Funciones Recursivas de Lista
(Homenaje a Giuseppe Jacopini)

Raúl Kantor - Giovanna Sontacchi

Junio 2026

Índice general

Prólogo

La ciencia de la computabilidad se construyó con la convergencia de diversos intentos de definición de lo que es “calculable”.

Se desarrollaron, en algunos casos en forma totalmente independiente, un mosaico de modelos que, pese a sus aparentes radicales diferencias de partida, demostraron una idéntica potencia.

En ese archipiélago de formalismos algunos de ellos fueron tomados por la academia como referentes principales y al mismo tiempo, el destino de otras ideas de equivalente poder expresivo fue un injusto, no intencional, olvido.

Estos criterios de preferencias globales pueden atribuirse a situaciones no necesariamente vinculadas a una superioridad estética o conceptual, sino a hechos dependientes de urgencias técnicas territoriales o, simplemente, del azar histórico.

El modelo de cálculo que se presenta en este libro fue creado por el extraordinario lógico **Giuseppe Jacopini** para los cursos de ”Lógica Matemática”que dictara en la Universidad de Roma a finales de la década del 70.

Su propuesta conceptual, de gran belleza, toma la lista finita como tipo de dato nativo del cálculo y reduce al mínimo (seis funciones base y dos operadores) las herramientas constructivas.

Por su sencillez y elegancia las **Funciones Recursivas de Lista** merecen un tratamiento similar al de otros modelos de difusión masiva.

Sin embargo, por lo que hemos podido relevar, sólo se ha preservado desde 1995, en la Licenciatura en Ciencias de la Computación de la Universidad Nacional de Rosario, una parte de ese legado.

Probablemente en algún repositorio de Italia se encuentre alguna referencia a este modelo[1]

Pero solo hemos encontrado dos fuentes en la red [2], en contraste con los millones que se pueden encontrar, por ejemplo, sobre Máquina de Turing.

Es por ello que rescatar la creación de Jacopini, a partir de nuestros lejanos recuerdos y dispersos apuntes de sus clases, no es un ejercicio de nostalgia, sino un acto de estricta justicia.

[1] Utilizamos en este trabajo partes de una ”dispensa”(apunte) de la Profesora Mentrasti, auxiliar de la cátedra. Lamentablemente tenemos una versión incompleta

[2] Listas y Funciones de Lista (Repositorio de la Universidad Nacional de La Plata) https://share.google/ozBGs2poodavbtgLj (presentación orientada a la enseñanza) Temas de Teoría de la Computación (Proyecto LATIN) https://rephip.unr.edu.ar/items/b1ae26a0-6129-454b-bed7-4ac5f9028150¨

Capítulo 1

Funciones Recursivas de Lista

1.1. Introducción

A partir de las **Funciones Recursivas de Lista (FRL)** pueden ser construidos todos los conceptos base (límites de los algoritmos, recursividad, semi-recursividad, etc.) de la fundamentación del cálculo.

Esbozaremos sólo algunas de las aplicaciones del modelo, que tiene, además, un fuerte potencial didáctico.

En capìtulos posteriores mostraremos como se relacionan con las **Funciones Recursivas (FR)** y las **Máquinas de Turing(MT)** [1], mostrando su equivalente capacidad de modelizar el cálculo.

Finalizamos con la construcción de una **FRL** "Universal"que permite asociar a todo proceso de cálculo un número natural.

1.2. Listas finitas

Definiciones 1.2.1.

Llamaremos $\mathcal{L}$ al conjunto de todas las listas finitas de elementos de los números naturales (incluyendo el 0).

Incluiremos en ese conjunto la ***lista vacía*** $\langle\ \rangle$

Representaremos tales listas con la notación

$< x_1, x_2, ..., x_m >$ *con* $x_i \in \mathbb{N}_0$

Emplearemos las letras mayúsculas para notar las listas en forma genérica

La ***longitud*** *de una lista es el número de elementos que posee.*

Para cada $m \in \mathbb{N}_0$ notaremos con $\mathcal{L}^m$ el conjunto de las listas que poseen exactamente m elementos.

Notaremos $\mathcal{L}^{\geq m}$ el conjunto de las listas que poseen al menos m elementos.

[1] En este trabajo suponemos que el lector está familiarizado con ambos modelos. Se hace un breve repaso de los fundamentos de estos en el APENDICE final (Capítulo 4)

Notaremos $\mathcal{L}^{\leq m}$ *el conjunto de las listas que poseen como máximo* m *elementos.*

La operación natural en $\mathcal{L}$ *es la* ***concatenación***
Dadas dos listas $X =< x_1, x_2, ..., x_m >$, $Y =< y_1, y_2, ..., y_k >$
llamamos ***concatenación*** *de* X *e* Y *la lista:*

$$< x_1, x_2, ..., x_m, y_1, y_2, ..., y_k >$$

Es sencillo mostrar que esta es una operación asociativa que tiene como elemento neutro la lista vacía

En algunos casos notaremos $X,\ Y$ *o* $< X,\ Y >$ *para indicar la lista obtenida al concatenar la lista* X *y la lista* Y

También usaremos, para distinguir en particular algunos elementos, notaciones del tipo $a,\ X,\ b,\ Y$ *o* $<\ a,\ X,\ \ b,\ Y >$ *para referirnos, por ejemplo, a la lista que se obtiene de concatenar las listas* $<\ a\ >,\ \ X,\ <\ b\ >,\ \ Y$.

Llamaremos ***funciones de listas*** *las funciones que van de* $\mathcal{L}$ *a* $\mathcal{L}$
Las notaremos con letras mayúsculas: F, G.
Dada una función de lista F *indicaremos el dominio de la función como* $\mathcal{D}_F$
De acuerdo al contexto para indicar como opera una función F *sobre una lista dada notaremos*

$$F(X) = Y$$

$$X \xrightarrow{F} Y$$

$$X \ \text{———} \ Y$$

1.3. Funciones Recursivas de Listas

A continuación, definiremos recursivamente un conjunto particular de funciones de lista a partir de seis funciones elementales y dos operadores.

Definiciones 1.3.1 (**Funciones base**)**.**

Llamaremos ***funciones base*** *las seis funciones siguientes:*

I) ***Cero a izquierda***

Esta función agrega un 0 *en el extremo izquierdo de la lista.*

La notaremos 0_i

$$0_i : \mathcal{L} \to \mathcal{L} \qquad 0_i < x_1, x_2, ..., x_k > \longmapsto < 0, x_1, x_2, ..., x_k >$$

$$X \xrightarrow{0_i} 0, X$$

II) ***Cero a derecha***

Esta función agrega un 0 *en el extremo derecho de la lista.*

La notaremos 0_d

$$0_d : \mathcal{L} \to \mathcal{L} \qquad 0_d < x_1, x_2, ..., x_k > \longmapsto < x_1, x_2, ..., x_k, 0 >$$

$$X \xrightarrow{0_d} X, 0$$

III) ***Borrar a izquierda***

Esta función, cuyo dominio son las listas no vacías, borra el elemento del extremo izquierdo de la lista.

La notaremos $\square_i$

$$\square_i : \mathcal{L}^{>0} \to \mathcal{L} \qquad \square_i < x_1, x_2, ..., x_k > \longmapsto < x_2, ..., x_k >$$

$$y, X \xrightarrow{\square_i} X$$

IV) ***Borrar a derecha***

Esta función, cuyo dominio son las listas no vacías, borra el elemento del extremo derecho de la lista.

La notaremos $\square_d$

$$\square_d : \mathcal{L}^{>0} \to \mathcal{L} \qquad \square_d < x_1, x_2, ..., x_{k-1}, x_k > \longmapsto < x_1, x_2, ..., x_{k-1} >$$

$$X, y \xrightarrow{\square_d} X$$

v) ***Sucesor a izquierda***

Esta función, cuyo dominio son las listas no vacías, incrementa en uno el elemento del extremo izquierdo de la lista.

La notaremos S_i

$$S_i : \mathcal{L}^{>0} \to \mathcal{L} \qquad S_i < x_1, x_2, ..., , x_k > \longmapsto < x_1 + 1, x_2, ..., x_k >$$

$$y, X \xrightarrow{S_i} y + 1, X$$

vi) ***Sucesor a derecha***

Esta función, cuyo dominio son las listas no vacías, incrementa en uno el elemento del extremo derecho de la lista.

La notaremos S_d

$$S_d : \mathcal{L}^{>0} \to \mathcal{L} \qquad S_d < x_1, x_2, ..., x_k > \longmapsto < x_1, x_2, ..., x_k + 1 >$$

$$X, y \xrightarrow{S_d} X, y + 1$$

Ejemplos 1.3.1.

1. $$0_i < 3, 4, 9 > \to < 0, 3, 4, 9 > \quad ; \quad 0_d < 3, 4, 9 > \to < 3, 4, 9, 0 >$$

2. $$\square_i < 3, 4, 9 > \to < 4, 9 > \quad ; \quad \square_d < 3, 4, 9 > \to < 3, 4 >$$

3. $$S_i < 3, 4, 9 > \to < 4, 4, 9 > \quad ; \; S_d < 3, 4, 9 > \to < 3, 4, 10 >$$

1.3.1. Operadores

Sólo utilizaremos dos operadores

Definición 1.3.1 (**Operador Composición**)**.**

Sean F, G *dos funciones de lista.*

$$F : \mathcal{D}_F \longrightarrow \mathcal{L} \qquad G : \mathcal{D}_G \longrightarrow \mathcal{L}$$

A partir de las mismas decimos que construimos por ***Composición*** *una nueva función de lista* H *(notaremos* $H = FG$*)*

$$H[X] = G[F[X]]$$

O sea que consiste en operar primero la función F *y al resultado aplicarle la función* G*.*

Con respecto al dominio de FG

$$X \in \mathcal{D}_{FG} \quad si \quad X \in \mathcal{D}_F \quad y \quad F[X] \in \mathcal{D}_G$$

Se puede observar que podemos pensar cualquier lista como el resultado de aplicar a la lista vacía, una adecuada **composición** de funciones base

$$Por\ \ ejemplo\ \ <2,\ 0,\ 1> =\ 0_i\ 0_i\ 0_i S_i\ S_i\ \ S_d\ <>$$

Por lo que la aplicación de cualquier función de lista **F** se puede pensar como la composición de dicha **F** con una determinada composición de funciones base aplicadas a la lista <> .

Definición 1.3.2 (**Operador Repetición**)**.**

Sea F una función de lista.
Diremos que a partir de la misma se construye por ***repetición*** *una nueva función H, que notaremos* $H = \{F\}$ *definida del siguiente modo:*

Sea una lista de la forma x, X, y *(la presentamos de modo tal de distinguir el primero y el último elemento de la lista) .*

$H = \{F\}$ *opera del siguiente modo:*

$$H = \{F\} < x, X, y >= \begin{cases} x, X, y & \text{si } x = y \\ \{F\}F < x, X, y > & \text{si x} \neq \text{y} \end{cases}$$

O sea $\{F\}$ *consiste en repetir la aplicación de F hasta que el primer componente de la n-upla dato sea igual al último. Observar que la parte que denominamos con X eventualmente puede ser la lista vacía, y que en el caso de una lista con un solo elemento la lista no se modifica. Por lo tanto* $X \in \mathcal{D}_{\{F\}}$ *si en la sucesión* $X, F[X], F[F[X]], ...$ *hay un elemento en el que la primera componente es igual a la última.*
Si eso no sucede, NO está definida la operación.
Por ejemplo, dada función $H = \{0_d\}$

$$X =< 1, 0 > \notin \mathcal{D}_H$$

A partir de las seis funciones base y los dos operadores construiremos en forma inductiva el conjunto de las **Funciones Recursivas de Lista**

Definición 1.3.3.

Definiremos el conjunto de ***Funciones Recursivas de Lista*** *de la siguiente manera:*

a) Las ***funciones base*** *definidas en 1.3.1 son* ***Funciones Recursivas de Lista*** *.*

b) Las funciones obtenidas a partir de ***Funciones Recursivas de Lista*** *aplicándoles un número finito de las operaciones de Composición y Repetición definidas en 1.3.1 y 1.3.2 son* ***Funciones Recursivas de Lista***

Las notaremos **FRL**.

Veamos algunos ejemplos de **FRL**

Ejemplos 1.3.2.

1. ***Pasar a la derecha***

   Esta función pasa el elemento que está en el extremo izquierdo de la lista al extremo derecho .

   La notaremos $\triangleright$

   Se puede ver que se construye a partir de las funciones base y los operadores del siguiente modo:

$$\triangleright \ = 0_d\big\{S_d\big\}\square_i$$

2. ***Pasar a la izquierda***

   Esta función, similar a la anterior, pasa el elemento que está en el extremo derecho de la lista al otro extremo .

   La notaremos $\triangleleft$

$$X, y \xrightarrow{\triangleleft} y, X$$

   Se puede construir del siguiente modo:

$$\triangleleft \ = 0_i\big\{S_i\big\}\square_d$$

3. ***Duplicar a izquierda***

   Esta función, que tiene como dominio las listas de al menos un elemento, duplica el primer elemento de la lista.

   La notaremos D_i

$$D_i \quad < x, X > \longrightarrow < x, x, X >$$
$$x, X \xrightarrow{D_i} x, x, X$$

   Se construye del siguiente modo:

$$D_i \quad = 0_d \ \big\{S_d\big\} \ \triangleleft$$

4. ***Duplicar a derecha***

   Similar a la anterior agrega al final de la lista una copia del último elemento de la lista.

   La notaremos D_d

$$D_d \quad < X, y > \longrightarrow < X, y, y >$$
$$X, y \xrightarrow{D_d} X, y, y$$

   Se construye del siguiente modo:

$$D_d \ = 0_i\big\{S_i\big\} \ \triangleright$$

5. ***Cambia extremos***

Esta función intercambia los extremos de la lista.

La notaremos $\leftrightarrow$

$$\leftrightarrow\ < x, X, y > \longrightarrow < y, X, x >$$

$$x, X, y \xrightarrow{\leftrightarrow} y, X, x$$

Se sugiere tratar de construir esta función y luego controlar con la propuesta que sigue:

$$\leftrightarrow\ =\ \rhd\, 0_i \lhd 0_i \{ S_i \rhd \rhd S_i \lhd \lhd \} \square_d \square_i \rhd$$

Podemos pensar cada una de las funciones recién vistas como "subrutinas"que nos servirán para construir nuevas funciones.

Lema 1.3.1. *Son* **FRL** *las funciones* Σ*,* Pd *,*Π*,* Fac*,* Exp *definidas del siguiente modo:*

I) $\Sigma(x, y, Y) \stackrel{def}{=} x + y, x, y, Y \quad (Suma)$

$$\mathcal{D}(\Sigma) = \mathcal{L}^{\geq 2}$$

II)

$$Pd(x, Y) \stackrel{def}{=} \begin{cases} 0, x, Y & \text{si } x = 0 \\ x - 1, x, Y & \text{si x} > 0 \end{cases} \quad (Predecesor)$$

$$\mathcal{D}(Pd) = \mathcal{L}^{\geq 1}$$

III) $\Pi(y, x, Y) \stackrel{def}{=} y.x, y, x, Y \quad (Producto)$

$$\mathcal{D}(\Pi) = \mathcal{L}^{\geq 2}$$

IV) $Fac(x, Y) \stackrel{def}{=} x!, x, Y \quad (Factorial)$

$$\mathcal{D}(Fac) = \mathcal{L}^{\geq 1}$$

V) $Exp(y, x, Y) \stackrel{def}{=} x^y, y, x, Y$ *(Exponencial) con la convención de que* $0^0 = 1$

$$\mathcal{D}(Exp) = \mathcal{L}^{\geq 2}$$

Las demostraciones de que estas funciones son **FRL** *son constructivas.*

Veamos, por ejemplo que la función Σ *se puede construir de la siguiente manera:*

$$\Sigma = \rhd\ D_d\ 0_d \left\{ \lhd\ S_d\ \rhd\ S_d \right\} \square_d \leftrightarrow \lhd$$

Veamos que Σ*, que fue construida como* **FRL** *, cumple con lo especificado :*

Apliquemos el primer tramo de la definición ($\triangleright\ D_d\ 0_d$ *)*

$$x, y, Y \xrightarrow{\triangleright\ D_d\ 0_d} y, Y, x, x, 0$$

Si $y = 0$ *, como los extremos de la lista son iguales no se aplica la parte que esta en el operador* $\{\ \}$ *y se pasa directamente al tercer tramo de la formula:*

$$y, Y, x, x, 0 \xrightarrow{\square_d \leftrightarrow \triangleleft} x, x, y, Y$$

Que es lo buscado, ya que

$$x + 0 = x \quad resulta \quad x + 0, x, 0, Y$$

Si $y \neq 0$ *entonces debemos aplicar la parte que esta entre* $\{\}$ *hasta que los extremos de la lista sean iguales, la primera aplicación produce el siguiente resultado:*

$$y, Y, x, x, 0 \xrightarrow{\triangleleft\ S_d\ \triangleright\ S_d} y, Y, x, x + 1, 1$$

Cesa de aplicarse cuando los extremos de la lista son iguales

$$y, Y, x, x + y, y$$

En ese caso se aplica el tercer tramo de la formula :

$$y, Y, x, x + y, y \xrightarrow{\square_d \leftrightarrow \triangleleft} x + y, x, y, Y$$

Por lo tanto $\Sigma \in$ **FRL**

Veamos que la función Pd (Predecesor) se puede construir de la siguiente manera:

$$Pd = \triangleright\ 0_i\ 0_i \left\{\ \triangleright \leftrightarrow\ D_i\ \leftrightarrow\ \square_i\ S_i\ \leftrightarrow \triangleleft\ \right\} \square_i \leftrightarrow\ \triangleleft$$

Aplicando el primer tramo de la función:

$$x, Y \xrightarrow{\triangleright\ 0_i\ 0_i} 0, 0, Y, x$$

Si $x = 0$ *como los extremos la lista son iguales no se aplica la parte que está en el operador* $\{\ \}$ *y se pasa directamente al tercer tramo de la formula:*

$$0, 0, Y, x \xrightarrow{\square_i \leftrightarrow\ \triangleleft} 0, x, Y$$

Se cumple que $Pd(0, Y) = 0, 0, Y$

Si $x \neq 0$ *entonces debemos aplicar la parte que está entre* $\{\}$ *hasta que los extremos de la lista sean iguales.*

La primera aplicación produce el siguiente resultado:

$$0,0,Y,x \xrightarrow{\triangleright \leftrightarrow\ D_i\ \leftrightarrow\ \square_i\, S_i\ \leftrightarrow \triangleleft} 1,0,Y,x$$

Se puede verificar que en cada paso el resultado es .

$$t, t-1, Y, x$$

Cuando los extremos son iguales ($t = x$) se aplica el tercer tramo de la función y obtenemos el resultado buscado.

$$Pd(0,Y) = 0,0,Y \quad ; \quad Pd(x+1,Y) = x, x+1, Y$$

Se dejan las demostraciones para las funciones iii), iv), v) como ejercicio.

1.4. Algunas propiedades

Mostraremos a continuación una serie de propiedades que serán útiles para encontrar nuevos miembros de la familia **FRL**.

Definición 1.4.1. *Dada una función numérica f con dominio contenido en una potencia cartesiana de $\mathbb{N}_0$ y cuyo codominio sea $\mathbb{N}_0$,*

$$f : X^k \ \to \mathbb{N}_0 \quad k \in \mathbb{N}_0.$$

$$\mathcal{D}(f) = X^k \sqsubseteq \mathbb{N}_0^k$$

*Diremos que la función f tiene **asociada** una función $F_f \in$ **FRL** si se cumple*

$$F_f(X^k, Y) = f(X^k), Y \quad \forall \quad X^k \in \mathcal{D}(f)$$

Definición 1.4.2. *Dada una función f*

$$f : \mathbb{N}_0 \longrightarrow \mathbb{N}_0$$

*Llamamos **potencia** n a la función que se obtiene aplicando n veces la función f (con la convención que aplicada 0 veces es la identidad). Esto nos permite presentar la potencia como una función numérica de dos variables $F^{(2)}$, llamada **potencia** de f, del siguiente modo:*

$$F(0,x) = x \quad \forall x \in \mathbb{N}_0$$

$$F(y+1,x) = f(F(y,x)) \quad \forall x \in \mathbb{N}_0$$

Notamos $F(y,x)$ como $f^y(x)$

Proposición 1.4.1. *Si una función $f : \mathbb{N}_0 \to \mathbb{N}_0$ tiene asociada una función $F_f \in$ **FRL** entonces la función potencia $F(y,x) = f^y(x)$ tiene asociada una función en* **FRL**

$$Si \;\; \exists \; F_f \in \mathbf{FRL} \;\; tal \;\; que \;\; F_f(x,Y) = f(x),Y$$

$$Entonces \; \exists \; G \in \mathbf{FRL} \;\; tal \;\; que \;\; G(y,x,Y) = f^y(x),Y$$

Demostración. Defino:

$$G = \; \rhd \; 0_i \left\{ \; \rhd \; F_f \; \lhd \; S_i \right\} \square_i \; \square_d$$

Se verifica que aplicando la primera parte de la función:

$$y,x,Y \xrightarrow{\rhd \; 0_i} \; 0,x,Y,y$$

Si $y = 0$ no se aplica la parte que está definida en el operador $\{ \; \}$ y se pasa a la parte donde se borran los extremos

$$G(0,x,Y) \; = \; x,Y = \; f^0(x),Y$$

Si y es distinto de cero, se aplica lo que está en el operador $\{ \; \}$ hasta que se cumpla la condición de igualdad de los extremos.

$$0,x,Y,y \xrightarrow{\rhd \; F_f \; \lhd \; S_i} \; 1,f(x),Y,y$$

Se repite hasta

$$t-1,f^{t-1}(x),Y,y \xrightarrow{\rhd \; F_f \; \lhd \; S_i} \; t,f^t(x),Y,y$$

Cuando

$$y = t \quad G(y,x,Y) = f^y(x),Y$$

□

Ejemplo 1.4.1. *Definiremos una función numérica $\hat{d}^{(2)}$ del siguiente modo:*

$$\hat{d}^{(2)} : \mathbb{N}_0^{(2)} \longrightarrow \mathbb{N}_0$$

$$\hat{d}^{(2)}(y,x) \overset{def}{=} \begin{cases} x-y & \text{si } x \geq y \\ 0 & \text{si x < y} \end{cases}$$

Si $x \geq y$ se resta, en otro caso el resultado es 0 .

(Se suele notar a $\hat{d}^{(2)}(y,x)$ como $x \dot{-} y$)

Se puede ver que la función $\hat{d}^{(2)}$ tiene asociada en **FRL** *la potencia de la función Pd_1*

$$Pd_1(x) \overset{def}{=} \begin{cases} 0 & \text{si } x = 0 \\ x-1 & \text{si x > 0} \end{cases}$$

Es inmediato que esta función tiene asociada una función en **FRL** *por lo visto en ii) de 1.3.1*

$$\hat{d}(y,x) = Pd^y(x)$$

Lo mismo se puede utilizar para mostrar que se puede construir una función Suma

$$Suma(y, x, Y) \stackrel{def}{=} y + x, Y$$

Utilizando la potencia de la función base S_i

$$Suma \;\; = \;\; {S_i}^y$$

Ejemplo 1.4.2. *Definiremos una función de lista* $D_{\boldsymbol{0}}$ *del siguiente modo:*

$$D_{\boldsymbol{0}}(y, X) \stackrel{def}{=} \begin{cases} 1, y, X & \text{si } y = 0 \\ 0, y, X & \text{si y} > 0 \end{cases}$$

Se trata de una función que distingue cuando el primer elemento de una lista es el 0.

Para mostrar que $D_{\boldsymbol{0}} \in$ **FRL** *la construimos de la manera siguiente:*

$$D_{\boldsymbol{0}} \; = D_i \;\rhd\; 0_i \; S_i \; 0_i \left\{ \;\rhd\; \square_i \; 0_i; S_d \;\lhd \right\} \square_i \;\; \square_d$$

Se verifica que aplicando la primera parte de la función:

$$x, Y \; \frac{D_i \;\rhd\; 0_i \; S_i \; 0_i}{} \;\; 0, 1, x, Y, x$$

Si $x = 0$ *no se aplica la parte que está definida en el operador* $\{\;\}$ *y se pasa a la parte donde se borran los extremos*

$$D_{\boldsymbol{0}} \; (0, Y) \;\; = \;\; 1, 0, Y$$

Si $x \neq 0$ *, se aplica lo que está en el operador hasta que se cumpla la condición de igualdad de los extremos.*

$$0, x, Y, x \;\; \frac{\rhd \;\square_i \; 0_i; S_d \;\lhd}{} \;\; 1, 0, x, Y, x$$

Se repite hasta

$$t - 1, 0, x, Y, x \;\; \frac{\rhd \;\square_i \; 0_i; S_d \;\lhd}{} \;\; t, 0, x, Y, x$$

Cuando $x = t$ *se borran los extremos y resulta Cuando*

$$D_{\boldsymbol{0}} \; (x, Y) \;\; = \;\; 0, x, Y \;\; x \neq 0$$

Observación : Aplicamos, a las funciones, el criterio extensional, considerando que dos funciones, F, G son equivalentes si tienen el mismo dominio y para todo elemento X que pertenezca a dicho dominio es $F(X) = G(X)$, **aunque utilicemos distintos procedimientos para calcularlas.**

Si una función pertenece a **FRL** todas las combinaciones de funciones base que la construyan son equivalentes independientemente del costo de su cálculo medido con algún criterio establecido de eficiencia.

Por ejemplo, en el caso de querer calcular $D_{\boldsymbol{0}}(1947, X)$ con la fórmula propuesta en 1.4.2 se deben realizar 1947 comparaciones para determinar el resultado... [2]

[2]Este trabajo se enfoca en la computabilidad (en "qué"se puede calcular). y no en el muy importante tema sobre establecer una jerarquía contemplando la eficiencia entre distintas funciones equivalentes, (el cómo")

1.5. Conjuntos recursivos

Definición 1.5.1. *Dado un subconjunto A de $\mathbb{N}_0^k$ definimos su* ***función característica de lista*** *de la siguiente manera*

$$\chi_A :\rightarrow \{0,1\}$$
$$\chi_A(X^k, Y) = \begin{cases} 1, X, Y & \text{si } X^k \in A \\ 0, X, Y & \text{si } X^k \notin A \end{cases}$$

Esto nos permite establecer una clasificación en los subconjuntos de $\mathbb{N}_0^k$

Definición 1.5.2.
Diremos que un subconjunto A de $\mathbb{N}_0^k$ es un ***conjunto recursivo*** *(notaremos* **CR***) si su función característica $\chi_A \in$* **FRL**

Proposición 1.5.1.
Sea $k \in \mathbb{N}$, y sean A y B dos subconjuntos de $\mathbb{N}_0^k$. Si A, B son **CR** *, el complemento $\neg A$, la intersección ($A \cap B$) y la unión ($A \cup B$) son* **CR**.

Demostración:
Como A, B son **CR** sus funciones características χ_A , $\chi_B \in$ **FRL**

(a)
$$\chi_{\neg A} = \chi_A \; D_{\mathbf{0}} \; \triangleright \square_i \; \triangleleft$$

(b) Utilizamos la función Π (Producto) que definimos en iii) de 1.3.1

$$\chi_{A \cap B} = \chi_A \; \triangleright \; \chi_B \; \triangleleft \; \Pi \; \triangleright \; \square_i \, \square_i \; \triangleleft$$

En a) y b) las funciones características se construyen a partir de **FRL**, por lo tanto sus conjuntos asociados son **CR**.

(c) $\chi_{A \cup B}$
Recordemos que $A \cup B = \neg[(\neg A) \cap (\neg B)]$ (De Morgan)
Aplicamos, entonces, lo visto en a) y b).

1.6. Relaciones recursivas

Las funciones constituyen un caso particular de las llamadas relaciones entre conjuntos.

Ante una expresión como "$a+b \leq c$" se puede, para cada terna $< x_1, x_2, x_3 >$ de $\mathbb{N}_0^3$ verificar si sus elementos cumplen o no con la expresión.

Se trata en este caso de una relación sobre $\mathbb{N}_0^3$.

En el mismo sentido una afirmación como "x es primo " es una relación definida sobre $\mathbb{N}_0^1$

Cada relación sobre $\mathbb{N}_0^k$ queda completamente determinada por el conjunto de las k-uplas que verifican la relación. (decimos que la relación es **verdadera** en esas k-uplas)

Es entonces que dada una relación α definida sobre $\mathbb{N}_0^k$ podemos asociar a la misma el subconjunto de $\mathbb{N}_0^k$, de los valores en que la relación es verdadera. Notaremos $D_\alpha \subset \mathbb{N}_0^k$ a dicho conjunto.

Así por ejemplo dada la relación $x = y$ el subconjunto asociado $D_=$ es la diagonal de $\mathbb{N}_0^2$

A partir de la asociación de una *relación* con el conjunto de los valores donde se cumple definimos:

Definición 1.6.1.
Dada una relación α definida en $\mathbb{N}_0^k$ diremos que es ***Relación Recursiva*** *(notaremos* **RR***) si su conjunto asociado D_α es* **CR**

Ejemplo 1.6.1.
Mostraremos que la relación '=' es recursiva
Llamaremos $D_=$ al conjunto asociado a esta relación.

$$D_= = \{(x, x) \;\; x \in \mathbb{N}_0\}$$

.

Recordemos que en base a lo visto en 1.4.1 la función numérica $R(x, y) = x \dot{-} y$ tiene asociada ***FRL*** *una función F_R tal que*

$$F_R(y, x, Z) \stackrel{def}{=} \begin{cases} x - y, y, x, Z & \text{si } x \geq y \\ 0, y, x, Z & \text{si x < y} \end{cases}$$

Observemos que la función

$$G_= F_R \; \rhd \; \rhd \leftrightarrow \lhd \; F_R \; \lhd \; \Sigma \; D_{\boldsymbol{0}} \; \rhd \; \square_i \, \square_i \, \square_i \; \rhd \leftrightarrow \lhd \lhd$$

$$(\; Si \;\; (x \dot{-} y + y \dot{-} x) \;\; = \; 0 \;\; \Longrightarrow x \; = \; y)$$

Esta es la función característica del conjunto $D_=$ y es inmediato que esta función es **FRL**, *por lo que el conjunto $D_=$ es* **CR** *y por lo tanto la relación '=' es* **RR**.

Definición 1.6.2. *Sean α, β relaciones definidas sobre $\mathbb{N}_0^k$*
Definimos :

I) $\neg\alpha(X)$ *la relación que es verdadera $\Leftrightarrow$ $\alpha(X)$ es falsa*

II) $(\alpha \wedge \beta)(X)$ *la relación que es verdadera $\Leftrightarrow$ $\alpha(X)$ y $\beta(X)$ son ambas verdaderas*

III) $(\alpha \vee \beta)(X)$ *la relación que es verdadera $\Leftrightarrow$ $\alpha(X)$ o $\beta(X)$ son verdaderas*

Proposición 1.6.1.
Si α, β son **RR**, $\Longrightarrow \;\; \neg\alpha, \; \alpha \wedge \beta, \; \alpha \vee \beta \; \in$ **RR**
La demostración es inmediata a partir de la proposición 1.5.1 .

Se puede avanzar con la definición de otros conceptos afines a los desarrollados en el estudio de las funciones recursivas.

En apariencia, se nota que las **FRL** tienen gran capacidad para modelizar cálculos.

En el Capítulo siguiente mostraremos el alcance de dicho poder

Capítulo 2

Poder de cálculo de las FRL

2.1. Las FRL y las Funciones Recursivas

Mostraremos que las **FRL** son, al menos, tan poderosas como las **Funciones Recursivas** (que notaremos **FR**) para representar procedimientos de cálculo.

Para ello nos basaremos en la definición inductiva del conjunto de **FR**, (ver el Capítulo 4) que se construían de la siguiente manera:

a) Las llamadas **funciones base** definidas en 4.1.1 (Ceros, Proyecciones, sucesor) , son **FR**.

b) Las funciones obtenidas a partir de **FR** aplicándoles un número finito de veces los operadores Composición, Recursión y Minimizador (Φ , R y M) (ver 4.1.2, 4.1.3 , 4.1.4) son **FR**.

2.1.1. Las FRL "representan" a las FR

A partir de las definiciones de de las **FR** mostraremos en forma inductiva el siguiente teorema:

Teorema 2.1.1.

Para toda función $g^k \in$ **FR** *existe una función en* **FRL** *, que notaremos* F_g *tal que:*
Dada la lista $X^k \in \mathcal{D}_g$ *y cualquier lista* $\boldsymbol{Y}$

$$F_g < X^k, Y > \longmapsto g(X^k),\ X^k,\ Y \qquad X^k, Y \in \mathcal{D}_{F_g} \iff X^k \in \mathcal{D}_g$$

$$X^k,\ Y \overset{F_g}{\longmapsto} g(X^k),\ X^k,\ Y$$

Diremos que F_g ***"representa"*** *a la función* g

Demostraremos este teorema a partir de los lemas siguientes:

Lema 2.1.1. ***Representación de las funciones base***

Las funciones base de las **FR** *:* ***funciones cero*** c^k*,* ***proyecciones*** $p_i^{(n)}$ *y* ***sucesor*** *son representadas en* **FRL**

1. *Funciones cero*

   Las funciones cero c^k *son, para cualquier k, representadas por la función* 0_i *En efecto:*

$$X,\ Y \overset{F_c}{\longmapsto} 0,\ X,\ Y$$

2. *Las proyecciones* $p_i^{(n)}$*, son representadas por las funciones* F_{p_i} *que se construyen de la siguiente manera :*

$$F_{p_i} = \underbrace{\rhd \rhd \dots}_{i\ -\ 1\ veces}\ D_i\ \underbrace{\leftrightarrow \lhd \leftrightarrow \lhd \dots}_{i\ -\ 1\ veces}$$

   Observar que estos representantes para las proyecciones $p_i^{(n)}$ *sólo dependen del valor de i .*

3. *El sucesor es representado por* $F_s = D_i\ S_i$

Se deja al lector la sencilla verificación de que las funciones presentadas cumplen con lo solicitado.

Luego todas las funciones base de **FR** *se pueden representar con* **FRL**

Lema 2.1.2. ***Composición***

La función que se construye por composición de funciones que tienen representación en **FRL***, tiene representación en* **FRL**

Sea la función numérica $f^{(n)}$ *y una familia de n funciones numéricas de orden k ,* $\{g_i^{(k)}\}_{i=1}^n$ *tales que tienen representación en* **FRL** *:* $F_f, F_{g_1}, F_{g_2}, ..., F_{g_n}$ *entonces*

$$h = \Phi\left(f^{(n)}, g_1^{(k)}, g_2^{(k)}, ..., g_n^{(k)}\right)$$

tiene representación en **FRL**

Se puede ver que la F_h *buscada es:*

$$F_h = F_{g_1} \rhd\ F_{g_2} \rhd\ \dots\ F_{g_n} \underbrace{\lhd \lhd \dots}_{n\ -\ 1\ veces}\ F_f \rhd\ \underbrace{\square_i\ \square_i \dots}_{n\ veces} \lhd$$

En efecto:

$$X, Y\ \ \xrightarrow{F_{g_1} \rhd\ F_{g_2} \rhd\ \dots\ F_{g_n} \lhd\lhd \dots;}\ \ g_1(X),\ g_2(X), \dots g_n(X), X, Y$$

$$g_1(X), g_2(X), \dots g_n(X), X, Y \xrightarrow{F_f \rhd\ \square_i\ \square_i \dots \lhd}\ \ f(X), X, Y$$

Lema 2.1.3. ***Recursión***

La función construida por recursión de funciones numéricas que tienen representación en las **FRL** *tiene representación en las* **FRL**

Sean las funciones numéricas $g^{(k)}$ y $h^{(k+2)}$ tales que existen F_g y F_h que las representan en las **FRL** *entonces $f^{(k+1)} = R(g^{(k)}, h^{(k+2)})$ es representable en* **FRL**

La función buscada es :

$$F_f \ = \rhd F_g \ 0_i \{ \quad \rhd \leftrightarrow \lhd F_h \rhd \ \square_i \ S_i \ \leftrightarrow \lhd \quad \} \ \square_i \ \leftrightarrow \lhd$$

Veamos los pasos:

$$y, X, Y \quad \xrightarrow{\rhd F_g \ 0_i} \quad 0, \ g(X), \ X, \ Y, \ y$$

Si $y = 0$ entonces, por ser iguales los extremos no se aplica la función entre $\{ \ \}$ y directamente se pasa a la parte que sigue:

$$0, \ g(X), \ X, \ Y, \ 0 \ \xrightarrow{\square_i \ \leftrightarrow \lhd} \ g(X), \ 0, \ X, \ Y$$

Si no son iguales los extremos se aplica la función del operador $\{ \ \}$

$$t, f(t,X), \ X, \ Y, \ y \ \xrightarrow{\rhd \leftrightarrow \lhd F_h \ \rhd \ \square_i \ S_i \ \leftrightarrow \lhd} \ t{+}1, \ f(t{+}1), \ X, \ Y, \ y$$

Esto se repite hasta que los extremos sean iguales:

$$y, f(y,X), \ X, \ Y, \ y \ \xrightarrow{\square_i \ \leftrightarrow \lhd} \ f(y,X), \ y, \ X, \ Y$$

Lema 2.1.4. ***Minimización***

La función numérica que se obtiene por minimización de una función representable en **FRL** *tiene representación en* **FRL**.

Sea $g^{(n+1)}$ que tiene representación F_g en ***FRL****, entonces $f^{(n)} = M[g]$ es representable.*
Se puede verificar que:

$$F_f \ = \ 0_i \ F_g \ 0_d \{ \ \square_i \ S_i \ F_g \ \} \square_i \ \square_d$$

$$X, \ Y \xrightarrow{0_i \ F_g \ 0_d} \ g(0,X), \ 0, \ X, \ Y, \ 0$$

Si $g(0,X) = 0$ como los extremos son iguales , no se aplica la función que figura entre $\{ \ \}$ y queda:

$$\underbrace{g(t,X)}_{\equiv\ 0},\ 0,\ X,\ Y,\ 0\ \xrightarrow{\square_i\ \square_d}\ 0\ ,\ X,\ Y\ =\ f(X),X,\ Y$$

Si no son iguales los extremos se repite la función entre $\{\ \}$ *hasta que esto suceda:*

$$\underbrace{g(t,X)}_{\neq\ 0},\ t,\ X,\ Y,\ 0\ \xrightarrow{\square_i\ S_i\ F_g}\ g(t+1)\ ,t+1,\ X,\ Y,0$$

Se repite hasta que $g(k,X)$ *sea igual a* 0 *(Si eso no sucede* $X \notin\ \mathcal{D}_{F_f}$*)*

$$\underbrace{g(k,X)}_{\equiv\ 0},\ k,\ X,\ Y,\ 0\ \xrightarrow{\square_i\ \square_d}\ k\ ,\ X,\ Y\ =\ f(X),X,\ Y$$

Como por definición las **FR** *se construyen a partir de las funciones base aplicando los operadores* Φ *,* R *y* M *, de los lemas vistos se concluye que toda* **FR** *tiene una representación en las* **FRL**

Esto implica que, como modelo de cálculo las **FRL** son entonces, al menos, tan potentes como las **FR**.

Observemos que además esta representación tiene la característica de conservar los datos.

Usando el símbolo $\sqsubseteq$ para establecer una jerarquía respecto a la capacidad de modelizar el cálculo entre los dos modelos se puede poner

$$\textbf{FR} \sqsubseteq\ \textbf{FRL}$$

Surge entonces la posibilidad que este modelo de cálculo sea incluso más poderoso que el de las **FR**.

En el próximo punto veremos que ambos modelos son equivalentes y, en consecuencia sigue vigente la llamada Tesis de Church

2.2. Las FRL y las Máquinas de Turing

Mostraremos que el poder de cálculo de las **Máquinas de Turing** (que notaremos **MT**)[1] es, al menos, tan poderoso, como el de las **FRL**.

Veremos que para toda una función recursiva de lista F que pertenece a **FRL** existe una máquina de Turing M_F que realiza el mismo cálculo.

Sin pérdida de generalidad utilizaremos un alfabeto de dos símbolos: el alfabeto $A = \{\square, \bullet\}$

[1] Una vision general de las **MT** se puede consultar en el Apendice del Capítulo 4

2.2.1. Representación de las FRL por las MT

Para comenzar debemos encontrar una manera de representar el dominio de las **FRL**, las listas de números, en la cinta de la máquina.

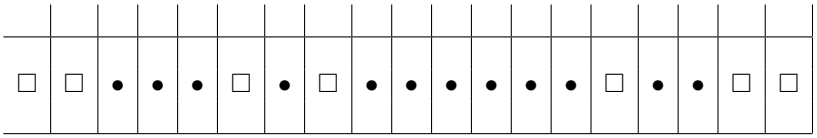
La manera más simple es emplear un sistema unario, asociando al número n la cantidad de $n+1$ $\bullet$.

Para representar una lista de números usaremos grupos de $\bullet$ separados por un blanco.

Por ejemplo, la lista

$$< 2, 0, 5, 1 >$$

se escribe en la cinta de la máquina del siguiente modo:



Notaremos la representación de una lista X de elementos pertenecientes a $\mathbb{N}_0$ en la cinta de una **MT** con $<< X >>$.

Diremos que dada una F que pertenece a **FRL** es **representada** por una Máquina s de Turing M_F si dada cualquier lista X que pertenezca al dominio de F, la M_F tomando como configuración inicial $<< X >>$, con la máquina observando la primera casilla vacía antes del primer elemento de la lista(si es la lista vacía, en cualquier lugar), luego de procesarla llega a la configuración final

$$<< F(X) >>$$

y posicionada en la primera casilla libre anterior a

$$<< F(X) >>$$

.

Teorema 2.2.1.

Dada una F que pertenece a las **FRL** *existe siempre una Máquina de Turing: M_F que realiza el mismo cálculo.*

Para demostrarlo usaremos un camino similar al que utilizamos para mostrar que toda **FR** *era representada por una función recursiva de lista apelando esta vez a la construcción recursiva del conjunto* **FRL**

Definiremos algunas **MT** *elementales que nos facilitarán la demostración.*

Definiciones 2.2.1. *1. Máquina "Mueve a derecha" (la notaremos: **d**)*

	q_1
$\square$	$\square, d, q_0$
$\bullet$	$\bullet, d, q_0$

*2. Máquina "Mueve a izquierda" (la notaremos: **i**)*

	q_1
$\square$	$\square, i, q_0$
$\bullet$	$\bullet, i, q_0$

3. Máquina "Blanco" (la notaremos: $\square$)

	q_1
$\square$	$\square, n, q_0$
$\bullet$	$\square, n, q_0$

4. Máquina "Punto" (la notaremos: $\bullet$)

	q_1
$\square$	$\bullet, n, q_0$
$\bullet$	$\bullet, n, q_0$

5. *Máquina "Nada" (la notaremos:* $\mathbf{n}$*)*

	q_1
$\square$	$\square, n, q_0$
$\bullet$	$\bullet, n, q_0$

Las dos primeras se mueven a la derecha o a la izquierda sin modificar el contenido de la celda observada. Las otras dos sustituyen el contenido de la celda observada incondicionalmente por el símbolo de su nombre. Finalmente la última, como su nombre lo indica, no hace nada.

6. *Máquina "Busca primer Blanco a Derecha" (la notaremos* $D^{\square}$ *)*
Esta máquina que avanza por la cinta hacia la derecha hasta que encuentra el primer $\square$
Se puede componer de la siguiente manera:

$$D^{\square} \stackrel{def}{=} \bullet \circlearrowright d \xrightarrow{\square} n$$

7. *Máquina "Busca primer* $\bullet$ *a derecha" (la notaremos* $D^{\bullet}$*)*
Se desplaza hacia la derecha hasta encontrar el primer $\bullet$

8. *"Busca primer Blanco a izquierda"(la notaremos* $I^{\square}$*)*
Se desplaza hacia la izquierda hasta encontrar el primer blanco

9. *"Busca primer* $\bullet$ *a izquierda"(la notaremos* $I^{\bullet}$*)*
Se desplaza hacia la izquierda hasta encontrar el primer $\bullet$

10. *Máquina "Busca primeros dos blancos a la derecha" (la notaremos* $D^{\square\,\square}$*)*
Esta máquina avanza hacia la derecha hasta que encuentra dos casillas vacías se construye de la siguiente manera :

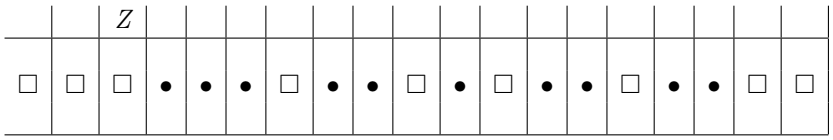
$$D^{\square\,\square} \stackrel{def}{=} D^{\square} \xrightarrow[\square]{n} d \xrightarrow[\bullet]{} \;\cdot$$

11. *En forma similar podemos definir la máquina* $I^{\square\,\square}$ *(que busca los dos primeros blancos que aparecen a la izquierda , y las máquinas* $D^{\bullet\,\bullet}$ *,* $I^{\bullet\,\bullet}$*, que hacen un trabajo similar pero buscando los primeros dos* $\bullet$ *que utilizaremos en nuestra demostración.*

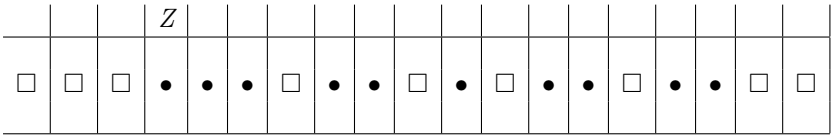
Definición 2.2.1. *La Máquina* ***Z***

La máquina ***Z*** *es una* **MT** *que realiza la siguiente tarea: Como dato de entrada se le debe presentar un conjunto de grupos de* • *separados por un solo* □*, con todo el resto de la cinta en blanco. (o sea el modo que representamos las listas)*
La máquina ***Z*** *parte en la primera celda vacía que está a la izquierda del primer grupo.*
La configuración final de la cinta es la misma que la inicial, pero la celda observada difiere de acuerdo a lo siguiente:
Si el primer grupo de • *tiene el mismo número de elementos que el último grupo entonces la casilla en que se para la máquina es la misma en la que partió (queda en la primera vacía a la izquierda). Si los grupos primero y último no tienen el mismo número de elementos, entonces la casilla en la que se detiene la máquina será la primera del primer grupo (es decir la del primer* • *)*

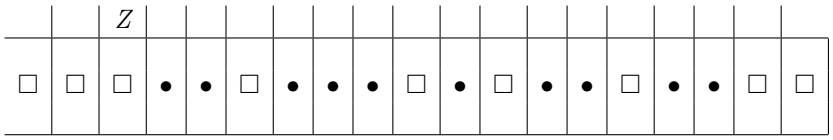
Ejemplo 2.2.1. *Cinta inicial*



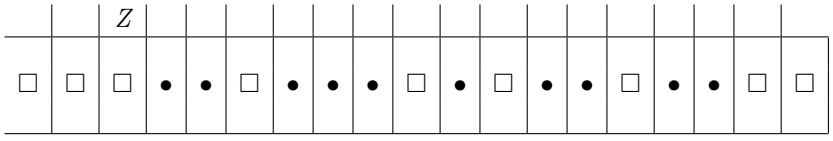
Cinta final



Ejemplo 2.2.2. *Cinta inicial*



Cinta final



Es un muy interesante ejercicio la construcción de esta máquina. Se recomienda fuertemente tratar de construir su diagrama. (En el Apéndice en 4.4 se encuentra un camino posible para dicha construcción).

Lema 2.2.1. ***Funciones Base***

Las funciones base $0_i,\ 0_d,\ \square_i,\ \square_d,\ S_i,\ S_d$ *de* **FRL** *son representadas por* **MT**

En efecto veamos los diagramas de las máquinas que representan a cada una de ellas:

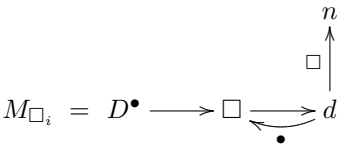
a) Cero a izquierda

$$M_{0_i} = i \longrightarrow \bullet \longrightarrow i$$

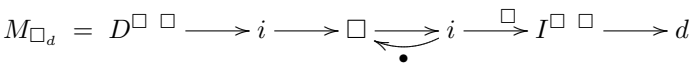
b) Cero a derecha

$$M_{0_d} = D^{\square\,\square} \longrightarrow \bullet \longrightarrow I^{\square\,\square} \longrightarrow d$$

c) Blanco a izquierda



d) Blanco a derecha



e) Sucesor a izquierda

$$M_{s_i} = D^{\bullet} \longrightarrow i \longrightarrow \bullet \longrightarrow i$$

f) Sucesor a derecha

$$M_{s_d} = D^{\square\,\square} \longrightarrow I^{\bullet} \longrightarrow d \longrightarrow \bullet \longrightarrow I^{\square\,\square} \longrightarrow d$$

Lema 2.2.2. ***Composición***

Dadas dos funciones de lista F *y* G *que son representadas por la Máquinas de Turing ,* M_F *y* M_G *existe una Máquina de Turing* M_{FG} *que representa a la composición* FG*.*

Es inmediato, M_{FG} *buscada es directamente la que se construye aplicando* M_F *y luego aplicando* M_G

$$M_{FG} = M_F \longrightarrow M_G$$

Lema 2.2.3. ***Operador Repetición***

Dada una función de lista F que es representada por una Máquina de Turing M_F existe una Máquina de Turing que representa a la función $\{F\}$

Se demuestra empleando la máquina Z que vimos en 2.2.1

$$M_{\{F\}} = \quad Z \xrightarrow{\bullet} i \quad \begin{array}{l} \uparrow \\ \downarrow \end{array} \square \quad M_F$$

Esta máquina aplica M_F mientras los extremos de la lista son distintos.

Luego, por el carácter recursivo de construcción de las **FRL**, *hemos mostrado que cualquiera sea la función* ***F*** *en* **FRL** *siempre existe una M_F en* **MT***que la representa.*

2.3. Vigencia de la Tesis de Turing

Utilizando el símbolo de jerarquía respecto a la capacidad de modelizar el cálculo que vimos en 2.1.1 se concluye que podemos poner :

$$\mathbf{FRL} \sqsubseteq \mathbf{MT}$$

Pero sabemos que las **MT** pueden ser representadas por las **FR** (Ver 4.3)

$$\mathbf{MT} \sqsubseteq \mathbf{FR}$$

Por lo tanto, a partir de lo visto:

$$\mathbf{FR} \sqsubseteq \mathbf{FRL} \sqsubseteq \mathbf{MT} \sqsubseteq \mathbf{FR}$$

Por lo que queda demostrado que los tres modelos son equivalentes y se mantiene la tesis de Church.

Capítulo 3

Godelización de las FRL

Mostraremos que podemos asociar a cada función de **FRL** un número $t \in \mathbb{N}_0$ en forma unívoca de modo tal que, conociendo dicho número, se pueda reconstruir la función a que está asociado.

Notaremos F_t la función de lista a la que le corresponda el numero t.

También usaremos la notación $[F]$ para referirnos al numero asociado a la función **F**.

La herramienta principal que utilizaremos para establecer esta función entre las funciones de lista y $\mathbb{N}_0$ será una fumción que a cada par de números asocia un número en forma tal que permita reconstruir los elementos del par.

Se tratara de una de las llamadas funciones Dupla que definiremos a continuación.

Definición 3.0.1. *Función Dupla*

Llamaremos ***Función Dupla****, una función numérica* $W^{(2)}$*,*

$$W^{(2)} : \mathbb{N}_0^{(2)} \longrightarrow \mathbb{N}_0$$

que tenga las siguientes propiedades:

I) $W(x,y) = W(m,n) \implies x = m \ , \ y = n$ *(unicidad)*

II) $x > m \implies W(x,y) > W(m,y)$ *(creciente respecto a la primera componente)*

III) $y > n \implies W(x,y) > W(x,n)$ *(creciente respecto a la segunda componente)*

A partir de estas propiedades se puede observar que si W es función dupla entonces

$$W(x,y) \geq x \quad W(x,y) \geq y$$

Por la propiedad i) vemos que podemos definir las inversas $U^{(1)}$ y $V^{(1)}$ tales que:

$$z = W(x,y) \implies U(z) = x \quad , \quad V(z) = y$$

y se cumple

1.
$$W(U(z), V(z)) = z$$

2.
$$U(W(x, y)) = x$$

3.
$$V(W(x, y)) = y$$

Observemos que por satisfacer las condiciones ii) e iii) de la definición 3.0.1 las funciones U y V pueden definirse:

$$U(z) = \mu x(\bigcup_{y=0}^{z} z = W(x, y)) \quad , U(z) \leq z$$

$$V(z) = \mu y(\bigcup_{x=0}^{z} z = W(x, y)) \quad , V(z) \leq z$$

Se puede mostrar si W es una función recursiva, tanto U como V también lo son

$$W \in \mathbf{FR} \Longrightarrow \quad U,\, V \in \mathbf{FR}$$

Ejemplo 3.0.1. *En este trabajo usaremos la siguiente* ***función dupla*** *:*

$$W(x, y) = \left[\frac{(x + y)(x + y + 1)}{2}\right] + x$$

(donde el valor entre corchetes indica que se debe tomar la parte entera)
Es inmediato que es una ***Función Recursiva***[1]

3.1. Las FRL como números naturales

Veamos a continuación como realizaremos la asociación de un números a cada función recursiva de lista.

A las seis funciones base le asociaremos los números del 0 al 5 del siguiente modo

1. **Borrar a izquierda**

   $F_0 = \square_i$

2. **Borrar a derecha**

   $F_1 = \square_d$

3. **Cero a izquierda**

   $F_2 = 0_i$

[1] En este caso se puede mostrar que se puede construir W empleando solo los operadores Composición y Recursión, por lo que se trata de una **Función Recursiva Primitiva**, y también lo son sus inversas U y V

4. **Cero a derecha**
   $F_3 = 0_d$
5. **Sucesor a izquierda**
   $F_4 = S_i$
6. **Sucesor a derecha**
   $F_5 = S_d$

Definición 3.1.1. *Composición*
Sean las funciones de recursivas de lista F_p y F_q (que tienen a p y q como sus números asociados)
Asociaremos a la composición de dichas funciones F_pF_q el numero t construido de la siguiente manera

$$t = 6 + 2 \times W(p, q)$$

Definición 3.1.2. *Repetición*

Sean una función de lista F_p
A la función $\{F_p\}$ le haremos corresponder el número t construido de la siguiente manera

$$t = 7 + 2p$$

Observemos que aplicando este mecanismo a cada $F \in$ **FRL** le podemos asociar un único numero $\in \mathbb{N}_0$

Ejemplo 3.1.1.
Sea la función

$$H = \ 0_i \ \{S_i\}$$

$$[0_i] = 0 \qquad [S_i] = 4$$

$$[\{S_i\}] \ = \ 7 + 2 \times 4 \ = \ 15$$

$$[H] = \ 6 + 2 \times \ W(0, 15)$$
$$[H] = 6 + 2 \times \ ((15 \times 16)/2) + 0) = \ 246$$

3.1.1. Casos

De este modo a cada $n \in \mathbb{N}_0$ se le puede asociar una función de lista de modo unívoco

Si el número es menor que seis le corresponde una de las seis funciones base

1. $F_0 < y, X >= X$

1 $F_1 < X, y >= X$

2 $F_2 < X >= 0, X$

3 $F_3 < X >= X, 0$

4 $F_4 < y, X >= y + 1, X$

5 $F_5 < X, y >= X, y + 1$

6 Si el número es mayor que cinco es **par** .

Lo podemos presentar entonces de la forma $6 + 2k$

Pero a partir de k podemos encontrar las dos variables

$p = U(k)$ y $q = V(k)$ tales que $k = W(p, q)$

En ese caso

$F_{6+2W(p,q)} < X >= F_p F_q < X >$

7 Si el numero mayor que cinco es **impar** se puede presentar como $7 + 2p$

$$F_{7+2p} < x, X, y >= \begin{cases} x, X, y & \text{si } x = y \\ \{F_p(F_{7+2p} < x, X, y) > \} & \text{si x} \neq \text{y} \end{cases}$$

Se puede observar que con este procedimiento se establece una biyección entre los $\mathbb{N}_0$ y las **FRL** ya que no solo identificamos cada proceso de cálculo con un único número sino que a cada número le podemos hacer corresponder unívocamente una función. (Si bien es cierto que algunas funciones así definidas tienen como dominio el conjunto vacío)

Ejemplo 3.1.2. *Sean las funciones :*

$$Inut_1 \;=\; 0_i\ 0_d\ S_i\ \{\ 0_i\}$$

$$Inut_2 \;=\; 0_i\ 0_d\ S_i\ \{\leftrightarrow\}$$

Estas funciones no están definidas para ningún valor[2]

[2]Notemos que la funciones tienen comportamientos distintos, mientras que $Inut_1$ crece en forma ininterrumpida hacia la izquierda, $Inut_2$, oscila permanentemente intercambiando extremos. Un tema interesante es el estudio de estos "objetos extraños" estableciendo algún tipo de jerarquía o clasificación

3.2. Función Universal

Definición 3.2.1. *Llamaremos* ***Función Universal****, (notaremos* ***U****) a una función de lista que dada la lista* t, X *devuelve la función* F_t *asociada al valor* t *aplicada al argumento* X *. En el caso de que* X *pertenezca al dominio de* F_t *el resultado es* $F_t < X >$*, quedando indefinida si no pertenece a dicho dominio*

$$t, X \xrightarrow{U} F_t < X > \quad si \quad X \in \mathcal{D}_{F_t}$$

$$t, X \in D(\boldsymbol{U}) \longleftrightarrow X \in \mathcal{D}_{F_t}$$

Teorema 3.2.1.

$$\boldsymbol{U} \in \mathbf{FRL}$$

Demostración

La demostración de este teorema es constructiva.

Veremos como hacemos para construir la función U *como una* ***FRL***

El procedimiento para calcular el valor que corresponde a la función de lista asociada a t*,* F_t *aplicada a la lista* X *lo podemos realizar del siguiente modo*

Comenzamos con la expresión $F_t < X >$

A partir del valor de t transformamos tal expresión de acuerdo a lo visto en los diferentes casos en 3.1.1

En los casos 0,1,2,3,4,5 y en la primera alternativa de 7 (los extremos de la lista son iguales) el cálculo termina.

Para el caso 6 y la segunda parte del caso 7 se nos crea una nueva expresión en la que figurarán dos nuevas funciones.

En ese caso se aplicará de nuevo el procedimiento .

Esto es en una fase genérica del procedimiento tendremos una secuencia de funciones en una expresión del tipo

$$F_t F_{t_1} F_{t_2} \dots F_{t_k} < X >$$

Que según el valor de t será sustituida por

1. $F_{t_1} F_{t_2} \dots F_{t_k} < Y > \quad si \ t = 0 \ \ y \quad X = x, Y$
2. $F_{t_1} F_{t_2} \dots F_{t_k} < Y > \quad si \ t = 1 \quad y \ X = Y, x$
3. $F_{t_1} F_{t_2} \dots F_{t_k} < 0, X > \quad si \ t = 2$
4. $F_{t_1} F_{t_2} \dots F_{t_k} < X, 0 > \quad si \ t = 3$
5. $F_{t_1} F_{t_2} \dots F_{t_k} < x + 1, Y > \quad si \ t = 4 \ \ y \ X = x, Y$
6. $F_{t_1} F_{t_2} \dots F_{t_k} < Y, x + 1 > \quad si \ t = 5 \ \ y \ X = Y, x$
7. $F_{U(t-6)/2} F_{V(t-6)/2} F_{t_1} F_{t_2} \dots F_{t_k} < X >$ *si* t *es par y mayor de 5*
8. $F_{t_1} F_{t_2} \dots F_{t_k} < X >$ *si* t *es impar, mayor de 5 y* $X = y, Z, x$ *con* $x = y$
9. $F_{(t-7)/2} F_t F_{t_1} F_{t_2} \dots F_{t_k} < X >$ *si* t *es impar, mayor de 5 y* $X = y, Z, x$ *con* $x \neq y$

Se repite el procedimiento tantas veces como sea necesario hasta llegar a la lista resultado del cálculo (observemos que es necesario que la lista inicial X pertenezca al dominio de la función de lista.

A la sucesión de funciones $F_t F_{t_1} F_{t_2} \dots F_{t_k}$

Le podemos asociar la lista

$$t, t_1, t_2, \dots t_k$$

Un problema que va a surgir es distinguir la lista que se va formando de los datos de la lista argumento X (reaparece, en forma inesperada, el viejo problema de distinguir datos de programa)

Lo resolveremos sumando 1 a cada uno de los t_i (lo que los hace a todos distintos de 0 y poniendo un 0, como marca al final de la lista).

En cualquier momento del cálculo tendríamos una lista

$$t+1, t_1+1, t_2+1, \dots t_k+1, 0, X$$

Como nuestra función solo toma en cuenta en cada aplicación el primer miembro de la lista, podemos presentar la misma de la forma

$$t+1, L, 0, X$$

donde $L = t_1+1, t_2+1, \dots t_k+1$ es una lista de números positivos. Para avanzar en la demostración usaremos el siguiente lema

Lema 3.2.1. *Existe una función recursiva de lista que llamaremos **Ev** que cumple lo siguiente*

1. $$1, L, 0, X \xrightarrow{Ev} L, 0, \square_i < X >$$
2. $$2, L, 0, X \xrightarrow{Ev} L, 0, \square_d < X >$$
3. $$3, L, 0, X \xrightarrow{Ev} L, 0, 0_i < X >$$
4. $$4, L, 0, X \xrightarrow{Ev} L, 0, 0_d < X >$$
5. $$5, L, 0, X \xrightarrow{Ev} L, 0, S_i < X >$$
6. $$6, L, 0, X \xrightarrow{Ev} L, 0, S_d < X >$$
7. *Caso en que t sea impar mayor que 5 . Lo notaremos : **I***

   $$\boldsymbol{I}, L, 0, X \xrightarrow{Ev} U((\boldsymbol{I}-7)/2)+1, V((\boldsymbol{I}-7)/2)+1, L, 0, < X >$$

8. *Caso en que t sea par mayor que 6 . Lo notaremos : **P***

   Existen dos posibilidades respecto al primero y último valor de la lista sobre la que estamos operando que presentamos en la forma

   $$y, X, z$$

8.i Si $y \neq z$

$$\boldsymbol{P}, L, 0, y, X, z \xrightarrow{Ev} (\boldsymbol{P} - 8)/2) + 1, \boldsymbol{P}, L, 0, y, X, z$$

8.ii Si $y = z$

$$\boldsymbol{P}, L, 0, y, X, z \xrightarrow{Ev} L, 0, y, X, z$$

Construiremos $\boldsymbol{Ev}$ *como la composición de funciones evaluadoras del valor inicial* $t+1$ *de la lista, que verifican con la función característica asociada a la relación que debe cumplir el valor de* $t+1$.

$$\boldsymbol{Ev} \;=\; \boldsymbol{Ev}'_1\ \boldsymbol{Ev}'_2\ \boldsymbol{Ev}'_3\ \boldsymbol{Ev}'_4\ \boldsymbol{Ev}'_5\ \boldsymbol{Ev}'_6\ \boldsymbol{Ev}'_I\ \boldsymbol{Ev}'_P\ \square_i$$

Cada una de las $\boldsymbol{Ev}_x$ *evalúa si cumple el valor de* $t+1$ *la relación que tiene asociada. La estructura general de cada uno de estos evaluadores es la siguiente:*

$$\boldsymbol{Ev}'_x \;=\; \boldsymbol{R}_x\ 0_d\ \{\square_i\ \square_d\ \boldsymbol{Ev}_x\ \ 0_i\ 0_i\ 0_d\}\ \square_d\ \square_i$$

Donde $\boldsymbol{R}_x$ *es la función característica de la relación asociada al evaluador* $\boldsymbol{Ev}_x$.

$$Para\ \ i \;=\; 1,2,3,4,5\ \ y\ \ 6\ \ al\ \ \boldsymbol{Ev}_i\ \ le\ \ \ corresponde\ \ \boldsymbol{R}_i \equiv\ (\ x \;=\; i\ )$$

$$Al\ \ evaluador\ \ \ \ \boldsymbol{Ev}_I\ \ le\ \ \ corresponde\ \ \boldsymbol{R}_I \equiv\ (\ x \geq 7\ \wedge\ Mod(x,2) \;=\; 1)$$

$$Al\ \ evaluador\ \ \ \boldsymbol{Ev}_P\ \ le\ \ \ corresponde\ \ \boldsymbol{R}_P \equiv\ (\ x \geq 7\ \wedge\ Mod(x,2) \;=\; 0)$$

Esta estructura hace que si el número a evaluar no cumple la relación, la lista queda como estaba y se pasa al evaluador siguiente. Si cumple con la condición, entonces realiza la acción que le corresponde y pasa al evaluador siguiente con un número inicial igual a 0 *por lo que no modifican la lista dato.*

Quedan por construir los $\boldsymbol{Ev}_x$ *que deben actuar en el caso que se cumpla la condición asociada*

Se recomienda intentar construir dichos evaluadores y compararlos con los que se presentan en este trabajo.

Veamos ejemplos de aplicación del $\boldsymbol{Ev}'_3$ *y del* $\boldsymbol{Ev}'_4$ *adelantando la definición de los respectivos evaluadores:*

$$\boldsymbol{Ev}_3 \;=\; \square_i\ 0_d\ 0_d\ \{\ \leftrightarrow\ \triangleright\ \}\square_d\ 0_i \quad 0_i\ \{\ \leftrightarrow\ \triangleleft\ \}\square_i\square_d$$

$$\boldsymbol{Ev}_4 \;=\; \square_i\ 0_d$$

Ejemplos 3.2.1. *1. Sea una lista cuyo primer elemento sea un* 3 .

$$\boldsymbol{Ev}\ (3,\ L,\ 0,\ X) \;=$$

$$\boldsymbol{Ev}'_1\ \boldsymbol{Ev}'_2\ \boldsymbol{Ev}'_3\ \boldsymbol{Ev}'_4\ \boldsymbol{Ev}'_5\ \boldsymbol{Ev}'_6\ \boldsymbol{Ev}'_I\ \boldsymbol{Ev}'_P\ \square_i\ (3,\ L,\ 0,\ X)$$

Apliquemos la primer función $\boldsymbol{Ev}'_1$

$$\boldsymbol{Ev}'_1\ (3,\ L,\ 0,\ X)\ =$$

$$\boldsymbol{R}_1\ \ 0_d\ \{\square_i\ \square_d\ \boldsymbol{Ev}_1\ \ 0_i\ 0_i\ 0_d\}\ \square_d\ \square_i\ (3,\ L,\ 0,\ X)$$

Aplicando el primer tramo de $\boldsymbol{Ev}'_1$*, la relación* $\boldsymbol{R}_1$*) aplicada al primer número,* 3*. devuelve falso, esto es* 0 *.*

$$\boldsymbol{R}_1\ 0_d\ (3,\ L,\ 0,\ X)\ =\ 0_d(\ 0,\ 3,\ L,\ 0,\ X)\ =\ \ 0,\ 3,\ L,\ 0,\ X\ ,0$$

Como los extremos son iguales NO corresponde aplicar la parte entre $\{\}$ *y pasamos directamente al tercer tramo de* $\boldsymbol{Ev}'_1$

$$\square_d\ \square_i\ (\ 0,\ 3,\ L,\ 0,\ X\,0\ )=\ 3,\ L,\ 0,\ X$$

Que devuelve exactamente la lista inicial. Se ve que al aplicar $\boldsymbol{Ev}'_2$ *sucede algo similar dejando también invariante la lista dato. Aplicamos entonces* $\boldsymbol{Ev}'_3$

$$\boldsymbol{Ev}'_3\ (3,\ L,\ 0,\ X)\ =$$

$$\boldsymbol{R}_3\ \ 0_d\ \{\square_i\ \square_d\ \boldsymbol{Ev}_3\ \ \ 0_i\ 0_i\ 0_d\}\ \square_d\ \square_i\ (3,\ L,\ 0,\ X)$$

Aplicando el primer tramo de $\boldsymbol{Ev}'_3$*, como la relación* $\boldsymbol{R}_3$ *es verdadera devuelve 1.*

$$\boldsymbol{R}_3\ 0_d\ (3,\ L,\ 0,\ X)\ =\ \ 1,\ 3,\ L,\ 0,\ X\ ,0$$

Como los extremos no son iguales corresponde aplicar la parte entre $\{\}$

$$\{\square_i\ \square_d\ \boldsymbol{Ev}_3\ \ 0_i\ 0_i\ 0_d\ \}(\ 1,\ 3,\ L,\ 0,\ X\ ,0)\ \ =$$

$$\{\square_i\ \square_d\ \underbrace{\square_i\ 0_d\ 0_d\ \{\ \leftrightarrow \rhd\ \}\ \square_d\ 0_i\ 0_i\{\ \ \leftrightarrow \lhd\ \}\ \square_i\square_d}_{\boldsymbol{Ev}_3}\ 0_i\ 0_i\ 0_d\ \}(1,\ 3,\ L,\ 0,\ X,\ 0)$$

Aplicamos la primera parte de la función $\square_i$
$square_d\square_i 0_d\ 0_d$ *y nos queda:*

$$\{\ \leftrightarrow \rhd\ \}\ \square_d\ 0_i\ \{\ \ \leftrightarrow \lhd\ \}\ \square_d\ 0_i\ 0_i\ 0_d\ (\ L,\ 0,\ X,\ 0,\ 0)$$

Observemos que al ser todos los elementos de la lista L positivos la aplicación de $\{\ \leftrightarrow \rhd\ \}$ *se repite hasta que toda la lista L pasa del lado izquierdo de los datos quedando:*

$$\square_d\ 0_i\ \{\ \ \leftrightarrow \lhd\ \}\ \square_d\ 0_i\ 0_i\ 0_d\ (\ 0,\ \ \ X,\ 0,\ L,\ 0)$$

Aplicando $\square_d\ 0_i$ *queda :*

$$\{\ \ \leftrightarrow \lhd\ \}\ \square_d\ 0_i\ 0_i\ 0_d\ (\ 0,\ 0,\ \ \ X,\ 0,\ L)$$

En este caso $\{\ \ \leftrightarrow \lhd\ \}$ *cumple la función de llevar nuevamente la lista L al principio:*

$$\square_d\ 0_i\ 0_i\ 0_d\ (\ 0,\ L,\ 0,\ \ \ X,\ 0)$$

Finalmente aplicando $\square_d\ 0_i\ 0_i\ 0_d$ *la lista queda:*

$$0,\ 0,\ L,\ 0,\ \ \ X,\ 0$$

Por lo que debo dejar de aplicar la parte entre $\{\}$ *aplicando el ultimo tramo de la función que borra los extremos quedando:*

$$0,\ L,\ 0,\ \ X$$

El proceso continua aplicando el resto de los evaluadores, pero como el primer numero es un 0 *, no modifican la lista. El ultimo elemento de la función* $\boldsymbol{Ev}$ *es* $\square_i$ *quedando entonces la lista*

$$L, 0, 0, X$$

O sea asocia al número 3 *la función base* 0_i

2. *Veamos un ejemplo con una lista que cuyo primer elemento sea un* 4*.*

$$4,\ L,\ 0,\ X$$

En este caso

$$\boldsymbol{Ev}_4\ =\ \square_i\ 0_d$$

Como hemos visto en el caso anterior los evaluadores $1, 2$ *y* 3 *dejaran invariante la lista*

Aplicamos entonces $\boldsymbol{Ev}'_4$

$$\boldsymbol{Ev}'_4\ (4,\ L,\ 0,\ X)\ =$$

$$\boldsymbol{R}_4\ \ 0_d\ \{\square_i\ \square_d\ \boldsymbol{Ev}_4\ \ 0_i\ 0_i\ 0_d\}\ \square_d\ \square_i\ (4,\ L,\ 0,\ X)$$

Aplicando el primer tramo de $\boldsymbol{Ev}'_4$

$$\boldsymbol{R}_4\ 0_d\ (4,\ L,\ 0,\ X)\ =\ 0_d\ (\ 1,\ 4,\ L,\ 0,\ X)\ =\ \ 1,\ 4,\ L,\ 0,\ X\ ,0$$

Como los extremos no son iguales corresponde aplicar la parte entre $\{\}$

$$\{\square_i\ \square_d\ \boldsymbol{Ev}_4\ \ 0_i\ 0_i\ 0_d\ \}(\ 1,\ 4,\ L,\ 0,\ X\ ,0)\ \ =$$

$$\{\square_i\ \square_d\ \underbrace{\square_i\ 0_d}_{\boldsymbol{Ev}_4}\ 0_i\ 0_i\ 0_d\ \}(1,\ 4,\ L,\ 0,\ X,\ 0)\ \ =$$

$$(0, 0,\ L,\ 0,\ X,\ 0,\ 0)$$

Los extremos son iguales, por lo que debo dejar de aplicar la parte entre $\{\}$ *aplicando el ultimo tramo de la función que borra los extremos queda:*

$$0,\ L,\ 0,\ \ X,\ 0$$

El proceso continua aplicando el resto de los evaluadores, pero como el primer numero es un 0 *, no modifican la lista. El ultimo elemento de la función* $\boldsymbol{Ev}$ *es* $\square_i$ *quedando entonces la lista*

$$L, 0,\ X,\ 0$$

O sea asocia al número 4 *la función base* 0_d

Para presentar los restantes evaluadores, nos conviene utilizar algunas funciones auxiliares que nos permitirán simplificar su descripción :

Definiciones 3.2.1.

$$\Rrightarrow \stackrel{def}{=} 0_d \left\{ \leftrightarrow \rhd \right\}\Box_d$$

$$\Lleftarrow \stackrel{def}{=} 0_i \left\{ \leftrightarrow \rhd \right\}\Box_i$$

Estas dos funciones las empleamos en el ejemplo visto, para pasar hacia la izquierda o la derecha los elementos de la lista L

$$\bowtie \stackrel{def}{=} 0_d\ S_d\ 0_d\ \left\{\Box_d\ S_i\ \rhd\ 0_d\right\}\ \Box_d\ 0_i\ S_i\ \left\{\Box_i\ \lhd\ Pd\ \rhd\ \Box_i\ \lhd\ 0_i\ S_i\right\}\ \Box_i\ \Box_d$$

$$\triangle \stackrel{def}{=} 0_d\, 0_d\, 0_d\ \Rrightarrow \rhd \rhd \rhd\ 0_i\ S_i \lhd \lhd \lhd \Lleftarrow \Box_d\ \lhd \left\{ \leftrightarrow \right\}\Box_i\ \Box_d\, 0_d\ \Rrightarrow \rhd \rhd \rhd\ \Box_i \lhd \lhd \lhd \Lleftarrow\ \Box_d$$

$$\mathcal{U}(x,\ Y)\ \stackrel{def}{=}\ x,\ U\left(\ \left[\frac{x\dot{-}7}{2}\right]\ \right) +\ 1, Y$$

$$\mathcal{V}(x,\ Y)\ \stackrel{def}{=}\ x,\ V\left(\ \left[\frac{x\dot{-}7}{2}\right]\ \right) +\ 1, Y$$

Donde U y V son las funciones definidas en 3.0.1 . Con $\left[\ \right]$ se indica que se debe tomar la parte entera de la división, A $\mathcal{U}$ y $\mathcal{V}$las utilizaremos al evaluar los valores impares mayores que 6.

$$\mathcal{M}(x,\ Y)\ \stackrel{def}{=}\ x,\ \left[\frac{x\dot{-}2}{2}\right] +\ 1, Y$$

Esta función la utilizaremos en el evaluador de los valores pares mayores que 6.

Queda como ejercicio la construcción de estas tres funciones.

Ademas utilizaremos la ya conocidas D_i, D_d Duplicadores a izquierda y derecha y Pd (Predecesor, definida en 1.3.1 ii)) ,

A partir de estas definiciones se puede mostrar que las funciones buscadas son :

1. $$\boldsymbol{Ev}_1\ =\ \Box_i\ 0_d\ \Rrightarrow\ \Box_i\ \Box_i\ 0_i \Lleftarrow \Box_d\ \bowtie$$

2. $$\boldsymbol{Ev}_2\ =\ \Box_i\ \Box_d \bowtie$$

3. $$\boldsymbol{Ev}_3\ =\ \Box_i\ 0_d\ \Rrightarrow\ \ 0_i\ \ \Lleftarrow \Box_d$$

4. $$\boldsymbol{Ev}_4\ =\ \Box_i\ 0_d$$

5.

$$\boldsymbol{Ev}_5 \;=\; \square_i\; 0_d \;\Rrightarrow\; \square_i \;\; S_i\; 0_i \Lleftarrow \square_d \bowtie$$

6.

$$\boldsymbol{Ev}_6 \;=\; \square_i\; S_d \;\bowtie$$

I

$$\boldsymbol{Ev}_I \;=\; \mathcal{U} \;\rhd\; \mathcal{V} \;\rhd \square_i \;\lhd \lhd$$

P

$$\boldsymbol{Ev}_P \;=\; \triangle\, 0_d \Rrightarrow \rhd\; D_i \;\leftrightarrow \lhd\; 0_i \big\{\, \square_i \leftrightarrow \lhd\; 0_i \big\} \square_i\, \square_d \;\big\{\, \rhd\; \mathcal{M} \;\lhd\; D_d \;\lhd \big\} \square_i\, \square_i$$

Se recomienda verificar cada una de las funciones propuestas (y, eventualmente, proponer mejoras)

Con la construcción de la función $\boldsymbol{Ev}$ *la demostración del teorema es inmediata ya que* $Ev \in \boldsymbol{FRL}$ *y es sencillo verificar que la función* $\boldsymbol{U}$ *es la siguiente*

$$\boldsymbol{U} = S_i \;\rhd\; 0_i \;\lhd\; 0_d \,\big\{\, \square_d\; \boldsymbol{Ev}\; 0_d \big\}\, \square_d\, \square_i$$

$$\boldsymbol{U} \in \mathbf{FRL}$$

Observación:
Todo calculo con funciones recursivas de lista $F(X)$ puede ser entonces pensado como una aplicación de la funcion **U** aplicada a la lista formada por el numero asociado a F y la lista X.

$$\mathbf{U}(t, X)$$

Pero vimos que toda lista puede ser pensada como una sucesión de funciones base aplicada a la lista vacía (ver1.4.1). Llamemos G_X a la función que produce la lista X al ser aplicada a la lista vacia.

$$X = G_X <>$$
$$F(X) \equiv G_x F <>$$

pero F y G_X tendrán asociados sus respectivos números t y g Luego la composición $G_X F$ tendrá asociado el número $k = 6 + 2W(g, t)$ por lo que todo cálculo recursivo (incluyendo datos y procedimientos) puede ser asociado a un número k y representado con la fórmula

$$\mathbf{U}\, k <>$$

3.3. Algunas consecuencias interesantes...

Observemos que existen infinitas funciones de lista que realizan el mismo cálculo.

Recordemos que si dos procedimientos comparten el mismo dominio y tienen el mismo resultado, los consideramos funciones equivalentes (equivalencia extensional, en lógica conocida como regla η)

O sea, para una misma función (desde el punto de vista extensional, pueden corresponder infinitos números diferentes)

A partir de esto se puede derivar una relación de equivalencia en $\mathbb{N}_0$

$$\mathbf{m} \equiv \mathbf{n}$$

Si y solo si la función de lista asociada a **m** aplicada a una lista cualquiera tiene el mismo resultado que la asociada a **n**

Ejemplo 3.3.1. *La función*

$$I = 0_i \ \square_i$$

es tal que $\forall\ X \in \mathcal{L} \quad I(X) = X$

Luego

$$\forall F \in \boldsymbol{FRL}, \quad Y \in \mathcal{D}_F \Longrightarrow FI(Y) = IF(Y) = F(Y)$$

Como el número que corresponde a I es

$$6 + 2 \times W(0,2) = 9$$

Para cualquier número m dados los números

$$n = 6 + 2 \times W(m,9) \ \ y \ p = 6 + 2 \times W(9,m)$$

se cumple

$$m \equiv n \equiv p$$

Lema 3.3.1. *Sea $f \in$* **FR** *de orden k*

$$f : \mathbb{N}_0^k \longrightarrow \mathbb{N}_0 \quad k \in \mathbb{N}_0.$$

Existe una función recursiva de lista que llamaremos F_f tal que si $X^k \in D(f)$

$$F_f(X^k) = f(X^k)$$

$$F_f(X^k, Y) \quad no \ \ esta \ \ definida \ \ si \ \ Y \neq <>$$

$$X^k \in \mathcal{D}(F_f) \iff X^k \in \mathcal{D}_f$$

Observemos que habíamos demostrado en el teorema 2.1.1 que a cada función recursiva f^k le podíamos asociar una función de lista G_f tal que si la lista X^k pertenece al dominio de la función:

$$G_f(X^k, Y) = f^k(X^k), X^k, \ Y \quad \forall \ \ Y \ \ con \quad X^k \in \mathcal{D}_f$$

En este caso buscamos una función de lista que solamente actue sobre las listas X^k que pertenezcan al dominio en que está definida la función f^k , devolviendo el valor $f^k(X^k)$

Para la demostración utilizaremos un conjunto especial de funciones de lista .

Definición 3.3.1.

Sea la familia de funciones I^k definidas de la siguiente manera

$$I^k = \underbrace{0_d \ 0_d \dots}_{k \text{ veces}} \underbrace{\rhd \ \rhd \dots}_{k \text{ veces}} S_i \ \underbrace{\lhd \ \lhd \dots}_{k \text{ veces}} \underbrace{\square_d \ \square_d \dots}_{k \text{ - } 1 \text{ veces}} 0_i \ S_i \{ \leftrightarrow \} \square_d \ \square_i$$

Se puede ver que estas funciones tienen dos propiedades

1. $$X =< x_1, x_2, ..., x_m > \qquad \in D(I^k) \Longleftrightarrow k = m$$

2. $$X \in D(I^k) \Longrightarrow I^k(X) = X$$

Se trata de funciones identidad que "controlan" que su dominio tenga exactamente un dado número de elementos. Observemos la presencia en la construcción de cada I^k de la función:

$$\{ \leftrightarrow \}$$

Que oscila eternamente si el numero de elementos no es k.

Por lo tanto la función buscada en el teorema para cada $f \in \boldsymbol{FR}$ es

$$F_f = I^k G_f \underbrace{\square_d \ \square_d \dots}_{k \text{ veces}}$$

Donde

$$G_f(X, Y) = f(X), X, Y$$

Cuya existencia probamos en 2.1.1

Teorema 3.3.1. *(Kleene)*

Para cada $\boldsymbol{n}$ existe una función $E_n \in \boldsymbol{FR}$ de orden $n+1$ tal que para toda función $f \in \boldsymbol{FR}$ de orden n existe un numero $\boldsymbol{t}$ tal que si $X^n \in \mathcal{D}(f)$ entonces

$$E_n(t, X^n) = f(X^n)$$

$$t, X^n \in \mathcal{D}(E_n) \Longleftrightarrow X^n \in \mathcal{D}(f)$$

Sea la función de lista

$$U_n = I^{n+1} \ U \ I^1$$

Donde U es la función universal de lista definida en 3.2 y las I^k son las definidas en 3.3.1. Por lo visto en el capitulo 2.2 existe una maquina de Turing

que la representa y, por lo tanto hay una función recursiva que responde de la misma manera que dicha MT Esta sera justamente la función E_n buscada

En efecto, dado que podemos asociar a cada función recursiva un numero vinculado con la función recursiva de lista que copia su comportamiento ese sera el numero ***t*** *(en realidad son infinitos los números que podemos emplear para cada función) a reemplazar en los datos de la función para que*

$$E_n(t, X^n) = f(X^n)$$

3.4. Una función NO recursiva

Definiremos una función **NoR** tomando como base la función E_1 definida en el teorema 3.3.1 de la siguiente manera;

$$\mathbf{NoR}(x, y) = \begin{cases} E_1(x, y) & \text{si } y \in \mathcal{D}(F_x) \\ 0 & \text{si } y \notin \mathcal{D}(F_x) \end{cases}$$

Esta función es total pero no puede ser recursiva.

Si **NoR** fuese recursiva también sería recursiva la función;

$$\Phi(x) = \mathbf{NoR}(x, x) + 1.$$

Pero si Φ es recursiva por lo visto en 3.3.1 existirá un valor t tal que

$$E_1(t, x) = \Phi(x)$$

Como $\Phi(x)$ es total $\Longrightarrow \forall x \;\; x \in \mathcal{D}(\Phi) \Longrightarrow \forall x \;\; x \in \mathcal{D}(E_1)$

Si calculamos $E_1(t, x)$ tomando ese valor t como segunda componente, llegamos al absurdo de

$$E_1(t, t) = \Phi(t) = E_1(t, t) + 1$$

Esta aplicación del "truco de la diagonal"muestra desde otro ángulo una versión del problema de la parada ...

Con este ultimo ejemplo entendemos que hemos dado un panorama bastante amplio de las posibilidades del hermoso modelo creado por Giuseppe Jacopini.

Más alla de su evidente potencia didáctica surgen muchas interesantes cuestiones, como por ejemplo:

1. Dada una función de lista, la llamaremos "elegante"si el numero asociado es el menor posible. Existirá una función recursiva de lista que indique ante una función si es elegante? [3]

2. Construir un "Compilador"que dada una función de lista y un dato nos devuelva su resultado. (su existencia hubiera facilitado la escritura de este trabajo).

3. Qué pasa si no nos limitamos a listas finitas?

Para quienes estén interesados en continuar trabajando sobre estos temas o similares se los invita a contactarse con `raulkantor+frl@gmail.com`

[3] Ver Chatin

Capítulo 4

APÉNDICE

Este capítulo lo dedicamos a repasar algunos conceptos utilizados en el trabajo.

4.1. Funciones Recursivas

Llamaremos función numérica a toda función f cuyo dominio sea una potencia cartesiana de $\mathbb{N}_0$ y cuyo codominio sea $\mathbb{N}_0$,

$$f : \mathbb{N}_0^k \longrightarrow \mathbb{N}_0 \quad k \in \mathbb{N}_0.$$

Consideramos el caso en que el valor de k sea cero, con la convención de identificar una función de cero variables con un número perteneciente a los $\mathbb{N}_0$.

Dentro del conjunto de las funciones numéricas, definiremos un subconjunto de funciones que llamaremos **Funciones Recursivas** [1] que se construye a partir de un conjunto de funciones llamadas **base** y la aplicación de tres operadores: **Composición**, **Recursion** y **Minimizador**

4.1.1. Funciones base

Definición 4.1.1 (**Funciones Cero**)**.**
Se trata de funciones constantes definidas sobre las potencias cartesianas de n-uplas de

$$c^{(k)} : \mathbb{N}_0^{(k)} \longrightarrow \mathbb{N}_0$$

$$X \in \mathbb{N}_0^{(k)} \implies c^{(k)}(X) = 0 \in \mathbb{N}_0$$

Son funciones muy simples, ya que sólo consisten en asociar a cualquier valor el número cero.

Definición 4.1.2 (**Proyecciones**)**.**
Definiremos una familia de funciones.
Dados $n \in \mathbb{N}$, $k \in \mathbb{N}$ tales que $1 \leq n$, y $1 \leq k \leq n$.

[1] Modelo desarrollado principalmente a partir de trabajos de Dedekind, Skolem, Peano, Hilbert, Ackerman y Church

Llamamos función proyección de orden n, en la k-ésima componente y notaremos

$$p_k^{(n)} : \mathbb{N}_0^n \longrightarrow \mathbb{N}_0$$

$$p_k^{(n)}(x_1, x_2, ..., x_n) \stackrel{def}{=} x_k \in \mathbb{N}_0$$

Se trata de una familia de funciones sencillas de calcular ya que seleccionan en una n-upla la k-ésima componente

Definición 4.1.3 (**Sucesor**)**.**
Llamamos función Sucesor a la función

$$s : \mathbb{N}_0 \longrightarrow \mathbb{N}_0$$

$$x \in \mathbb{N}_0 \longmapsto s(x) \stackrel{def}{=} x + 1$$

Es decir, s produce el sucesor de su valor de entrada. También en este caso, se trata de una función elemental.

Llamaremos **funciones base** a las funciones $c^{(k)}$ con $k \geq 0$, a las proyecciones $p_k^{(n)}$ con $n, k \in \mathbb{N}$ tal que $1 \leq k \leq n$ y a la función sucesor $s^{(1)}$.

La idea es agregar herramientas de construcción que nos permitan combinando estas funciones bases, como ladrillos elementales, obtener funciones más complejas.

4.1.2. Operador Composición

El modo más inmediato de combinar funciones es "encadenándolas " de modo que los resultados de la aplicación de un grupo de ellas sean del dominio de otra.

Dados $n \geq 1$, $k \geq 0$, una función numérica $f^{(n)}$ y una familia de n funciones numéricas de orden k , $\{g_i^{(k)}\}_{i=1}^n$, diremos que la función h tal que

$$h : \mathbb{N}_0^k \longrightarrow \mathbb{N}_0$$

$$X \in \mathbb{N}_0^k \longmapsto \quad h(X) \stackrel{\text{def}}{=} f(g_1(X), g_2(X), ..., g_n(X)) \in \mathbb{N}_0$$

es definida por la **Composición** de las funciones $f^{(n)}$, $\{g_i^{(k)}\}_{i=1}^n$ y notaremos:

$$h = \Phi\left(f^{(n)}, g_1^{(k)}, g_2^{(k)}, ..., g_n^{(k)}\right)$$

En algunos casos al operador se le agregan los índices $n+1$ y k, $\quad \Phi_k^{n+1}$, para hacer referencia a las $n+1$ funciones sobre las que opera, y a las k-uplas de su dominio.

Φ_k^{n+1} no está definido para todas las $n+1$-uplas de funciones, sino sólo para aquellas n+1-uplas en las cuales la primera sea una función de n-variables y las n restantes sean de orden k.

4.1.3. Operador Recursión

Definiremos un nuevo operador R que llamaremos **Recursión** que a partir de dos funciones: $g^{(k)}$ y $h^{(k+2)}$ construye una nueva función $f^{(k+1)}$

$$f^{(k+1)} = R(g^{(k)}, h^{(k+2)})$$

definida para cada k+1-upla, del siguiente modo:
Si el primer elemento de la $k+1$upla es igual a cero:

$$f(0, X^k) \stackrel{\text{def}}{=} g(X^k)$$

Si el primer elemento es distinto de cero (lo presentamos como $y+1$) el valor de la función se calcula recursivamente del modo siguiente:

$$f(y+1, X^k) \stackrel{\text{def}}{=} h(y, X^k, f(y, X^k))$$

4.1.4. Minimizador

Dada $f^{(n+1)}$, decimos que $g^{(n)}$ se construye por **minimización** de f y lo notamos $M[f]$, cuando g es definida del modo siguiente:

$$g^{(n)}(X) = M[f](X) = \mu_t(f(t, X) = 0)$$

es decir, para cada X^n el valor de $g(X)$ que se le asocia es *(si existe)* el mínimo t tal que $f(t, X) = 0$

Observemos que nada garantiza que exista tal valor de t , por lo que las funciones construidas con M pueden no estar definidas para todos los valores de la n-uplas.

Definiremos el conjunto de **Funciones Recursivas** de la siguiente manera:

a) Las funciones base definidas en 4.1.1 son **Funciones Recursivas**.

b) Las funciones obtenidas a partir de **Funciones Recursivas** aplicándoles un número finito de veces los operadores Composición, Recursión y Minimizador (Φ , R y M) son **Funciones Recursivas**.

Las notaremos: **FR**.

4.2. Maquina de Turing

Existen diversas descripciones, todas en esencia equivalentes, del autómata creado por el genial lógico **Alan Turing** a las que notaremos genéricamente **MT**

La idea base es suponer que todo cálculo puede ser realizado “mecánicamente” aplicando rigurosa y acríticamente un conjunto de reglas ante determinadas situaciones.

4.2.1. Descripción de la Máquina de Turing

La información que procesa esta máquina ideal se encuentra en una cinta dividida en casillas en las que hay símbolos de un alfabeto finito:

$$A = \{s_0, s_1, s_2, \ldots s_n\}$$

Este alfabeto tiene al menos dos símbolos.

Uno de tales símbolos, que notaremos con s_0 se usa para indicar que la casilla está vacía. A veces utilizaremos el nombre "blanco" para referirnos al mismo, y también lo notaremos con $\square$

La cinta se considera infinita, pero se conviene que sólo un conjunto finito de casillas tienen, al iniciar el proceso, símbolos diferentes de s_0 (es decir no están vacías).

La **MT** en cada momento se considera que está "observando " o "leyendo" una casilla (se puede pensar en un visor que se desliza por la cinta) y se encuentra en un "estado" q_j que pertenece a un conjunto finito de estados Q.

$$Q = \{q_0, q_1, q_2, \ldots q_m\}$$

El conjunto de estados tiene siempre, al menos, dos estados: uno llamado 'estado inicial'. que es en el que se encuentra la **MT** al comenzar el cálculo (por convención lo codificaremos con q_1) y uno llamado "estado final" (convenimos en codificarlo q_0) en el cual la máquina se detiene y da por terminado el cálculo.

Dependiendo del estado en que se encuentra (supongamos q_r) y del símbolo que esta en la casilla observada (supongamos s_i) la **MT** realiza consecutivamente tres operaciones:

a) Símbolo: Sustituye el símbolo observado s_i por otro pertenenciente al alfabeto A. (eventualmente puede ser el mismo símbolo)

b) Movimiento: Se mueve a la celda derecha o a la celda izquierda o permanece en la misma celda. (notaremos d, i, n las tres alternativas)

c) Estado : Sustituye el estado en que se encuentra q_r por otro pertenenciente al conjunto de estados Q.(eventualmente puede ser el mismo estado)

Partiendo de una configuración dada de la cinta que llamamos : *configuración inicial* y del estado inicial (q_1), se repiten los pasos vistos hasta que la **MT** alcance el que llamamos "estado final"(q_0) y la **MT** se detiene. (lo que no siempre sucede necesariamente).

Si se llega a ese estado q_0, la configuración de la cinta en ese estado es el resultado del cálculo y se le llama *configuración final*.

Por lo tanto una **MT** queda totalmente determinada por su alfabeto, sus estados y una función que le indica que hacer en cada momento a partir del estado en que se encuentra y el símbolo que está observando.

$$f_{MT} : A \times (Q - q_0) \longrightarrow A \times Q \times M$$

Donde con A representamos el alfabeto, $Q - q_0$ los estados distintos del final y M es el conjunto de los posibles movimientos $M = \{d, i, n\}$

Esta función puede ser representada por una tabla de $(n+1)$ filas correspondientes a cada uno de los elementos del alfabeto y m columnas correspondientes a los estados q_1 a q_m

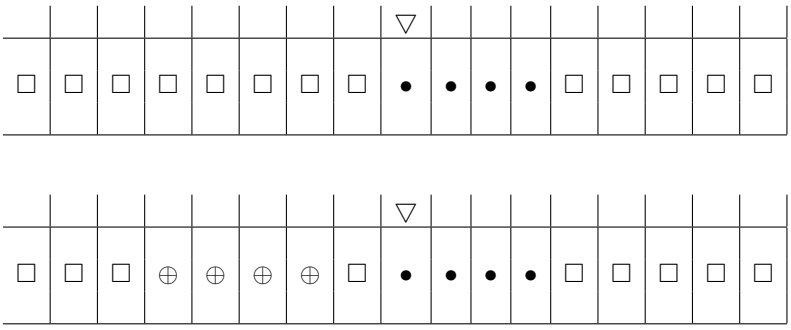
En cada una de las casillas i, j , asociadas al símbolo s_i del alfabeto y al estado q_j aparecerá una terna (s_m, m_k, q_h) que indicará respectivamente el símbolo a escribir, el movimiento a ejecutar y el estado sucesivo que corresponda.

La primera columna de la tabla (correspondiente al estado q_1) es la del estado en que se encuentra la máquina en el momento de comenzar a trabajar.

Ejemplo 4.2.1. *Tomando como alfabeto* $A = \{\square, \bullet, \oplus\}$
La máquina que tiene esta tabla

	q_1	q_2	q_3	q_4	q_5	q_6
$\bullet$	$\oplus, d, q_1$	err	$\bullet, i, q_3$	$\bullet, i, q_4$	$\bullet, d, q_5$	$\bullet, i, q_2$
$\oplus$	err	$\bullet, n, q_3$	$\oplus, i, q_3$	$\oplus, i, q_4$	$\oplus, d, q_5$	$\oplus, d, q_6$
$\square$	$\square, i, q_2$	$\square, d, q_0$	$\square, i, q_4$	$\oplus, n, q_5$	$\square, d, q_6$	$\square, n, q_2$

En el estado inicial en la cinta hay escrito una cantidad de "$\bullet$ " y la casilla observada es la del primer $\bullet$, al final quedara un grupo de la misma cantidad de $\oplus$ separados por un blanco.



4.2.2. Composición de Máquinas de Turing

Sean dos máquinas M_1 y M_2, *ambas con el mismo alfabeto.*

Se define la máquina N composición de M_1 y M_2 (notamos $N = M_1\ M_2$), a la máquina que realiza la operación de aplicar primero la máquina M_2 y al resultado aplicar la M_1.

Es sencillo construir, a partir de las tablas de las máquinas M_1 y M_2 la tabla de la N.

Se adjunta inmediatamente a la tabla de M_2, la tabla de M_1 cambiando los nombres de los estados para que no se confundan (por ej. haciendo $q_i = q_i^{'}$) y

reemplazando en la tabla de la M_2 cada aparición del estado final (q_0) con el inicial de la máquina M_1 , ($q_1{}'$).

	q_1	q_2	q_3	q_4
s_0	...		...	..
s_1	...		$..q_0$	..
s_2	...		...	$..q_0$

	q_1	q_2	q_3
s_0	...	...	...
s_1	...	...	...
s_2	...	...	...

	q_1	q_2	q_3	q_4	$q_1{}'$	$q_2{}'$	$q_3{}'$
s_0	...	...	...	...	...	...	...
s_1	...	$q_1{}'$	...	...	...	...	...
s_2	...	...	$q_1{}'$	...	...	...	...

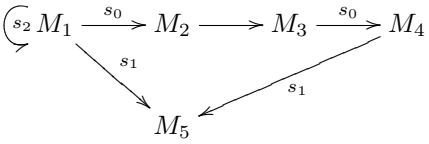
4.2.3. Diagramas de composición

Veamos una forma más general de componer máquinas.

Ante el estado final de una máquina se toma en consideración el símbolo observado en ese estado y se la conecta con una determinada máquina.(eventualmente la misma)

Esto se diagrama con una flecha, marcada con el símbolo que parte de la máquina que ha terminado hacia la que continua con el proceso.

Supongamos varias máquinas: M_1,M_2, M_3,M_4,M_5, todas sobre el mismo alfabeto (en este ejemplo $[s_0, s_1, s_2]$).



En este ejemplo si M_1 alcanza el estado q_0 con s_2 vuelve a su estado q_1, si termina con s_0 va al estado q_1 de M_2 y si finaliza con s_1 al estado q_1 de M_5.

La tabla de la máquina compuesta se construirá renombrando los estados de cada máquina para evitar coincidencias, agrupando las tablas de las máquinas y en cada tabla donde aparece el estado q_0, reemplazarlo, si fuera necesario, de acuerdo al símbolo del alfabeto, con el estado ${q_1}^j$ de la máquina j que le corresponde.

4.3. Representación de MT por FR

Para mostrar que las **MT** pueden ser representadas por **FR** usaremos como herramienta una de las llamadas funciones de Gödel

4.3.1. Funciones de Gödel

Dado que el conjunto de las listas finitas de números de $\mathbb{N}_0$ es numerable, implica que su cardinalidad es la misma de $\mathbb{N}_0$ y por lo tanto es posible encontrar funciones que asocien en forma unívoca un valor de $\mathbb{N}_0$ a cada lista finita de números naturales.

Definición 4.3.1. *Diremos que una función $G^{(2)}$ es función de* ***Gödel*** *si para cada lista finita $< a_0, a_1, ...a_s >$ de elementos de $\mathbb{N}_0$, existe un valor z (al que se le llama número de Gödel de la lista) que también notaremos:*

$$z = \overline{< a_0, a_1, ...a_s >}$$

tal que:

I)

$$G(i, z) = a_i \quad i = 0, 1, 2, ...s$$

II)

$$Se\ \ cumple\ \ que\ \ \overline{< x_0, x_1, ...x_s >} < \overline{< y_0, y_1, ...y_s, y_{s+1}, ...y_k >}$$

$$con \quad s < k, \quad x_i = y_i \ \ para \ \ i = 0, ...s$$

O sea que el número de Gödel asociado a cualquier lista finita es siempre menor que el que se le asocia al de la lista que se forma agregando nuevos elementos a la primera.

Ejemplo 4.3.1.

Sea una función ***dupla*** W *y sus inversas U ,V definidas en 3.0.1*

A cualquier sucesión finita $< a_0, a_1, ...a_k >$ se le puede asociar unívocamente el número:

$$z = W(a_0, W(a_1, ...(W(a_k, 0))))$$

Entonces tenemos la función de Gödel:

$$G(i, z) = U(V^i(z))$$

Utilizando la función dupla vista en el ejemplo 3.0.1 se ve que esta función de Gödel es ***FR***

Teorema 4.3.1. *Para toda* ***MT*** *se puede encontrar una función recursiva que realiza el mismo cálculo.*

Debemos encontrar una forma de representar la configuración de la cinta de la ***MT*** *como un número.*

Sin perdida de generalidad mostraremos el teorema para las ***MT*** *que tienen un alfabeto de diez símbolos $(s_0, s_1, s_2, \ldots, s_9)$ y $p + 1$ estados.*

En este caso asociamos el $\square$ al símbolo s_0

En cada momento la situación de una dada ***MT*** *está completamente definida por los datos que contiene la cinta, la casilla observada y el estado en que se encuentra*

Consideramos la cinta dividida en tres partes: la celda observada, y las zonas que se encuentran a izquierda y derecha de esta celda.

Supongamos una configuración como la siguiente:

								Obs										
s_0	s_0	s_0	s_5	s_3	s_8	s_4	s_2	s_3	s_1	s_7	s_9	s_2	s_0	s_0	s_0	s_0	s_0	s_0

Asociamos a la parte izquierda de la cinta el número que tiene como cifras decimales los índices de los símbolos contenidos en las casillas a la izquierda de la casilla observada, teniendo en cuenta que no consideramos los ceros a la izquierda. Llamamos a este número C_i

En nuestro ejemplo será $C_i = 53842$

A la casilla observada le asociamos como número, el índice del símbolo que contiene. Notamos ese número Ob.

En nuestro ejemplo será $Ob = 3$

Finalmente asociamos a la parte derecha de la cinta el número que forman los subíndices de los símbolos que contienen pero leídos de derecha a izquierda. Con lo que los que son los infinitos "ceros a la derecha" los que no cuentan en este caso. Notaremos ese número como C_d

En nuestro ejemplo será $C_d = 2971$

La situación en que se encuentra la ***MT*** *en cada momento queda completamente determinada por el contenido de la cinta , el valor de la celda observada y el estado* q_e *en que se encuentra. Por lo tanto los valores:* C_i, Ob, C_d *y el número del estado (un valor entre* 1 *y* p *que notaremos:* e*) describen la situación de la cinta en cada momento.*

Asociemos a estos cuatro valores un número de Gödel (como se vio en 4.3.1 que llamaremos N

$$N = \overline{< C_i, Ob, C_d, e >}$$

Aplicando una función G *como la vista en 4.3.1 se puede reconstruir:*

$$C_i = G(0, N) \quad Ob = G(1, N) \quad C_d = G(2, N) \quad e = G(3, N)$$

A partir del estado en que se encuentra y del valor observado la máquina realiza lo que le indica su tabla y alcanza una nueva configuración y estado,

$$C_i^{'}, Ob^{'}, C_d^{'}, e^{'}$$

al que podemos asociar un $N^{'}$.

$$N^{'} = \overline{< C_i^{'}, Ob^{'}, C_d^{'}, e^{'} >}$$

Queremos ver cual es la naturaleza de función que llamaremos F_M *que va de* N *a* $N^{'}$

A partir del valor observado y el estado en la tabla de la Maquina se obtienen tres datos: el símbolo que sustituye al observado (lo llamaremos S_k *y le asociamos el valor* k*), el nuevo estado* q_f *(al que asociamos el valor* f*) y el movimiento a realizar* (d, i, n).

Asociemos el valor 1 *al movimiento hacia derecha, el valor* 2 *al movimiento a la izquierda y el valor* 3 *al no-movimiento. Llamamos* m *al valor del movimiento.*

Los valores k, p *y* m *están determinados por la tabla de la máquina.*

Podemos pensar esta Tabla como la unión de tres funciones $S^{(2)}$, $E^{(2)}$ *y* $M^{(2)}$ *definidas para los valores* 0..,9 *de los símbolos y para los valores* 1...p

de los estados. Estas funciones las podemos extender para todo $\mathbb{N}_0^2$*, del modo siguiente:*

$$S(i,j) \ = E(i,j) \ = M(i,j) \ = \ 0 \ \ \forall\, i > 9 \ \forall \ j > p$$

(p es el número de estados).

Se observa que las funciones $S^{(2)}$*,* $E^{(2)}$ *y* $M^{(2)}$ *son* ***Funciones Recursivas*** *ya que son distintas de cero en un conjunto finito de su dominio.*

Veamos ahora en que modo se modifican los valores que describen la situación de la ***MT*** *cuando se realiza un movimiento hacia la derecha y se sustituye el símbolo observado con el de la función*

$$S(Ob, e)$$

Se puede ver que los nuevos valores que asociamos a las partes izquierda y derecha la cinta, al valor observado y al nuevo estado son :

$$C_i^{'} = C_i \times 10 + S(Ob, e)$$
$$C_d^{'} = Q(C_d, 10)$$
$$Ob^{'} = Mod(C_d, 10)$$
$$e^{'} = E(Ob, e)$$

Donde S *y* E *son las funciones de la tabla y*

$$Q(x, y) = [x/y]$$

es la parte entera de la división.(Con la convención $Q(x, 0) = 0$*) y*

$$Mod(x, y)$$

es el resto de la división de x por y.(Con la convención $Mod(x, 0) = 0$*) En el ejemplo visto, si suponemos que el valor que sustituye al* 3 *es* 7 *y el movimiento es hacia la derecha:*

									Ob									
s_0	s_0	s_0	s_5	s_3	s_8	s_4	s_2	s_7	s_1	s_7	s_9	s_2	s_0	s_0	s_0	s_0	s_0	s_0

$$C_i^{'} = 53842 \times 10 + 7 = 538427$$
$$C_d^{'} = Q(2971, 10) = 297$$
$$Ob^{'} = Mod(2971, 10) = 1$$
$$e^{'} = E(Ob, e)$$

Si el movimiento es hacia la izquierda:

							Ob											
s_0	s_0	s_0	s_5	s_3	s_8	s_4	s_2	s_7	s_1	s_7	s_9	s_2	s_0	s_0	s_0	s_0	s_0	s_0

$$C_i^{'} = Q(C_i, 10) \quad en \quad el \quad ejemplo = 5384$$

$$C_d^{'} = C_d \times 10 + S(Ob, e) \quad en \quad el \quad ejemplo = 29717$$

$$Ob^{'} = Mod(C_i, 10) \quad en \quad el \quad ejemplo = 2$$

$$e^{'} = E(Ob, e)$$

y si no se mueve:

$$C_i^{'} = C_i \quad en \quad el \quad ejemplo = 53842$$

$$C_d^{'} = C_d \quad en \quad el \quad ejemplo = 2971$$

$$Ob^{'} = S(Ob, e) \quad en \quad el \quad ejemplo = 7$$

$$e^{'} = E(Ob, e)$$

Veamos ahora como calculamos $N^{'}$ en cada caso :

Movimiento hacia la derecha ($m = 1$):

$$N_1^{'} = \overline{< C_i^{'}, Ob^{'}, C_d^{'}, e^{'} >} = \overline{< C_i \times 10 + S(Ob, e), Mod(C_d, 10), Q(C_d, 10), E(Ob, e) >}$$

$$N_1^{'} = \overline{< G(0,N) \times 10 + S(G(1,N), G(3,N)), Mod(G(2,N), 10), Q(G(2,N), 10), E(G(1,N), G(3,N)) >}$$

Movimiento hacia la izquierda ($m = 2$):

$$N_2^{'} = \overline{< Q(G(0,N), 10), Mod(G(0,N), 10), G(2,N) \times 10 + S(G(1,N), G(3,N)), E(G(1,N), G(3,N)) >}$$

No se mueve ($m = 3$):

$$N_3^{'} = \overline{< G(0,N), S(G(1,N), G(3,N)), G(2,N), E(G(1,N), G(3,N)) >}$$

Para contemplar las tres posibilidades basta usar el valor del movimiento para anular los casos que no nos interesan:

$$N^{'} = (N_1^{'} \times (m-2) \times (m-3)) + (N_2^{'} \times (m-1) \times (m-3)) + (N_3^{'} \times (m-1) \times (m-2))$$

Pero $m = M(Ob, e) = M(G(1,N), G(3,N))$

*Luego el valor $N^{'}$ se puede obtener como resultado de aplicar una **FR** [2] que llamaremos F_M al número de Gödel N de que describe la configuración inicial de la **MT** y repetir la aplicación de la misma hasta que (si es posible el cálculo) se llegue al estado q_0*

[2] Se trata de una función Recursiva Primitiva, ya que para su calculo solo se emplea la composición y la recursión

Podemos crear una función Numero de pasos $\mathcal{N}_p$ definida como el mínimo valor t para aplicar F_M al numero que representa la configuración inicial hasta que $G({F_M}^t(N), 3) = 0$

$$\mathcal{N}_p(N) = \mu_t(G({F_M}^t(N, 3) = 0)$$

*Si el calculo se completa y se se encuentra es mínimo valor t , la función ${F_M}^t$ es la **FR** buscada .*

Observacion : La condición de que el alfabeto tenga diez valores no es ninguna limitación. En el caso de un alfabeto de k elementos basta tomar el sistema numérico de base k . El valor 10 *de las fórmulas sigue valiendo pues en cada caso es la forma de expresar el valor de la base)*

4.4. La Maquina Z

Se puede construir con un alfabeto de tres símbolos $\{\bullet, \square, \oplus\}$ El agregado de un símbolo auxiliar simplifica (no aparece en el resultado final). Se puede mostrar que se puede construir una con idéntico comportamiento utilizando solo dos símbolos.(es un poco más complicada)

De todos modos, la configuración inicial y la final tienen solo dos símbolos ($\square$, $\bullet$)

Utilizaremos como base la Máquina del ejemplo 4.2.1 que duplicaba a la izquierda de una serie de $\bullet$ el mismo número de $\oplus$. Llamaremos a esta Máquina DUP_i.

Es inmediato que podemos construir una máquina (que llamaremos DUP_d que realice similar tarea , pero dejando a la derecha la serie de $\oplus$.

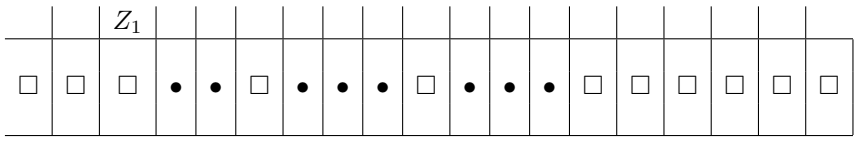
Es entonces que una primera parte de nuestra máquina **Z** :

$$Z_1 = d \rightarrow DUP_i \rightarrow D^{\square\,\square} \rightarrow i \rightarrow i \rightarrow DUP_d$$

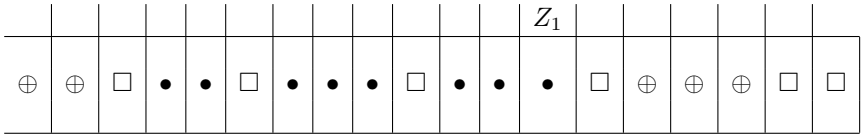
La máquina d la ubica en el primer elemento del primer grupo de $\bullet$ de la lista, luego la DUP_i duplica ese primer grupo de la lista con $\oplus$ a la izquierda.

La $D^{\square\,\square}$ avanza hasta encontrar dos $\square$ (lo que indica el final de la lista) y luego se retrocede dos lugares a la izquierda (para poder aplicar la DUP_d)

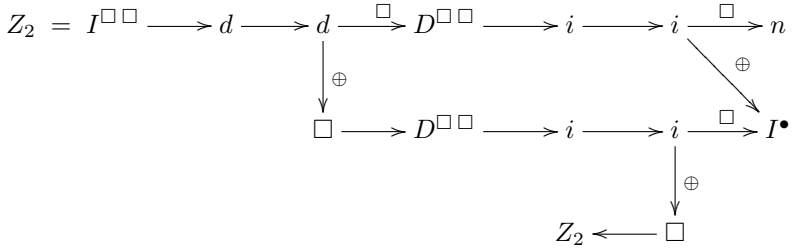
Ejemplo 4.4.1. *Si tenemos como cinta inicial la lista :*



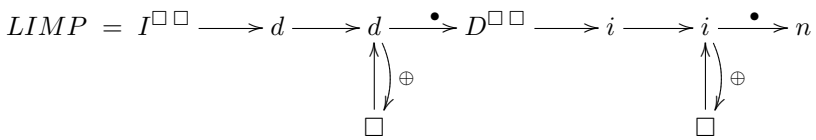
Luego de la aplicación de Z_1



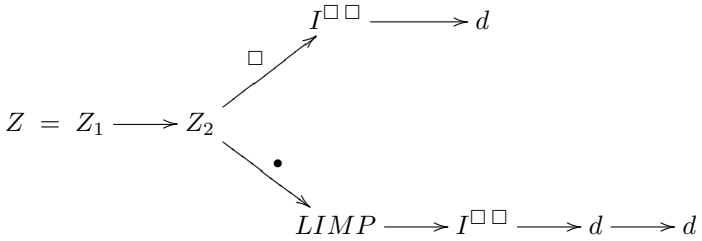
Ahora construimos una Z_2 que va borrando uno de cada lado de los $\oplus$ hasta que no quede ninguno en uno de los extremos (si sucede lo mismo en el otro extremo se termina en un $\square$, son iguales), si esto no sucede se posiciona en un $\bullet$



En el caso que los valores sean diferentes se deben borrar los $\oplus$ que quedan:



Finalmente según el resultado se posiciona.



1. **Espero comentarios...**