F(MT)	*REC*	*F(WHILE)*
DEC(MT)	*DECI*	*DEC(WHILE)*
ENU(MT)	*ENUM*	*ENU(WHILE)*
funciones Turing-computables totales	*TREC*	*T-WHILE*
predicados Turing-decidibles	*PRED(TREC)*	*PRED(T-WHILE)*
predicados Turing-enumerables	*PRED(REC)*	*PRED(WHILE)*

$$codi : Cód\_While \rightarrow \mathsf{N}$$

$$codi(s) = \begin{cases} 5(i\text{-}1) & \text{si} \quad s = X_i{:=}0 \\ 5\sigma_1^{\,2}(i\text{-}1,\ j\text{-}1)+1 & \text{si} \quad s = X_i{:=}X_j \\ 5\sigma_1^{\,2}(i\text{-}1,\ j\text{-}1)+2 & \text{si} \quad s = X_i{:=}X_j+1 \\ 5\sigma_1^{\,2}(i\text{-}1,\ j\text{-}1)+3 & \text{si} \quad s = X_i{:=}X_j-1 \\ 5\sigma_1^{\,2}(i\text{-}1,\ Codi(s_1))+4 & \text{si} \quad s = while\ X_i \neq 0\ do\ s_1\ od \end{cases}$$

$$CODI : WHILE \rightarrow \mathsf{N}$$

$$CODI(Q) = \sigma_1^{\,3}(n,\, p - max\{n,k\},\, Codi(s))$$

donde $k = max\{\, m \in \Sigma_d^{+} \mid X_m \infty\, s \,\}$

$$decodi(z) = \begin{cases} X_i{:=}0 & \text{si} \;\; tipo(z) = 0 \\ X_i{:=}X_j & \text{si} \;\; tipo(z) = 1 \\ X_i{:=}X_j+1 & \text{si} \;\; tipo(z) = 2 \\ X_i{:=}X_j-1 & \text{si} \;\; tipo(z) = 3 \\ while \;\; X_i \neq 0 \;\; do \;\; DeCodi(j\text{-}1) \;\; od & \text{si} \;\; tipo(z) = 4 \end{cases}$$

Definición 12.9: *Función decodificación de códigos (DeCodi)*

Definimos la función decodificación de codigos, notada *DeCodi*, como:

$DeCodi : \mathsf{N} \rightarrow Cód\_While$

$DeCodi(z) =$

$decodi(degod(z+1,1))\,;\, decodi(degod(z+1,2))\,;\, ...;\, decodi(degod(z+1,l(z)))$

Definición 12.10: *Función decodificación de programas (DECODI)*

Definimos la función decodificación de programas, notada *DECODI*, como:

$DECODI : \mathsf{N} \rightarrow WHILE$

$DECODI(z) = (\ \sigma_{3,1}{}^{1}(z)\ ,\ \ \sigma_{3,2}{}^{1}(z) + max\{\sigma_{3,1}{}^{1}(z), k\}\ ,\ \ DeCodi(\sigma_{3,3}{}^{1}(z))\ )$

donde $k = max\{\ m \in \Sigma_d^{\ +}\ |\ X_m \prec DeCodi(\sigma_{3,3}{}^{1}(z))\ \}$

Definición 12.11: *Función universal (U)*

Sea F un conjunto numerable de funciones de N^n en N.

Diremos que la función $U[F] : \mathsf{N}^{n+1} \rightarrow \mathsf{N}$ es universal para la clase F sii:

$$\exists\, h : \mathsf{N} \rightarrow F \text{ indexación de } F \quad | \quad U[F](i, \underline{x}) = h(i)(\underline{x}) \qquad \forall i \in \mathsf{N} \wedge \forall \underline{x} \in \mathsf{N}^n$$

Notas: $h(i) \in F$

Cuando escogemos una indexación concreta h para $U[F]$, decimos que $U[F]$ es la función universal para F bajo la indexación h.

Definición 12.13: *Programa universal (U)*

$U(z, \underline{x})$

$m := god(\underline{x})$; (* inicializa m, la variable de la memoria *)

$m := Simular(z, m)$; (* simula el programa z con la memoria m *)

$X_1 := degod(m,1)$ (* obtiene de la memoria la salida de z *)

Definición 12.14: *Programa Simular*

```
Simular(z, m)
    z := z+1 ;                                   (* sumamos uno ya que al codificar restamos uno *)
    while l(z) ≠ 0 do                            (* mientras haya sentencias pendientes *)
        s := degod(z,1) ;                        (* obtengo el número de la primera sentencia *)
        if tipo(s) ≤ 3 then                      (* si es una asignación *)
            m := Ejecutar(s, m) ;                (* ejecutarla *)
            z := Reducir(z) ;                    (* y eliminar la sentencia de z *)
        else                                     (* si es una sentencia while *)
            if degod(m, extr(s,1)) ≠ 0 do        (* si la variable de control no es nula *)
                z := Añadir(z, extr(s,2))        (* añade a z el cuerpo del bucle *)
            else                                 (* si la variable de control es nula *)
                z := Reducir(z)                  (* eliminar la sentencia de z *)
            fi
        fi
    od ;
    X1 := m
```

Definición 12.15: *Función reemplazar (reem)*

Definimos la función reemplazar, notada *reem*, como:

$reem : \mathsf{N}^3 \rightarrow \mathsf{N}$

$$reem(z,k,x) = \begin{cases} god(degod(z,1), \ldots, degod(z,l(z)), 0, \ldots, 0, x) & \text{si } k > l(z) \wedge l(z) > 0 \\ god(0, \ldots, 0, x) & \text{si } k > l(z) \wedge l(z) = 0 \\ z & \text{si } k = 0 \\ god(degod(z,1), \ldots, degod(z,k-1), x, degod(z,k+1), \ldots, degod(z,l(z))) & \\ & \text{si } k \leq l(z) \wedge k \neq 0 \end{cases}$$

Definición 12.16: *Programa Ejecutar*

```
Ejecutar(s, m)
    if tipo(s) = 0 then
        m := reem(m, extr(s,1), 0)
    fi ;
    if tipo(s) = 1 then
        m := reem(m, extr(s,1), degod(m, extr(s,2)))
    fi ;
    if tipo(s) = 2 then
        m := reem(m, extr(s,1), degod(m, extr(s,2))+1)
    fi ;
    if tipo(s) = 3 then
        m := reem(m, extr(s,1), degod(m, extr(s,2))-1)
    fi ;
    X1 := m
```

Definición 12.17: *Programa Reducir*

Reducir(z)

$$
\begin{array}{l}
\quad if\ \ l(z) < 2\ \ then\ \ z := 0 \\
\quad else\ \ z := god(degod(z,2),\ degod(z,3),\ \ldots,\ degod(z,l(z))) \\
\quad fi\ ; \\
\quad X_1 := z
\end{array}
$$

Definición 12.18: *Programa Añadir*

Añadir(z,s)

$$X_1 := god(degod(s,1),\ \ldots,\ degod(s,l(s)),\ degod(z,1),\ \ldots,\ degod(z,l(z)))$$

problema	predicado	función
resoluble	decidible	$\in TREC$
parcialmente resoluble	enumerable	$\in REC$
no resoluble	no decidible	$\notin TREC$
totalmente no resoluble	no enumerable	$\notin REC$