# Combo 1

July 3, 2024

# 1 Proposición 1: Caracterización de conjuntos p.r.

## 1.1 Enunciado

Un conjunto $S$ es $\Sigma$-p.r. sii $S$ es el dominio de alguna funcion $\Sigma$-p.r..

Nota: en la inducción de la prueba hacer solo el caso de la composición.

## 1.2 Demostración

Supongamos que $S \subseteq \omega^n \times \Sigma^{*m}$.

($\Rightarrow$) Note que $S = D_{Pred \circ \chi_S^{\omega^n \times \Sigma^{*m}}}$.

($\Leftarrow$) Probaremos por induccion en $k$ que $D_F$ es $\Sigma$-p.r., para cada $F \in \mathrm{PR}_k^\Sigma$. El caso $k = 0$ es facil. Supongamos el resultado vale para un $k$ fijo y supongamos $F \in \mathrm{PR}_{k+1}^\Sigma$. Veremos entonces que $D_F$ es $\Sigma$-p.r.. Hay varios, pero sólo consideraremos el de la composición.

Supongamos ahora que $F = g \circ [g_1, ..., g_r]$ con $g, g_1, ..., g_r \in \mathrm{PR}_k^\Sigma$. Si $F = \emptyset$, entonces es claro que $D_F = \emptyset$ es $\Sigma$-p.r.. Supongamos entonces que $F$ no es la funcion $\emptyset$. Tenemos entonces que $r$ es de la forma $n + m$ y

$$\begin{aligned} g &: D_g \subseteq \omega^n \times \Sigma^{*m} \to O \\ g_i &: D_{g_i} \subseteq \omega^k \times \Sigma^{*l} \to \omega,\ i = 1, ..., n \\ g_i &: D_{g_i} \subseteq \omega^k \times \Sigma^{*l} \to \Sigma^*, i = n+1, ..., n+m \end{aligned}$$

con $O \in \{\omega, \Sigma^*\}$ y $k, l \in \omega$. Por Lema de Extensión, sabemos que hay funciones $\Sigma$-p.r. $\bar{g}_1, ..., \bar{g}_{n+m}$ las cuales son $\Sigma$-totales y cumplen

$$g_i = \bar{g}_i|_{D_{g_i}}, \text{ para } i = 1, ..., n+m.$$

Por hipotesis inductiva los conjuntos $D_g$, $D_{g_i}$, $i = 1, ..., n+m$, son $\Sigma$-p.r. y por lo tanto

$$S = \bigcap_{i=1}^{n+m} D_{g_i}$$

lo es. Notese que

$$\chi_{D_F}^{\omega^k \times \Sigma^{*l}} = (\chi_{D_g}^{\omega^n \times \Sigma^{*m}} \circ [\bar{g}_1, ..., \bar{g}_{n+m}] \wedge \chi_S^{\omega^k \times \Sigma^{*l}})$$

lo cual nos dice que $D_F$ es $\Sigma$-p.r..

# 2 Teorema 2: Neumann vence a Godel

## 2.1 Enunciado

Si $h$ es $\Sigma$-recursiva, entonces $h$ es $\Sigma$-computable.

Nota: en la inducción de la prueba hacer solo el caso $h = R(f, \mathcal{G})$

## 2.2 Demostración

Probaremos por induccion en $k$ que

(*) Si $h \in \mathrm{R}_k^\Sigma$, entonces $h$ es $\Sigma$-computable.

El caso $k = 0$ es trivial. Supongamos (*) vale para $k$, veremos que vale para $k + 1$. Sea $h \in \mathrm{R}_{k+1}^\Sigma - \mathrm{R}_k^\Sigma$. Hay varios casos, pero solo consideraremos la recursión primitiva con variable y valor alfabéticos. Supongamos $h = R(f, \mathcal{G})$, con

$$\begin{aligned} f &: S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^* \\ \mathcal{G}_a &: S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^*,\ a \in \Sigma \end{aligned}$$

elementos de $\mathrm{R}_k^\Sigma$. Sea $\Sigma = \{a_1, ..., a_r\}$. Por hipotesis inductiva, las funciones $f$, $\mathcal{G}_a$, $a \in \Sigma$, son $\Sigma$-computables y por lo tanto podemos hacer el siguiente programa $\mathcal{P}$ via el uso de macros

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

Luego, si comprobamos que este programa computa $h$, demostraríamos este caso. Para ello, notemos que tenemos que ver que $h = \Psi_{\mathcal{P}}^{n,m,*}$:

- $D_h = D_{\Psi_{\mathcal{P}}^{n,m,*}}$: Es fácil de ver porque se hace uso de $\mathcal{G}$ en las macros y, en caso que no pertenezca al dominio de $h$, entonces estas macros no se dentendrían, por lo que no estaría en el dominio de $\Psi_{\mathcal{P}}^{n,m,*}$ tampoco. En todos los demás casos sí se detiene.

- $\forall(\vec{x}, \vec{\alpha}, \alpha) \in D_h, h(\vec{x}, \vec{\alpha}, \alpha) = \Psi_{\mathcal{P}}^{n,m,*}(\vec{x}, \vec{\alpha}, \alpha)$: Notemos que se cumple porque se hace desde el caso base de la recursión en adelante, manteniendo en los valores contenidos en las variables $P\overline{m+3}, P\overline{m+2}$ el valor actual de la recursión y la palabra por la que "iteramos". Además, se aplica siempre la correcta $\mathcal{G}_a (a \in \Sigma)$ para cada caso y se deja almacenado en $P1$ el valor de retorno correspondiente cuando $\mathcal{P}$se detiene.

El caso de recursión primitiva de variable alfabética y valor numérico es totalmente análogo.