Tema 1
Programas y Funciones Computables

Alberto García González

Universidad de Cádiz

Modelos de Computación - Universidad de Cádiz

Contenidos

1. Funciones numéricas
2. El lenguaje $\mathscr{L}$
3. Ejemplos de instrucciones del modelo
4. Modelo semántico de $\mathscr{L}$
5. Funciones $\mathscr{L}$-computables
6. Macros
7. El Modelo URM
8. El Modelo While-Loop

Funciones numéricas

Definición (Función numérica)

Una función numérica es una aplicación

$$f : \mathbb{N}^k \to \mathbb{N}, \text{ para } k \in \mathbb{N} \text{ y } k \geqslant 1$$

Ejemplo (Factorial)

$$fac : \mathbb{N} \to \mathbb{N}$$
$$fac(n) = n \cdot (n-1) \cdot \ldots \cdot 1$$

Ejemplo (Suma)

$$g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$g(x, y) = x + y$$

El lenguaje $\mathscr{L}$

Definición (Memoria)

La estructura de memoria de $\mathscr{L}$ es un conjunto de variables de la forma:

- De entrada: $\mathscr{V}_E = \{X_1, X_2, X_3, ...\}$ con $X_i \in \mathbb{N}$, para todo i
- Locales: $\mathscr{V}_L = \{Z_1, Z_2, Z_3, ...\}$ con $Z_j \in \mathbb{N}$, para todo j
- De salida: $Y \in \mathbb{N}$

Definición (Etiquetas)

$\mathscr{L}$ dispone de las siguientes etiquetas

$$\mathscr{L}ab = \{A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, D_2, E_2, ...\}$$

El lenguaje $\mathscr{L}$

Definición (Instrucciones)

Instrucción	Significado
`V ← V+1`	Incrementa a V la unidad.
`V ← V-1`	Decrementa a V la unidad. Si $V = 0$ su valor no cambia.
`V ← V`	Intrucción "dummy". No hace nada.
`IF V ≠ 0 GOTO L`	Si $V \neq 0$, salta a la instrucción etiquetada con $[L]$. Si $V = 0$, continúa con la siguiente instrucción. Si no hay instrucción etiquetada con $[L]$, el programa acaba.

El lenguaje $\mathscr{L}$

Definición (Valores iniciales de las variables)

- Entrada: $X_1 = x_1, X_2 = x_2, X_3 = x_3, ..., X_k = x_k$. El resto toman valor 0.
- Locales: $Z_j = 0, \forall j \in \mathbb{N}$
- Salida: $Y = 0$

Ejemplos de instrucciones del modelo

Ejemplo (Función pseudoidentidad)

```
[A] X ← X-1
    Y ← Y + 1
    IF X ≠ 0 GOTO A
```

$$f : \mathbb{N} \to \mathbb{N}$$

$$f(x) = \begin{cases} x, & \text{si } x \neq 0 \\ 1, & \text{si } x = 0 \end{cases}$$

Ejemplos de instrucciones del modelo

Ejemplo (Función identidad)

```
[A] IF X ≠ 0 GOTO B
    Z ← Z + 1
    IF Z ≠ 0 GOTO E
[B] X ← X - 1
    Y ← Y + 1
    Z ← Z + 1
    IF Z ≠ 0 GOTO A
```

$$g : \mathbb{N} \to \mathbb{N}$$

$$g(x) = x, \forall x \in \mathbb{N}$$

Ejemplo (Macro de bifurcación incondicional)

$$\texttt{GOTO E} \leftrightarrow \begin{array}{l} \texttt{Z} \leftarrow \texttt{Z + 1} \\ \texttt{IF Z} \neq \texttt{0 GOTO E} \end{array}$$

Ejemplos de instrucciones del modelo

Ejemplo (Macro de asignación)

Se define la macro V′ ← V como sigue:

```
[A]   IF X ≠ 0 GOTO B
      GOTO C
[B]   X ← X - 1
      Y ← Y + 1
      Z ← Z + 1
      GOTO A
[C]   IF Z ≠ 0 GOTO D
      GOTO E
[D]   Z ← Z - 1
      X ← X + 1
      GOTO C
```

Esta macro se ayuda de la variable local Z para no destruir X.

Ejemplos de instrucciones del modelo

Ejemplo (Función Suma)

Se define la macro $\texttt{V} \leftarrow \texttt{V}' + \texttt{V}''$ como sigue:

$$
\begin{array}{ll}
 & \texttt{Y} \leftarrow \texttt{X}_1 \\
 & \texttt{Z} \leftarrow \texttt{X}_2 \\
\texttt{[B]} & \texttt{IF Z} \neq \texttt{0 GOTO A} \\
 & \texttt{GOTO E} \\
\texttt{[A]} & \texttt{Z} \leftarrow \texttt{Z - 1} \\
 & \texttt{Y} \leftarrow \texttt{Y + 1} \\
 & \texttt{GOTO B}
\end{array}
$$

Ejemplos de instrucciones del modelo

Ejemplo (Función Resta Parcial)

Se define la macro $\mathtt{V} \leftarrow \mathtt{V}' \dot{-} \mathtt{V}''$ como sigue:

```
        Y ← X1
        Z ← X2
[C]     IF Z ≠ 0 GOTO A
        GOTO E
[A]     IF Y ≠ 0 GOTO B
        GOTO A
[B]     Y ← Y - 1
        Z ← Z - 1
        GOTO C
```

Ejemplos de instrucciones del modelo

Ejemplo (Función Producto)

Se define la macro $\mathtt{V} \leftarrow \mathtt{V}' \cdot \mathtt{V}''$ como sigue:

$$
\begin{array}{ll}
 & \mathtt{Z}_2 \leftarrow \mathtt{X}_2 \\
[\mathtt{B}] & \mathtt{IF}\ \mathtt{Z}_2 \neq 0\ \mathtt{GOTO}\ \mathtt{A} \\
 & \mathtt{GOTO}\ \mathtt{E} \\
[\mathtt{A}] & \mathtt{Z}_2 \leftarrow \mathtt{Z}_2 - 1 \\
 & \mathtt{Z}_1 \leftarrow \mathtt{X}_1 + \mathtt{Y} \\
 & \mathtt{Y} \leftarrow \mathtt{Z}_1 \\
 & \mathtt{GOTO}\ \mathtt{B}
\end{array}
$$

Modelo semántico de $\mathscr{L}$

Definición (Modélo semántico)

Un modelo semántico atribuye significado a los $\mathscr{L}$-programas.

Definición (Tipos de semántica)

De menor a mayor nivel de abstracción:

- Operacional: Trata de estudiar el comportamiento de una rutina estudiando directamente su código.
- Denotacional: Trata de estudiar el comportamiento de una rutina a nivel funcional. Se trata como una caja negra, analizando la salida en base a la entrada.
- Axiomática: Trata de estudiar el comportamiento de una rutina mediante la aplicación de reglas lógicas.

Tanto la semántica operacional como la denotacional son usadas a nivel práctico, mientras la semántica axiomática es usada únicamente a nivel teórico.

Modelo semántico de $\mathscr{L}$

Definición (Estado)

Un estado es una lista de ecuaciones de la forma $V = m$, donde V es una variable y $m \in \mathbb{N}$.

Ejemplo

- $< X_1 = 1, X_2 = 1, Z_1 = 0, Z_2 = 0, Y = 0 >$
- $< X_1 = 250, X_2 = 154, Z_1 = 3, Z_2 = 6, Y = 10000 >$

Modelo semántico de $\mathscr{L}$

Definición (Configuración)

Dado un $\mathscr{L}$-programa $\mathscr{P}$, una configuración es un par de la forma (i,σ) donde $i \in \mathbb{N}$, $(1 \leqslant i \leqslant long(P) + 1)$ y σ es un estado de $\mathscr{P}$.

Ejemplo

- $(1, < X_1 = 1, X_2 = 1, Z_1 = 0, Z_2 = 0, Y = 0 >)$
- $(7, < X_1 = 250, X_2 = 154, Z_1 = 3, Z_2 = 6, Y = 10000 >)$

Modelo semántico de $\mathscr{L}$

Definición (Sentencias de $\mathscr{L}$)

- $\texttt{V} \leftarrow \texttt{V + 1}$
- $\texttt{V} \leftarrow \texttt{V - 1}$
- $\texttt{V} \leftarrow \texttt{V}$
- $\texttt{IF V} \neq \texttt{0 GOTO L}$

Definición (Instrucciones de $\mathscr{L}$)

Una instrucción de un $\mathscr{L}$-programa $\mathscr{P}$ es una sentencia de la forma `I` o `[L] I`, donde `I` es una sentencia con una variable concreta tomada en $\mathscr{V}_E \cup \mathscr{V}_L \cup \{Y\}$ y $L \in \mathscr{L}ab$.

Modelo semántico de $\mathscr{L}$

Definición (Configuración sucesora)

Sea $\mathscr{P}$ un $\mathscr{L}$-programa y sean (i, σ) y (j, τ) dos configuraciones de $\mathscr{P}$. Se dice que (j, τ) es sucesora de (i, σ) si:

- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\texttt{V} \leftarrow \texttt{V}$, entonces $j = i + 1$ y $\tau = \sigma$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\texttt{V} \leftarrow \texttt{V + 1}$, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $V = m$ por $V = m + 1$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\texttt{V} \leftarrow \texttt{V - 1}$, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $V = m$ por $V = m - 1$. Si $m = 0$, no hay cambios.

Modelo semántico de $\mathscr{L}$

Definición (Configuración sucesora (cont.))

- Si la i-ésima instrucción es de la forma `IF V` $\neq$ `0 GOTO L`, entonces
  - Si $V \neq 0$, entonces $\tau = \sigma$ y j:
    - Si no hay instrucciones etiquetadas con $[L]$, entonces $j = long(\mathscr{P}) + 1$.
    - Si hay una única instrucción etiquetada con $[L]$ en la k-ésima posición de $\mathscr{P}$, entonces $j = k$.
    - Si hay varias instrucciones etiquetadas con $[L]$ en las posiciones $k_1, k_2, ..., k_n$ del programa, $j = min\{k_1, k_2, ..., k_n\}$.
  - Si $V = 0$, entonces $\tau = \sigma$ y $j = i + 1$.
- Notación: $(i, \sigma) \sim (j, \tau)$: (j, τ) es sucesora de (i, σ).

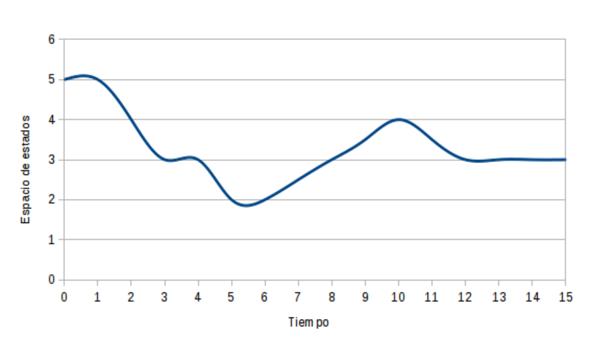
Definición (Computación)

Dado un $\mathscr{L}$-programa $\mathscr{P}$, una computación es una sucesión finita de configuraciones $S_1, S_2, ..., S_k$, donde $S_i \sim S_{i+1}$ para $i = 1, 2, ..., k - 1$ y S_k es final.

Modelo semántico de $\mathscr{L}$

Ejemplo

Una computación puede ser representada como una trayectoria en el espacio de estados, donde los valores tomados en el eje y indican la configuración adoptada en cada instante de tiempo.



Funciones $\mathscr{L}$-computables

Definición (Función $\mathscr{L}$-computable)

Sea un $\mathscr{L}$-programa $\mathscr{P}$. Sean $X_1, X_2, ..., X_k$ sus variables de entrada. Se construye la siguiente configuración inicial:

$$(1, < X_1 = r_1, X_2 = r_2, ..., X_k = r_k, Z_1 = 0, ..., Z_m = 0, Y = 0 >)$$

donde $r_i \in \mathbb{N}$ para $1 \leqslant i \leqslant k$. Se comienza la ejecución y entonces:

- Existe una computación $S_1, S_2, ..., S_k$ de P, donde S_k es terminal. Diremos que

  $$Y = \varphi_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k)$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.
- No existe tal computación. Diremos que

  $$\varphi_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k) = \uparrow$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.

Definición (Función parcialmente computable)

Una función $f : \mathbb{N}^k \to \mathbb{N}$ es parcialmente computable si existe un $\mathscr{L}$-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, ..., x_k) = \varphi_{\mathscr{P}}^{(k)}(x_1, x_2, ..., x_k)$$

Ejemplo

La resta restringida es parcialmente computable.

Funciones $\mathscr{L}$-computables

Definición (Función computable)

Una función $f : \mathbb{N}^k \to \mathbb{N}$ es totalmente computable si existe un $\mathscr{L}$-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, \dots, x_k) = \varphi_{\mathscr{P}}^{(k)}(x_1, x_2, \dots, x_k), \forall (x_1, x_2, \dots, x_k) \in \mathbb{N}^k$$

Ejemplo

La suma y el producto son funciones totalmente computables.

Macros

Sea $\mathscr{P}$ un $\mathscr{L}$-programa cualquiera, y sea $f : \mathbb{N}^k \to \mathbb{N}$ una función tal que

$$f(x_1, x_2, ..., x_k) = \varphi^k_{\mathscr{P}}(x_1, x_2, ..., x_k)$$

Nuestro objetivo es disponer de macros de la forma $w \leftarrow f(V_1, V_2, ..., V_k)$. Escribimos $\mathscr{P} = \mathscr{P}(Y, X_1, ..., X_n, Z_1, ..., Z_k; E, A_1, ..., A_l)$ de forma que podemos representar programas obtenidos de $\mathscr{P}$ reemplazando las variables y etiquetas por otras. En particular, escribiremos
$\mathscr{Q}_m = \mathscr{P}(Z_m, Z_{m+1}, ..., Z_{m+n}, Z_{m+n+1}, ..., Z_{m+n+k}; E_m, A_{m+1}, ..., A_{m+l})$ para cualquier m. Habrá que preparar las variables para un estado inicial.

Macros

- Preparación de las variables para la expansión de una macro

$$
\begin{array}{rl}
 & \mathtt{Z}_m \leftarrow 0 \\
 & \mathtt{Z}_{m+1} \leftarrow \mathtt{V}_1 \\
 & \mathtt{Z}_{m+2} \leftarrow \mathtt{V}_2 \\
 & \vdots \\
 & \mathtt{Z}_{m+n} \leftarrow \mathtt{V}_n \\
 & \mathtt{Z}_{m+n+1} \leftarrow 0 \\
 & \mathtt{Z}_{m+n+2} \leftarrow 0 \\
 & \vdots \\
 & \mathtt{Z}_{m+n+k} \leftarrow 0 \\
 & \mathscr{Q}_m \\
[\mathtt{E}_m] & \mathtt{W} \leftarrow \mathtt{Z}_m
\end{array}
$$

Macros

- Otras macros

$$\texttt{IF P(V}_1\texttt{,V}_2\texttt{,...,V}_k\texttt{) GOTO L} \leftrightarrow \begin{array}{l} \texttt{W} \leftarrow \texttt{P(V}_1\texttt{,V}_2\texttt{,...,V}_k\texttt{)} \\ \texttt{IF W} \neq \texttt{0 GOTO L} \end{array}$$

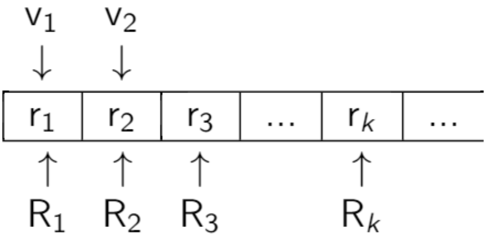
El Modelo URM

El modelo URM (Unlimited Register Machine) representa una abstracción de una máquina de registros ilimitados.

Definición (Memoria)

La memoria de URM se compone de una serie de celdas llamadas registros que pueden almacenar cualquier número natural. Un URM-programa puede hacer uso de un número finito de registros. Los registros serán referenciados con las letras $R_1, R_2, ..., R_k$, y sus valores correspondientes pueden ser referenciados con la letra minúscula correspondiente: $r_1, r_2, ..., r_k$.
Al inicio de un URM-programa, los valores de entrada $v_1, v_2, ..., v_k$ son almacenados en los registros $R_1, R_2, ..., R_k$, respectivamente, y el resto de registros son inicializados con valor 0.
Al finalizar la ejecución del programa, el valor de salida es almacenado en el registro R_1.

El Modelo URM



Definición (Instrucciones de URM)

Instrucción	Significado
`Z(n)`	Reemplaza el valor del registro R_n por 0.
`S(n)`	Incrementa el registro R_n en la unidad.
`T(m,n)`	Copia el valor del registro R_m en el registro R_n.
`J(m,n,i)`	Si los registros R_m y R_n tienen el mismo valor, entonces se salta a la i-ésima instrucción. Si su valor es distinto, se continúa con la siguiente instrucción. Si no existe la i-ésima instrucción, el programa termina.

Ejemplo (Suma)

```
J(2,3,0)
S(1)
S(3)
J(1,1,1)
```

Ejemplo (Resta Restringida)

```
J(1,2,5)
S(2)
S(3)
J(1,1,1)
T(3,1)
```

Ejemplo (Producto)

```
J(1,3,0)
J(2,4,10)
Z(5)
J(1,5,8)
S(3)
S(5)
J(1,1,4)
S(4)
J(1,1,2)
T(3,1)
```

El Modelo URM

Definición (Estado)

Un estado es una lista de ecuaciones de la forma $R_i = m$, donde R_i es un registro, $i \in \mathbb{Z}$ y $m \in \mathbb{N}$.

Definición (Configuración)

Dado un URM-programa $\mathscr{P}$, una configuración es un par de la forma (i,σ) donde $i \in \mathbb{N}$, $(1 \leqslant i \leqslant long(P) + 1)$ y σ es un estado de $\mathscr{P}$.

El Modelo URM

Definición (Sentencias de URM)

- `Z(n)`
- `S(n)`
- `T(m,n)`
- `J(m,n,i)`

Definición (Instrucciones de URM)

Una instrucción de un URM-programa $\mathscr{P}$ es una sentencia donde cada uno de los registros a los que se hace referencia pertenecen a $\mathbb{N}$.

El Modelo URM

Definición (Configuración sucesora)

Sea $\mathscr{P}$ un URM-programa y sean (i, σ) y (j, τ) dos configuraciones de $\mathscr{P}$. Se dice que (j, τ) es sucesora de (i, σ) si:

- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\mathtt{Z(n)}$, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $R_n = v$ por $R_n = 0$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\mathtt{S(n)}$, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $R_n = v$ por $R_n = v + 1$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma $\mathtt{T(m,n)}$, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $R_n = v$ por $R_n = w$, siendo w el valor del registro R_m.

El Modelo URM

Definición (Configuración sucesora (cont.))

- Si la i-ésima instrucción es de la forma `J(m,n,k)`, entonces
  - Si $R_m \neq R_n$, entonces $\tau = \sigma$ y $j = i + 1$.
  - Si $R_m = R_n$, entonces $\tau = \sigma$ y:
    - Si existe una k-ésima instrucción, entonces $j = k$.
    - Si $k = 0$ o $k > long(\mathscr{P})$, entonces el programa termina.

Definición (Computación)

Dado un URM-programa $\mathscr{P}$, una computación es una sucesión finita de configuraciones $S_1, S_2, ..., S_k$, donde $S_i \sim S_{i+1}$ para $i = 1, 2, ..., k-1$ y S_k es final.

El Modelo URM

Definición (Función URM-computable)

Sea un URM-programa $\mathscr{P}$. Se construye la siguiente configuración inicial:

$$(1, < R_1 = r_1, R_2 = r_2, ..., R_k = r_k >)$$

donde $r_i \in \mathbb{N}$ para $1 \leqslant i \leqslant k$. Se comienza la ejecución y entonces:

- Existe una computación $S_1, S_2, ..., S_k$ de P, donde S_k es terminal. Diremos que

  $$Y = \mu_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k)$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.
- No existe tal computación. Diremos que

  $$\mu_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k) = \uparrow$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.

Definición (Función parcialmente computable)

Una función $f : \mathbb{N}^k \to \mathbb{N}$ es parcialmente computable si existe un URM-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, ..., x_k) = \mu^{(k)}_{\mathscr{P}}(x_1, x_2, ..., x_k)$$

Definición (Función computable)

Una función $f : \mathbb{N}^k \to \mathbb{N}$ es totalmente computable si existe un URM-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, ..., x_k) = \mu^{(k)}_{\mathscr{P}}(x_1, x_2, ..., x_k), \forall (x_1, x_2, ..., x_k) \in \mathbb{N}^k$$

El modelo While-Loop, el cual estudiaremos mediante el lenguaje L (no confundir con $\mathscr{L}$) es capaz de computar exactamente las funciones primitivas recursivas. Los programas escritos en este lenguaje suelen llamarse *loop*-programas.

Definición (Memoria)

La estructura de memoria de L es similar a la vista anteriormente en el modelo $\mathscr{L}$. Se define como un conjunto de variables de la forma:

- De entrada: $\mathscr{V}_E = \{X_1, X_2, X_3, ...\}$ con $X_i \in \mathbb{N}$, para todo i
- Locales: $\mathscr{V}_L = \{Z_1, Z_2, Z_3, ...\}$ con $Z_j \in \mathbb{N}$, para todo j
- De salida: $Y \in \mathbb{N}$

El Modelo While-Loop

Definición (Instrucciones de While-Loop)

Instrucción	Significado
`V ← 0`	Asigna a `V` el valor 0.
`V ← V+1`	Incrementa `V` en la unidad.
`V ← V'`	Copia en `V` el valor de `V'`.
`LOOP V`	Determina el comienzo de un bucle.
`END`	Determina el final de un bucle.

El Modelo While-Loop

- Las instrucciones `LOOP V` y `END` siempre deben aparecer por pares.
- Su funcionamiento es el siguiente: en el momento en el que la ejecución pasa por una instrucción `LOOP V`, las instrucciones comprendidas entre ésta y la instrucción `END` correspondiente serán ejecutadas tantas veces como indique el valor de `V` en ese momento, sin importar si se modifica su valor en el interior del bloque `LOOP - END`.
- Este comportamiento implica que todas las funciones modeladas bajo este modelo son totalmente computables.

El Modelo While-Loop

Ejemplo (Suma)

```
Z ← X_1
LOOP X_2
Z ← Z + 1
END
Y ← Z
```

Ejemplo (Producto)

```
LOOP X_1
LOOP X_2
Y ← Y + 1
END
END
```

El Modelo While-Loop

Definición (Valores iniciales de las variables)

- Entrada: $X_1 = x_1, X_2 = x_2, X_3 = x_3, ..., X_k = x_k$. El resto toman valor 0.
- Locales: $Z_j = 0, \forall j \in \mathbb{N}$
- Salida: $Y = 0$

Definición (Estado)

Un estado es una lista de ecuaciones de la forma $V = m$, donde V es una variable y $m \in \mathbb{N}$.

Definición (Configuración)

Dado un *loop*-programa $\mathscr{P}$, una configuración es un par de la forma (i,σ) donde $i \in \mathbb{N}$, $(1 \leqslant i \leqslant long(P) + 1)$ y σ es un estado de $\mathscr{P}$.

El Modelo While-Loop

Definición (Sentencias de L)

- `V ← 0`
- `V ← V + 1`
- `V ← V'`
- `LOOP`
- `END`

Definición (Instrucciones de L)

Una instrucción de un *loop*-programa $\mathscr{P}$ es una sentencia del lenguaje con una variable concreta tomada en $\mathscr{V}_E \cup \mathscr{V}_L \cup \{Y\}$.

El Modelo While-Loop

Definición (Configuración sucesora)

Sea $\mathscr{P}$ un *loop*-programa y sean (i, σ) y (j, τ) dos configuraciones de $\mathscr{P}$. Se dice que (j, τ) es sucesora de (i, σ) si:

- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma `V ← 0`, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $V = m$ por $V = 0$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma `V ← V + 1`, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $V = m$ por $V = m + 1$.
- Si la i-ésima instrucción de $\mathscr{P}$ es de la forma `V ← V'`, entonces $j = i + 1$ y τ se obtiene de σ cambiando la ecuación $V = m$ por $V = V'$.

El Modelo While-Loop

Definición (Configuración sucesora (cont.))

- Si la i-ésima instrucción es de la forma `LOOP V`, entonces $j = i + 1$ y $\tau = \sigma$. Las instrucciones comprendidas entre ésta y su correspondiente instrucción `END` serán ejecutadas tantas veces como indique el valor de `V`.
- Si la i-ésima instrucción es de la forma `END`, entonces $\tau = \sigma$ y $j = k$, donde k es el número de la instrucción `LOOP V` emparejada con esta instrucción.

Definición (Computación)

Dado un *loop*-programa $\mathscr{P}$, una computación es una sucesión finita de configuraciones $S_1, S_2, ..., S_k$, donde $S_i \sim S_{i+1}$ para $i = 1, 2, ..., k - 1$ y S_k es final.

Definición (Función *loop*-computable)

Sea un *loop*-programa $\mathscr{P}$. Sean $X_1, X_2, ..., X_k$ sus variables de entrada. Se construye la siguiente configuración inicial:

$$(1, < X_1 = r_1, X_2 = r_2, ..., X_k = r_k, Z_1 = 0, ..., Z_m = 0, Y = 0 >)$$

donde $r_i \in \mathbb{N}$ para $1 \leqslant i \leqslant k$. Se comienza la ejecución y entonces:

- Existe una computación $S_1, S_2, ..., S_k$ de P, donde S_k es terminal. Diremos que

  $$Y = \delta_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k)$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.
- No existe tal computación. Diremos que

  $$\delta_{\mathscr{P}}^{(k)}(X_1, X_2, ..., X_k) = \uparrow$$

  donde $X_1, X_2, ..., X_k$ son valores concretos y corresponden a $r_1, r_2, ..., r_k$.

Definición (Función parcialmente computable)

Una función $f : \mathbb{N}^k \rightarrow \mathbb{N}$ es parcialmente computable si existe un *loop*-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, ..., x_k) = \delta_{\mathscr{P}}^{(k)}(x_1, x_2, ..., x_k)$$

Definición (Función computable)

Una función $f : \mathbb{N}^k \rightarrow \mathbb{N}$ es totalmente computable si existe un *loop*-programa $\mathscr{P}$, tal que:

$$f(x_1, x_2, ..., x_k) = \delta_{\mathscr{P}}^{(k)}(x_1, x_2, ..., x_k), \forall (x_1, x_2, ..., x_k) \in \mathbb{N}^k$$