec{x}, \vec{\alpha}, \alpha) = 1$.

Definimos $M^{\leq}(P) = \lambda \vec{x} \vec{\alpha} [\min_{\alpha}^{\leq} P(\vec{x}, \vec{\alpha}, \alpha)]$

Diremos que $M^{\leq}(P)$ es obtenida por minimización de variable alfabética a partir de P .

Obs: $M^{\leq}(P)$ esta definida solo para aquellas $(n + m)$ -uplas $(\vec{x}, \vec{\alpha})$ para las cuales hay al menos un α tq se da $P(\vec{x}, \vec{\alpha}, \alpha) = 1$

1.12. Combo 12: Defina cuando un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ es llamado Σ -computable, cuando es llamado Σ -enumerable y defina «el programa P enumera a S »

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -computable cuando la función $\chi_S^{\omega^n \times \Sigma^{*m}}$ sea Σ -computable.

Un conjunto $S \subseteq \omega^n \times \Sigma^{*m}$ sera llamado Σ -enumerable cuando sea vacío o haya una función sobreyectiva $F : \omega \rightarrow S$ tq $F_{(i)}$ sea Σ -computable, para cada $i \in \{1, \dots, n + m\}$.

Nótese que, un conjunto no vacío $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ -enumerable sii hay programas P_1, \dots, P_{n+m} con dato de entrada $x \in \omega$ tales que:

$$S = \text{Im} [\Psi_{P_1}^{1,0,\#}, \dots, \Psi_{P_n}^{1,0,\#}, \Psi_{P_{n+1}}^{1,0,*}, \dots, \Psi_{P_{n+m}}^{1,0,*}]$$

Como puede notarse, los programas P_1, \dots, P_{n+m} puestos secuencialmente a funcionar desde el estado $\|x\|$ producen, en forma natural, un procedimiento efectivo que enumera a S . Es decir que los programas P_1, \dots, P_{n+m} enumeran a S .

1.13. Combo 13

Defina:

1.13.1. $i^{n,m}$

$$i^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \omega$$

$$(0, \vec{x}, \vec{\alpha}, P) \rightarrow 1$$

$$(t, \vec{x}, \vec{\alpha}, P) \rightarrow j \text{ tq } \overbrace{S_P(\dots S_P(S_P(1, \vec{x}, \vec{\sigma})))}^{t \text{ veces}} = (j, \vec{u}, \vec{\eta})$$

1.13.2. $E_{\#}^{n,m}$

$$E_{\#}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \omega^{[\mathbb{N}]}$$

$$(0, \vec{x}, \vec{\alpha}, P) \rightarrow (x_1, \dots, x_n, 0, \dots)$$

$$(t, \vec{x}, \vec{\alpha}, P) \rightarrow \vec{u} \text{ tq } \overbrace{S_P(\dots S_P(S_P(1, \vec{x}, \vec{\sigma})))}^{t \text{ veces}} \dots = (j, \vec{u}, \vec{\eta})$$

1.13.3. $E_{\#}^{n,m} + E_{*}^{n,m}$

$$E_{*}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \Sigma^{*[\mathbb{N}]}$$

$$(0, \vec{x}, \vec{\alpha}, P) \rightarrow (\alpha_1, \dots, \alpha_n, \varepsilon, \dots)$$

$$(t, \vec{x}, \vec{\alpha}, P) \rightarrow \vec{\eta} \text{ tq } \overbrace{S_P(\dots S_P(S_P(1, \vec{x}, \vec{\sigma})))}^{t \text{ veces}} \dots = (j, \vec{u}, \vec{\eta})$$

1.13.4. $E_{\#_j}^{n,m}$

$$E_{\#_j}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \omega$$

$$E_{\#_j}^{n,m} = p_j^{n,m} \circ E_{\#}^{n,m}$$

1.13.5. $E_{*_j}^{n,m}$

$$E_{*_j}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \Sigma^{*}$$

$$E_{*_j}^{n,m} = p_j^{n,m} \circ E_{*}^{n,m}$$

1.13.6. $\text{Halt}^{n,m}$

$$\text{Halt}^{n,m} : \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma} \rightarrow \{0, 1\}$$

$$(t, \vec{x}, \vec{\sigma}, P) \rightarrow i^{n,m}(t, \vec{x}, \vec{\alpha}, P) = n(P) + 1$$

1.13.7. $T^{n,m}$

$$\text{Dom}_{T^{n,m}} = \{(\vec{x}, \vec{\sigma}, P) : P \text{ se detiene partiendo de } \parallel x_1, \dots, x_n, \alpha_1, \dots, \alpha_m \parallel\}$$

$$T^{n,m} : \text{Dom}_{T^{n,m}} \rightarrow \omega$$

$$(t, \vec{x}, \vec{\sigma}, P) \rightarrow \min_t(\text{Halt}^{n,m}(t, \vec{x}, \vec{\sigma}, P))$$

1.13.8. AutoHalt^{Σ}

$$\text{Dado } \Sigma \supseteq \Sigma_p$$

$$\text{AutoHalt}^{\Sigma} : \text{Pro}^{\Sigma} \rightarrow \{0, 1\}$$

$$P \rightarrow (\exists t \in \omega) \text{Halt}^{0,1}(t, P, P)$$

1.13.9. Los conjuntos A y N

$$\text{Dado } \Sigma \supseteq \Sigma_p$$

$$A = \{P \in \text{Pro}^{\Sigma} : \text{AutoHalt}^{\Sigma}(P)\}$$

$$N = \{P \in \text{Pro}^\Sigma : \neg \text{AutoHalt}^\Sigma(P)\}$$

1.14. Combo 14: Explique en forma detallada la notación lambda

Usamos la notación lambda de Church de la forma descrita a continuación.

Esta notación se define en función de un alfabeto finito previamente fijado, que denotaremos por Σ .

Solo se usan expresiones tq:

1. Variables permitidas:

- Se usan **variables numéricas** que se valúan en números de (ω) , y se denotan por letras como $x, y, z, u, v, w, n, m, k, \dots$
- Se usan **variables alfabéticas** que se valúan en palabras sobre el alfabeto Σ . Se denotan por letras como $\alpha, \beta, \gamma, \delta, \varepsilon, \psi, \eta, \dots$

2. **Dominio parcial:** Las expresiones lambda pueden ser **parcialmente definidas**. Es decir, puede haber valuaciones de sus variables para las cuales la expresión no este definida.

3. **Libertad sintáctica:** Las expresiones pueden ser descritas informalmente.

4. **Valores booleanos:** Consideramos que las expresiones booleanas toman valores en el conjunto $\{0, 1\} \subseteq \omega$ (usando 0 para falso y 1 para verdadero).

Dado un alfabeto Σ a las expresiones que cumplan las características dadas anteriormente las llamaremos lambdificables con respecto a Σ .

1.15. Combo 15: Dada una función $f : \text{Dom}_f \subseteq \omega \times \Sigma^* \rightarrow \omega$, describa qué tipo de objeto es y qué propiedades debe tener el macro: $[V2 \leftarrow f(V1, W1)]$

Dada una función $f : \text{Dom}_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ Σ -computable, escribiremos $\overline{Vn+1} \leftarrow f(V1, \dots, \overline{Vn}, W1, \dots, \overline{Wm})$ para denotar un macro M que cumple las propiedades:

1. Las variables oficiales de M son: $V1, \dots, \overline{Vn}, \overline{Vn+1}, W1, \dots, \overline{Wm}$
2. M no tiene labels oficiales.
3. Si reemplazamos:
 1. Las variables oficiales por variables concretas $\overline{Nk_1}, \dots, \overline{Nk_n}, \overline{Nk_{n+1}}, \overline{Pj_1}, \dots, \overline{Pj_m}$
 2. Las variables auxiliares de M por variables concretas todas distintas entre sí y distintas de las ya mencionadas.
 3. Los labels auxiliares por labels concretos, también distintos entre sí.

Entonces, la palabra así obtenida es un programa de S^Σ que denotamos $\overline{Nk_{n+1}} \leftarrow f(\overline{Nk_1}, \dots, \overline{Nk_n}, \overline{Pj_1}, \dots, \overline{Pj_m})$ y debe satisfacer que si lo corremos partiendo de un estado e que le asigne a las variables $\overline{Nk_1}, \dots, \overline{Nk_n}, \overline{Nk_{n+1}}, \overline{Pj_1}, \dots, \overline{Pj_m}$ valores

$x_1, \dots, x_n, \alpha_1, \dots, \alpha_m$, entonces independientemente de los valores que les asigne e al resto de las variables se dará que:

1. Si $(\vec{x}, \vec{\alpha}) \notin \text{Dom}_f$, entonces el programa no se detiene.
2. Si $(\vec{x}, \vec{\alpha}) \in \text{Dom}_f$, entonces:
 - El programa se detiene;
 - Llega a un estado e' tq:
 1. e' asigna a $N\overline{k}_{n+1}$ el valor $f(\vec{x}, \vec{\alpha})$;
 2. e' solo difiere de e en el valor de $N\overline{k}_{n+1}$ y en aquellas variables que reemplazaron a las variables auxiliares de M .

La palabra $N\overline{k}_{n+1} \leftarrow f(N\overline{k}_1, \dots, N\overline{k}_n, P\overline{j}_1, \dots, P\overline{j}_m)$ se denomina la expansión del macro $V\overline{n} + 1 \leftarrow f(V1, \dots, V\overline{n}, W1, \dots, W\overline{m})$ respecto de la elección concreta de variables y labels realizada.

1.16. Combo 16: Dado un predicado $p : D_f \subseteq \omega \times \Sigma^* \rightarrow \omega$, describa qué tipo de objeto es y qué propiedades debe tener el macro: [IF P(V1,W1) GOTO A1]

Dado un predicado $P : \text{Dom}_P \subseteq \omega^n \times \Sigma^{*m} \rightarrow \{0, 1\}$ Σ -computable, escribiremos [IF $P(V1, \dots, V\overline{n}, W1, \dots, W\overline{m})$ GOTO A1] para denotar un macro M que cumple las siguientes propiedades:

1. Las variables oficiales de M son: $V1, \dots, V\overline{n}, W1, \dots, W\overline{m}$
2. A1 es el único label oficial de M .
3. Si reemplazamos:
 1. Las variables oficiales por variables concretas: $N\overline{k}_1, \dots, N\overline{k}_n, P\overline{j}_1, \dots, P\overline{j}_m$
 2. El label oficial A1 por un label concreto $L\overline{k}$
 3. Las variables auxiliares por variables concretas, todas distintas entre sí y distintas de las ya mencionadas
 4. Los labels auxiliares por labels concretos, todos distintos entre sí y distintos de $L\overline{k}$

Entonces, la palabra así obtenida es un programa de S^Σ (salvo por la ley del GOTO respecto de $L\overline{k}$) que denotaremos $[\text{IF } P(N\overline{k}_1, \dots, N\overline{k}_n, P\overline{j}_1, \dots, P\overline{j}_m) \text{ GOTO } L\overline{k}]$ y debe satisfacer que si lo corremos partiendo de un estado e que le asigne a las variables $N\overline{k}_1, \dots, N\overline{k}_n, P\overline{j}_1, \dots, P\overline{j}_m$ valores $x_1, \dots, x_n, \alpha_1, \dots, \alpha_m$, entonces independientemente de los valores que les asigne e al resto de las variables se dará que

1. Si $(\vec{x}, \vec{\alpha}) \notin \text{Dom}_P$, el programa no se detiene.
2. Si $(\vec{x}, \vec{\alpha}) \in \text{Dom}_P$ y $P(\vec{x}, \vec{\alpha}) = 1$, entonces:
 - Luego de una cantidad finita de pasos, se salta al label $L\overline{k}$.
 - El estado alcanzado e' también solo puede diferir de e en las variables que reemplazaron a las auxiliares de M , y no modifica las demás.
3. Si $(\vec{x}, \vec{\alpha}) \in \text{Dom}_P$ y $P(\vec{x}, \vec{\alpha}) = 0$, entonces:
 - Luego de una cantidad finita de pasos, el programa se detiene.

- El estado alcanzado e' también solo puede diferir de e en las variables que reemplazaron a las auxiliares de M , y no modifica las demás.

La palabra $\left[\text{IF } P(\overline{Nk_1}, \dots, \overline{Nk_n}, \overline{Pj_1}, \dots, \overline{Pj_m}) \text{ GOTO } L\overline{k} \right]$ se denomina la expansión del macro $\left[\text{IF } P(V1, \dots, V\overline{n}, W1, \dots, W\overline{m}) \text{ GOTO } A1 \right]$ respecto de la elección concreta de variables y labels realizada.

1.17. Combo 17: Defina el concepto de función y desarrolle las tres Convenciones Notacionales asociadas a dicho concepto

Una función es un conjunto de pares tq, si $(x, y) \in f$ y $(x, z) \in f$, entonces $y = z$.

Dada una función f , definimos:

- $\text{Dom}_f = \{x : (x, y) \in f \text{ para algún } y\}$
- $\text{Im}_f = \{y : (x, y) \in f \text{ para algún } x\}$

Dado $x \in \text{Dom}_f$, usaremos $f(x)$ para denotar al único $y \in \text{Im}_f$ tq $(x, y) \in f$.

Nótese que \emptyset es una función y que $\text{Dom}_{\emptyset} = \text{Im}_{\emptyset} = \emptyset$.

Escribimos $f : S \subseteq A \rightarrow B$ para expresar que f es una función tq $\text{Dom}_f = S \subseteq A$ y $\text{Im}_f \subseteq B$. También escribimos $f : A \rightarrow B$ si $S = A$. En tal contexto llamaremos a B conjunto de llegada.

Muchas veces para definir una función f , lo haremos dando su dominio y su regla de asignación.

Ademas, para hacer mas intuitiva aun la descripción de la función, también incluiremos un conjunto de llegada de dicha función y a la regla de asignación la escribiremos usando una flecha.

2. Combos de teoremas

2.1. Combo 1

1. **Proposición** (Caracterización de conjuntos Σ -p.r.): Un conjunto S es Σ -p.r. sii S es el dominio de alguna función Σ -p.r. (En la inducción de la prueba hacer solo el caso de la composición)
2. **Teorema** (Neumann vence a Gödel): Si h es Σ -recursiva, entonces h es Σ -computable. (En la inducción de la prueba hacer solo el caso $h = R(f, G)$, con $I_h \subseteq \omega$)

2.2. Combo 2

1. **Lema** (Lema de división por casos para funciones Σ -p.r.): Supongamos $f_i : D_{f_i} \subseteq \omega^n \times \Sigma^{*m} \rightarrow \Sigma^*$, $i = 1, \dots, k$, son funciones Σ -p.r. tales que $D_{f_i} \cap D_{f_j} = \emptyset$ para $i \neq j$. Entonces $f_1 \cap \dots \cap f_k$ es Σ -p.r. (Hacer el caso $k = 2$, $n = 2$ y $m = 1$)
2. **Proposición** (Caracterización básica de conjuntos Σ -enumerables): Sea $S \subseteq \omega^n \times \Sigma^{*m}$ un conjunto no vacío. Entonces son equivalentes:
 1. S es Σ -enumerable

2. Hay un programa $P \in \text{Pro}^\Sigma$ tq:
 1. Para cada $x \in \omega$, P se detiene partiendo desde el estado $\llbracket x \rrbracket$ y llega a un estado de la forma $((x_1, \dots, x_n, y_1, \dots), (\alpha_1, \dots, \alpha_m, \beta_1, \dots))$, donde $(x_1, \dots, x_n, \alpha_1, \dots, \alpha_m) \in S$
 2. Para cada $(x_1, \dots, x_n, \alpha_1, \dots, \alpha_m) \in S$ hay un $x \in \omega$ tq P se detiene partiendo desde el estado $\llbracket x \rrbracket$ y llega a un estado como en $((x_1, \dots, x_n, y_1, \dots), (\alpha_1, \dots, \alpha_m, \beta_1, \dots))$

(Hacer el caso $n = 2$ y $m = 1$)

2.3. Combo 3

1. **Teorema** (Gödel vence a Neumann): Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \Sigma^*$ es Σ -computable, entonces f es Σ -recursiva
2. **Teorema** (Caracterización de conjuntos Σ -efectivamente computables): 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
 (Hacer solo (b) \rightarrow (a))

2.4. Combo 4

1. **Proposición** (Caracterización básica de conjuntos Σ -enumerables): (igual a Combo 2, hacer caso $n = 2, m = 1$)
2. **Lema** (Lema de la sumatoria): Sea Σ un alfabeto finito. Si $f : \omega \times S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m \rightarrow \omega$ es Σ -p.r., con $S_i \subseteq \omega$ y $L_j \subseteq \Sigma^*$ no vacíos, entonces

$\lambda x y \vec{x} \vec{\alpha}. \sum_t = x^y f(t, \vec{x}, \vec{\alpha})$ es Σ -p.r.

2.5. Combo 5

1. **Lema**: Sea $\Sigma = @, \%, !$ y $f : S_1 \times S_2 \times L_1 \times L_2 \rightarrow \omega$, con $S_1, S_2 \subseteq \omega$ y $L_1, L_2 \subseteq \Sigma^*$ no vacíos. Sea G una familia Σ -indexada de funciones $G_a : \omega \times S_1 \times S_2 \times L_1 \times L_2 \times \Sigma^* \rightarrow \omega$ para cada $a \in \Sigma$.

Si f y cada G_a son Σ -efectivamente computables, entonces $R(f, G)$ lo es. (Ejercicio de la Guía 5)

1. **Lema** (Lema de cuantificación acotada): Sea $p : S \times S_1 \times \dots \times S_n \times L_1 \times \dots \times L_m \rightarrow \omega$ un predicado Σ -p.r., y $\bar{S} \subseteq S$ un conjunto Σ -p.r. Entonces

$\lambda x \vec{x} \vec{\alpha} \left[\left(\forall t \in \bar{S} \right)_{t \leq x} P(t, \vec{x}, \vec{\alpha}) \right]$ es Σ -p.r.

2.6. Combo 6

1. **Lema**: Si $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ -efectivamente computable, entonces S es Σ -efectivamente enumerable
2. **Teorema** (Caracterización de conjuntos Σ -r.e.): Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Son equivalentes:

(1) S es Σ -recursivamente enumerable (2) $S = \text{IF}$, para alguna $F : D_F \subseteq \omega^k \times \Sigma^{*l} \rightarrow \omega^n \times \Sigma^{*m}$ tq cada $F(i)$ es Σ -recursiva (3) $S = D_f$, para alguna función Σ -recursiva f (Hacer la prueba de (2) \rightarrow (3), con $k = l = 1$ y $n = m = 2$)

2.7. Combo 7

1. **Lema** (Lema de minimización acotada): Sean $n, m \geq 0$. Sea $p : D_p \subseteq \omega \times \omega^n \times \Sigma^{*m} \rightarrow \omega$ un predicado Σ -p.r.

(a) $M(P)$ es Σ -recursiva (b) Si existe una función $f : \omega^n \times \Sigma^{*m} \rightarrow \omega$ Σ -p.r. tq $M(P)(\vec{x}, \vec{\alpha}) = \min_t P(t, \vec{x}, \vec{\alpha}) \leq f(\vec{x}, \vec{\alpha})$, entonces $M(P)$ es Σ -p.r.

1. **Lema**: Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow O$ es Σ -recursiva y $S \subseteq D_f$ es Σ -r.e., entonces $f|_S$ es Σ -recursiva

(Hacer solo el caso S no vacío, $n = m = 1$ y $O = \Sigma^*$)

2.8. Combo 8

1. **Lema**: Si $\Sigma \supseteq \Sigma_p$, entonces AutoHalt^Σ no es Σ -recursivo

2. **Teorema**: Si $\Sigma \supseteq \Sigma_p$, entonces AutoHalt^Σ no es Σ -efectivamente computable

3. **Lema**: Sea $A = p \in \text{Pro}^\Sigma : \text{AutoHalt}^{\Sigma(P)} = 1$, entonces A es Σ -r.e. y no Σ -recursivo

Además, el conjunto $N = p \in \text{Pro}^\Sigma : \text{AutoHalt}^{\Sigma(P)} = 0$ no es Σ -r.e.

1. **Teorema** (Neumann vence a Gödel): Si h es Σ -recursiva, entonces h es Σ -computable

(Hacer solo el caso $h = M(P)$)

2.9. Combo 9

1. **Lema** (Lema de división por casos para funciones Σ -recursivas): Supongamos $f_i : D_{f_i} \subseteq \omega^n \times \Sigma^{*m} \rightarrow O$ para $i = 1, \dots, k$, tales que $D_{f_i} \cap D_{f_j} = \emptyset$ para $i \neq j$. Entonces $f_1 \sqcup \dots \sqcup f_k$ es Σ -recursiva

(Hacer el caso $k = 2$, $n = m = 1$ y $O = \omega$)

1. **Teorema** (Gödel vence a Neumann): Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \rightarrow \omega$ es Σ -computable, entonces f es Σ -recursiva