neumann-v-godel-c1

> Si h es Σ-*recursiva* entonces es Σ-*computable*

Prueba por inducción

> (*) Caso k : si $h \in R_k^\Sigma$ entonces es Σ-*computable*

Caso k = 0

> Altamente trivial

Sea $h \in R_0^\Sigma$
Sean los siguientes programas los que computan las funciones básicas del paradigma de godel:

- $Succ : N1 \leftarrow N1 + 1$
- $Pred : N1 \leftarrow N1 \dot{-} 1$
- $C_0^{0,0} : N1 \leftarrow 0$
- $C_\epsilon^{0,0} : P1 \leftarrow \varepsilon$
- $d_a \quad a \in \Sigma : P1 \leftarrow P1.a$
- $p_j^{n,m} : N1 \leftarrow N\overline{j} \quad j \in 1, \ldots, n$
- $p_j^{n,m} : P1 \leftarrow P\overline{j - n} \quad j \in n + 1, \ldots, n + m$

Caso k+1

Sea $h \in R_{k+1}^\Sigma - R_k^\Sigma$

Caso h=R(f,G) (combo 1)

> Usando la hipótesis inductiva se obtienen las macros necesarias
> Luego realizar un programa que según el primer caractér de la variable de recursión realize la G apropiada
> Para el caso alfabético y numérico cambiar la variable de resultado segun sea apropiado

(Alfabético)

Supongamos $h = R(f, \mathcal{G})$ con

$$\begin{aligned} f &: S_1 \times \cdots \times S_n \times L_1 \times \cdots \times L_m \to \Sigma^* \\ \mathcal{G} &: S_1 \times \cdots \times S_n \times L_1 \times \cdots \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^* \end{aligned}$$

elementos de R_k^Σ. Sea $\Sigma = \{a_1, \dots, a_r\}$. Por (*) hay macros que computan las funciones f y $\mathcal{G}_a \quad a \in \Sigma$.

Damos el siguiente programa que computa su recursion:

$$\begin{array}{rl}
 & [P\overline{m+3} \leftarrow f(N1, \dots, N\bar{n}, P1, \dots, P\bar{m})] \\
L\overline{r+1} & IF\ P\overline{m+1}\ BEGINS\ a_1\ GOTO\ L\bar{1} \\
 & \vdots \\
 & IF\ P\overline{m+1}\ BEGINS\ a_r\ GOTO\ L\bar{r} \\
 & GOTO\ L\overline{r+2} \\
L1 & P\overline{m+1} \leftarrow {}^\curvearrowright P\overline{m+1} \\
 & [P\overline{m+3} \leftarrow \mathcal{G}_1(N1, \dots, N\bar{n}, P1, \dots, P\bar{m}, P\overline{m+2}, P\overline{m+3})]] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.\, a_1 \\
 & GOTO\ L\overline{r+1} \\
 & \vdots \\
Lr & P\overline{m+1} \leftarrow {}^\curvearrowright P\overline{m+1} \\
 & [P\overline{m+3} \leftarrow \mathcal{G}_r(N1, \dots, N\bar{n}, P1, \dots, P\bar{m}, P\overline{m+2}, P\overline{m+3})]] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.\, a_r \\
 & GOTO\ L\overline{r+1} \\
L\overline{r+2} & P1 \leftarrow P\overline{m+3}
\end{array}$$

(Numérico)

Muy similar al caso anterior, **solo requiere realizar un cambio de variables para el resultado**.
Supongamos $h = R(f, \mathcal{G})$ con

$$\begin{aligned} f &: S_1 \times \cdots \times S_n \times L_1 \times \cdots \times L_m \to \omega \\ \mathcal{G} &: \omega \times S_1 \times \cdots \times S_n \times L_1 \times \cdots \times L_m \times \Sigma^* \to \omega \end{aligned}$$

elementos de R_k^Σ.

Sea $\Sigma = \{a_1, \ldots, a_r\}$. Por (*) hay macros que computan las funciones f y $\mathcal{G}_a \quad a \in \Sigma$.
Damos el siguiente programa que computa su recursion:

$$\begin{array}{rl}
 & [N\overline{n+1} \leftarrow f(N1, \ldots, N\overline{n}, P1, \ldots, P\overline{m})] \\
L\overline{r+1} & IF\ P\overline{m+1}\ BEGINS\ a_1\ GOTO\ L\bar{1} \\
 & \vdots \\
 & IF\ P\overline{m+1}\ BEGINS\ a_r\ GOTO\ L\bar{r} \\
 & GOTO\ L\overline{r+2} \\
L\bar{1} & P\overline{m+1} \leftarrow {}^\curvearrowright P\overline{m+1} \\
 & [P\overline{m+3} \leftarrow \mathcal{G}_1(N\overline{n+1}, N1, \ldots, N\overline{n}, P1, \ldots, P\overline{m}, P\overline{m+2})]] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.\, a_1 \\
 & GOTO\ L\overline{r+1} \\
 & \vdots \\
L\bar{r} & P\overline{m+1} \leftarrow {}^\curvearrowright P\overline{m+1} \\
 & [N\overline{n+1} \leftarrow \mathcal{G}_r(N\overline{n+1}, N1, \ldots, N\overline{n}, P1, \ldots, P\overline{m}, P\overline{m+2})]] \\
 & P\overline{m+2} \leftarrow P\overline{m+2}.\, a_r \\
 & GOTO\ L\overline{r+1} \\
L\overline{r+2} & N1 \leftarrow N\overline{n+1}
\end{array}$$