Lenguajes Formales y Computabilidad

12 de diciembre de 2018

Índice general

Parte I

Principio de induccion y cardinalidad

Capítulo 1

Principio de induccion

1.1. Definiciones

1.1.1. Conjunto Inductivo

Una definicion inductiva de un conjunto A comprende base, induccion y clausura:

base conjunto de uno o mas elementos «*iniciales*» de A.

induccion una o mas reglas para construir «*nuevos*» elementos de A a partir de «*viejos*» elementos de A.

clausura determinar que A consiste exactamente de los elementos obtenidos a partir de los basicos y aplicando las reglas de induccion, sin considerar elementos «*extra*».

La forma de clausurar es pedir que A sea el minimo conjunto que satisface las condiciones de base e induccion o en forma equivalente, definir a A como la interseccion de todos los conjuntos que satisfacen dichas condiciones.

Definicion formal Sean U un conjunto que llamaremos universo, B un subconjunto de U que llamaremos base y K un conjunto no vacio de funciones que llamaremos constructor.

Diremos que un conjunto A esta definido inductivamente por B, K, U si es el minimo conjunto que satisface:

- $B \subseteq A$.
- Si $f \in K \wedge a_1, \ldots, a_n \in A$ entonces $f(a_1, \ldots, a_n) = a \in A$.

1.1.2. Secuencia de formacion

Sean U, B, K como en la definicion anterior. Una secuencia $a_1, \ldots, a_m$ de elementos de U es una secuencia de formacion para a_m si $\forall i = 1, \ldots, m$ se verifica que:

- $a_i \in B$ o bien,
- $\exists f \in K$ con $ar(f) = n$ y $0 < i_1, \ldots, i_n < i$ tales que $f(a_{i_1}, \ldots, a_{i_n}) = a_i$

Notemos que el conjunto A tiene todos los elementos de U que poseen una secuencia de formacion.

Diremos que B y K definen una gramatica para las cadenas sintacticamente correctas del lenguaje A.

1.2. Demostraciones

1.2.1. Pertenencia

Para probar que un elemento pertenece a un conjunto inductivo, debemos dar su secuencia de formacion.

Ejemplo Sea L el minimo conjunto que satisface:

- $\lambda, 0, 1 \in L$.
- $a \in L \wedge b \in \{0, 1\} \Rightarrow bab \in L$

Probaremos que $110111011 \in L$. En efecto posee la siguiente secuencia de formacion: $1 \Rightarrow 111 \Rightarrow 1011101 \Rightarrow 110111011$.

1.2.2. No pertenencia

Para probar que un elemento no pertenece a un conjunto inductivo, podemos:

- Mostrar que no existe una secuencia de formacion para el elemento.
- Mostrar que si se quita al elemento del conjunto se siguen cumpliendo las clausulas.
- *Probar cierta propiedad del conjunto que sirva para excluir al elemento.*

Por ejemplo, para probar que $110111010 \notin L$ (definido en el apartado anterior) podriamos demostrar que todas las cadenas de L comienzan y terminan con el mismo caracter.

Para demostrar este tipo de propiedades podemos valernos del principio de induccion primitiva que se detalla a continuacion.

1.2.3. Principio de induccion primitiva

Enunciado Sea $A \subseteq U$ definido inductivamente por la base B y el constructor K, si:

1. vale $P(x)\ \forall x \in B$ y si
2. para cada $f \in K$ resulta: $P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n) \Rightarrow P[f(a_1, a_2, \ldots, a_n)]$

entonces vale $P(x)\ \forall x \in A$.

Demostracion Sea C el conjunto de todos los elementos de A que satisfacen una propiedad P, queremos probar que $C = A$.

- $C \subseteq A$ es trivial por definicion.

- Veamos que C satisface las clausulas de la definicion inductiva de A:
  - Sea $x \in B$, luego por *(1)* vale $P(x)$ y entonces $x \in C$ por lo que $B \subseteq C$.
  - Sean $f \in K$, $a_1, a_2, \ldots, a_n \in C$ y $f(a_1, a_2, \ldots, a_n) = a$ queremos probar que $a \in C$:
    - Por definicion de C valen $P(a_1), P(a_2), \ldots, P(a_n)$.
    - Por *(2)* vale $P(a)$.

    Luego por definicion de C resulta $a \in C$.

  Dado que A es el minimo conjunto que cumple las clausulas de su definicion inductiva concluimos que $A \subseteq C$.

Puesto que $C \subseteq A$ y $A \subseteq C$ entonces debe ser $A = C$.

Capítulo 2

Cardinalidad

2.1. Definiciones

2.1.1. Funcion inyectiva

Decimos que $f : X \to Y$ es *inyectiva* si: $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$ o bien $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$.

2.1.2. Funcion sobreyectiva

Decimos que $f : X \to Y$ es *sobreyectiva* si: $\forall y \in Y \ \exists x \in X / f(x) = y$.

2.1.3. Funcion biyectiva

Decimos que $f : X \to Y$ es *biyectiva* si es inyectiva y sobreyectiva.

2.1.4. Conjuntos equipotentes

Dos conjuntos A y B tienen la misma cardinalidad (son *equipotentes*) si existe una funcion biyectiva de A en B y lo notaremos: $\#A = \#B$, $A \sim B$.

2.1.5. Cardinalidad precedente

La cardinalidad de un conjunto A es anterior a la de un conjunto B si existe una funcion inyectiva f de A en B y lo notaremos $\#A \preceq \#B$.

Si ademas ninguna de las funciones inyectivas de A en B es sobreyectiva entonces: $\#A \prec \#B$.

2.1.6. Conjuntos finitos

Un conjunto es finito cuando es vacio o equipotente a $[\![1, n]\!]$ para algun $n \in \mathbb{N}$. En caso contrario se dice infinito.

2.1.7. Conjuntos numerables

Diremos que un conjunto A es numerable si es finito, o bien resulta que $A \sim \mathbb{N}$ en cuyo caso se dice que A es infinito numerable. Si nada de lo anterior aplica se dice que A no es numerable.

2.1.8. Familia de conjuntos

Un conjunto F se dice una famlia de conjuntos si sus elementos son conjuntos. Diremos que F es una familia indexada de conjunto indice I (no vacio) si existe una funcion con dominio I y recorrido F.

Llamando S_α (con $\alpha \in I$) a los elementos de la familia F, podemos entonces decir que $F = \{S_\alpha / \alpha \in I\}$.

2.1.9. Conjunto de partes

Dado un conjunto S, el conjunto de partes de S denotado por $\mathcal{P}(S)$ es el conjunto de todos los subconjuntos de S.

2.2. Teoremas

2.2.1. Teorema de Cantor-Schroder-Bernstein

Enunciado Si $\#A \preceq \#B$ y $\#B \preceq \#A$ entonces $A \sim B$. En otras palabras: si existe una funcion inyectiva de A en B y otra de B a A entonces existe una funcion biyectiva de A a B.

Demostracion Consultar bibliografia [4] y [5] paginas 322 y 232.

2.2.2. Formulaciones equivalentes

Enunciado Sea A un conjunto, luego las siguientes expresiones son equivalentes:

1. A es numerable.
2. $A = \emptyset$ o existe una funcion sobreyectiva $f : \mathbb{N} \to A$.
3. Existe una funcion $g : A \to \mathbb{N}$ que es inyectiva.

Demostracion Consultar bibliografia [4] pag. 310.

2.2.3. Cardinalidad de $\mathbb{N} \times \mathbb{N}$

Enunciado El producto cartesiano $\mathbb{N} \times \mathbb{N}$ es infinito numerable.

Demostracion Sea $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ dada por $f[(i,j)] = \frac{1}{2}(i+j-1)(i+j-2)+i$, observemos primero que

1. $\forall j \in \mathbb{N}$ resulta:

$$\begin{aligned} f[(1,j+1)] - f[(1,j)] &= \left[\frac{1}{2}(1+j)(j)+1\right] - \left[\frac{1}{2}(j)(j-1)+1\right] \\ &= \frac{1}{2}j + \frac{1}{2}j^2 + 1 - \frac{1}{2}j^2 + \frac{1}{2}j - 1 = j \end{aligned}$$

2. $\forall i,j \in \mathbb{N}$ resulta $f[(1,i+j-1)] \leq f[(i,j)] < f[(1,i+j)]$ luego $i+j$ es el menor numero natural n tal que $f[(i,j)] < f[(1,n)]$.

- Veamos que f es inyectiva: supongamos $f[(i_1, j_1)] = f[(i_2, j_2)]$. Luego por *(2)*, $i_1 + j_1$ es el menor natural n tal que $f[(i_1, j_1)] < f(1, n)$ y por nuestra suposicion tambien tenemos que $f[(i_2, j_2)] < f(1, n)$ por lo que $i_1 + j_1 = i_2 + j_2$. Usando la definicion de f obtenemos:

$$\begin{aligned} i_1 &= f[(i_1, j_1)] - \frac{1}{2}(i_1 + j_1 - 2)(i_1 + j_1 - 1) = \\ &= f[(i_2, j_2)] - \frac{1}{2}(i_2 + j_2 - 2)(i_2 + j_2 - 1) = i_2 \end{aligned}$$

  y puesto que $i_1 + j_1 = i_2 + j_2$ concluimos que $j_1 = j_2$ y en consecuencia $(i_1, j_1) = (i_2, j_2)$.

- Para ver que f es suryectiva supongamos $n \in \mathbb{N}$. Sea k el menor natural tal que $f[(1, k)] > n$. Notemos que $f[(1, 1)] = 1 \leq n$ por lo que $k \geq 2$. Como k es el menor tenemos que $f[(1, k-1) \leq N]$ y por (1) resulta:

$$0 \leq n - f[(1, k-1)] < f[(1, k)] - f[(1, k-1)] = k - 1$$

  y agregando 1 miembro a miembro obtenemos $1 \leq n - f[(1, k-1)] + 1 < k$. Sea $i = n - f[(1, k-1)] + 1$ entonces $1 \leq i < k$; y sea $j = k - i$. Notese que $i, j \in \mathbb{N}$. Con estas elecciones de i, j concluimos:

$$\begin{aligned} f[(i, j)] &= \frac{1}{2}(i + j - 2)(i + j - 1) + i \\ &= \frac{1}{2}(k - 2)(k - 1) + n - f[(1, k-1)] + 1 \\ &= \frac{1}{2}(k - 2)(k - 1) + n - \left[\frac{1}{2}(k - 2)(k - 1) + 1\right] + 1 = n \end{aligned}$$

Observacion Observemos una tabla de valores para comprender mejor esta funcion:

$f[(i, j)]$	1	2	3	4	5	6
1	1	2	4	7	11	↙
2	3	5	8	12	↙	
3	6	9	13	↙		
4	10	14	↙			
5	15	↙				
6	↙					

2.2.4. Corolario

Enunciado $\mathbb{N}^d \sim \mathbb{N}$.

Demostracion Lo demostraremos por induccion:

Para $d = 1$ vale trivialmente. Veamos ahora que si $\mathbb{N}^d \sim \mathbb{N} \Rightarrow \mathbb{N}^{d+1} \sim \mathbb{N}$.

Escribamos $\mathbb{N}^{d+1} = \mathbb{N}^d \times \mathbb{N}$. Como $\mathbb{N}^d$ es numerable (por hipotesis inductiva) podemos listar a sus elementos: $\mathbb{N}^d = \{a_1, a_2, a_3, \ldots\}$.

Sea $f : \mathbb{N}\times\mathbb{N} \to \mathbb{N}^{d+1}$ dada por $f(i,j) = (a_i, j)$ resulta $\mathbb{N}^{d+1} \sim \mathbb{N}\times\mathbb{N} \sim \mathbb{N}$.

2.2.5. Union numerable de conjuntos numerables

Enunciado Sean S_α conjuntos numerables (finitos o infinitos) y un conjunto indice I tambien numerable (finito o infinito) entonces la union de los elementos de la familia $F = \{S_\alpha : \alpha \in I\}$, es decir $S = \bigcup_{\alpha\in I} S_\alpha$ sera tambien numerable.

Demostracion Nos pondremos en el peor caso posible: supondremos que tanto los conjuntos S_α como el conjunto indice I son infinito numerables.

Dado que el conjunto indice I es infinito numerable, sin perder generalidad podemos considerar de aqui en mas que $I = \mathbb{N}$. Luego podemos escribir entonces $F = \{S_\alpha : \alpha \in I\} = \{S_i : i \in \mathbb{N}\}$.

Dado que S_i es infinito numerable, podemos escribir $S_i = \{a_{ij}/j \in \mathbb{N}\} = \{a_{i1}, a_{i2}, a_{i3}, \ldots\}$. Observemos que podemos organizar los elementos de la union de acuerdo a la siguiente tabla:

$$\begin{array}{ccccc} a_{11} & a_{12} & a_{13} & a_{14} & \cdots \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots \\ a_{31} & a_{32} & a_{33} & a_{34} & \cdots \\ a_{41} & a_{42} & a_{43} & a_{44} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Luego la funcion $f : S \to \mathbb{N} \times \mathbb{N}$ dada por $f(a_{ij}) = (i,j)$ es inyectiva y como $\mathbb{N} \times \mathbb{N} \sim \mathbb{N}$ resulta que S es numerable.

2.2.6. Cardinalidad infinita mas pequeña

Enunciado Para todo conjunto infinito A, resulta: $\#\mathbb{N} = \aleph_0 \preceq \#A$.

Demostracion Sea A un conjunto infinito:

- Como A es infinito resulta $A \neq \emptyset \Rightarrow \exists x_1 \in A$.
- Como A es infinito resulta $A \neq \{x_1\} \Rightarrow \exists x_2 \in A / x_2 \neq x_1$.
- Como A es infinito resulta $A \neq \{x_1, x_2\} \Rightarrow \exists x_3 \in A / x_3 \neq x_1, x_2$.
- Como A es infinito resulta $A \neq \{x_1, x_2, x_3\} \Rightarrow \exists x_4 \in A / x_4 \neq x_1, x_2, x_3$.

De esta forma se puede construir una sucesion $(x_n)_{n\geq 1}$ de elementos de A tales que $x_i \neq x_j$ si $i \neq j$.

Definimos $f : \mathbb{N} \to A$ dada por $f(i) = x_i$ para todo $i \in \mathbb{N}$. Como f es inyectiva resulta que $\aleph_0 \preceq \#A$.

2.2.7. Cardinalidad del conjunto de partes

Enunciado Para todo conjunto S, resulta: $\#S \prec \#\mathcal{P}(S)$.

Demostracion La funcion $f(x) = \{x\}$ es inyectiva de S en $\mathcal{P}(S)$ por lo que $\#S \preceq \#\mathcal{P}(S)$. Veamos ahora que no existe funcion sobreyectiva de S en $\mathcal{P}(S)$.

Supongamos existe $g : S \to \mathcal{P}(S)$ sobreyectiva y definamos $B = \{x \in S / x \notin g(x)\} \subseteq S$ el conjunto de los elementos de S que no pertenecen a su imagen a traves de g. Como g es sobreyectiva y $B \in \mathcal{P}(S)$ sabemos que $\exists x \in S / g(x) = B$.

- Si $x \in B$: por definicion de B resulta $x \notin g(x) = B$. Contradiccion.
- Si $x \notin B$: por definicion de B resulta $x \in g(x) = B$. Contradiccion.

Por lo tanto g no es sobreyectiva.

2.2.8. Innumerabilidad del continuo

Enunciado El conjunto de los numeros reales no es numerable.

Demostracion Alzanca con probar que el intervalo $(0,1)$ no es numerable pues $(0,1) \sim \mathbb{R}$. En efecto $f(x) = tan\left(x\pi - \frac{\pi}{2}\right)$ o bien $g(x) = ln\left(\frac{1}{x} - 1\right)$ demuestran este hecho.

Representemos los elementos de $(0,1)$ por su expansion decimal infinita, por ejemplo $0,229384112598\ldots$. Supongamos que $(0,1)$ es numerable, habra entonces un primer elemento, segundo, etc. Listemoslos del siguiente modo:

$$\begin{array}{cccccc} 0,\boldsymbol{a_{11}} & a_{12} & a_{13} & a_{14} & a_{15} & \ldots \\ 0,a_{21} & \boldsymbol{a_{22}} & a_{23} & a_{24} & a_{25} & \ldots \\ 0,a_{31} & a_{32} & \boldsymbol{a_{33}} & a_{34} & a_{35} & \ldots \\ 0,a_{41} & a_{42} & a_{43} & \boldsymbol{a_{44}} & a_{45} & \ldots \\ 0,a_{51} & a_{52} & a_{53} & a_{54} & \boldsymbol{a_{55}} & \ldots \\ & & \vdots & & & \end{array}$$

Consideremos ahora el numero $b = 0,b_1b_2b_3b_4b_5\ldots$ donde cada digito b_i puede ser cualqueir digito excepto a_{ii} (es decir los numeros en negrita ubicados en la diagonal). Es claro que $b \in (0,1)$ pero es distinto a todos los numeros del listado ya que difiere de cada numero en por lo menos un digito. Esto constituye una contradiccion, luego el intervalo $(0,1) \sim \mathbb{R}$ no es numerable.

2.2.9. Cardinalidad del continuio

Enunciado La cardinalidad de $\mathbb{R}$ es igual a la de $\mathcal{P}(\mathbb{N})$.

Demostracion Veamos primero que $[0,1) \sim (0,1)$. En efecto $f : [0,1) \to (0,1)$ dada por $f(x) = \frac{1}{4} + \frac{1}{2}x$ es inyectiva y $g : (0,1) \to [0,1)$ definida por $g(x) = x$ tambien; luego por el teorema de Cantor-Schroder-Bernstein $[0,1) \sim (0,1)$.

Probaremos entonces que $\mathbb{R} \sim (0,1) \sim [0,1) \sim \mathcal{P}(\mathbb{N})$. Representemos los elementos de $[0,1)$ por su expansion decimal infinita: $0,b_1b_2b_3\ldots$.

Definimos $f : [0,1) \to \mathcal{P}(\mathbb{N})$ dada por $f\left(0,a_1a_2a_3\ldots\right) = \{10a_1, 10^2a_2, 10^3a_3, \ldots\}$. Por ejemplo $f\left(0,\overline{12}\right) = \{10, 200, 1000, 20000, 10000, \ldots\}$ o $f\left(0,05\right) = \{0, 500\}$.

Sean dos numeros *distintos* $b = 0,b_1b_2b_3\ldots$ y $c = 0,c_1c_2c_3\ldots$ luego $b_i \neq c_i$ para algun i. Claramente $b_i10^i \in f(b)$ pero $b_i10^i \notin f(c)$ por lo que $f(b) \neq f(c)$, es decir f es inyectiva.

Ahora definimos $g : \mathcal{P}(\mathbb{N}) \to [0,1)$ dada por $g(X) = 0, a_1 a_2 a_3 \ldots$ donde $a_i = 1$ si $i \in X$ y $a_i = 0$ si $i \notin X$. Por ejemplo $g(\{1,3\}) = 0,101\overline{0}$ o $g(\{2,4,6,8,\ldots\}) = 0,\overline{01}$. Notemos que $g(\emptyset) = 0$ y $g(\mathbb{N}) = 0,\overline{1}$.

Sean $X \neq Y$ entonces existe al menos un i que pertenece a uno de los conjuntos, pero no al otro y en consecuencia $g(X) \neq g(Y)$ pues difieren en el iesimo lugar decimal; es decir g es inyectiva.

Finalmente por el teorema de Cantor-Schroder-Bernstein $[0,1) \sim \mathcal{P}(\mathbb{N})$.

2.3. Ejemplos

2.3.1. Cadinalidad de $\mathbb{Z}$

Puesto que $f : \mathbb{N} \to \mathbb{Z}$ definida por $f(n) = n/2$ (si n es par) y $f(n) = (1-n)/2$ (si n es impar) es biyectiva, resulta $\mathbb{Z} \sim \mathbb{N}$.

En forma alternativa $\mathbb{Z} = \{\ldots, -2, -1\} \cup \{0\} \cup \{1, 2, \ldots\}$ es u. n. c. n.

2.3.2. Cadinalidad de $\mathbb{Q}$

Sea $f : \mathbb{Z} \times \mathbb{N} \to \mathbb{Q}$ dada por $f[(p,q)] = p/q$. Claramente f es sobreyectiva.

Como $\mathbb{N} \sim \mathbb{N} \times \mathbb{N} \sim \mathbb{Z} \times \mathbb{N}$ sabemos existe $g : \mathbb{N} \to \mathbb{Z} \times \mathbb{N}$ biyectiva, luego la funcion $f \circ g : \mathbb{N} \to \mathbb{Q}$ es sobreyectiva y por formulaciones equivalentes $\mathbb{Q}$ es numerable. Claramente $\mathbb{Q}$ no es finito, por lo que $\mathbb{Q} \sim \mathbb{N}$.

En forma alternativa, escribimos a $\mathbb{Q}$ como una u. n. c. n.: $\mathbb{Q} = \bigcup_{k \in \mathbb{N}} A_k$ con $A_k = \left\{\ldots, -\frac{2}{k}, -\frac{1}{k}, \frac{0}{k}, \frac{1}{k}, \frac{2}{k}, \ldots\right\}$.

2.3.3. Cantidad de funciones de X en $\{0,1\}$

Enunciado La cardinalidad del conjunto de funciones de X en $\{0,1\}$ es igual a la cardinalidad de $\mathcal{P}(X)$.

Definiciones Sean $B = \{0, 1\}$, $\mathcal{F}(A, B)$ el conjunto de todas las funciones de A en B y para cada $S \in P(A)$ consideremos la funcion:

$$\begin{aligned} \chi_S : A &\to B \\ x &\to \chi_S(x) = \begin{cases} 1 & x \in S \\ 0 & x \notin S \end{cases} \end{aligned}$$

Definimos:

- $$\begin{aligned} f : \mathcal{P}(A) &\to \mathcal{F}(A, B) \\ S &\to f(S) = \chi_S \end{aligned}$$
- $$\begin{aligned} g : \mathcal{F}(A, B) &\to \mathcal{P}(A) \\ F &\to g(F) = F^{-1}(\{1\}) \end{aligned}$$

  (donde $F^{-1}(\{1\})$ es el conjunto de todas las preimagenes de 1 a traves de F)

Demostracion Analicemos la funcion $\chi_{F^{-1}(\{1\})}$. Tenemos:

$$\chi_{F^{-1}(\{1\})}(x) = \begin{cases} 1 & x \in F^{-1}(\{1\}) \\ 0 & x \notin F^{-1}(\{1\}) \end{cases} = \begin{cases} 1 & F(x) = 1 \\ 0 & F(x) = 0 \end{cases} = F(x)$$

Ahora $(f \circ g)(F) = f[F^{-1}(\{1\})] = \chi_{F^{-1}(\{1\})} = F$ es decir: $(f \circ g)$ es la identidad en $\mathcal{F}(A, B)$. *(*)*

Sabemos que la preimagen de $\{1\}$ a traves de χ_S es justamente S. Por lo tanto:

$$\forall S \in \mathcal{P}(A) : (g \circ f)(S) = g[f(S)] = g(\chi_S) = \chi_S^{-1}(\{1\}) = S$$

por lo que $(g \circ f)$ es la identidad en $\mathcal{P}(A)$. *(**)*

De *(*)* y *(**)* resulta que f y g son biyectivas, es decir, existen la misma cantidad de subconjuntos de A que de funciones de A en B.

2.3.4. Ejercicios

Para los siguientes ejercicios consideraremos la funcion $\pi(n)$ que calcula el n-esimo numero primo. Es decir: $\pi(1) = 2, \pi(2) = 3, \pi(3) = 5, \ldots$

2.3.4.1. Pares ordenados

Enunciado Demuestre utilizando numeracion de Godel que $\mathbb{N} \sim \mathbb{N} \times \mathbb{N}$.

Solucion Sea $f : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ dada por $f(n) = (n, 1)$. Esta funcion es trivialmente inyectiva.

Sea $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ dada por $g(a, b) = 2^a 3^b$. El teorema fundamental de la aritmetica nos permite asegurar que esta funcion es inyectiva.

Luego por el teorema de Cantor-Schroder-Bernstein concluimos que $\mathbb{N} \sim \mathbb{N} \times \mathbb{N}$.

2.3.4.2. Funciones de dominio finito

Enunciado Demuestre utilizando numeracion de Godel que si $\#F = n \in \mathbb{N}$ entonces existe una cantidad numerable de funciones de F en $\mathbb{N}$.

Si $F = \{\oplus, \ominus, \otimes, \oslash\}$, ¿que numero le corresponde a la funcion h dada por $h(\oplus) = 6$, $h(\ominus) = 3$, $h(\otimes) = 3$ y $h(\oslash) = 2$?

Solucion Puesto que $F \sim [\![1, n]\!]$ sabemos que existe una funcion biyectiva $f : F \to [\![1, n]\!]$. Sea g una funcion de F en $\mathbb{N}$ podemos asignar a esta funcion el numero: $\prod_{i=1}^{n} \pi(i)^{g[f(i)]}$.

Como por cada funcion diferente obtenemos un numero distinto concluimos que $\#\{f / f : F \to \mathbb{N}\} \preceq \#\mathbb{N}$.

El numero correspondiente a h es: $2^6 3^3 5^3 7^2 = 10.584.000$.

2.3.4.3. Intervalos de extremos racionales

Enunciado Demuestre utilizando numeracion de Godel que el conjunto de todos los intervalos reales con extremos racionales tiene la menor cardinalidad infinita. ¿Que numero le corresponde al intervalo $\left(-\frac{7}{9}, \frac{11}{13}\right]$?

Solucion Sea I un intervalo real con extremos $\frac{p}{q} < \frac{r}{s}$ racionales, asignamos a este intervalo el numero:

$$2^{\chi_I(p/q)}3^{sgn(p/q)+1}5^{|p|}7^{|q|}11^{sgn(r/s)+1}13^{|r|}17^{|s|}19^{\chi_I(r/s)}$$

El numero asignado a $\left(-\frac{7}{9}, \frac{11}{13}\right]$ es $2^0 3^0 5^7 7^9 11^2 13^{11} 17^{13} 19^1$.

2.3.4.4. Imagenes monocromaticas

Enunciado Llamaremos imagen monocromatica de mn pixeles a un elemento de $\mathcal{M}_{m\times n}(\{\blacksquare, \square\})$. Demuestre utilizando numeracion de Godel que existe una cantidad numerable de imagenes monocromaticas. Dibuje la imagen 7776.

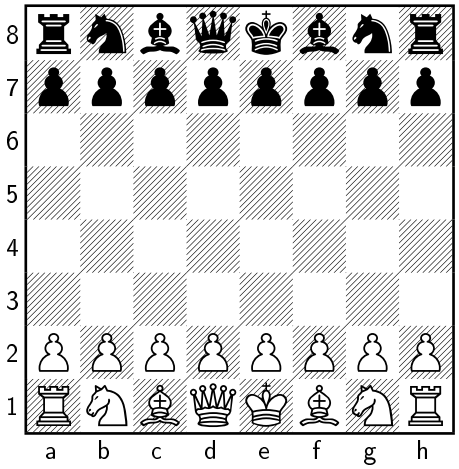
Solucion Sea $X \in \mathcal{M}_{m\times n}(\{\blacksquare, \square\})$ asigmamos a esta matriz el numero:

$$2^m 3^n \prod_{k=1}^{mn} \pi\,(k+2)^{f(x_{ij})}$$

donde $f(\blacksquare) = 0, f(\square) = 1$. La imagen 7776 corresponde a un cuadrado negro de 5 filas y 5 columnas.

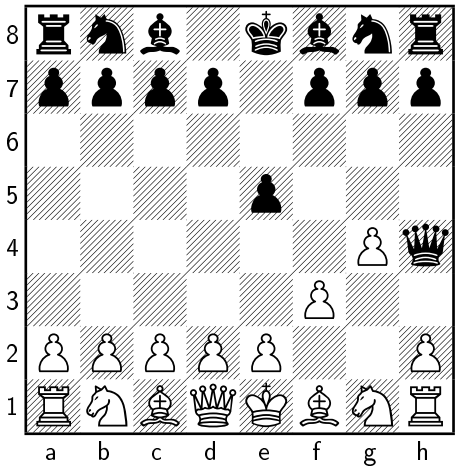
2.3.4.5. Partidas de ajedrez

Enunciado Observe la siguiente disposicion inicial de un tablero de ajedrez:



Sea $C = \{a1, a2, \ldots, a8, b1, \ldots, b8, \ldots, h8\}$ llamaremos *movimiento* a un elemento de $M = (C \times C) \cup \{(+,+)\}$. Definimos entonces una partida de ajedrez de n movimientos como un elemento de M^{2n} y llamaremos $\mathcal{P}$ al conjunto de todas las partidas (validas o invalidas) posibles.

Por ejemplo tras la partida $((f2, f3), (e7, e5), (g2, g4), (d8, h4))$ el tablero se encontraria en la siguiente situacion:



Demuestre mediante numeracion de Godel que existen una cantidad numerable de partidas (validas o invalidas) de ajedrez. ¿Que numero le corresponde a la anterior partida?

Solucion Numeremos las casillas del 1 al 64 (de izquierda a derecha, de arriba hacia abajo). Luego podemos identificar univocamente a una partida de ajedrez $X \in M^{2n}$ mediante el numero:

$$\prod_{i=0}^{2n-1} \pi\left(2i+1\right)^{f(X,2i+1)} \pi\left(2i+2\right)^{f(X,2i+2)}$$

donde f es la funcion que extrae de una partida, el numero correspondiente a un determinado movimiento.

Como a diferentes partidas le correspondes diferentes numeros resulta que $\mathcal{P} \preceq \mathbb{N}$.

A la partida anterior le corresponde el numero $2^{46}3^{38}5^{13}7^{29}11^{55}13^{39}17^{4}19^{40}$.

Parte II

Modelos de calculo

Capítulo 3

Funciones recursivas primitivas

3.1. Definiciones

3.1.1. Aridad

Sea $f : A_1 \times A_2 \times \ldots \times A_n \to B$ llamaremos aridad al numero de argumentos que toma la funcion, es decir n y notaremos $f^{(n)}$.

3.1.2. Funcion numerica

Llamaremos funcion numerica a toda funcion $f : \mathbb{N}^k \to \mathbb{N}$ con $k \in \mathbb{N}$. Si $k = 0$ identificaremos a dicha funcion con un numero perteneciente a $\mathbb{N}$.

3.1.3. Funciones base

Llamaremos funciones base a las siguientes tres funciones:

- La funcion cero $c^{(n)} : \mathbb{N}^n \to \mathbb{N}$ definida por $c^{(n)}(X) = 0$.
- Las funciones proyeccion $p_k^{(n)} : \mathbb{N}^n \to \mathbb{N}$ definidas por $p_k^{(n)}(x_1, x_2, \ldots, x_n) = x_k$.
- La funcion sucesor $s^{(1)} : \mathbb{N} \to \mathbb{N}$ definida por $s^{(1)}(x) = x + 1$.

3.1.4. Operadores

Definiremos dos operadores que nos permitiran construir nuevas funciones:

- El operador de composicion Φ que dada una funcion numerica $f^{(n)}$ y n funciones numericas de aridad k, construye la funcion numerica h definida como:

$$\begin{aligned} h : \mathbb{N}^k &\to \mathbb{N} \\ X^k &\to h\left(X^k\right) = f\left[g_1\left(X^k\right), g_2\left(X^k\right), \ldots, g_n\left(X^k\right)\right] \end{aligned}$$

  y que notaremos $h = \Phi\left(f, g_1, g_2, \ldots, g_n\right)$.

- El operador de recursion R que dadas dos funciones numericas $g^{(k)}$ y $h^{(k+2)}$ construye una nueva funcion numerica $f^{(k+1)}$ definida como

$$f\left(y, X^k\right) = \begin{cases} g\left(X^k\right) & y = 0 \\ h\left[y-1, X^k, f\left(y-1, X^k\right)\right] & y > 0 \end{cases}$$

  y notaremos $f = R(g, h)$.

3.1.5. Definicion inductiva

Definimos inductivamente el conjunto de funciones recursivas primitivas (FRP) como el menor conjunto tal que:

- Las funciones base pertenecen a FRP.
- Las funciones obtenidas aplicando un numero finito de operaciones de composicion y recursion sobre elementos de FRP tambien pertenecen a FRP.

3.2. Ejemplos

3.2.1. Predecesor natural

La funcion $\widehat{Pd}^{(1)}(y) = \begin{cases} 0 & y = 0 \\ y-1 & y > 0 \end{cases}$ es FRP pues:

1. $\widehat{Pd}^{(1)}(0) = 0 = c^{(0)}()$.
2. $\widehat{Pd}^{(1)}(y) = y - 1 = p_1^{(2)}\left[y-1, \widehat{Pd}^{(1)}(y-1)\right]$.

por lo que $\widehat{Pd}^{(1)} = R\left(c^{(0)}, p_1^{(2)}\right)$.

3.2.2. Suma

La funcion $\Sigma^{(2)}(y,x) = y + x$ es FRP pues:

1. $\Sigma^{(2)}(0,x) = 0 + x = x = p_1^{(1)}(x)$.
2. $$\begin{aligned}\Sigma^{(2)}(y,x) &= y + x = y + x + 1 - 1 = (y-1) + x + 1 = s^{(1)}\left[\Sigma^{(2)}(y-1,x)\right] = \\ &= s^{(1)}\left\{p_3^{(3)}\left[y-1, x, \Sigma^{(2)}(y-1,x)\right]\right\} = \Phi\left(s^{(1)}, p_3^{(3)}\right).\end{aligned}$$

y en consecuencia $\Sigma^{(2)} = R\left[p_1^{(1)}, \Phi\left(s^{(1)}, p_3^{(3)}\right)\right]$.

3.2.3. Diferencia natural

La funcion $\hat{d}^{(2)}(y,x) = \begin{cases} 0 & x < y \\ x - y & x \geq y \end{cases}$ que notaremos como $x \dot{-} y$ es FRP pues:

1. $\hat{d}^{(2)}(0,x) = x \dot{-} 0 = x = p_1^{(1)}(x)$.
2. $$\begin{aligned}\hat{d}^{(2)}(y,x) =& x \dot{-} y = x \dot{-} (y-1) - 1 = \widehat{Pd}^{(1)}\left[\hat{d}^{(2)}(y-1,x)\right] = \\ =& \widehat{Pd}^{(1)}\left\{p_3\left[y-1, x, \hat{d}^{(2)}(y-1,x)\right]\right\} = \\ =& \Phi\left(\widehat{Pd}^{(1)}, p_3^{(3)}\right).\end{aligned}$$

luego $\hat{d}^{(2)} = R\left[p_1^{(1)}, \Phi\left(\widehat{Pd}^{(1)}, p_3^{(3)}\right)\right]$.

3.2.4. Producto

La funcion $\Pi^{(2)}(y,x) = yx$ es FRP pues:

1. $\Pi^{(2)}(0,x) = 0x = 0 = c^{(1)}(x)$.

2. $$\begin{aligned}\Pi^{(2)}(y,x) &= yx = (y-1)x + x = \Sigma^{(2)}\left[x, \Pi(y-1,x)\right] = \\ &= \Sigma^{(2)}\left\{p_2^{(3)}\left[y-1, x, \Pi^{(2)}(y-1,x)\right], p_3^{(3)}\left[y-1, x, \Pi^{(2)}(y-1,x)\right]\right\} = \\ &= \Phi\left(\Sigma^{(2)}, p_2^{(3)}, p_3^{(3)}\right).\end{aligned}$$

entonces $\Pi^{(2)} = R\left[c^{(1)}, \Phi\left(\Sigma^{(2)}, p_2^{(3)}, p_3^{(3)}\right)\right]$.

3.2.5. Factorial

La funcion $Fac^{(1)}(y) = y!$ es FRP pues:

1. $Fac^{(1)}(0) = 0! = 1 = 0 + 1 = s^{(1)}\left[c^{(0)}()\right] = \Phi\left(s^{(1)}, c^{(0)}\right)$.

2. $$\begin{aligned}Fac^{(1)}(y) &= y! = y(y-1)! = [y-1]!\,[(y-1)+1] = \\ &= \Pi^{(2)}\left\{p_2^{(2)}\left[y-1, Fac^{(1)}(y-1)\right], \Phi\left(s^{(1)}, p_1^{(2)}\left[y-1, Fac^{(1)}(y-1)\right]\right)\right\} = \\ &= \Phi\left[\Pi^{(2)}, p_2^{(2)}, \Phi\left(s^{(1)}, p_1^{(2)}\right)\right].\end{aligned}$$

resultando $Fac^{(1)} = R\left\{\Phi\left(s^{(1)}, c^{(0)}\right), \Phi\left[\Pi^{(2)}, p_2^{(2)}, \Phi\left(s^{(1)}, p_1^{(2)}\right)\right]\right\}$.

3.2.6. Exponenciacion total

La funcion $\widehat{Exp}^{(2)}(y,x) = 1$ si $x = y = 0$ o x^y en caso contrario, es FRP pues:

1. $\widehat{Exp}^{(2)}(0,x) = x^0 = 1 = 0 + 1 = s^{(1)}\left[c^{(1)}(x)\right] = \Phi\left(s^{(1)}, c^{(1)}\right)$.

2. $$\begin{aligned}\widehat{Exp}^{(2)}(y,x) &= x^y = x^{y-1}x = \Pi^{(2)}\left[x, \widehat{Exp}^{(2)}(y-1,x)\right] = \\ &= \Pi^{(2)}\left\{p_2^{(3)}\left[y-1, x, \widehat{Exp}^{(2)}(y-1,x)\right], p_3^{(3)}\left[y-1, x, \widehat{Exp}^{(2)}(y-1,x)\right]\right\} = \\ &= \Phi\left(\Pi^{(2)}, p_2^{(3)}, p_3^{(3)}\right).\end{aligned}$$

por lo que $\widehat{Exp}^{(2)} = R\left[\Phi\left(s^{(1)}, c^{(1)}\right), \Phi\left(\Pi^{(2)}, p_2^{(3)}, p_3^{(3)}\right)\right]$.

3.2.7. Distinguidora del cero

La funcion $D_0^{(1)}(y) = \begin{cases} 0 & y \neq 0 \\ 1 & y = 0 \end{cases}$ es FRP pues:

1. $D_0^{(1)}(0) = 1 = 0 + 1 = s^{(1)}\left[c^{(0)}()\right] = \Phi\left(s^{(1)}, c^{(0)}\right)$.
2. $D_0^{(1)}(y) = 0 = c^{(2)}\left[y-1, D_0^{(1)}(y-1)\right]$.

y en consecuencia $D_0^{(1)} = R\left[\Phi\left(s^{(1)}, c^{(0)}\right), c^{(2)}\right]$.

En forma alternativa $D_0^{(1)}(y) = \widehat{Exp}(0, y) = \Phi\left(\widehat{Exp}^{(2)}, c^{(1)}, p_1^{(1)}\right)$.

3.2.8. Signo

La funcion $Sgn^{(1)}(y) = \begin{cases} 1 & y \neq 0 \\ 0 & y = 0 \end{cases}$ es FRP pues:

1. $Sgn^{(1)}(0) = 0 = c^{(0)}()$.
2. $Sgn^{(1)}(y) = 1 = \Phi\left(s^{(1)}, c^{(2)}\right)$.

por lo que $Sgn^{(1)} = R\left[c^{(0)}, \Phi\left(s^{(1)}, c^{(2)}\right)\right]$.

Tambien podemos definirla como $Sgn^{(1)}(y) = D_0^{(1)}\left[D_0^{(1)}(y)\right] = \Phi\left(D_0^{(1)}, D_0^{(1)}\right)$.

3.2.9. Maximo

La funcion $Max^{(2)}(y, x) = \begin{cases} x & x \geq y \\ y & y \geq x \end{cases}$ es FRP pues:

$$Max^{(2)}(y,x) \underbrace{=}_{(*)} \left(x \dot{-} y\right) + y = \hat{d}^{(2)}(y,x) + y = \Phi\left(\Sigma^{(2)}, d^{(2)}, p_1^{(2)}\right)$$

Justificamos $(*)$ ya que:

- $x > y \Rightarrow \left(x \dot{-} y\right) + y = x - y + y = x$.
- $x < y \Rightarrow \left(x \dot{-} y\right) + y = 0 + y = y$.
- $x = y \Rightarrow \left(x \dot{-} y\right) + y = 0 + y = y = x$.

3.2.10. Distancia

La funcion $k^{(2)}(x,y) = |x - y|$ es FRP pues:

$$k^{(2)}(x,y) = \begin{cases} x \dot{-} y & x > y \\ y \dot{-} x & x \leq y \end{cases} = \left(x \dot{-} y\right) + \left(y \dot{-} x\right) = \hat{d}^{(2)}\left(p_2^{(2)}, p_1^{(2)}\right) + \hat{d}^{(2)}\left(p_1^{(2)}, p_2^{(2)}\right) =$$
$$= \Phi\left(\hat{d}^{(2)}, p_2^{(2)}, p_1^{(2)}\right) + \Phi\left(\hat{d}^{(2)}, p_1^{(2)}, p_2^{(2)}\right) =$$
$$= \Phi\left[\Sigma^{(2)}, \Phi\left(\hat{d}^{(2)}, p_2^{(2)}, p_1^{(2)}\right), \Phi\left(\hat{d}^{(2)}, p_1^{(2)}, p_2^{(2)}\right)\right].$$

3.2.11. Equivalencia

La funcion $E^{(2)}(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$ es FRP pues:

$$E^{(2)}(x,y) = D_0^{(1)}\left[k^{(2)}(x,y)\right] = \Phi\left(D_0^{(1)}, k^{(2)}\right)$$

3.2.12. Inequivalencia

La funcion $\neg E(x,y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$ es FRP pues:

$$\neg E(x,y) = D_0^{(1)}\left[E^{(2)}(x,y)\right] = \Phi\left(D_0^{(1)}, E^{(2)}\right)$$

3.2.13. Menor o igual

La funcion $Leq^{(2)}(y,x) = \begin{cases} 1 & x \leq y \\ 0 & x > y \end{cases}$ es FRP pues:

$$Leq^{(2)}(y,x) = D_0^{(1)}\left[\hat{d}^{(2)}(y,x)\right] = \Phi\left(D_0^{(1)}, \hat{d}^{(2)}\right).$$

3.2.14. Mayor

La funcion $Mayor^{(2)}(x,y) = \begin{cases} 1 & x > y \\ 0 & x \leq y \end{cases}$ es FRP pues:

$$Mayor^{(2)}(y,x) = \Phi\left(D_0^{(1)}, Leq^{(2)}\right)$$

3.2.15. Funcion potencia

Dada una funcion $f^{(1)}$ definimos $F^{(2)}$ llamada potencia de f como:

$$F(y,x) = \begin{cases} x & y = 0 \\ f\left[F(y-1,x)\right] & y > 0 \end{cases}$$

y notaremos $F(y,x) = f^y(x)$.
La funcion $f^y(x)$ es FRP pues:

1. $F^{(2)}(0,x) = x = p_1^{(1)}(x)$.
2. $F^{(2)}(y,x) = f^{(1)}\left[F(y-1,x)\right] = f^{(1)}\left\{p_3^{(3)}\left[y-1,x,F(y-1,x)\right]\right\} = \Phi\left(f^{(1)}, p_3^{(3)}\right)$.

entonces $F^{(2)} = R\left[p_1^{(1)}, \Phi\left(f^{(1)}, p_3^{(3)}\right)\right]$.

3.2.16. Sumatoria

Sea $f^{(2)} \in FRP$, la funcion $F^{(2)}(y,x) = \sum_{z=0}^{y} f(z,x)$ es FRP pues:

1. $F^{(2)}(0,x) = \sum_{z=0}^{0} f(z,x) = f^{(2)}(0,x) = \Phi\left(f^{(2)}, c^{(1)}, p_1^{(1)}\right)$.
2. $$\begin{aligned} F^{(2)}(y,x) =& \sum_{z=0}^{y} f(z,x) = \underbrace{\sum_{z=0}^{y-1} f(z,x)}_{p_3^{(3)}} + f^{(2)}(y,x) = \\ =& \Phi\left\{\Sigma^{(2)}, p_3^{(3)}, \Phi\left[f^{(2)}, \Phi\left(s^{(1)}, p_1^{(3)}\right), p_2^{(3)}\right]\right\}. \end{aligned}$$

resultando $F^{(2)}(y,x) = R\left(\Phi\left[f^{(2)}, c^{(1)}, p_1^{(1)}\right], \Phi\left\{\Sigma^{(2)}, p_3^{(3)}, \Phi\left[f^{(2)}, \Phi\left(s^{(1)}, p_1^{(3)}\right), p_2^{(3)}\right]\right\}\right)$.

3.3. Conjuntos

3.3.1. Conjunto recursivo primitivo

Diremos $A \subseteq \mathbb{N}^k$ es un conjunto recursivo primitivo (CRP) si su funcion caracteristica $\chi_A : \mathbb{N}^k \to \{0,1\}$ es FRP.

3.3.2. Relaciones recursivas primitivas

Una relacion $R \subseteq \mathbb{N} \times \mathbb{N}$ se dice recursiva primitiva (RRP) si es un CRP.

3.4. Teoremas

3.4.1. Conjuntos unitarios

Enunciado Todo subconjunto unitario de $\mathbb{N}^k$ es CRP.

Demostracion Lo haremos por induccion en k. Sea $P(k)$: «todo subconjunto unitario de $\mathbb{N}^k$ es CRP».

1. Veamos que es valido para $k = 1$, es decir que todo subconjutno unitario de $\mathbb{N}$ es CRP. Sean $A = \{n\}$ y $f_n^{(1)} = n$ entonces:

$$\chi_A^{(1)}(y) = \begin{cases} 1 & y \in A \\ 0 & y \notin A \end{cases} = \begin{cases} 1 & y = n \\ 0 & y \neq n \end{cases}$$

   es decir $\chi_A^{(1)}(y) = E^{(2)}\left[p_1^{(1)}(y), f_n^{(1)}(y)\right] = \Phi\left(E^{(2)}, p_1^{(1)}, f_n^{(1)}\right)$.

2. Supongamos que es valido para $k = m$, es decir que todo subconjutno unitario de $\mathbb{N}^m$ es CRP. Queremos ver si todo subconjunto unitario de $\mathbb{N}^{m+1}$ es CRP.
   Sean $B = \{(a_1, a_2, \ldots, a_m, n)\}$ y $A = \{(a_1, a_2, \ldots, a_m)\}$. Por hipotesis inductiva $\chi_A^{(m)}$ es FRP. Luego:

$$\chi_B^{(m+1)}[(y_1, y_2, \ldots, y_m, y_{m+1})] = \Pi\left\{\chi_A\left[\left(p_1^{(m+1)}, p_2^{(m+1)}, \ldots, p_m^{(m+1)}\right)\right], E\left[p_{m+1}^{(m+1)}, f_n^{(m+1)}\right]\right\}$$

3.4.2. Operaciones sobre CRP

Enunciado Sean $k \in \mathbb{N}$ y $A, B \subseteq \mathbb{N}^k$ entonces si A y B son CRP el complemento $\overline{A}$, la interseccion $A \cap B$ y la union $A \cup B$ son CRP.

Demostracion Como A, B son CRP sus funciones caracteristicas χ_A y χ_B son FRP. Luego:

- $\chi_{\overline{A}} = \Phi\left(D_0, \chi_A\right)$.
- $\chi_{A \cap B} = \Phi\left(\Pi, \chi_A, \chi_B\right)$.
- $\chi_{A \cup B} = \Phi\left[sgn^{(1)}, \Phi\left(\Sigma, \chi_A, \chi_B\right)\right]$.

3.4.3. Conjuntos finitos

Enunciado Todo subconjunto finito de $\mathbb{N}^k$ es CRP.

Demostracion Queda como ejercicio al lector, completar esta demostracion por induccion, valiendose del teorema anterior para la union.

Capítulo 4

Funciones recursivas

4.1. Introduccion

4.1.1. Funcion parcial y funcion total

Una funcion numerica $f : \mathbb{N}^k \to \mathbb{N}$ se dice *parcia*l si no esta definida sobre todos los elementos de $\mathbb{N}^k$. Si esta definida para todos los elementos de $\mathbb{N}^k$ se dice *total*.

4.1.2. Funcion mayora

Decimos que una funcion $f^{(1)}$ mayora a otra funcion $g^{(n)}$ si $\forall\,(x_1, x_2, \ldots, x_n) \in dom\,(g)$ se verifica que $g\,(x_1, x_2, \ldots, x_n) \leq f\,[max\,(x_1, x_2, \ldots, x_n)]$ y lo notamos $f^{(1)} \uparrow g^{(n)}$.

4.1.3. Serie de Ackermann

Consideremos la siguiente sucesion de funciones que llamaremos sucesion de Ackermann:

- $f_0(x) = s(x) = x + 1$.
- $f_1(x) = f_0^{x+2}(x) = s^{x+2}(x) = x + (x + 2) = 2x + 2$.
- $f_2(x) = f_1^{x+2}(x)$.
- $\vdots$
- $f_k(x) = f_{k-1}^{x+2}(x)$.
- $\vdots$

Calculemos algunos valores de $f_2(x)$:

- $f_2(0) = f_1^2(0) = f_1[f_1(0)] = f_1[f_0^2(0)] = f_1[2] = f_0^4(2) = 6$.
- $f_2(1) = f_1^3(1) = f_1\{f_1[f_1(1)]\} = f_1\left\{f_1\left[f_0^3(1)\right]\right\} = f_1\{f_1[4]\} = f_1\left\{f_0^6[4]\right\} = f_1\{10\} = f_0^{12}\{10\} = 22$.
- $f_2(2) = f_1^4(2) = 2\{2[2(2\cdot 2+2)+2]+2\}+2 = 62$.
- $f_2(3) = 158$.

4.1.4. Funcion de Ackermann

Definimos una funcion que llamaremos ACK de la siguiente manera: $ACK(x) = f_x(x)$. O sea para encontrar su valor imagen para un determinado valor x, tomamos la x-esima función de Ackerman y la calculamos en dicho valor. Por ejemplo:

- $ACK(0) = f_0(0) = 1$.
- $ACK(1) = f_1(1) = 4$.
- $ACK(2) = f_2(2) = 62$.

$$
\begin{aligned}
ACK(3) =& f_3(3) = f_2^5(3) = f_2[f_2(f_2\{f_2[f_2(3)]\})] = f_2[f_2(f_2\{f_2[158]\})] \\
=& f_2\left[f_2\left(f_2\left\{f_1^{160}[158]\right\}\right)\right] = f_2\left[f_2\left(f_2\left\{f_1^{159}[318]\right\}\right)\right] = \\
=& f_2\left[f_2\left(f_2\left\{f_1^{158}[638]\right\}\right)\right] = f_2\left[f_2\left(f_2\left\{f_1^{157}[1278]\right\}\right)\right] = \cdots = \\
=& f_2\left[f_2\left(f_2\left\{f_1^{150}[163.838]\right\}\right)\right] = \cdots = f_2\left[f_2\left(f_2\left\{f_1^{140}[167.772.158]\right\}\right)\right] = \cdots \\
=& f_2\left[f_2\left(f_2\left\{f_1^{137}[1.342.177.278]\right\}\right)\right] = \cdots
\end{aligned}
$$

4.2. Teoremas

4.2.1. Totalidad de las FRP

Enunciado Si $f \in FRP$ entonces f es una funcion total.

Demostracion Lo demostraremos por induccion sobre el conjunto FRP.

- Caso base: Las funciones bases son totales por definicion.
- Composicion: Supongamos que $f^{(n)}, g_1^{(k)}, \ldots, g_n^{(k)}$ son totales. Veremos que $h = \Phi\left(f^{(n)}, g_1^{(k)}, \ldots, g_n^{(k)}\right)$ tambien lo es.
  Sea $X \in \mathbb{N}^k$ podemos calcular $Y = (g_1[X], \ldots, g_n[X])$ puesto que cada g_i es total por hipotesis inductiva. Ademas f tambien es total por lo que podemos calcular $f(Y)$.
  Es decir, existe un numero natural $z = f(Y) = h(X)$.
- Recursion: Supongamos que $g^{(k)}, h^{(k+2)}$ son totales. Veremos que $f\left(y, X^k\right) = R(g,h)$ tambien lo es, por induccion en y.
  1. Caso base $y = 0$: $f(0, X^k) = g\left(X^k\right)$ que es total por hipotesis.
  2. Caso inductivo $y = p$: Supongamos $f\left(p, X^k\right)$ es total, luego:

$$
f\left(p+1, X^k\right) = h\left[p, X^k, \underbrace{f\left(p, X^k\right)}_{\text{total por HI}}\right]
$$

  y como h es total resulta que f es total.

4.2.2. Propiedades de Ackermann

Enunciado

1. $\forall k \in \mathbb{N} \Rightarrow f_k \in FRP$.
2. $\forall x, k \in \mathbb{N} \Rightarrow f_k(x) > x$.
3. $\forall x_1, x_2, k \in \mathbb{N}$, si $x_1 < x_2$ entonces $f_k(x_1) < f_k(x_2)$.
4. $\forall x, k \in \mathbb{N} \Rightarrow f_k(x) < f_{k+1}(x)$.

Demostracion

1. Lo demostraremos por induccion en k:
   a) Caso base $k = 0$: $f_0(x) = s(x)$.
   b) Caso inductivo $k = h$: Supongamos que f_h es FRP. Queremos ver si f_{h+1} tambien lo es. En efecto $f_{h+1}(x) = f_h^{x+2}(x) = f_h^{s[s(x)]}(x)$.
2. Consultar bibliografia [3], pag. 56.
3. Consultar bibliografia [3], pag. 56.
4. $f_{k+1}(x) = f_k^{x+2}(x) = f_k^{x+1}[f_k(x)] \overbrace{>}^{(2)(3)} f_k^{x+1}(x) > \ldots > f_k(x)$

4.2.3. Mayorabilidad de las FRP

Enunciado Sea $g^{(n)} \in FRP$ entonces existe f_k de la serie de Ackermann tal que $f_k \uparrow g$.

Demostracion Lo demostraremos por induccion:

1. Caso base: Todas las funciones bases son mayoradas por f_0.
2. Caso inductivo:

a) Composicion: Sean las funciones $C^{(m)}, h_1^{(n)}, \ldots, h_m^{(n)}$ tales que f_k mayora a todas ellas, definimos $g^{(n)} = \Phi\left(C^{(m)}, h_1^{(n)}, \ldots h_m^{(n)}\right)$. Veremos que $f_{k+1} \uparrow g^{(n)}$.
Sabemos que $h_i^{(n)}(X) \leq f_k[max(X)]$ (por ser mayorada por f_k) luego $max\left\{h_1^{(n)}(X), \ldots, h_m^{(n)}(X)\right\} \leq f_k[max(X)]$ *(*)*.
Ademas como $f_k \uparrow C^{(m)}$ resulta:

$$g(X) = C\left[h_1^{(n)}(X), \ldots, h_m^{(n)}(X)\right] \leq f_k\left[max\left(h_1^{(n)}(X), \ldots, h_m^{(n)}(X)\right)\right]$$

y por *(*)* y propiedad *(3)* tenemos

$$f_k\left[max\left(h_1^{(n)}(X), \ldots, h_m^{(n)}(X)\right)\right] \leq f_k[f_k(max[X])]$$

ahora aplicamos varias veces las propiedades *(2)* y *(3)* obteniendo:

$$g(X) \leq f_k[f_k(max[X])] \leq f_k^{max(X)+2}[max(X)] = f_{k+1}[max(X)].$$

b) Recursion: Sea $g^{(n+1)} = R\left[B^{(n)}, h^{(n+2)}\right]$, veremos que $f_k \uparrow B \wedge f_k \uparrow h \Rightarrow f_{k+1} \uparrow g^{(n)}$.
Por hipotesis $g(0, X) = B(X) \leq f_k[max(X)]$ *(**)* y ademas:

$$g(1, X) = h[0, X, g(0, X)] \leq f_k[max(0, X, g[0, X])]$$

luego aplicando *(**)* y propiedad *(3)*

$$f_k[max(0, X, g[0, X])] \leq f_k[max(0, X, f_k[max(X)])]$$

y puesto que $f_k[max(X)] > max(X, 0)\ \forall k$ resulta

$$g(1, X) \leq f_k[max(0, X, f_k[max(X)])] \leq f_k[f_k(max[X])]$$

Puede demostrarse por induccion que $g(y, X) \leq f_k^{(y+1)}(max[X])$.
Finalmente:

$$f_k^{(y+1)}(max[X]) \leq f_k^{(y+1)}(max[y, X]) \leq f_k^{max(y,X)+1}(max[y, X])$$

y como $f_k^{max(y,X)+1}(max[y, X]) \leq f_k^{max(y,X)+2}(max[y, X]) = f_{k+1}[max(y, X)]$ resulta $g(y, X) \leq f_{k+1}[max(y, X)]$.

4.2.4. No primitividad de ACK

Enunciado La funcion $ACK(x)$ no es FRP.

Demostracion Supongamos que $ACK(x) \in FRP$. Luego $ACK(x)+1 \in FRP$. Por el teorema de mayorabilidad existe f_m en la serie de Ackermann que la mayora, es decir: $\forall x \in \mathbb{N}$ resulta:

$$ACK(x)+1 \leq f_m(x) \iff f_x(x)+1 \leq f_m(x)$$

y tomando $x = m$ obtenemos $f_m(m)+1 \leq f_m(m)$. ¡Absurdo! Por lo tanto $ACK(x) \notin FRP$.

4.3. Definiciones

4.3.1. Operador de minimizacion

Dada $h^{(n+1)}$, decimos que $g^{(n)}$ se construye por *minimizacion* de h (y lo notaremos $g = M[h]$) cuando g se define del modo siguiente:

$$g(X) = M[h](X) = \mu_t[h(t, X) = 0]$$

donde $\mu_t[h(t, X) = 0]$ es, si existe, el minimo valor de t tal que $h(t, X) = 0$.

4.3.2. Definicion inductiva

Definimos inductivamente el conjunto de Funciones Recursivas (FR) como el menor conjunto tal que:

- Las funciones base pertenecen a FR.
- Las funciones obtenidas aplicando un numero finito de operaciones de composicion, recursion y minimizacion sobre elementos de FR tambien pertenecen a FR.

4.4. Ejemplos

4.4.1. Predecesor

La funcion numerica $Pd(x) = x - 1$ solo esta definida si existe t tal que $t = x - 1 \iff t + 1 = x$. Buscamos entonces $Pd(x) = \mu_t [h(t, x) = 0]$. Sea entonces $h(t, x) = \neg E [s(t), x]$. Veamos algunos ejemplos:

- $Pd(2) = \mu_t [h(t, 2) = 0]$:
  - $t = 0$: $h(0, 2) = \neg E [s(0), 2] = \neg E [1, 2] = 1 \neq 0$.
  - $\boxed{t = \mathbf{1} : h(1, 2) = \neg E [s(1), 2] = \neg E [2, 2] = 0.}$
- $Pd(0) = \mu_t [h(t, 0) = 0]$:
  - $t = 0$: $h(0, 0) = \neg E [s(0), 0] = \neg E [1, 0] = 1 \neq 0$.
  - $t = 1$: $h(1, 0) = \neg E [s(1), 0] = \neg E [2, 0] = 1 \neq 0$.
  - $\vdots$

4.4.2. Diferencia

La funcion numerica $d(x, y) = x - y$ solo esta definida si existe t tal que $t = x - y \iff t + y = x$. Buscamos entonces $d(x, y) = \mu_t [h(t, x, y) = 0]$. Sea entonces $h(t, x) = \neg E [\Sigma(t, y), x]$. Veamos algunos ejemplos:

- $d(3, 1) = \mu_t [h(t, 3, 1) = 0]$:
  - $t = 0$: $h(0, 3, 1) = \neg E [0 + 1, 3] = \neg E [1, 3] = 1 \neq 0$.
  - $t = 1$: $h(1, 3, 1) = \neg E [1 + 1, 3] = \neg E [2, 3] = 1 \neq 0$.
  - $\boxed{t = \mathbf{2} : h(2, 3, 1) = \neg E [2 + 1, 3] = \neg E [3, 3] = 0.}$
- $d(1, 3) = \mu_t [h(t, 1, 3) = 0]$:
  - $t = 0$: $h(0, 1, 3) = \neg E [0 + 3, 1] = \neg E [3, 1] = 1 \neq 0$.
  - $t = 1$: $h(1, 1, 3) = \neg E [1 + 3, 1] = \neg E [4, 1] = 1 \neq 0$.
  - $\vdots$
- $d(0, 0) = \mu_t [h(t, 0, 0) = 0]$:
  - $\boxed{t = \mathbf{0} : h(0, 0, 0) = \neg E [0 + 0, 0] = \neg E [0, 0] = 0.}$

4.4.3. Division

La funcion numerica $div(x,y) = x/y$ solo esta definida si existe t tal que $ty = x$. Buscamos entonces $div(x,y) = \mu_t[h(t,x,y) = 0]$. Sea entonces $h(t,x,y) = \neg E[\Pi(t,y), x]$. Veamos algunos ejemplos:

- $div(25,5) = \mu_t[h(t,25,5) = 0]$.
  - $t = 0$: $h(0,25,5) = \neg E[\Pi(0,5), 25] = \neg E[0,25] = 1 \neq 0$.
  - $t = 1$: $h(1,25,5) = \neg E[\Pi(1,5), 25] = \neg E[5,25] = 1 \neq 0$.
  - $t = 2$: $h(2,25,5) = \neg E[\Pi(2,5), 25] = \neg E[10,25] = 1 \neq 0$.
  - $t = 3$: $h(3,25,5) = \neg E[\Pi(3,5), 25] = \neg E[15,25] = 1 \neq 0$.
  - $t = 4$: $h(4,25,5) = \neg E[\Pi(4,5), 25] = \neg E[20,25] = 1 \neq 0$.
  - $\boxed{t = \mathbf{5} : h(5,25,5) = \neg E[\Pi(5,5), 25] = \neg E[25,25] = 0.}$
- $div(4,3) = \mu_t[h(t,4,3) = 0]$.
  - $t = 0 : h(0,4,3) = \neg E[\Pi(0,3), 4] = \neg E[0,4] = 1 \neq 0$.
  - $t = 1 : h(1,4,3) = \neg E[\Pi(1,3), 4] = \neg E[3,4] = 1 \neq 0$.
  - $t = 2 : h(2,4,3) = \neg E[\Pi(2,3), 4] = \neg E[6,4] = 1 \neq 0$.
  - $t = 3 : h(3,4,3) = \neg E[\Pi(3,3), 4] = \neg E[9,4] = 1 \neq 0$.
  - $\vdots$
- $div(2,0) = \mu_t[h(t,2,0) = 0]$.
  - $t = 0 : h(0,2,0) = \neg E[\Pi(0,0), 2] = \neg E[0,2] = 1 \neq 0$.
  - $t = 1 : h(1,2,0) = \neg E[\Pi(1,0), 2] = \neg E[0,2] = 1 \neq 0$.
  - $\vdots$
- $div(0,0) = \mu_t[h(t,0,0) = 0]$.
  - $t = 0 : h(0,0,0) = \neg E[\Pi(0,0), 0] = \neg E[0,0] = 0$. ERROR.

Nuestra funcion h falla en el caso extremo $0/0$ pero podemos arreglarlo si la redefinimos como: $h(t,x,y) = \neg E\{\Pi[t,y] + D_0[y], x\}$.

Observemos que el termino que agregamos $D_0[y]$ da siempre 0 salvo cuando $y = 0$ en cuyo caso garantiazamos que $h(t,x,0) > 1$ con lo que la funcion no se detiene. Veamos que pasa ahora:

- $div\,(0,0) = \mu_t\,[h\,(t,0,0) = 0]$.
  - $t = 0 : h\,(0,0,0) = \neg E\,\{\Pi\,[0,0] + D_0\,[0+0]\,,0\} = \neg E\,\{0+1,0\} = \neg E\,\{1,0\} = 1.$
  - $t = 1 : h\,(0,0,0) = \neg E\,\{\Pi\,[0,0] + D_0\,[0+0]\,,0\} = \neg E\,\{0+1,0\} = \neg E\,\{1,0\} = 1.$
  - $\vdots$

Alternativamente podriamos usar esta otra definicion: $h\,(t,x,y) = d\,[x,\Pi\,(t,y)] + D_0\,(y)$.

4.4.4. Logaritmo

La funcion numerica $Log\,(x,y) = log_x y$ solo esta definida si existe t tal que $x^t = y$. Buscamos entonces $Log\,(x,y) = \mu_t\,[h\,(t,x,y) = 0]$. Sea entonces $h\,(t,x,y) = \neg E\left\{\widehat{Exp}\,[x,t] + D_0\,[x] + D_1\,[x]\,,y\right\}$.

Notense los terminos $D_0\,[x]$ y $D_1\,[x]$ que logran «indefinir» la funcion para las bases correspondientes.

Veamos algunos ejemplos:

- $Log\,(0,0) = \mu_t\,[h\,(t,0,0) = 0]$:
  - $t = 0$: $h\,(0,0,0) = \neg E\left\{\widehat{Exp}\,(0,0) + 1 + 0,0\right\} = \neg E\,\{1+1+0,0\} = 1.$
  - $t = 1$: $h\,(1,0,0) = \neg E\left\{\widehat{Exp}\,(0,1) + 1 + 0,0\right\} = \neg E\,\{0+1+0,0\} = 1.$
  - $\vdots$
- $Log\,(1,0) = \mu_t\,[h\,(t,1,0) = 0]$:
  - $t = 0$: $h\,(0,1,0) = \neg E\left\{\widehat{Exp}\,(1,0) + 0 + 1,0\right\} = \neg E\,\{1+0+1,0\} = 1.$
  - $t = 1$: $h\,(1,1,0) = \neg E\left\{\widehat{Exp}\,(1,1) + 0 + 1,0\right\} = \neg E\,\{1+0+1,0\} = 1.$
  - $\vdots$

- $Log\,(10,100) = \mu_t\,[h\,(t,10,100) = 0]$:
  - $t = 0$: $h\,(0,10,100) = \neg E\left\{\widehat{Exp}\,(10,0) + 0 + 0, 100\right\} = \neg E\,\{1 + 0 + 0, 100\} = 1.$
  - $t = 1$: $h\,(1,10,100) = \neg E\left\{\widehat{Exp}\,(10,1) + 0 + 0, 100\right\} = \neg E\,\{10 + 0 + 0, 100\} = 1.$
  - $\boxed{t = \mathbf{2} : h\,(2,10,100) = \neg E\left\{\widehat{Exp}\,(10,2) + 0 + 0, 100\right\} = \neg E\,\{100 + 0, 100\} = 0.}$
- $Log\,(2,3) = \mu_t\,[h\,(t,2,3) = 0]$:
  - $t = 0$: $h\,(0,2,3) = \neg E\left\{\widehat{Exp}\,(2,0) + 0 + 0, 3\right\} = \neg E\,\{1 + 0 + 0, 3\} = 1.$
  - $t = 1$: $h\,(1,2,3) = \neg E\left\{\widehat{Exp}\,(2,1) + 0 + 0, 3\right\} = \neg E\,\{2 + 0 + 0, 3\} = 1.$
  - $t = 2$: $h\,(2,2,3) = \neg E\left\{\widehat{Exp}\,(2,2) + 0 + 0, 3\right\} = \neg E\,\{4 + 0 + 0, 3\} = 1.$
  - $\vdots$

4.4.5. Raiz cuadrada

La funcion numerica $Sqrt\,(x) = \sqrt{x}$ solo esta definida si existe t tal que $t = \sqrt{x} \iff t^2 = x$. Buscamos entonces $Sqrt\,(x) = \mu_t\,[h\,(t,x) = 0]$. Sea entonces $h\,(t,x,y) = \neg E\,\{\Pi\,[t,t]\,, x\}$.

Veamos algunos ejemplos:

- $Sqrt\,(4) = \mu_t\,[h\,(t,4) = 0]$:
  - $t = 0$: $h\,(0,4) = \neg E\,\{\Pi\,(0,0)\,, 4\} = \neg E\,\{0,4\} = 1.$
  - $t = 1$: $h\,(1,4) = \neg E\,\{\Pi\,(1,1)\,, 4\} = \neg E\,\{1,4\} = 1.$
  - $\boxed{t = \mathbf{2} : h\,(2,4) = \neg E\,\{\Pi\,(2,2)\,, 4\} = \neg E\,\{4,4\} = 0.}$
- $Sqrt\,(3) = \mu_t\,[h\,(t,3) = 0]$:
  - $t = 0$: $h\,(0,3) = \neg E\,\{\Pi\,(0,0)\,, 3\} = \neg E\,\{0,3\} = 1.$
  - $t = 1$: $h\,(1,3) = \neg E\,\{\Pi\,(1,1)\,, 3\} = \neg E\,\{1,3\} = 1.$
  - $t = 2$: $h\,(2,3) = \neg E\,\{\Pi\,(2,2)\,, 3\} = \neg E\,\{4,3\} = 1.$
  - $t = 3$: $h\,(3,3) = \neg E\,\{\Pi\,(3,3)\,, 3\} = \neg E\,\{9,3\} = 1.$
  - $\vdots$

4.4.6. Piso de la raiz cuadrada

Definiremos la funcion numerica $FSq\left(x\right) = \lfloor \sqrt{x} \rfloor$. Buscamos un t tal que $t \leq \sqrt{x} < t{+}1 \iff t^2 \leq x < \left(t+1\right)^2$. Sea entonces $h\left(t,x\right) = Leq\left\{\Pi\left[t+1, t+1\right], x\right\}$. Veamos algunos ejemplos:

- $FSq\left(4\right) = \mu_t\left[h\left(t,4\right) = 0\right]$:
  - $t = 0$: $h\left(0,4\right) = Leq\left\{1,4\right\} = 1$.
  - $t = 1$: $h\left(1,4\right) = Leq\left\{4,4\right\} = 1$.
  - $\boxed{t = \mathbf{2} : h\left(2,4\right) = Leq\left\{9,4\right\} = 0.}$
- $FSq\left(5\right) = \mu_t\left[h\left(t,5\right) = 0\right]$:
  - $t = 0$: $h\left(0,5\right) = Leq\left\{1,5\right\} = 1$.
  - $t = 1$: $h\left(1,5\right) = Leq\left\{4,5\right\} = 1$.
  - $\boxed{t = \mathbf{2} : h\left(2,5\right) = Leq\left\{9,5\right\} = 0.}$

4.4.7. Piso del cociente

Definiremos la funcion numerica $Fdiv\left(x,y\right) = \lfloor x/y \rfloor$. Buscamos un t tal que $t \leq x/y < t{+}1 \iff ty \leq x < ty{+}y$. Sea entonces $h\left(t,x,y\right) = Leq\left\{\Pi\left(t,y\right) + y, x\right\}$. Veamos algunos ejemplos:

- $Fdiv(0,0) = \mu_t\left[h\left(t,0,0\right) = 0\right]$:
  - $t = 0$: $h\left(0,0,0\right) = Leq\left\{\Pi\left(0,0\right) + 0, 0\right\} = Leq\left\{0+0,0\right\} = 1$.
  - $t = 1$: $h\left(1,0,0\right) = Leq\left\{\Pi\left(1,0\right) + 0, 0\right\} = Leq\left\{0+0,0\right\} = 1$.
  - $\vdots$
- $Fdiv(5,0) = \mu_t\left[h\left(t,5,0\right) = 0\right]$:
  - $t = 0$: $h\left(0,5,0\right) = Leq\left\{\Pi\left(0,0\right) + 0, 5\right\} = Leq\left\{0+0,5\right\} = 1$.
  - $t = 1$: $h\left(1,5,0\right) = Leq\left\{\Pi\left(1,0\right) + 0, 5\right\} = Leq\left\{0+0,5\right\} = 1$.
  - $\vdots$

- $Fdiv(6,3) = \mu_t [h(t,6,3) = 0]$:
  - $t = 0$: $h(0,6,3) = Leq\{\Pi(0,3) + 3, 6\} = Leq\{0 + 3, 6\} = 1$.
  - $t = 1$: $h(1,6,3) = Leq\{\Pi(1,3) + 3, 6\} = Leq\{3 + 3, 6\} = 1$.
  - $\boxed{t = \mathbf{2} : h(2,6,3) = Leq\{\Pi(2,3) + 3, 6\} = Leq\{6 + 3, 6\} = 0.}$
- $Fdiv(4,3) = \mu_t [h(t,4,3) = 0]$:
  - $t = 0$: $h(0,4,3) = Leq\{\Pi(0,3) + 3, 4\} = Leq\{0 + 3, 4\} = 1$.
  - $\boxed{t = \mathbf{1} : h(1,4,3) = Leq\{\Pi(1,3) + 3, 4\} = Leq\{3 + 3, 4\} = 0.}$

Si quisieramos definir $Fdiv(0,0) = 0$ bastaria con redefinir h como:

$$h(t,x,y) = Leq\left\{\Pi[t,y] + y + \underbrace{\Pi[D_0(x), D_0(y)]}_{=1 \Longleftrightarrow x=y=0}, x\right\}$$

4.4.8. Resto

Definiremos la funcion numerica $mod(x,y)$ que da el resto de la division entera entre x e y. Buscamos un t tal que $Fdiv(x,y) \times y + t = x$. Sea entonces $h(t,x,y) = \neg E\{\Pi[Fdiv(x,y), y] + t, x\}$.

Veamos algunos ejemplos:

- $mod(x,0) = \mu_t [h(t,x,0) = 0]$: no termina pues $Fdiv(x,0)$ no termina.
- $mod(7,2) = \mu_t [h(t,7,2) = 0]$:
  - $t = 0$: $h(0,7,2) = \neg E\{\Pi[3,2] + 0, 7\} = \neg E\{6,7\} = 1$.
  - $\boxed{t = \mathbf{1} : h(1,7,2) = \neg E\{\Pi[3,2] + 1, 7\} = \neg E\{7,7\} = 0.}$
- $mod(10,4) = \mu_t [h(t,10,4) = 0]$:
  - $t = 0$: $h(0,10,4) = \neg E\{\Pi[2,4] + 0, 10\} = \neg E\{8,10\} = 1$.
  - $t = 1$: $h(1,10,4) = \neg E\{\Pi[2,4] + 1, 10\} = \neg E\{9,10\} = 1$.
  - $\boxed{t = \mathbf{2} : h(2,10,4) = \neg E\{\Pi[2,4] + 2, 10\} = \neg E\{10,10\} = 0.}$

Capítulo 5

Funciones de lista

5.1. Definiciones

5.1.1. Lista

Una lista es una secuencia ordenada y finita de cero o mas elementos de $\mathbb{N}_0$.

Notacion

- $[x_1, x_2, \ldots, x_k]$ para una lista de longitud k.
- [] para la lista vacia.
- Notaremos con $\mathcal{L}$ al conjunto de todas las listas.
- Con $\mathcal{L}^n$ indicaremos el conjunto de listas que poseen exactamente n elementos.
- Con $\mathcal{L}^{\geq n}$ indicaremos el conjunto de listas con al menos n elementos.

5.1.2. Concatenacion

Dadas dos listas $X = [x_1, \ldots, x_m]$ e $Y = [y_1, \ldots, y_n]$ llamamos concatenacion de X e Y a la lista $[x_1, \ldots, x_m, y_1, \ldots, y_n]$ y lo notaremos $[X, Y]$.

Para distinguir ciertos elementos de intereses escribiremos $[a, X, b, Y]$. La concatenacion es una operacion asociativa cuyo elemento neutro es la lista vacia.

5.1.3. Funciones de lista

Las funciones de listas son funciones que van de $\mathcal{L}$ en $\mathcal{L}$. Para indicar que una funcion F asigna a la lista X, la lista Y escribimos $F[X] = Y$ o bien $FX = Y$. Notese la omision de los parentesis sobre los argumentos.

Observacion Podemos pensar a las funciones numericas $f : \mathbb{N}_0^k \to \mathbb{N}_0$ como funciones de listas $F : \mathcal{L}^k \to \mathcal{L}^1$.

5.1.4. Funciones base

Llamaremos funciones base a las siguientes 6 funciones:

- Cero a izquierda: $0_i[x_1, x_2, \ldots, x_k] = [\mathbf{0}, x_1, x_2, \ldots, x_k]$.
- Cero a derecha: $0_d[x_1, x_2, \ldots, x_k] = [x_1, x_2, \ldots, x_k, \mathbf{0}]$.
- Borrar a izquierda: $\square_i[\boldsymbol{x_1}, x_2, \ldots, x_k] = [x_2, \ldots, x_k]$.
- Borrar a derecha: $\square_d[x_1, x_2, \ldots, \boldsymbol{x_k}] = [x_1, x_2, \ldots, x_{k-1}]$.
- Sucesor a izquierda: $S_i[x_1, x_2, \ldots, x_k] = [\boldsymbol{x_1} + \mathbf{1}, x_2, \ldots, x_k]$.
- Sucesor a derecha: $S_d[x_1, x_2, \ldots, x_k] = [x_1, x_2, \ldots, \boldsymbol{x_k} + \mathbf{1}]$.

Observacion Notese que $dom(0_i) = dom(0_d) = \mathcal{L}$ y que $dom(\square_i) = dom(\square_d) = dom(S_i) = dom(S_d) = \mathcal{L}^{\geq 1}$.

5.1.5. Operadores

Definiremos dos operadores que nos permitiran construir nuevas funciones:

- El operador de composicion que dadas dos funciones F y G construye la funcion $H = F \circ G$ que notaremos simplemente $H = FG$ definida como $HX = G\,[FX]$. Notese que contrariamente a lo usual, primero se aplica la funcion F y al resultado se aplica la funcion G.

- El operador de repeticion que dada una funcion F construye la funcion $\langle F \rangle$ definida como:

$$\langle F \rangle\,[x, Y, z] = \begin{cases} [x, Y, z] & x = z \\ F\,\langle F \rangle\,[x, Y, z] & x \neq z \end{cases}$$

  es decir que la funcion $\langle F \rangle$ actua aplicando F hasta que el primer y ultimo elemento de su argumento sean iguales.

Dominio
El dominio de la composicion es $dom\,(FG) = \{X \in \mathcal{L} / X \in dom\,(F) \wedge FX \in dom\,(G)\}$.

Observemos que $\langle F \rangle$ esta definida sobre listas de la forma $[x, Y, z]$, es decir, que tienen al menos dos elementos. Mas precisamente $X \in dom\,(\langle F \rangle)$ si en la sucecion $\{X, FX, FFX, FFFX, \ldots\}$ existe una lista cuyo primer y ultimo elemento son iguales.

5.1.6. Definicion inductiva

Definimos inductivamente el conjunto de funciones recursivas de listas (FRL) como el menor conjunto tal que:

- Las funciones base pertenecen a FRL.

- Las funciones obtenidas aplicando un numero finito de operaciones de composicion y repeticion sobre elementos de FRL tambien pertenecen a FRL.

5.2. Ejemplos

5.2.1. Pasar a izquierda

La funcion $\triangleleft = 0_i \langle S_i \rangle \square_d$ pasa el elemento de la derecha a la izquierda. Observemos una traza:

	$[X, y]$
0_i	$[0, X, y]$
$\langle S_i \rangle$	$[y, X, y]$
$\square_d$	$[y, X]$

5.2.2. Pasar a derecha

La funcion $\triangleright = 0_d \langle S_d \rangle \square_i$ pasa el elemento de la izquierda a la derecha.

5.2.3. Duplicar a izquierda

La funcion $D_i = 0_d \langle S_d \rangle \triangleleft$ duplica el elemento de la izquierda (a la izquierda). Observemos una traza:

	$[y, X]$
0_d	$[y, X, 0]$
$\langle S_d \rangle$	$[y, X, y]$
$\triangleleft$	$[y, y, X]$

5.2.4. Duplicar a derecha

La funcion $D_d = 0_i \langle S_i \rangle \triangleright$ duplica el elemento de la derecha (a la derecha).

5.2.5. Intercambiar extremos

La funcion $\leftrightarrow = \triangleright 0_i \triangleleft 0_i \langle S_i \triangleright \triangleright S_i \triangleleft \triangleleft \rangle \square_d \square_i \triangleright$ intercambia los extremos de una lista. Observemos una traza:

	$[x, Y, z]$
$\triangleright$	$[Y, z, x]$
0_i	$[0, Y, z, x]$
$\triangleleft$	$[x, 0, Y, z]$
0_i	$[0, x, 0, Y, z]$
$\langle S_i \triangleright \triangleright S_i \triangleleft \triangleleft \rangle$	$[z, x, z, Y, z]$
$\square_d$	$[z, x, z, Y]$
$\square_i$	$[x, z, Y]$
$\triangleright$	$[z, Y, x]$

5.2.6. Predecesor izquierda

La funcion $Pd_i\,[x, Y] = [x - 1, Y]$ es igual a $0_d 0_d S_d\,\langle S_d \triangleleft S_d \triangleright \rangle\,\square_i \square_d \triangleleft$. Observemos una traza:

	$[x, Y]$
0_d	$[x, Y, 0]$
0_d	$[x, Y, 0, 0]$
S_d	$[x, Y, 0, 1]$
$\langle S_d \triangleleft S_d \triangleright \rangle$	$[x, Y, x - 1, x]$
$\square_i$	$[Y, x - 1, x]$
$\square_d$	$[Y, x - 1]$
$\triangleleft$	$[x - 1, Y]$

5.2.7. Predecesor natural izquierda

La funcion $\widehat{Pd_i}\,[x, Y] = [x - 1, Y]$ si $x \neq 0$ y 0 en caso contrario, es igual a $0_d S_d S_i\,\langle P_i \square_d D_i \triangleright \rangle\,\square_d Pd_i$. Observemos una traza:

	$[x, Y]$
0_d	$[x, Y, 0]$
S_d	$[x, Y, 1]$
S_i	$[x + 1, Y, 1]$
$\langle P_i \square_d D_i \triangleright \rangle$	$[x, Y, x]$
$\square_d$	$[x, Y]$
Pd_i	$[x - 1, Y]$

5.2.8. Suma izquierda

La funcion $\Sigma_i\,[x, y, Z] = [x + y, Z]$ es igual a $\triangleright 0_d\,\langle S_d \triangleleft S_d \triangleright\rangle\,\square_i \square_d \triangleleft$. Observemos una traza:

	$[x, y, Z]$
$\triangleright$	$[y, Z, x]$
0_d	$[y, Z, x, 0]$
$\langle S_d \triangleleft S_d \triangleright\rangle$	$[y, Z, x + y, y]$
$\square_i$	$[Z, x + y, y]$
$\square_d$	$[Z, x + y]$
$\triangleleft$	$[x + y, Z]$

5.2.9. Suma izquierda persistente

La funcion $\widetilde{\Sigma}_i\,[x, y, Z] = [x + y, x, y, Z]$ es igual a $D_i \triangleright^2 D_i \leftrightarrow \Sigma_i \leftrightarrow \triangleleft \leftrightarrow \triangleleft$. Observemos una traza:

	$[x, y, Z]$
$D_i \triangleright^2$	$[y, Z, x, x]$
$D_i \leftrightarrow$	$[x, y, Z, x, y]$
Σ_i	$[x + y, Z, x, y]$
$\leftrightarrow$	$[y, Z, x, x + y]$
$\triangleleft$	$[x + y, y, Z, x]$
$\leftrightarrow$	$[x, y, Z, x + y]$
$\triangleleft$	$[x + y, x, y, Z]$

5.2.10. Resta izquierda

La funcion $dif_i\,[x, y, Z] = [x - y, Z]$ es igual a $\triangleright \triangleright 0_i \langle Pd_d \triangleleft Pd_d \triangleright \rangle \square_i \square_d \triangleleft$. Observemos una traza:

	$[x, y, Z]$
$\triangleright$	$[y, Z, x]$
$\triangleright$	$[Z, x, y]$
0_i	$[0, Z, x, y]$
$\langle Pd_d \triangleleft Pd_d \triangleright \rangle$	$[0, Z, x - y, 0]$
$\square_i$	$[Z, x - y, 0]$
$\square_d$	$[Z, x - y]$
$\triangleleft$	$[x - y, Z]$

5.2.11. Resta izquierda persistente

La funcion $\widetilde{dif}_i\,[x, y, Z] = [x - y, x, y, Z]$ es igual a $D_i \triangleright^2 D_i \leftrightarrow dif_i \leftrightarrow \triangleleft \leftrightarrow \triangleleft$.

5.2.12. Resta izquierda natural

La funcion $\widehat{dif}_i\,[x, y, Z] = [x - y, Z]$ si $x > y$ o 0 en caso contrario; es igual a $\triangleright \triangleright 0_i \left\langle \widehat{Pd}_d \triangleleft \widehat{Pd}_d \triangleright \right\rangle \square_i \square_d \triangleleft$.

Notese que esta funcion es igual a la resta izquierda, solo que en lugar de utilizar el predecesor utilizamos el predecesor natural.

5.2.13. Repetir izquierda

La funcion $rep_i\,[x, y, Z] = \left[\underbrace{y, y, \ldots, y}_{x \text{ veces}}, Z\right]$ es igual a $\triangleright 0_i\,\langle \square_i D_i 0_i P d_d\rangle\,\square_i^2 \square_d \triangleleft$.
Observemos una traza:

	$[x, y, Z]$
$\triangleright$	$[y, Z, x]$
0_i	$[0, y, Z, x]$
$\langle \square_i D_i 0_i P d_d\rangle$	$\left[0,\ \underbrace{y, \ldots, y}_{x+1 \text{ veces}}, Z, 0\right]$
$\square_i^2$	$\left[\underbrace{y, \ldots, y}_{x \text{ veces}}, Z, 0\right]$
$\square_d$	$\left[\underbrace{y, \ldots, y}_{x \text{ veces}}, Z\right]$

5.2.14. Naturales izquierda

La funcion $nat_i\,[x, Y] = [1, 2, \ldots, x, Y]$ es igual a $D_i \triangleright 0_i\,\langle \square_i D_i P d_i 0_i P d_d\rangle\,\square_i^2 \square_d$.
Observemos una traza:

	$[x, Y]$
D_i	$[x, x, Y]$
$\triangleright$	$[x, Y, x]$
0_i	$[0, x, Y, x]$
$\langle \square_i D_i P d_i 0_i P d_d\rangle$	$[0, 0, 1, 2, \ldots, x, Y, 0]$
$\square_i^2$	$[1, 2, \ldots, x, Y, 0]$
$\square_d$	$[1, 2, \ldots, x, Y]$

5.2.15. Producto izquierda

La funcion $\Pi_i\,[x,y,Z]=[xy,Z]$ es igual a $\triangleright D_i 0_i \left\langle S_i \triangleright \widetilde{\Sigma}_i \triangleright \square_i \triangleleft^2 \right\rangle \square_i dif_i \square_d$. Observemos una traza:

	$[x,y,Z]$
$\triangleright$	$[y,Z,x]$
D_i	$[y,y,Z,x]$
0_i	$[0,y,y,Z,x]$
$\left\langle S_i \triangleright \widetilde{\Sigma}_i \triangleright \square_i \triangleleft^2 \right\rangle$	$[x,xy+y,y,Z,x]$
$\square_i$	$[xy+y,y,Z,x]$
dif_i	$[xy,Z,x]$
$\square_d$	$[xy,Z]$

5.2.16. Exponencial izquierda

La funcion $Exp_i\,[x,y,Z]=[x^y,Z]$ es igual a $\triangleright \leftrightarrow 0_i S_i 0_i \left\langle S_i \triangleright \widetilde{\Pi}_i \triangleright \square_i \triangleleft^2 \right\rangle \square_i \square_d \triangleright \square_i \triangleleft$. Observemos una traza:

	$[x,y,Z]$
$\triangleright \leftrightarrow$	$[x,Z,y]$
$0_i S_i$	$[1,x,Z,y]$
0_i	$[0,1,x,Z,y]$
$\left\langle S_i \triangleright \widetilde{\Pi}_i \triangleright \square_i \triangleleft^2 \right\rangle$	$[y,x^y,x,Z,y]$
$\square_i$	$[x^y,x,Z,y]$
$\square_d$	$[x^y,x,Z]$
$\triangleright$	$[x,Z,x^y]$
$\square_i$	$[Z,x^y]$
$\triangleleft$	$[x^y,Z]$

Desafortunadamente para la funcion asi definida resulta: $Exp_i\,[0,0,Z]=[1,Z]$, pero podemos solucionarlo si la redefinimos como:

$$\underbrace{\widetilde{\Sigma}_i 0_d S_d \left\langle S_d \right\rangle \square_i \square_d}_{indefinidor} \triangleright \leftrightarrow 0_i S_i 0_i \left\langle S_i \triangleright \widetilde{\Pi}_i \triangleright \square_i \triangleleft^2 \right\rangle \square_i \square_d \triangleright \square_i \triangleleft$$

5.2.17. Distinguidora del 0

La funcion $D_0[x,Z]=[1,Z]$ si $x=0$ o $[0,Z]$ en caso contrario es igual a $0_i\widehat{Exp_i}$.

5.2.18. Raiz x-esima izquierda

La funcion $Rad_i[x,y,Z]=\left[\sqrt[x]{y},Z\right]$ es igual a $\triangleright\leftrightarrow 0_i\widetilde{Exp_i}\left\langle\square_i S_i\widetilde{Exp_i}\right\rangle\square_i\square_d\triangleright\square_i\triangleleft$. Observemos una traza:

	$[x,y,Z]$
$\triangleright\leftrightarrow$	$[x,Z,y]$
0_i	$[0,x,Z,y]$
$\widetilde{Exp_i}\left\langle\square_i S_i\widetilde{Exp_i}\right\rangle$	$\left[y,\sqrt[x]{y},x,Z,y\right]$
$\square_i$	$\left[\sqrt[x]{y},x,Z,y\right]$
$\square_d$	$\left[\sqrt[x]{y},x,Z\right]$
$\triangleright$	$\left[x,Z,\sqrt[x]{y}\right]$
$\square_i$	$\left[Z,\sqrt[x]{y}\right]$
$\triangleleft$	$\left[\sqrt[x]{y},Z\right]$

5.3. Representacion de FR mediante FRL

Observemos que las funciones recursivas y las funciones de listas tienen distintos dominios y codominios, por lo que resultara util establecer una correspondencia entre ambas.

5.3.1. Representabilidad

Sea $g:\mathbb{N}_0^k\to\mathbb{N}_0$ una funcion recursiva entonces si existe una funcion de listas F_g tal que $\forall X^k\in Dom(g)$ y $\forall Y$ resulta $F_g[X,Y]=[g(X),X,Y]$ diremos entonces que F_g representa a la funcion recursiva g como funcion de listas.

5.3.2. Casos base

Enunciado Para toda funcion recursiva base g existe una funcion de listas F_g que la representa.

Demostracion

- Funciones cero $c^{(n)}$: son iguales a 0_i.
- Funciones proyeccion $p_k^{(n)}$: son iguales a $\triangleright^{k-1} D_i \left(\leftrightarrow \triangleleft\right)^{k-1}$.
- Funcion sucesor: es igual a $D_i S_i$.

5.3.3. Composicion

Enunciado Dadas las funciones recursivas $f^{(n)}$ y $\left\{g_i^{(k)}\right\}_{i=1}^{n}$ con representaciones en FRL dadas por F_f y $\left\{F_{gi}^{(k)}\right\}_{i=1}^{n}$, la composicion $h = \Phi\left(f, g_1, \ldots, g_n\right)$ tambien tiene representacion en funciones de listas.

Demostracion Veamos que $F_h = F_{g1} \triangleright F_{g2} \triangleright \ldots F_{gn} \triangleleft^{n-1} F_f \triangleright \square_i^n \triangleleft$ representa dicha composicion:

	$[X, Y]$
F_{g1}	$[\boldsymbol{g_1}\left(\boldsymbol{X}\right), X, Y]$
$\triangleright$	$[X, Y, \boldsymbol{g_1}\left(\boldsymbol{X}\right)]$
F_{g2}	$[\boldsymbol{g_2}\left(\boldsymbol{X}\right), X, Y, g_1\left(X\right)]$
$\triangleright$	$[X, Y, g_1\left(X\right), \boldsymbol{g_2}\left(\boldsymbol{X}\right)]$
$\ldots F_{gn}$	$[\boldsymbol{G_n}\left(\boldsymbol{X}\right), X, Y, g_1\left(X\right), g_2\left(X\right), \boldsymbol{\ldots}, \boldsymbol{g_{n-1}}\left(\boldsymbol{X}\right)]$
$\triangleleft^{n-1}$	$[\boldsymbol{g_1}\left(\boldsymbol{X}\right), \boldsymbol{g_2}\left(\boldsymbol{X}\right), \boldsymbol{\ldots}, \boldsymbol{g_n}\left(\boldsymbol{X}\right), X, Y]$
F_f	$[\boldsymbol{f}\left\{\boldsymbol{g_1}\left(\boldsymbol{X}\right), \ldots, \boldsymbol{g_n}\left(\boldsymbol{X}\right)\right\}, g_1\left(X\right), \ldots, g_n\left(X\right), X, Y]$
$=$	$[\boldsymbol{h}\left(\boldsymbol{X}\right), g_1\left(X\right), \ldots, g_n\left(X\right), X, Y]$
$\triangleright$	$[g_1\left(X\right), \ldots, g_n\left(X\right), X, Y, \boldsymbol{h}\left(\boldsymbol{X}\right)]$
$\square_i^n$	$[X, Y, h\left(X\right)]$
$\triangleleft$	$[h\left(X\right), X, Y]$

5.3.4. Recursion

Enunciado Sean las funciones recursivas $g^{(k)}$ y $h^{(k+2)}$ con representaciones en FRL dadas por F_g y F_h, entonces la funcion $f^{(k+1)} = R(g,h)$ tambien tiene representacion en FRL.

Demostracion Veamos que $F_f = \triangleright F_g 0_i \langle \triangleright \leftrightarrow \triangleleft F_h \triangleright \square_i S_i \leftrightarrow \triangleleft \rangle \square_i \leftrightarrow \triangleleft$ representa dicha recursion. Aplicando el primer bloque de funciones:

	$[y, X, Y]$
$\triangleright$	$[X, Y, y]$
F_g	$[\boldsymbol{g\,(X)}, X, Y, y]$
0_i	$[\mathbf{0}, g(X), X, Y, y]$
	$[0, \boldsymbol{f\,(0, X)}, X, Y, y]$

A partir de aqui hay 2 casos:

1. $y = 0$:

	$[0, f(0,X), X, Y, y]$
$\langle \ldots \rangle$	$[0, f(0,X), X, Y, 0]$
$\square_i$	$[f(0,X), X, Y, 0]$
$\leftrightarrow$	$[0, X, Y, f(0,X)]$
$\triangleleft$	$[f(y,X), y, X, Y]$

2. $y \neq 0$:

	$[0, f(0,X), X, Y, y]$
$\langle \triangleright$	$[f(0,X), X, Y, y, \mathbf{0}]$
$\leftrightarrow$	$[\mathbf{0}, X, Y, y, \boldsymbol{f\,(0, X)}]$
$\triangleleft$	$[\boldsymbol{f\,(0, X)}, 0, X, Y, y]$
F_h	$[\boldsymbol{h\,(f\,(0, X)\,, 0, X)}, f(0,X), 0, X, Y, y]$
$(*)$	$[\boldsymbol{f\,(1, X)}, f(0,X), 0, X, Y, y]$
$\triangleright$	$[f(0,X), 0, X, Y, y, \boldsymbol{f\,(1, X)}]$
$\square_i$	$[0, X, Y, y, f(1,X)]$
S_i	$[\mathbf{1}, X, Y, y, f(1,X)]$
$\leftrightarrow$	$[\boldsymbol{f\,(1, X)}, X, Y, y, \mathbf{1}]$
$\triangleleft \rangle$	$[\mathbf{1}, f(1,X), X, Y, y]$

Si continuamos podemos ver que el resultado de la repeticion sera: $[y, f\,(y, X)\,, X, Y, y]$. Luego aplicando el ultimo bloque de funciones:

	$[y, f\,(y, X)\,, X, Y, y]$
$\square_i$	$[f\,(y, X)\,, X, Y, y]$
$\leftrightarrow$	$[\boldsymbol{y}, X, Y, \boldsymbol{f\,(y, X)}]$
$\triangleleft$	$[\boldsymbol{f\,(y, X)}, y, X, Y]$

5.3.5. Minimizacion

Enunciado Sea la funcion recursiva $h^{(n+1)}$ y F_h su representacion en FRL entonces la funcion $g^{(n)} = M\,[h]$ tambien tiene representacion en FRL.

Demostracion Veamos que $F_g = 0_i F_h 0_d \,\langle \square_i S_i F_h \rangle\, \square_i \square_d$ representa dicha minimizacion. Aplicando el primer bloque de funciones:

	$[X, Y]$
0_i	$[\mathbf{0}, X, Y]$
F_h	$[\boldsymbol{h\,(0, X)}, 0, X, Y]$
0_d	$[h\,(0, X)\,, 0, X, Y, 0]$

A partir de aqui hay 2 casos:

1. $h\,(0, X) = 0$:

	$[h\,(0, X)\,, 0, X, Y, 0]$
$\langle \ldots \rangle$	$[h\,(0, X)\,, 0, X, Y, 0]$
$\square_i$	$[h\,(0, X)\,, X, Y, 0]$
$\square_d$	$[h\,(0, X)\,, X, Y]$
	$[\mu_t\,(h\,(t, X) = 0)\,, X, Y]$
	$[g\,(X)\,, X, Y]$

2. $h\,(0, X) \neq 0$:

	$[h\,(0, X)\,, 0, X, Y, 0]$
$\langle \square_i$	$[0, X, Y, 0]$
S_i	$[\mathbf{1}, X, Y, 0]$
$F_h \rangle$	$[\boldsymbol{h\,(1, X)}, 1, X, Y, 0]$

Vemos que en general, el efecto de la repeticion es transformar $[h\,(t,X)\,,t,X,Y,0]$ en $[h\,(t+1,X)\,,t+1,X,Y,0]$. La repeticion se aplicara un numero k de veces hasta que $h\,(k,X)=0$ y a partir de ahi aplicando el ultimo bloque de funciones:

	$[h\,(k,X)\,,k,X,Y,0]$
	$[0,k,X,Y,0]$
$\square_i$	$[k,X,Y,0]$
$\square_d$	$[k,X,Y]$
	$[\mu_t\,(h\,(t,X)=0)\,,X,Y]$
	$[g\,(X)\,,X,Y]$

5.3.6. Conclusion

Con todo lo anterior hemos demostrado que toda funcion recursiva puede ser representada por una funcion de listas, con lo que las FRL son un modelo de calculo tan poderoso como el de las FR.

Parte III

Lenguajes formales

Capítulo 6

Gramaticas y expresiones regulares

6.1. Definiciones

6.1.1. Alfabeto

Un alfabeto es un conjunto finito y no vacio, cuyos elementos reciben el nombre de simbolos, letras o caracteres.

Notaremos a los alfabetos con la letra sigma, por ejemplo $\Sigma = \{l_1, l_2, \ldots, l_k\}$.

6.1.2. Cadena

Una cadena o palabra sobre un alfabeto Σ es una secuencia finita de simbolos de Σ. Mas precisamente se la define como la funcion $p : [\![1, n]\!] \to \Sigma$ donde el entero positivo n es la longitud de la palabra, que se denota tambien como $|p|$.

Ademas definimos la palabra especial $\lambda : \{\} \to \Sigma$ llamada nula, o vacia, considerada de longitud 0.

Dada la palabra $p : [\![1, n]\!] \to \Sigma$ por simplicidad de notacion escribiremos la cadena p como $p = p(1)\, p(2) \ldots p(n) = l_1 l_2 \ldots l_n$.

6.1.3. Clausuras de Kleene

Dado un alfabeto Σ definimos $\Sigma^0 = \{\lambda\}$ y $\Sigma^n = \{p/p : [\![1, n]\!] \to \Sigma\}$ para todo $n \geq 1$, es decir, el conjunto de todas las palabra de una determinada longitud.

Definimos ademas dos operadores: $\Sigma^* = \bigcup_{n\geq 0} \Sigma^n$ y $\Sigma^+ = \bigcup_{n\geq 1} \Sigma^n$, es decir que Σ^* es el conjunto de todas las palabras sobre un alfabeto Σ, incluyendo la palabra vacia.

Observaciones

- Dado un conjunto $A = \emptyset$ resulta $A^* = \{\lambda\}$.
- Si $A \neq \emptyset$ resulta A^* es infinito numerable pues es u.n.c.n.

6.1.4. Concatenacion

Dado un alfabeto Σ y dos palabras $p : [\![1, m]\!] \to \Sigma$ y $q : [\![1, n]\!] \to \Sigma$ definimos la concatenacion de p y q (denotada pq) como la palabra $pq : [\![1, m + n]\!] \to \Sigma$ tal que:

$$pq\,(i) = \begin{cases} p\,(i) & 1 \leq i \leq m \\ q\,(i-m) & m+1 \leq i \leq m+n \end{cases}$$

Observaciones

- En particular $p\lambda = \lambda p = p$.
- Es facil ver que $p\,(qr) = (qp)\,r$ pero $pq \neq qr$, es decir, la concatenacion es asociativa pero no conmutativa.

6.1.5. Cadena potencia

Dada una cadena $p \in \Sigma^*$ definimos su potencia p^n como:

$$p^n = \begin{cases} \lambda & n = 0 \\ p^{n-1}p & n \geq 1 \end{cases}$$

6.1.6. Cadena reversa

Dada una cadena $p \in \Sigma^*$ y un caracter $c \in \Sigma$ definimos inductivamente la cadena reversa p^R como:

$$\begin{aligned} \lambda^R &= \lambda \\ (pc)^R &= cp^R \end{aligned}$$

6.1.7. Subcadenas

Diremos que:

- s es *subcadena* de p si existen $q, r \in \Sigma^*$ tales que: $p = qsr$.
- s es *prefijo* de p si es una subcadena tal que: $p = \lambda sr$.
- s es sufijo de p si es una subcadena tal que: $p = qs\lambda$.
- s es *subcadena propia* de p si es subcadena de p y ademas $s \neq p$.

6.2. Lenguajes

6.2.1. Definicion

Un lenguaje sobre un alfabeto Σ es un subconjunto de Σ^*. Para cada alfabeto Σ llamaremos $\mathcal{L}$ al conjunto de todos los lenguajes sobre Σ, es decir, $\mathcal{L} = \mathcal{P}(\Sigma^*)$.

Observacion Para cualquier alfabeto, sabemos que $\#\Sigma^* = \aleph_0$ y en consecuencia $\#\mathcal{L} = \#\mathcal{P}(\Sigma^*) = \aleph_1$.

6.2.2. Union

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la union de los lenguajes como: $L_1 \cup L_2 = \{p \in \Sigma^* / p \in L_1 \vee p \in L_2\}$.

6.2.3. Interseccion

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la interseccion de los lenguajes como: $L_1 \cap L_2 = \{p \in \Sigma^* / p \in L_1 \wedge p \in L_2\}$.

6.2.4. Diferencia

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la diferencia de los lenguajes como: $L_1 - L_2 = \{p \in \Sigma^* / p \in L_1 \wedge p \notin L_2\}$.

6.2.5. Complemento

Para cada lenguaje L, definimos el lenguaje complemento como: $\bar{L} = \Sigma^* - L$.

6.2.6. Concatenacion

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la concatenacion de los lenguajes como: $L_1 L_2 = \{p \in \Sigma^* / \exists q \in L_1, \exists r \in L_2, p = qr\}$.

Observaciones

- $L\emptyset = \emptyset L = \emptyset$.
- $L\{\lambda\} = \{\lambda\} L = L$.
- En general $L_1 L_2 \neq L_2 L_1$.

6.2.7. Potencia

Dado un lenguaje L definimos inductivamente su potencia L^n como

$$\begin{array}{rl} L^0 = & \{\lambda\} \\ L^{n+1} = & L^n L \end{array}$$

6.2.8. Clausuras de Kleene

Dados un alfabeto Σ y un lenguaje L sobre Σ definimos: $L^* = \bigcup_{n \geq 0} L^n$ y $L^+ = \bigcup_{n \geq 1} L^n$, es decir que L^* es la union infinita:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots = \{\lambda\} \cup L \cup L^2 \cup \cdots \cup L^n \cup \cdots$$

mientras que L^+ es la union infinita:

$$L^+ = L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots = L \cup L^2 \cup \cdots \cup L^n \cup \cdots$$

Observaciones

- Tanto L^* como L^+ contienen a L.
- Dado que $L^0 = \{\lambda\}$ resulta que $L^* = L^+ \cup \{\lambda\}$.
- $\emptyset^* = \{\lambda\}$.
- $L^+ = L^*L$.
- $(L^*)^* = L^*$.
- $L^*L^* = L^*$.

Al contrario que con los operadores anteriores, las clausuras de Kleen construyen siempre lenguajes infinitos.

6.3. Gramaticas

6.3.1. Definicion

Una gramatica es una tupla (N, T, P, σ) donde:

- N es un conjunto finito de simbolos llamados *no terminales*.
- T es un conjunto finito de simbolos llamados *terminales*, o alfabeto, tal que $N \cap T = \emptyset$.
- P es un conjunto finito de *reglas de produccion* donde $P \subseteq [(N \cup T)^* - T^*] \times [N \cup T]^*$.
- $\sigma \in N$ es el llamado *simbolo inicial*.

Observaciones

- Notese que $[(N \cup T)^* - T^*]$ es el conjunto de cadenas de no terminales y terminales que contienen al menos un no terminal. Dado que $\lambda \notin [(N \cup T)^* - T^*]$ no puede haber reglas del tipo (λ, β).
- A una regla de produccion (α, β) la notaremos: $\alpha \to \beta$.
- Una regla del tipo $\alpha \to \lambda$ recibe el nombre de regla λ.

6.3.2. Derivacion

Si $\alpha \to \beta$ es una regla de produccion y $\gamma\alpha\delta \in [(N \cup T)^* - T^*]$, entonces diremos que $\gamma\beta\delta$ deriva directamente de $\gamma\alpha\delta$ o que $\gamma\alpha\delta$ produce directamente $\gamma\beta\delta$ y lo notaremos: $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$.

Si vale que $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow a_{n-1} \Rightarrow \alpha_n$ donde $\alpha_i \in [(N \cup T)^* - T^*]$ y $\alpha_n \in [N \cup T]^*$ diremos que α_n deriva de α_1 o que α_1 produce α_n y lo notaremos: $\alpha_1 \Rightarrow^* \alpha_n$.

6.3.3. Lenguajes generados

Definimos el lenguaje generado por una gramatica $G = (N, T, P, \sigma)$ como $L(G) = \{p \in T^* / \sigma \Rightarrow^* p\}$.

6.3.4. Gramaticas regulares

Las gramaticas regulares (tambien llamadas del tipo 3 o lineales) pueden ser clasificadas como derechas o izquierdas.

Las reglas de produccion de una gramatica regular derecha tienen las siguientes restricciones:

1. El lado izquierdo debe consistir en un solo no terminal.

2. El lado derecho esta formado por un simbolo terminal que puede estar seguido o no, por un simbolo no terminal o la cadena vacia.

Es decir que las producciones de una gramatica regular derecha tienen la forma $A \to a$, $A \to aB$ o $A \to \lambda$, donde $A, B \in N$ y $a \in T$.

Alternativamente, en una gramatica regular izquierda las reglas de produccion son de la forma: $A \to a$, $A \to Ba$ o $A \to \lambda$.

Observaciones Toda gramatica regular derecha puede ser convertida en una gramatica regular izquierda (y viceversa). Algunas definiciones permiten reemplazar a por una cadena de uno o mas terminales, siendo ambas definiciones equivalentes.

6.3.5. Gramaticas libres de contexto

Las gramaticas libres o independientes del contexto son iguales a las regulares, pero sin restricciones sobre el lado derecho. Es decir que las reglas de produccion tienen la forma $A \to \alpha$ donde $A \in N$ y $\alpha \in (N \cup T)^*$.

Observaciones

- Estas gramaticas reciben ese nombre debido a que en el lado izquierdo el no terminal aparece solo, por lo que la regla se puede aplicar sin importar el contexto en el que aparece dicho no terminal, al contrario de reglas de la forma $\gamma N \delta \to \gamma \alpha \delta$ donde el no terminal N solo se puede reemplazar por α cuando se encuentre en el contexto de γ y δ.
- Dado que no existen restricciones sobre el lado derecho, podria ocurrir que en el aparezcan mas de uno como en $A \to XY$. En esta situacion el enfoque mas comun consiste en reemplazar en el paso siguiente el no terminal situado mas a la izquierda.
- Alternativamente se podria reemplazar no terminales desde la derecha u algun otro patron pues resultara que el orden en el que se apliquen las reglas, no afecta la determinacion de si una cadena puede ser generada por la gramatica.

6.3.6. Gramaticas sensibles al contexto

Las reglas de produccion de una gramatica dependiente del contexto son del tipo $\alpha A \beta \to \alpha \delta \beta$ donde $A \in N$, $\alpha, \beta \in [N \cup T]^*$ y $\delta \in [N \cup T]^+$. Adicionalmente se permite la regla $\sigma \to \lambda$ donde σ es el simbolo inicial siempre que σ no aparezca en el lado derecho de otra produccion.

Quizas a simple vista podria parecer que las gramaticas sensibles al contexto son mas restrictivas que las libre de contexto, sin embargo basta tomar $\alpha = \lambda = \beta$ para ver que las sensibles al contexto abarcan a las libres de contexto.

6.3.7. Gramaticas estructuradas por frases

Finalmente las gramaticas estructuradas por frases o irrestrictas, son aquellas que no tienen restricciones sobre la forma de las reglas de produccion. Es decir que sus reglas son de la forma $\alpha \to \beta$ donde $\alpha \in [(N \cup T)^* - T^*]$ y $\beta \in [N \cup T]^*$.

Observacion Dado que $\lambda \notin [(N \cup T)^* - T^*]$ no puede haber reglas del tipo $\lambda \to \beta$.

6.3.8. Relacion entre gramaticas

Llamando $\mathcal{G}_i = \{G/G \text{ es del tipo } i\}$ con $i = 0, 1, 2, 3$, se tiene que: $\mathcal{G}_3 \subset \mathcal{G}_2 \subset \mathcal{G}_1 \subset \mathcal{G}_0$.

6.3.9. Tipos de lenguajes

Diremos que un lenguaje L es de tipo i (con $i = 0, 1, 2, 3$) si y solo si existe una gramatica G del tipo i tal que $L(G) = L$.

6.3.10. Ejemplos

6.3.10.1. Identificacion de gramaticas

Gramaticas regulares

La gramatica $(N = \{\sigma, A\}, T = \{a, b\}, P, \sigma)$ donde P esta dada por las siguientes reglas de produccion

1. $\sigma \to b\sigma$.
2. $\sigma \to aA$.
3. $\sigma \to b$.
4. $A \to a\sigma$.
5. $A \to bA$.
6. $A \to a$.

es regular pues las reglas 3 y 4 son de la forma $A \to a$ y las restantes de la forma $A \to aB$.

La cadena *bbabbab* es producida de la siguiente forma:

$\sigma \Rightarrow^{(1)} b\sigma \Rightarrow^{(1)} \Rightarrow bb\sigma \Rightarrow^{(2)} bbaA \Rightarrow^{(5)} bbabA \Rightarrow^{(5)} bbabbA \Rightarrow^{(4)} bbabba\sigma \Rightarrow^{(3)} bbabbab.$

Gramaticas libre de contexto

La gramatica $(N = \{\sigma, A, B\}, T = \{a, b, c\}, P, \sigma)$ donde P esta dada por las siguientes reglas de produccion

1. $\sigma \to BAB$.
2. $\sigma \to ABA$.
3. $A \to aA$.
4. $A \to ab$.
5. $A \to AB$.
6. $B \to b$.
7. $B \to BA$.

no es regular pues la regla 4 no lo es. Es una gramatica libre de contexto pues todas sus reglas comienzan con un solo no terminal y derivan en una cadena de terminales y no terminales.

La cadena *abbabb* es producida de la siguiente forma:

$\sigma \Rightarrow^{(2)} ABA \Rightarrow^{(5)} \Rightarrow ABAB \Rightarrow^{(4)} abBAB \Rightarrow^{(6)} abbAB \Rightarrow^{(4)} abbabB \Rightarrow^{(6)} abbabb.$

Gramaticas libres de contexto

La gramatica $(N = \{\sigma, A, B\}, T = \{a, b\}, P, \sigma)$ donde P esta dada por las siguientes reglas de produccion

1. $\sigma \to A$.
2. $\sigma \to AAB$.
3. $A \to aa$.
4. $B \to b$.
5. $Aa \to ABa$.
6. $Bb \to ABb$.
7. $AB \to ABB$.

no es regular ni libre de contexto pues la regla 1 no lo es. Es una gramatica sensible al contexto, realizando las siguientes substituciones en la definicion:

1. Reemplazando α y β por λ, A por σ y γ por A.
2. Reemplazando α y β por λ, A por σ y γ por AAB.
3. Reemplazando α y β por λ, A por A y γ por aa.
4. Reemplazando α y β por λ, A por B y γ por b.
5. Reemplazando α por λ, β por a, A por A y γ por AB.

6. Reemplazando α por λ, β por b, A por B y γ por AB.

7. Reemplazando α por λ, β por B, A por A y γ por AB.

La cadena $aabaab$ es producida de la siguiente forma:

$\sigma \Rightarrow^{(2)} AAB \Rightarrow^{(3)}\Rightarrow AaaB \Rightarrow^{(5)} ABaaB \Rightarrow^{(4)} ABaab \Rightarrow^{(4)} Abaab \Rightarrow^{(3)} aabaab.$

Gramaticas estructurada por frases

La gramatica $(N = \{\sigma, A, B\}, T = \{a, b, c\}, P, \sigma)$ donde P esta dada por las siguientes reglas de produccion

1. $\sigma \to AB$.
2. $A \to aA$.
3. $A \to a$.
4. $B \to Bb$.
5. $B \to b$.
6. $AB \to BA$.

no es regular pues la regla 1 no lo es. Tampoco es libre ni sensible de contexto pues la regla 6 no lo es. Por lo tanto es estructurada por frases.

La cadena $abab$ es producida de la siguiente forma:

$\sigma \Rightarrow^{(1)} AB \Rightarrow^{(2)}\Rightarrow aAB \Rightarrow^{(4)} aABb \Rightarrow^{(6)} aBAb \Rightarrow^{(5)} abAb \Rightarrow^{(3)} abab.$

6.3.10.2. Generacion de lenguajes

Gramaticas regulares

- Cadenas sobre $\{a, b\}$ que comienzan con a:

  - $\sigma \to a$.
  - $\sigma \to aA$.
  - $A \to aA$.
  - $A \to bA$.
  - $A \to a$.
  - $A \to b$.

- Cadenas sobre $\{a, b\}$ que contengan exactamente una a y terminen con al menos una b:

  - $\sigma \to aA.$
  - $A \to b.$
  - $A \to bA.$
  - $\sigma \to bB.$
  - $B \to bB.$
  - $B \to aA.$

- Cadenas sobre $\{a, b\}$ que terminan con ba:

  - $\sigma \to bB.$
  - $B \to a.$
  - $B \to bB.$
  - $B \to aA.$
  - $A \to aA.$
  - $A \to bB.$
  - $\sigma \to aA.$

- Cadenas sobre $\{a, b\}$ de la forma $a^n b^m$:

  - $\sigma \to \lambda.$
  - $\sigma \to a.$
  - $\sigma \to aA.$
  - $A \to aA.$
  - $A \to \lambda.$
  - $A \to b.$
  - $A \to bB.$
  - $B \to bB.$
  - $B \to b.$

Gramaticas libre de contexto

- Cadenas sobre $\{a, b\}$ de la forma $a^n b^n$:

  - $\sigma \to a\sigma b.$
  - $\sigma \to \lambda.$

- Cadenas de la forma $a^n b^n c^k$:
  - $\sigma \to \lambda$.
  - $\sigma \to aAb\mathrm{C}$.
  - $A \to aAb$.
  - $A \to \lambda$.
  - $C \to \lambda$.
  - $C \to cC$.
- Cadenas sobre $\{a, b\}$ que empiezan y terminan con el mismo simbolo:
  - $\sigma \to aA$.
  - $A \to aA$.
  - $A \to bA$.
  - $A \to a$.
  - $\sigma \to bB$.
  - $B \to aB$.
  - $B \to bB$.
  - $B \to b$.
- Cadenas sobre $\{a, b\}$ de longitud impar:
  - $\sigma \to aI$.
  - $\sigma \to bI$.
  - $I \to \lambda$.
  - $I \to aP$.
  - $I \to bP$.
  - $P \to aI$.
  - $P \to bI$.
- Cadenas sobre $\{a, b\}$ de la forma $a^m b^n$ con $m < n$:
  - $\sigma \to aAB$.
  - $A \to aAB$.
  - $A \to B$.
  - $B \to b$.
  - $B \to bB$.

- Cadenas sobre $\{a, b, c\}$ que empiezan y terminan con a; y entre cada aparicion de a y la siguiente, hay un numero par de b o impar de c:

  - $\sigma \to a.$
  - $\sigma \to aPBPC.$
  - $PBPC \to bIBPC.$
  - $PBPC \to cPBIC.$
  - $PBIC \to bIBIC.$
  - $PBIC \to cIBPC.$
  - $IBPC \to bPBPC.$
  - $IBPC \to cIBIC.$
  - $IBIC \to bPBIC.$
  - $IBIC \to cIBPC.$
  - $PBPC \to aPBPC.$
  - $PBIC \to aPBPC.$
  - $IBIC \to aPBPC.$
  - $PBPC \to a.$
  - $PBIC \to a.$
  - $IBIC \to a.$

6.4. Expresiones regulares

6.4.1. Definicion

Una expresion regular (ER) sobre un alfabeto Σ es una cadena sobre el alfabeto $\Sigma \cup \{(,), \circ, \cup, *, \emptyset\}$ definida inductivamente como:

- $\emptyset \in ER.$
- $x \in \Sigma \Rightarrow x \in ER.$
- $p, q \in ER \Rightarrow (p \cup q) \in ER.$
- $p, q \in ER \Rightarrow (p \circ q) \in ER.$
- $p \in ER \Rightarrow (p^*) \in ER.$

Para reducir el numero de parentesis, convenimos el siguiente orden de precedencia: $*, \circ, \cup$.

Observacion Notese que una expresion regular es una cadena de simbolos, no un lenguaje.

6.4.2. Lenguaje asociado

El lenguaje asociado a una expresion esta dado por la funcion $L : ER \to \mathcal{L}$ que tiene las siguientes propiedades:

- $L(\emptyset) = \emptyset$.
- $x \in ER \wedge x \in \Sigma \Rightarrow L(x) = \{x\}$.
- $p, q \in ER \Rightarrow L(p \cup q) = L(p) \cup L(q)$.
- $p, q \in ER \Rightarrow L(p \circ q) = L(p) \circ L(q)$.
- $p \in ER \Rightarrow L(p^*) = L(p)^*$.

Definimos ademas $L_{ER} = \{L \in \mathcal{L} / \exists e \in ER : L(e) = L\}$, el conjunto de todos los lenguajes asociados a expresiones regulares.

6.4.3. Relacion entre lenguajes asoc. a ER y $\mathcal{L}_3$

Enunciado El conjunto de todos los lenguajes asociados a expresiones regulares es igual al conjunto de todos los lenguajes regulares.

Demostracion Deberemos probar una doble contencion:

- $L_{ER} \subseteq \mathcal{L}_3$: Lo demostraremos por induccion sobre ER
  - $e = \emptyset \Rightarrow L(e) = \emptyset$: Construimos una gramatica regular cuya unica regla de produccion es $\sigma \to a\sigma$.
  - $e = x \Rightarrow L(e) = \{x\}$: Construimos una gramatica regular cuya unica regla de produccion es $\sigma \to x$.
  - $e = p \cup q \Rightarrow L(p \cup q) = L(p) \cup L(q)$: Sean $P = (N_p, T_p, P_p, \sigma_p)$ la gramatica que genera $L(p)$, $Q = (N_q, T_q, P_q, \sigma_q)$ la gramatica que genera $L(q)$ y asumiendo que $N_p \cap N_t = \emptyset$ construimos la gramatica $G = (N_p \cup N_q, T_p \cup T_q, P_g, \sigma_p)$ donde P_g son todas las reglas de produccion P_p mas todas las de P_q en donde reemplazamos cada ocurrencia de σ_q por σ_p.

- $e = p \circ q \Rightarrow L(p \circ q) = L(p) \circ L(q)$: Sean $P = (N_p, T_p, P_p, \sigma_p)$ la gramatica que genera $L(p)$, $Q = (N_q, T_q, P_q, \sigma_q)$ la gramatica que genera $L(q)$ y asumiendo que $N_p \cap N_t = \emptyset$ construimos la gramatica $G = (N_p \cup N_q, T_p \cup T_q, P_g, \sigma_p)$ donde P_g son todas las reglas de produccion P_q mas todas las de P_p en donde reemplazamos cada regla de la forma $X \to x$ por otra de la forma $X \to x\sigma_q$.
- $e = p^* \Rightarrow L(p^*) = L(p)^*$: Sean $P = (N, T, P, \sigma)$ la gramatica que genera $L(p)$ construimos la gramatica $G = (N, T, P', \sigma)$ donde P' son todas las reglas de produccion P en donde agregamos por cada regla de la forma $X \to x$, otra de la forma $X \to x\sigma$.

- $\mathcal{L}_3 \subseteq L_{ER}$: Queda como ejercicio al lector completar esta demostracion.

6.4.4. Ejemplos

6.4.4.1. Descripcion de lenguajes

- $[x \circ (y \circ z^*)] = \{xyz^n / n \in \mathbb{N}_0\}$.
- $[x^* \circ (y \circ z)] = \{x^n yz / n \in \mathbb{N}_0\}$.
- $[(x \cup y) \circ x] = \{xx, yx\}$.
- $(z \cup y)^* = \{\alpha_1 \alpha_2 \ldots \alpha_n / \alpha_i \in \{z, y\}\}$.
- $(y \circ y)^* = \{(yy)^n / n \in \mathbb{N}_0\}$.
- $(x^* \cup y^*) = \{\alpha^n / \alpha = x \vee \alpha = y, n \in \mathbb{N}_0\}$.
- $[(x \circ x) \cup z] = \{xx, z\}$.
- $[(z \cup y) \cup x] = \{z, y, x\}$.
- $(z \cup y)^* \circ x = \{\alpha_1 \ldots \alpha_n x / \alpha_i \in \{z, y\}, n \in \mathbb{N}_0\}$.
- $[(x \circ x^*) \circ y \circ y^*] = \{x^n y^m / n, m \in \mathbb{N}\}$.
- $[(x \circ x^*) \cup (y \circ y^*)] = \{\alpha^n / \alpha = x \vee \alpha = y, n \in \mathbb{N}\}$.
- $[(x^* \circ y^*) \circ z^*] = \{x^n y^m z^p / n, m, p \in \mathbb{N}_0\}$.

6.4.4.2. Interseccion de lenguajes

- $(x \cup y^*)$ y $(x \cup y)^*$: $x \cup y^*$.
- $[x \circ (x \cup y)^*]$ y $[(x \cup y)^* \circ y]$: $x \circ (x \cup y)^* \circ y$.
- $([(x \cup y) \circ y] \circ [x \cup y]^*)$ y $[y \circ (x \cup y)^*]$: $yy \circ (x \cup y)^*$.
- $[(x \cup y) \circ (x \cup \emptyset)]$ y $[(x \cup y) \circ (x \circ y)^*]$: $x \cup y \cup (x \circ x) \cup (y \circ x)$.

6.4.4.3. Generacion de lenguajes

Cadenas con un numero impar de x: $(x \circ x)^* \circ x$

Cadenas tales que cada y se encuentra entre un par de x: $([x \circ (y \circ x)^*] \cup x^*)^*$

Cadenas que contienen ab o ba: $[(a \cup b)^* \circ a \circ b \circ (a \cup b)^*] \cup [(a \cup b)^* \circ b \circ a \circ (a \cup b)^*]$.

Cadenas que contienen ab y ba:

$[(a \cup b)^* \circ ab \circ (a \cup b)^* \circ ba \circ (a \cup b)^*] \cup [(a \cup b)^* \circ ba \circ (a \cup b)^* \circ ab \circ (a \cup b)^*] \cup [(a \cup b)^* \circ aba \circ (a \cup b)^*]$.

Cadenas que no contienen ab: $a^* \cup (b^* \circ a^*)$.

Cadenas con un numero impar de 1 y par de 0:

$(00 \cup 11 \cup [(01 \cup 10) \circ (00 \cup 11)^* \circ (10 \cup 01)])^* \circ (1 \cup [(01 \cup 10) \circ (11 \cup 00)^* \circ 0])$.

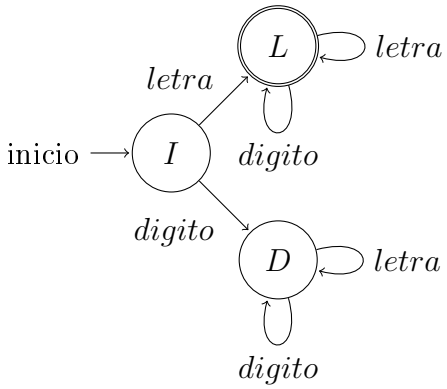
Capítulo 7

Teoria de automatas

7.1. Automatas finitos

7.1.1. Diagrama de transiciones

El siguiente diagrama representa a un automata que acepta cadenas que comienzan con una letra, y siguen con letras o numeros:



Diremos que una cadena es aceptada si sus simbolos corresponden a una secuencia de arcos (flechas) que conducen del circulo inicial a uno doble.

Un diagrama de transiciones puede ser usado como herramienta de diseño para producir rutinas de analisis lexico.

7.1.2. Automata de estado finito determinista

7.1.2.1. Definicion

Un automata de estado FINITO determinista es una quintupla $(\Sigma, S, f, Ac, \sigma)$ donde:

- Σ es un conjunto finito de *simbolos de entradas*.
- S es un conjunto FINITO de *estados*.
- $f : S \times \Sigma \to S$ es una *funcion de transicion*.
- $Ac \subseteq S$ es un conjunto de *estados de aceptacion*.
- $\sigma \in S$ es el estado inicial.

La funcion de transicion del diagrama anterior estaria definida de la siguiente forma:

- $f(\sigma_1, digito) = \sigma_2$.
- $f(\sigma_1, letra) = \sigma_3$.
- $f(\sigma_2, digito) = \sigma_2$.
- $f(\sigma_2, letra) = \sigma_2$.
- $f(\sigma_3, digito) = \sigma_3$.
- $f(\sigma_3, letra) = \sigma_3$.

Observacion Notese que cada estado del diagrama de transiciones de un automata de estado finito DETERMINISTA solo debe tener un arco que salga para cada simbolo del alfabeto; de lo contrario, un automata que llega a ese estado se enfrentara a una eleccion de cual debe ser el arco a seguir. Ademas, dicho diagrama debera estar completamente definido, es decir, debe existir por lo menos un arco para cada simbolo del alfabeto; de lo contrario, un automata que llega a ese estado puede enfrentarse a una situacion donde no puede aplicar ninguna transicion.

Estos requisistos estan reflejados en la definicion formal primero porque f es una funcion y, segundo porque su dominio es $S \times \Sigma$.

7.1.2.2. Palabra aceptada por un AEF

Sean $A = (\Sigma, S, f, Ac, \sigma)$ un AEF y $p = p(1)\,p(2)\ldots p(n)$ una palabra sobre Σ, diremos que dicha palabra es aceptada por el automata A si existen $\sigma_0, \sigma_1, \ldots, \sigma_n \in S$ tales que:

1. $\sigma_0 = \sigma$.
2. $\sigma_i = f(\sigma_{i-1}, p(i))$ para $i = 1, \ldots, n$.
3. $\sigma_n \in Ac$.

Observacion La palabra vacia es aceptada si y solo si el estado inicial es de aceptacion ($\sigma \in Ac$).

7.1.2.3. Funcion de transicion extendida

Dado $A = (\Sigma, S, f, Ac, \sigma)$ un AEF, se define $F : S \times \Sigma^* \to S$ sobre una palabra $p = cl$ donde c es una cadena y l un caracter de forma recursiva:

- $F(s, \lambda) = s$.
- $F(s, p) = f[F(s, c), l]$.

7.1.2.4. Lenguaje aceptado por un AEF

Definimos al lenguaje aceptado por un automata A como el conjunto: $\mathcal{AC}(A) = \{p \in \Sigma^* / p \text{ es aceptada por } A\}$.

7.1.2.5. Equivalencia de automatas

Sean A_1, A_2 automatas de estado finito, diremos que A_1 es equivalente a A_2 y lo notaremos $A_1 \equiv A_2$ si y solo si $\mathcal{AC}(A_1) = \mathcal{AC}(A_2)$, es decir, si y solo si aceptan el mismo lenguaje.

7.1.2.6. Regularidad de lenguajes aceptados por AEF

Enunciado Todo lenguaje aceptado por un AEF es regular, es decir:

$$\{L \in \mathcal{L} / L = \mathcal{AC}(A) \text{ para algun } A \in AEF\} \subseteq \mathcal{L}_3$$

Demostracion Sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF, definimos $G = (N, T, P, \sigma)$ donde $N = S, T = \Sigma, \sigma = \sigma_0$ y reglas de produccion:

- $A \to xB \iff f(A, x) = B$.
- $A \to \lambda \iff A \in Ac$

Debemos probar que $L(G) = \mathcal{AC}(A)$ es decir que:

$$\sigma \Rightarrow^* l_1 l_2 \dots l_n \iff F(\sigma_0, l_1, \dots, l_n) \in Ac$$

Queda como ejercicio al lector completar esta demostracion.

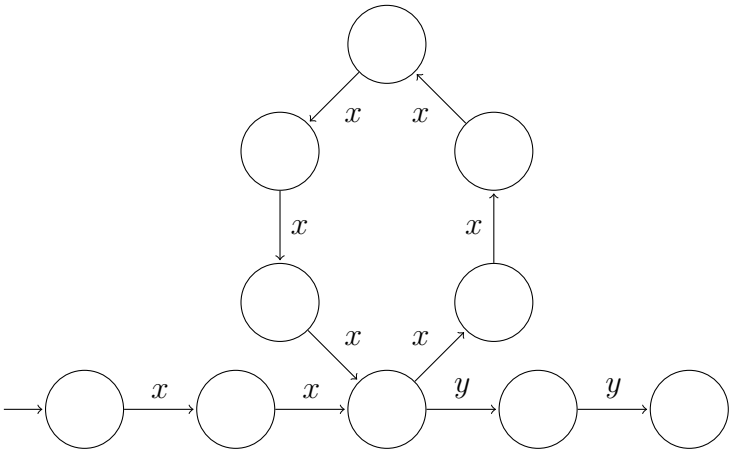
Observacion Este teorema nos indica que los AEF no sirven para reconocer lenguajes independientes de contexto.

7.1.2.7. Lema del bombeo para AEF

Enunciado Sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF, $L = \mathcal{AC}(A)$ y $x, y \in \Sigma$, si L contiene infinitas cadenas de la forma $x^n y^n$ entonces tambien contiene infinitas cadenas de la forma $x^m y^n$ con $m \neq n$.

Demostracion El numero de estados de A ($|S|$) es finito. Sea $n > |S| / x^n y^n \in L$. Observemos que durante el proceso de aceptacion de la palabra $x^n y^n$ (en particular, de la subcadena x^n) el automata A debera pasar sucesivamente por n estados conectados por el caracter x.

Los estados visitados en dicha secuencia no tienen por que ser, en general, necesariamente distintos. Sin embargo, como el numero de transiciones es mayor que el de estados disponibles, al menos un estado debe aparecer repetido como muestra el siguiente diagrama:



De aqui concluimos que A acepta tambien la palabra $x^{n+m}y^n$ (donde m denota la longitud del bucle).

7.1.2.8. No regularidad del lenguaje a^nb^n

Enunciado No existe un AEF tal que $\mathcal{AC}(A) = \{a^nb^n/n \in \mathbb{N}_0\}$.

Demostracion Supongamos por el absurdo que $\exists A \in AEF/\mathcal{AC}(A) = \{a^nb^n/n \in \mathbb{N}_0\}$. Dado que $\mathcal{AC}(A)$ contiene infinitas cadenas de la forma a^nb^n, por el lema del bombeo concluimos que $\mathcal{AC}(A)$ tambien contiene cadenas de la forma a^mb^n con $m \neq n$ lo cual contradice la definicion del lenguaje.

Observacion Una consecuencia de este teorema es que los AEF no sirven como analizadores lexicos de expresiones aritmeticas que contienen parentesis, pues el automata deberia aceptar cadenas de la forma $(^n)^n$ pero de ser asi tambien aceptaria por ejemplo $((^n)^n$ que es incorrecta.

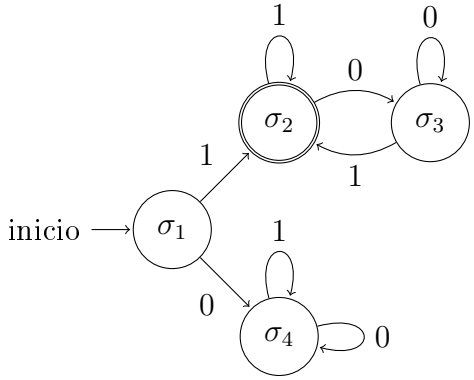
7.1.2.9. Ejemplos

Automata que acepta cadenas binarias que empiezan y terminan en 1

Definimos $\Sigma = \{0, 1\}, S = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}, Ac = \{\sigma_2\}, \sigma = \sigma_1$ y la siguiente funcion de transicion f:

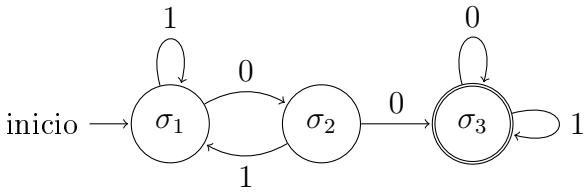
- $f(\sigma_1, 0) = \sigma_4,\ f(\sigma_4, 0) = \sigma_4,\ f(\sigma_4, 1) = \sigma_4.$
- $f(\sigma_1, 1) = \sigma_2,\ f(\sigma_2, 1) = \sigma_2,\ f(\sigma_2, 0) = \sigma_3.$
- $f(\sigma_3, 0) = \sigma_3,\ f(\sigma_3, 1) = \sigma_2.$

La anterior definicion se corresponde con el siguiente diagrama de transiciones:



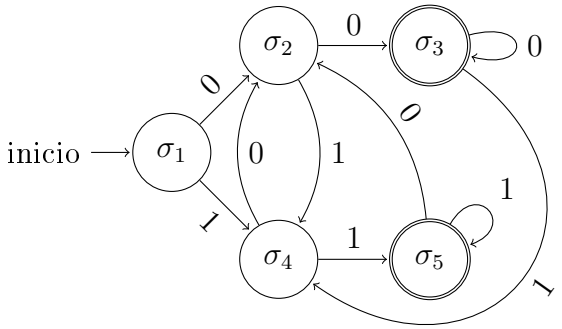
Automata que acepta cadenas binarias que tienen al menos dos ceros seguidos

Definimos el automata mediante el siguiente diagrama de transiciones:



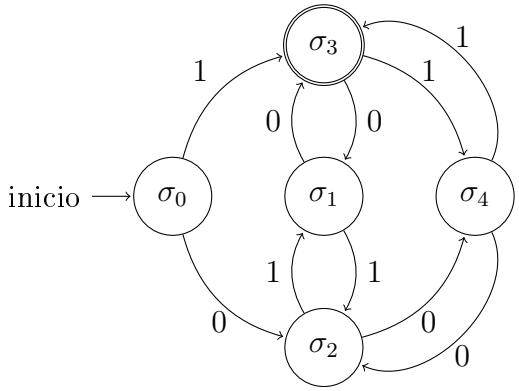
Automata que acepta cadenas binarias que terminan en 00 **o en** 11

Definimos el automata mediante el siguiente diagrama de transiciones:



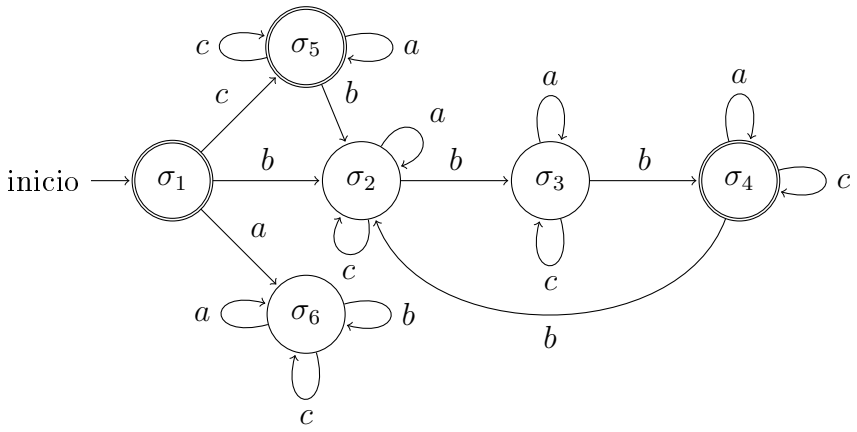
Automata que acepta cadenas binarias con un numero impar de 1 **y par de** 0

Definimos el automata mediante el siguiente diagrama de transiciones:



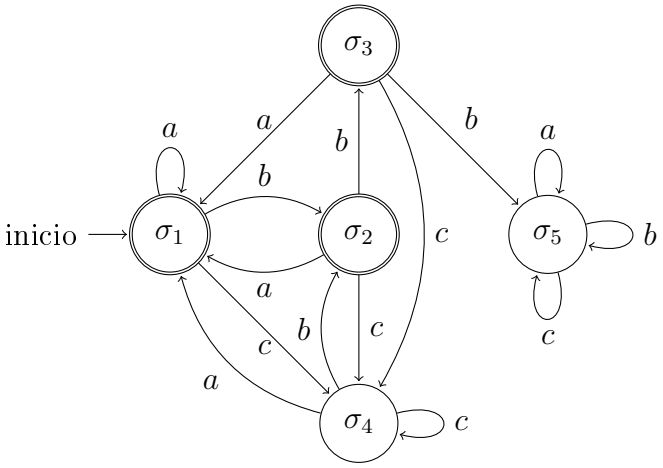
Automata que acepta cadenas sobre $\{a, b, c\}$ con un numero de b que sea multiplo de 3 y no empiece por a

Definimos el automata mediante el siguiente diagrama de transiciones:

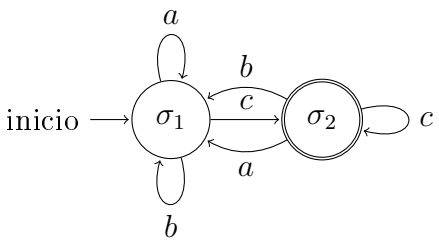


Automata que acepta cadenas sobre $\{a, b, c\}$ con un numero de b consecutivas ≤ 2 pero que no terminen en c

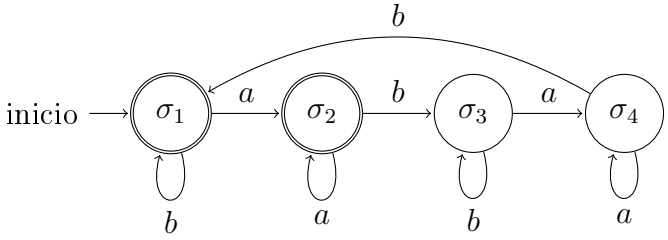
Definimos el automata mediante el siguiente diagrama de transiciones:



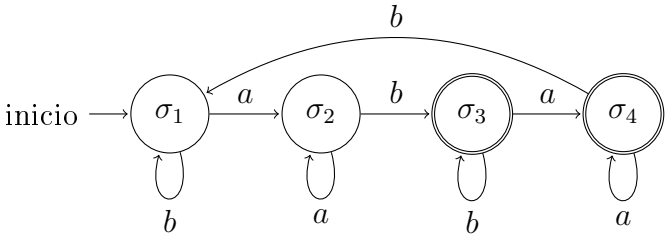
Automata que acepta cadenas sobre $\{a, b, c\}$ que terminan en c



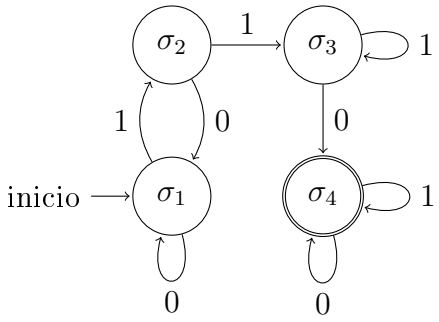
Automata que acepta cadenas sobre $\{a, b\}$ que tienen un numero par de subcadenas ab



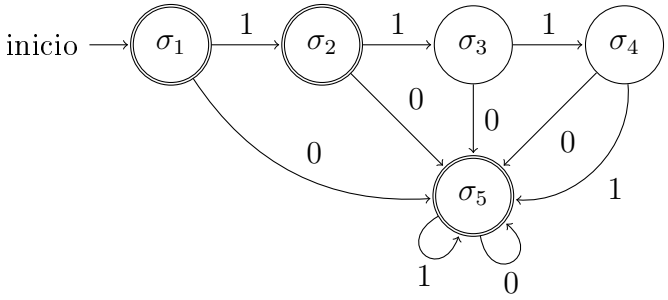
Automata que NO acepta cadenas sobre $\{a, b\}$ que tienen un numero par de subcadenas ab



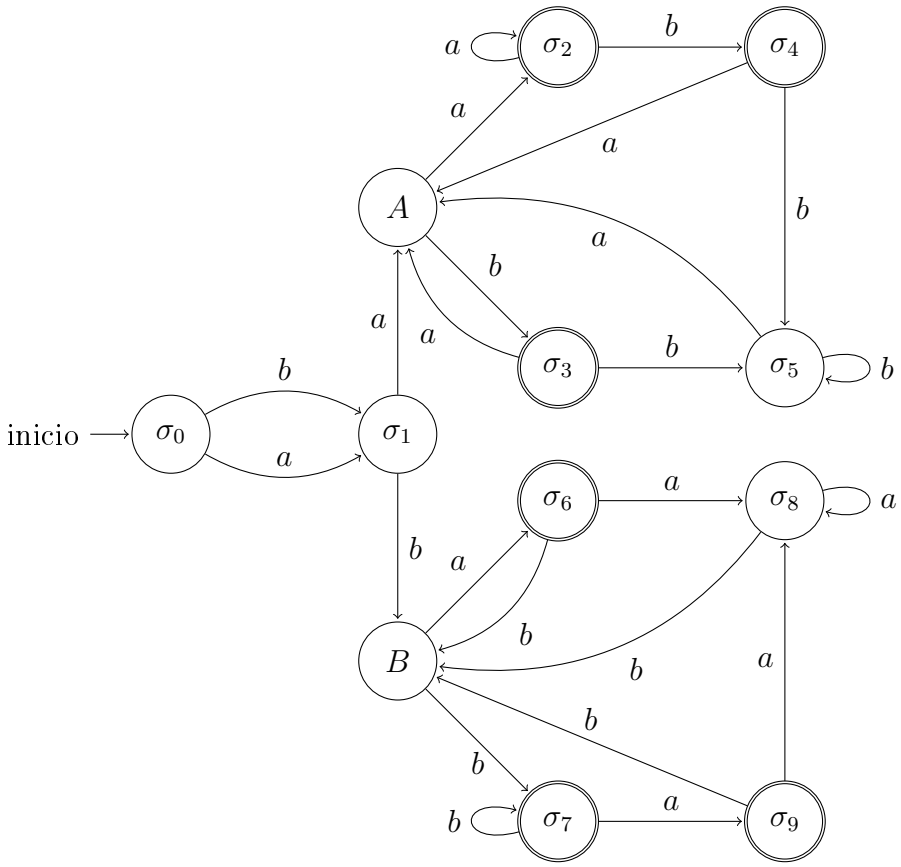
Automata que acepta cadenas binarias que contienen 110



Automata que acepta cadenas binarias distintas de 11 **y** 111



Automata que acepta cadenas de longitud minima 3 sobre $\{a, b\}$ cuya segunda letra es igual a la penultima



7.1.3. Automata de estado finito no determinista

7.1.3.1. Definicion

Un AEF no determinista ($AEFND$) es una quintupla $A = (\Sigma, S, R, Ac, \sigma)$ donde:

- Σ es un conjunto finito de *simbolos de entrada*.
- S es un conjunto finito de *estados*.
- $R \subseteq S \times \Sigma \times S$ es una *relacion de transicion*.
- $Ac \subseteq S$ es un conjunto de *estados de aceptacion*.
- $\sigma \in S$ es el *estado inicial*.

Alternativamente podriamos pensar a R como una funcion $g : S \times \Sigma \to \mathcal{P}(S)$, es decir $g(\sigma, c) = \{\sigma' \in S / (\sigma, c, \sigma') \in R\}$.

Observacion Como hemos cambiado una relacion por una funcion, ahora al leer un nuevo simbolo se podran seguir multiples caminos.

Ademas no es necesario que desde cada estado, se puedan leer todas las letras del alfabeto.

7.1.3.2. Lenguaje aceptado por un $AEFND$

Sea $A = (\Sigma, S, R, Ac, \sigma)$ un $AEFND$ y $p \in \Sigma^*$, luego:

- $p = \lambda \in \mathcal{AC}(A) \iff \sigma \in Ac$.
- $p = p(1)\, p(2) \ldots p(n) \in \mathcal{AC}(A) \iff \exists \sigma_0, \sigma_1, \ldots, \sigma_n$ tales que:
  1. $\sigma_0 = \sigma$.
  2. $(\sigma_{i-1}, p(i), \sigma_i) \in R \ \forall i = 1, \ldots, n$.
  3. $\sigma_n \in Ac$.

A toda secuencia $(\sigma_0, \sigma_1, \ldots, \sigma_n)$ que cumple con (1) y (2) pero no necesariamente con (3) se la llama secuencia que representa a p.

En otras palabras, en un $AEFND$ una cadena p sera:

- Aceptada si existe un camino que la represente y termine en Ac.
- No aceptada si no existe camino que la represente o bien todo camino que la representa termina en $s \in S - Ac$.

7.1.3.3. Equivalencia entre AEF y $AEFND$

Dado que toda funcion es a su vez una relacion, resulta que todo AEF es tambien un $AEFND$, por lo que:

$$\{L/\exists A \in AEF : \mathcal{AC}(A) = L\} \subseteq \{L/\exists A \in AEFND : \mathcal{AC}(A) = L\}$$

Mas precisamente sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF definimos $A' = (\Sigma, S, R, Ac, \sigma_0)$ con R dado por $(P, l, Q) \in R \iff f(P, l) = Q$, luego A' es $AEFND$ y $\mathcal{AC}(A') = \mathcal{A}(A)$.

Veremos a continuacion que en efecto, ambos conjuntos son identicos.

Teorema de Kleene-Rabin-Scott Para cada $AEFND$ existe un AEF que acepta el mismo lenguaje.

Demostracion Sea $A = (\Sigma, S, R, Ac, \sigma_0)$ un $AEFND$ y $A' = (\Sigma, S', f, Ac', s'_0)$ donde:

- $S' = \mathcal{P}(S)$.
- $Ac' = \{s' \in \mathcal{P}(S) / s' \cap Ac \neq \emptyset\}$.
- $s'_0 = \{s_0\}$.
- $f : S'\Sigma \to S'$ tal que $f(s', l) = \{s \in S/\exists u \in s' : (u, l, s) \in R\}$.

Debemos mostrar ahora que $\mathcal{AC}(A) = \mathcal{AC}(A')$.

Para mostrar que $p \in \mathcal{AC}(A) \iff p \in \mathcal{AC}(A')$ haremos induccion sobre la cantidad de letras de p, mas precisamente mostraremos que $\forall n \in \mathbb{N}_0$ vale el siguiente enunciado: «*Para cada ruta en A que va de su estado inicial s_0 a un estado s_n, existe una ruta en A' que va de su estado inicial $s'_0 = \{s_0\}$ a un estado s'_n tal que $s_n \in s'_n$. Reciprocamante, para cada ruta en A' que va de s'_0 a un estado s'_n , y para cada $s_n \in s'_n$, existe una ruta en A que va de s_0 a s_n*».

- Caso base $n = 0$: Trivial. En A, $s_n = s_0$ corresponde al caso en que la entrada es λ y en A' $s'_n = s'_0$.
- Paso inductivo: Supongamos que vale E_n y veamos que vale E_{n+1}. Consideremos una ruta en A de la forma $s_0, \ldots, s_n, s_{n+1}$ que recorre los arcos rotulados $l_1, \ldots l_{n+1}$. Para todas las transiciones sobre esta ruta

tenemos que $(s_i, l_{i+1}, s_{i+1}) \in R$; en particular para la ultima de ellas $(s_n, l_{n+1}, s_{n+1}) \in R$.
Por H. I. existe una ruta en A' de la forma $s'_0, \ldots, s'_n$ tal que $s_n \in s'_n$ y dado que $(s_n, l_{n+1}, s_{n+1}) \in R$ existe un arco en A' rotulado l_{n+1} de s'_n a un estado que contiene a s_{n+1}. Llamemoslo s'_{n+1}. Por lo tanto existe una ruta en A', de s'_0 a s'_{n+1} tal que $s_{n+1} \in s'_{n+1}$.
Reciprocamente consideremos una ruta en A' de la forma $s'_0, \ldots, s'_n, s'_{n+1}$ que recorre los arcos rotulados $l_1, \ldots, l_{n+1}$. Para todas las transiciones sobre esta ruta tenemos que $f(s'_i, l_{i+1}) = s'_{i+1}$; en particular, para la ultima de ellas $f(s'_n, l_{n+1}) = s'_{n+1}$.
Por H. I., para cada $s_n \in s'_n$ debe existir una ruta en A que va de s_0 a s_n y dado que $f(s'_n, l_{n+1}) = s'_{n+1}$, por definicion de f tenemos que s'_{n+1} es el conjunto de estados $s \in S$ a los que se puede llegar desde un estado de s'_n siguiendo un arco con etiqueta l_{n+1}. Por lo tanto, para cada $s \in s'_{n+1}$ existe una ruta en A que va desde s_0 a s.

7.1.3.4. Relacion entre AEF, $AEFND$ y $\mathcal{L}_3$

Puesto que toda funcion es ademas relacion sabiamos que $\mathcal{AC}(AEF) \subseteq \mathcal{AC}(AEFND)$ y por el teorema de Kleene-Rabin-Scott sabemos que $\mathcal{AC}(AEFND) \subseteq \mathcal{AC}(AEF)$ por lo que $\mathcal{AC}(AEF) = \mathcal{AC}(AEFND)$.

Ademas tambien probamos que dado un AEF existe una gramatica regular que genera el mismo lenguaje. En sintesis: $\mathcal{AC}(AEFND) = \mathcal{AC}(AEF) \subseteq \mathcal{L}_3$. A continuacion veremos que estos tres conjuntos son iguales.

Enunciado Para todo lenguaje regular, existe un $AEFND$ que lo acepta.

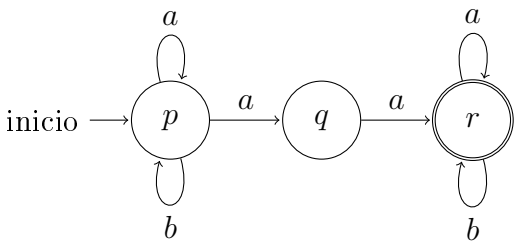
Demostracion Queda como ejercicio al lector completar esta demostracion.

Conclusion Como $\mathcal{AC}(AEFND) = \mathcal{AC}(AEF) \subseteq \mathcal{L}_3$ y acabamos de probar $\mathcal{L}_3 \subseteq \mathcal{AC}(AEFND)$ resulta: $\mathcal{AC}(AEF) = \mathcal{AC}(AEFND) = \mathcal{L}_3$.

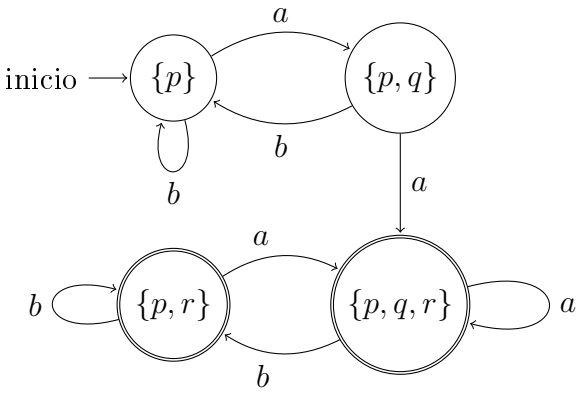
7.1.3.5. Ejemplos

Construccion de un AEF a partir de un $AEFND$

Consideremos el siguiente $AEFND$:



Es equivalente al siguietne AEF:



7.2. Automatas de pila

7.2.1. Introduccion

Los automatas de pila son automatas que cuentan con las mismas caracteristicas que un $AEFND$ pero ademas disponen de una pila de memoria en la que podran leer o escribir simbolos, de forma de poder saber en cada momento si el automata ya realizo alguna transicion en el pasado.

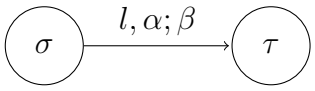
Durante cada transicion el automata ejecuta la siguiente secuencia:

1. Leer un simbolo de entrada.
2. Extraer un simbolo de la pila.
3. Insertar un simbolo en la pila.
4. Pasar a un nuevo estado.

A este proceso lo representaremos con la notacion $(\sigma, l, \alpha; \beta, \tau)$ donde:

- σ es el estado actual.
- l es el simbolo del alfabeto que se lee en la entrada.
- α es el simbolo que se extrae de la pila.
- β es el simbolo que se inserta en la pila.
- τ es el nuevo estado al que pasa el automata.

Representaremos este proceso mediante el siguiente diagrama de transiciones:



Puesto que permitimos que l, α, β sean la cadena vacia, podemos en cualquier transicion no leer un caracter, no extraer nada de la pila o no escribir nada en la pila si asi lo necesitaramos.

Llamaremos a las transiciones de la forma $(\sigma, \lambda, \lambda; \lambda, \tau)$ transiciones espontaneas.

Observacion Notese que las transiciones representada por $(\sigma, l, \lambda; \lambda, \tau)$ son las que realiza un $AEFND$. Por lo tanto los $AEFND$ son un caso particular de AP y en consecuencia los lenguajes regulares son un subconjunto de los lenguajes aceptados por AP, es decir: $\mathcal{L}_3 \subseteq \mathcal{AC}(AP)$.

7.2.2. Definicion formal

Un automata de pila (AP) es una sextupla $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ donde:

- S es un conjunto finito de *estados*.
- Σ es un conjunto finito de *simbolos de entrada*.
- Γ es un conjunto finito de *simbolos de pila*.
- $T \subseteq S \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times S$ es una *relacion de transicion*.
- $\sigma \in S$ es el *estado inicial*.
- $Ac \subseteq S$ es un conjunto de estados de aceptacion.

7.2.3. Configuracion

Una configuracion de un automata de pila $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ es un elemento de $\mathcal{C}_A = S \times \Sigma^* \times \Gamma^*$.

Dicha configuracon (σ, p, γ) indica que el automata esta en el estado σ, le falta ller la cadena p de la entrada y el contenido completo de la pila es γ.

7.2.4. Relacion entre configuraciones

Para una automata de pila $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ definimos la relacion «lleva en un paso» (que notaremos $\Rightarrow_A$) entre configuraciones, de la siguiente forma: $(\sigma, lp, \alpha\gamma) \Rightarrow_A (\tau, p, \beta\gamma) \iff (\sigma, l, \alpha; \beta, \tau) \in T$.

La relacion «lleva en uno o mas pasos» ($\Rightarrow_A^*$) define recursivamente a partir de la relacion $\Rightarrow_A$ de forma analoga a lo hecho para la funcion de transicion f de los AEF.

7.2.5. Lenguaje aceptado

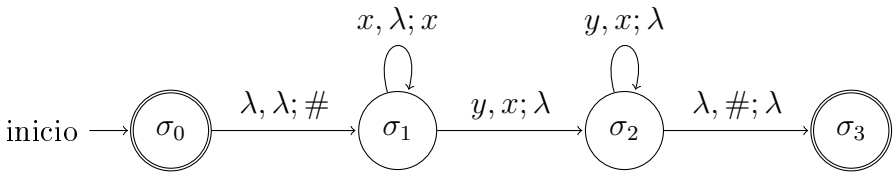
Sea $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ un automata de pila, el lenguaje aceptado por A es el conjunto: $\mathcal{AC}(A) = \{p \in \Sigma^* / (\sigma, p, \lambda) \Rightarrow^* (\tau, \lambda, \gamma) : \gamma \in \Gamma^* \wedge \tau \in Ac\}$.

Es decir, una cadena p sera aceptada por un AP si, arrancando desde su estado inicial y con la pila vacia, es posible que el automata llegue a un estado de aceptacion despues de leer toda la cadena.

Observacion No necesariamente se llegara a un estado de aceptacion luego de leer el ultimo caracter de una palabra pues a continuacion el automata podria realizar transiciones de la forma $(\sigma, \lambda, \alpha; \beta, \tau)$ y llegar luego al estado de aceptacion.

7.2.6. Relacion entre $\mathcal{L}_3$ y $\mathcal{AC}(AP)$.

Ya hemos visto que $\mathcal{L}_3 \subseteq \mathcal{AC}(AP)$. Sin embargo esta inclusion es estricta, pues como veremos, el lenguaje $\{x^n y^n / n \in \mathbb{N}_0\}$ es aceptado por el siguiente automata de pila:



7.2.7. Teorema del vaciado de pila

Enunciado Para cada $A \in AP$ existe $A' \in AP$ tal que A' vacia su pila y ademas $\mathcal{AC}(A') = \mathcal{AC}(A)$.

Demostracion Sea $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ un AP, fabricaremos A' de la siguiente manera:

- El estado inicial de A deja de serlo, pues introducimos un nuevo estado inicial y una transicion del nuevo al anterior que lo unico que hace es insertar en la pila un marcador # (suponiendo que $\# \notin \Gamma$).
- Los estados de aceptacion de A dejan de serlo e introduciremos un nuevo estado P junto con transiciones espontaneas que pasan de cada uno de los antiguos estados de aceptacion al nuevo estado P.
- Vaciamos la pila sin salir del estado P introduciendo transiciones de la forma $(P, \lambda, x; \lambda, P)$ para cada $x \in \Gamma$.
- Agregamos un nuevo y unico estado de aceptacion Q junto a la transicion $(P, \lambda, \#; \lambda, Q)$.

Formalmente definimos $A' = (S', \Sigma, \Gamma', T', \sigma', Ac')$ donde:

- $S' = S \cup \{R, P, Q\}$ donde $R, P, Q \notin S$.
- $\Gamma' = \Gamma \cup \{\#\}$ donde $\# \notin \Gamma$.
- $\sigma'_0 = R$.
- $Ac' = \{Q\}$.
- $$\begin{aligned} T' = T &\cup \{(R, \lambda, \lambda; \#, \sigma)\} && (1)\\ &\cup \{(\tau, \lambda, \lambda; \lambda, P) / \tau \in Ac\} && (2)\\ &\cup \{(P, \lambda, \alpha; \lambda, P) / \alpha \in \Gamma\} && (3)\\ &\cup \{(P, \lambda, \#; \lambda, Q)\}. && (4) \end{aligned}$$

Veamos ahora que $\mathcal{AC}(A) = \mathcal{AC}(A')$:

- $\subseteq$: Sea $\alpha \in \mathcal{AC}(A)$. Sabemos que partiendo del estado inicial σ con la pila vacia, α nos lleva en uno o mas pasos a un estado de aceptacion. En terminos de configuraciones: $(\sigma, \alpha, \lambda) \Rightarrow^* (\tau, \lambda, \gamma)$ donde $\tau \in Ac, \gamma \in \Gamma^*$. Luego en A' tenemos la derivacion:

  $$(R, \alpha, \lambda) \Rightarrow^{(1)} (\sigma, \alpha, \#) \Rightarrow^* (\tau, \lambda, \gamma\#) \Rightarrow^{(2)} (P, \lambda, \gamma\#) \Rightarrow^{(3)*} (P, \lambda, \#) \Rightarrow^{(4)} (Q, \lambda, \lambda)$$

  Como $Q \in Ac'$ concluimos que $\alpha \in \mathcal{AC}(A')$ y la pila queda vacia.

- $\supseteq$: Analogo.

7.2.8. Relacion entre $\mathcal{L}_2$ y automatas de pila

Enunciado

1. Sea G una gramatica independiente de contexto entonces existe un automata de pila A tal que $L(G) = \mathcal{AC}(A)$.
2. Sea A un automata de pila entonces existe una gramatica independiente del contexto G tal que $L(G) = \mathcal{AC}(A)$.

Es decir $\mathcal{L}_2 = \mathcal{AC}(AP)$.

Demostracion

1. Consultar bibliografia [2], pag 85.
2. Consultar bibliografia [2], pag 90.

Conclusion De todo lo que hemos visto hasta ahora sabemos que:

$$\mathcal{L}_3 = \mathcal{AC}(AEF) = \mathcal{AC}(AEFND) = L(ER) \subset \mathcal{L}_2 = \mathcal{AC}(AP)$$

7.2.9. Lema de bombeo para automatas de pila

Enunciado Sea $L \in \mathcal{L}_2$, luego si L es infinito existe $p \in L$ de la forma $p = xuyvz$ donde $uv \neq \lambda$ tal que $xu^n yv^n z \in L \; \forall n \in \mathbb{N}$.

Demostracion Consultar bibliografia [2] pag. 102.

7.2.10. Corolario

Enunciado Existen lenguajes sensibles al contexto.

Demostracion Probaremos que $L = \{a^n b^n c^n / n \in \mathbb{N}\} \notin \mathcal{L}_2$.

Como L es inifinito el lema de bombeo nos permite asegurar que existe una cadena $xuyvz \in L$ tal que $xu^n yv^n z \in L \; \forall n \in \mathbb{N}$ con $uv \neq \lambda$. Supongamos sin perder generalidad que $u \neq \lambda$ luego hay dos posibilidades:

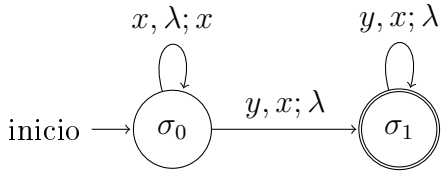
- u esta formado por un solo simbolo y al bombearlo se obtiene una palabra que no mantiene la igualdad entre exponentes resultando no pertenecer al lenguaje.
- u esta formado por mas de un caracter y al bombearlo se obtiene una palabra que altera el orden de los simbolos y por lo tanto no pertenece al lenguaje.

Llegamos al absurdo por cualquiera de las dos posibilidades, por lo que $L \notin \mathcal{L}_2$.

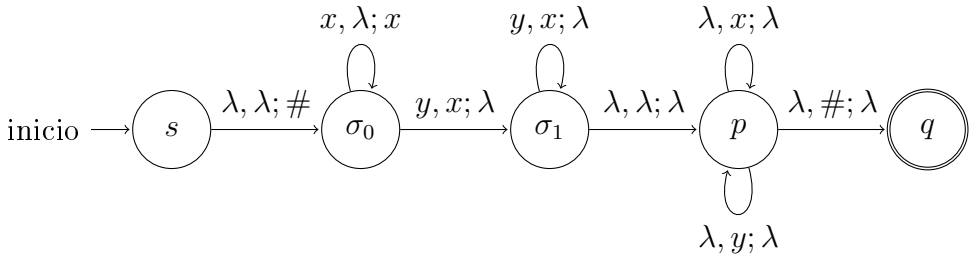
7.2.11. Ejemplos

7.2.11.1. Automatas de pila

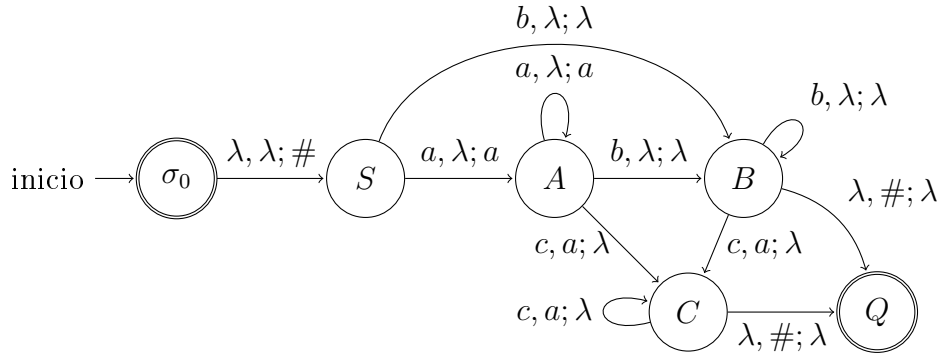
Automata que acepta el lenguaje $\{x^n y^k / n, k \in \mathbb{N}, k \leq n\}$



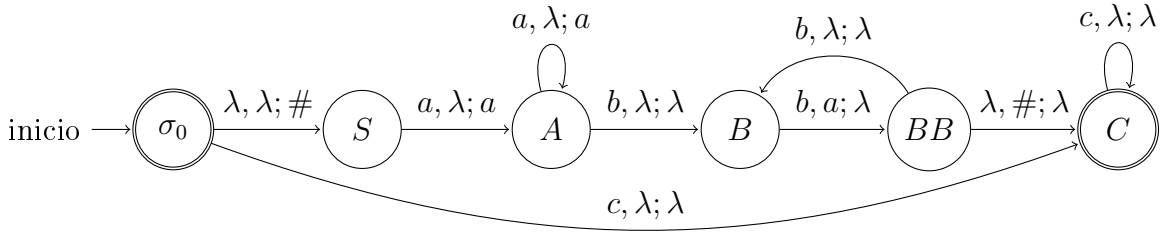
Automata que acepta el lenguaje $\{x^n y^k / n, k \in \mathbb{N}, k \leq n\}$ y vacia la pila



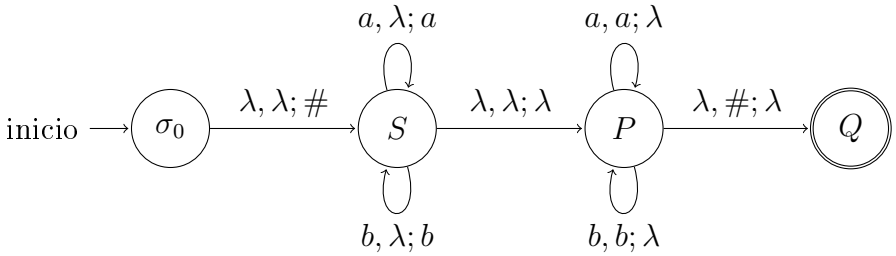
Automata que acepta el lenguaje $\{a^n b^m c^n / n, m \in \mathbb{N}_0\}$ y vacia la pila



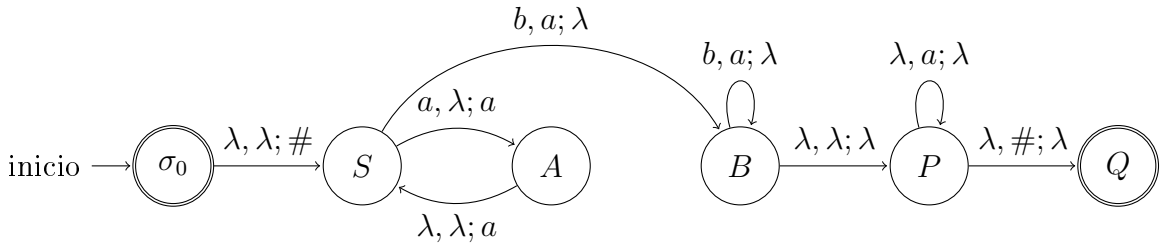
Automata que acepta el lenguaje $\{a^n b^{2n} c^m / n, m \in \mathbb{N}_0\}$ y vacia la pila



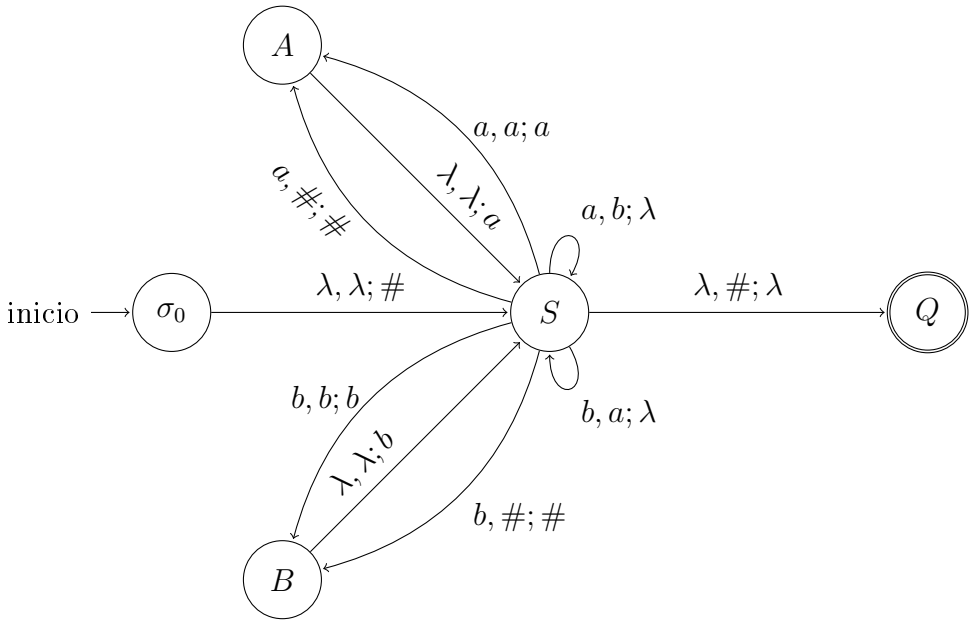
Automata que acepta el lenguaje $\left\{ww^R / w \in \{a, b\}^*\right\}$ y vacia la pila



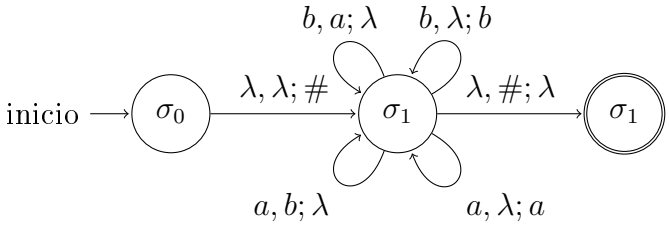
Automata que acepta el lenguaje $\{a^m b^n / m, n \in \mathbb{N}_0 \wedge n \leq 2m\}$ y vacia la pila



Automata que acepta el lenguaje $\{p \in \{a,b\}^* / N_a(p) = N_b(p)\}$ y vacia la pila



O en forma mas sencilla:



7.2.11.2. Lenguajes no libres de contexto

- El lenguaje $L = \{a^n b^n c^n / n \in \mathbb{N}_0\}$ no es libre de contexto. En efecto, supongamos que lo es y consideremos la cadena $xuyvz \in L / uv \neq \lambda$:
  - Si u o v tienen mas de un simbolo, al bombearlos se obtendran subcadenas del tipo *abab*, *bcbc* o *abcabc*. Contradiccion.
  - Si $u = v$ tienen un solo simbolo entonces al bombearlos, podemos incrementar el exponente de este simbolo, independientemente de los otros. Contradiccion.
  - Si $u \neq v$ tienen un solo simbolo entonces al bombearlos, los restantes tendran un exponente diferente. Contradiccion.
- El lenguaje $L = \{0^n \# 0^{2n} \# 0^{3n} / n \in \mathbb{N}_0\}$ no es libre de contexto. En efecto, supongamos lo es y consideremos la cadena $xuyvz \in L / uv \neq \lambda$. Observemos que ninguna palabra termina ni empieza con # por lo que sabemos que $x \neq \#$ y $z \neq \#$. Luego:
  - Si u o v tienen un simbolo # entonces al bombearlo podemos generar mas de 2 simbolos #. Contradiccion.
  - Si $u = \lambda \Rightarrow xuyvz = 0^i yvz$ y puesto que toda palabra tiene dos simbolos # entonces $y = z = \#$. Contradiccion.
  - Analogamente si $v = \lambda \Rightarrow xuyvz = xuy0^i$, entonces $x = y = \#$. Contradiccion.
  - La posibilidad restante es $xuyvz = 0^i 0^j y 0^k 0^l \Rightarrow xuyvz = 0^i 0^j \# 0^k 0^l$. Contradiccion.

7.3. Maquinas de Turing

7.3.1. Descripcion

Al igual que los demas automatas que hemos estudiado, la maquina de Turing contiene un mecanismo de control que en cualquier momento puede encontrarse en uno de entre un numero finito de estados. Uno de estos estados se denomina estado inicial y representa el estado en el cual la maquina comienza los calculos. Por convencion lo notaremos q_1.

Otro de los estados se conoce como estado de parada; una vez que la maquina llega a ese estado, terminan todos los calculos. De esta manera, el estado de parada de una maquina de Turing difiere de los estados de aceptacion de los automatas de estados finito y los automatas de pila en que estos pueden continuar sus calculos despues de llegar a un estado de aceptacion, mientras que en una maquina de Turing debe detenerse en el momento en que llegue a su estado de parada. Notaremos a este estado como q_0.

Notese que con base en la definicion anterior, el estado inicial de una maquina de Turing no puede ser a la vez el estado de parada; por lo tanto, toda maquina de Turing debe tener cuanto menos dos estados.

Una diferencia mas importante entre una maquina de Turing y los automatas de los capitulos anteriores es que la maquina de Turing puede leer y escribir en su medio de entrada. Para ser mas precisos, la maquina de Turing esta equipada con un cabezal que puede emplearse para leer y escribir simbolos en la cinta de la maquina, que es infinita a izquierda y derecha, pero se conviene que solo un conjunto finito de casillas no estan vacias. Asi una maquina de Turing puede emplear su cinta como almacenamiento auxiliar tal como lo hacen los automatas de pila. Sin embargo con este almacenamiento una maquina de Turing no se limita a las operaciones de insercion y extraccion, sino que puede rastrear los datos de la cinta y modificar las celdas que desee sin alterar las demas.

Al utilizar la cinta para fines de almacenamiento auxiliar, es conveniente que una maquina de Turing emplee marcas especiales para distinguir porciones de la cinta. Para esto, permitimos que una maquina de Turing lea y escriba simbolos que no aparecen en los datos de entrada; es decir, hacemos una distincion entre el conjunto (finito) de simbolos, llamado alfabeto de la maquina, en el que deben estar codificados los datos de entrada iniciales, y un conjunto, posiblemente mayor (tambien finito), de simbolos de la cinta, que la maquina puede leer y escribir. Esta distintion es similar a la que se establece entre el alfabeto de un automata y sus simbolos de pila. El simbolo blanco es un simbolo de la cinta que esta en cualquier celda de la cinta que no este ocupada. Notaremos a este simbolo como $\square$.

Las accion especifica que realiza una maquina de Turing consiste en tres operaciones consecutivas:

1. Substituye el simbolo apuntado por el cabezal por otro del alfabeto de la cinta que eventualmente podra ser el mismo.

2. Mueve el cabezal hacia la derecha, izquierda o permanece en la misma posicion (notaremos d, i, n respectivamente).

3. Cambia el estado en que se encuentra por otro perteneciente al conjunto de estados que eventualmente podra ser el mismo.

La accion que se ejecutara en un momento dado dependera del simbolo apuntado por el cabezal asi como del estado actual del mecanismo de control de la maquina. Si representamos con Γ al conjunto de simbolos de la cinta, con S al conjunto de estados y $S' = S - \{q_0\}$ entonces es posible representar las transiciones de la maquina mediante una funcion $f : S' \times \Gamma \to \Gamma \times \{d, i, n\} \times S$. Podemos entonces interpretar $f(q_r, \alpha) = (\beta, d, q_n)$ como «si el estado actual es q_r y el simbolo actual es α: reemplazar α por β, mover el cabezar a la derecha y pasar al estado q_n».

Esta funcion, de dominio finito, puede ser representada por una tabla de $|S|$ columnas y $|\Gamma|+1$ filas. En cada una de las casillas i, j asociadas al estado q_j y el simbolo s_i aparecera una terna que indicara que simbolo escribir, el movimiento a ejecutar y el estado al que pasar. La primer columna de la tabla correspondera al estado q_1.

Notese que al describir con una funcion las transiciones de una maquina de Turing, la maquina es determinista. Para ser mas precisos, existe una y solo una transicion asociada a cada par estado-simbolo, donde el estado no es el de detencion.

Durante la operacion normal, una maquina de Turing ejecuta transiciones repetidamente hasta llegar al estado de parada. Esto quiere decir que en ciertas condiciones es posible que nunca se detengan los calculos de una maquina de Turing, ya que su programa interno puede quedar atrapado en un ciclo sin fin.

7.3.2. Definicion formal

Una maquina de Turing (MT) es una septupla $M = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ donde:

- S es un conjunto finito de estados.
- Σ es un conjunto finito de simbolos llamado alfabeto de la maquina.
- Γ es un conjunto finito de simbolos llamados simbolos de la cinta, tal que $\Sigma \subseteq \Gamma$.
- $\square \in \Gamma$ es el simbolo blanco.
- $q_1 \in S$ es el estado inicial.
- $q_0 \in S$ es el estado de parada.
- $f : S - \{q_0\} \times \Gamma \to \Gamma \times \{d, i, n\} \times S$ es la funcion de transicion de la maquina.

7.3.3. Maquinas elementales

Es posible demostrar que con un alfabeto de un solo simbolo (ademas del blanco), se puede codificar cualquier alfabeto finito; por lo tanto, de ahora en adelante trabajaremos con maquinas de Turing sobre el alfabeto $\Sigma = \{\bullet\}$. Definiremos a continuacion las siguientes maquinas elementales:

- Maquina «mover a la derecha» (que notaremos D):

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

- Maquina «mover a la izquierda» (que notaremos I):

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

- Maquina «blanco» (que notaremos $\square$):

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

- Maquina «punto» (que notaremos $\bullet$):

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

- Maquina «nada» (que notaremos N):

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

A partir de estas maquinas elementales y realizando composiciones se podra construir cualquier maquina de Turing sobre el alfabeto dado.

7.3.4. Composicion

Sean $S = \{q_0, q_1, \ldots, q_n\}$, $S' = \{q'_0, q'_1, \ldots, q'_m\}$, $\Sigma = \{\bullet\}$ y $\Gamma = \{\bullet, \square\}$ definimos $M = (S, \Sigma, \Gamma, f_M, \square, q_1, q_0)$ y $M' = (S', \Sigma, \Gamma, f_{M'}, \square, q'_1, q'_0)$. Llamaremos composicion de M con M' a la maquina $MM' = (S \cup S' - \{q_0\}, \Sigma, \Gamma, f, \square, q_1, q'_0)$ que realiza la operacion equivalente a aplicar primero la maquina M y al resultado aplicar M', donde f esta dada por:

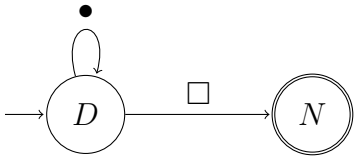
$$f(q,s) \begin{cases} f_M(s,q)[q_0|q'_1] & q \in S \\ f_{M'}(s,q) & q \in S' \end{cases}$$

Es sencillo construir, a partir de las tablas de las maquinas M y M' la tabla de MM'. Para ello se adjunta inmediatamente a la tabla de M, la tabla de M' y se reemplaza en la tabla de M cada aparicion de q_0 por q'_1.

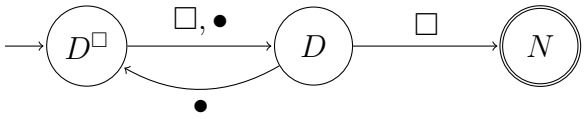
7.3.5. Diagramas de composicion

Veamos una forma mas general de componer maquinas. En este caso ante el estado final de una maquina se toma en consideracion el símbolo observado, y dependiendo del mismo, se la conecta con el estado inicial de una determinada maquina (eventualmente la misma). Esto se diagrama con una flecha, marcada con el simbolo que parte de la maquina que ha terminado hacia la que continua con el proceso. Si no se etiqueta la flecha de salida implica que en todos los casos no etiquetados se pasa al estado inicial de la maquina a la que apunta la flecha. Veamos algunos ejemplos:

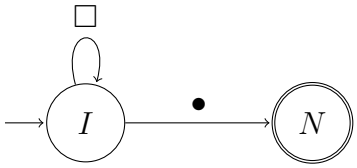
- Maquina «derecha hasta blanco» (que notaremos $D^{\square}$):



- Maquina «derecha hasta dos blancos» (que notaremos $D^{\square\square}$):



- Maquina «izquierda hasta punto» (que notaremos $I^{\bullet}$):



De forma analoga definimos las maquinas $I^{\square}, I^{\bullet}, D^{\bullet}, \ldots$.

7.3.6. Lenguajes recursivos y recursivamente enumerables

Diremos que un lenguale $\mathcal{L}$ sobre un alfabeto Σ es decidible (o recursivo), si existe una maquina de Turing $M_{\mathcal{L}} = (S, \Sigma, \Sigma \cup \{\square, \circ, \bullet\}, f, \square, q_1, q_0)$ tal que si la cinta inicial contiene una palabra $p \in \Sigma^*$ entonces la cinta final sera $\ldots \square \circ \square \ldots$ si $p \in \mathcal{L}$ o bien $\ldots \square \bullet \square \ldots$ si $p \notin \mathcal{L}$.

Cualquier maquina de Turing reconocedora de cadenas es una maquina de Turing calculadora de la funcion caracteristica del lenguaje que reconoce: es una funcion que asocia el valor $\circ$ a las cadenas que pertenecen al lenguaje y $\bullet$ a las cadenas que no pertenecen al lenguaje. Ademas cualquier maquina de Turing calculadora de funciones es una maquina de Turing reconocedora de lenguajes, ya que se puede representar cada funcion por el lenguaje formado por las tuplas que se pueden formar con sus parametros de entrada y de salida.

Por ejemplo la funcion suma se puede representar como el conjunto de tripletas ordenadas $\{(0,0,0), (0,1,1), (0,2,2), \ldots, (1,9,10), \ldots\}$. Este len-

guaje estara formado por las cadenas tales que el tercer caracter representa la suma de los dos primeros.

Un lenguaje recursivamente enumerable es un lenguaje formal para el cual existe una máquina de Turing que acepta y se detiene con cualquier cadena del lenguaje. Pero que puede parar y rechazar, o bien iterar indefinidamente, con una cadena que no pertenece al lenguaje, en contraposición a los lenguajes recursivos en cuyo caso se requiere que la máquina de Turing pare en todos los casos.

Todos los lenguajes regulares, independientes de contexto, dependientes de contexto y recursivos son recursivamente enumerables.

7.3.7. Representacion de FRL con MT

Mostraremos que dada una funcion de lista, existe una maquina de Turing sobre $\Sigma = \{\square, \bullet\}$ que la representa. Para ello definiremos primero una representacion de una lista, en la cinta de la maquina.

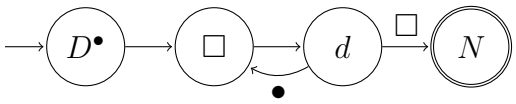
Representaremos entonces las listas como grupos de $\bullet$ separados por un solo simbolo blanco, donde por cada numero n en la lista, tendremos $n+1$ puntitos. Por ejemplo la lista $[2, 0, 5, 1]$ sera representada por la cinta $\bullet\bullet\bullet\square\bullet\square\bullet\bullet\bullet\bullet\bullet\bullet\square\bullet\bullet$. Notaremos la representacion de una lista $X \in \mathcal{L}$ en una cinta de una maquina de Turing como $\langle\langle X \rangle\rangle$.

Diremos que dada una funcion de lista F, una maquina de Turing (que notaremos M_F) la representa si dada cualquier lista $X \in Dom\,(F)$, la maquina M_F tomando como configuracion inicial $\langle\langle X \rangle\rangle$ y con el cabezal en la primer casilla vacia antes del primer $\bullet$, luego de procesarla llega a la configuracion final $\langle\langle FX \rangle\rangle$, observando la primer casilla libre de dicha configuracion.

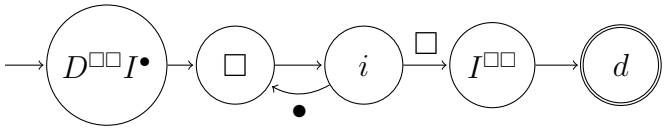
7.3.7.1. Funciones base

- La funcion 0_i esta representada por la maquina $i \bullet i$.
- La funcion 0_d esta representada por la maquina $D^{\square\square} \bullet I^{\square\square} d$.
- La funcion S_i esta representada por la maquina $D^{\bullet} i \bullet i$.
- La funcion S_d esta representada por la maquina $D^{\square\square} I^{\bullet} d \bullet I^{\square\square} d$.

- La funcion $\square_i$ esta representada por la maquina:



- La funcion $\square_d$ esta representada por la maquina:



7.3.7.2. Composicion

Sean $f, g \in FL/\exists M_f, M_g \in MT$, entonces la funcion fg esta representada por la maquina $M_f M_g$.

7.3.7.3. Maquina Z

Introduccion La maquina Z es una maquina de Turing sobre $\Sigma = \{\bullet, \square\}$ que parte en la primer celda vacia a la izquierda del primer caracter, cuyo comportamiento es el siguiente:

- Si la cinta inicial no contiene al menos dos grupos de $\bullet$ separados por un solo $\square$, entonces la maquina no para.
- Si el primer grupo de $\bullet$ tiene la misma cantidad de elementos que el ultimo, entonces la configuracion final es igual a la inicial.
- Si los grupos primero y ultimo no tienen el mismo numero de elementos, entonces la configuracion final es igual a la inicial, pero con el cabezal apuntando al primer $\bullet$.

Definiciones Definiremos primero cinco maquinas:

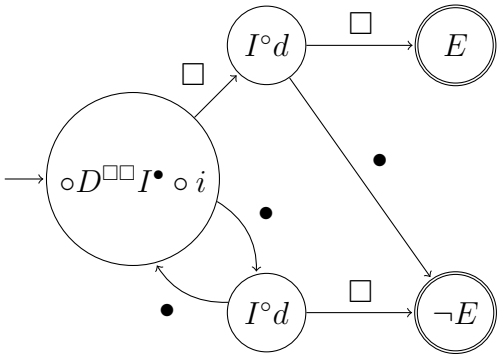
1. La maquina preambulo P, que no para si la entrada no es correcta.
2. La maquina contadora C, que vacia los puntitos de a pares, mientras se pueda.
3. La maquina igualdad E, que actua si el conteo fue igual.

4. La maquina desigualdad $\neg E$, que actua si el conteo fue desigual.
5. La maquina restauradora R, que vuelve a rellenar los puntitos.

Maquina P Definimos $P := D^{\square}D^{\bullet}I^{\square\square}dd$. Veamos algunos ejemplos:

- $\check{\square}\bullet\bullet\bullet\underbrace{\Rightarrow}_{D^{\square}}\bullet\bullet\bullet\check{\square}\underbrace{\Rightarrow}_{D^{\bullet}}\dots$
- $\check{\square}\bullet\bullet\square\bullet\bullet\underbrace{\Rightarrow}_{D^{\square}}\bullet\bullet\check{\square}\bullet\bullet\underbrace{\Rightarrow}_{D^{\bullet}}\bullet\bullet\square\check{\bullet}\bullet\underbrace{\Rightarrow}_{I^{\square\square}}\check{\square}\square\bullet\bullet\square\bullet\bullet\underbrace{\Rightarrow}_{dd}\check{\bullet}\bullet\square\bullet\bullet$

Maquina C Definimos C mediante el siguiente diagrama:



Veamos algunos ejemplos:

- $\check{\bullet}\bullet\square\bullet\square\bullet\bullet \Rightarrow \circ\bullet\square\bullet\square\check{\bullet}\circ \Rightarrow \circ\check{\bullet}\square\bullet\square\bullet\circ \Rightarrow \circ\circ\square\bullet\check{\square}\circ\circ \Rightarrow \circ\circ\check{\square}\bullet\square\circ\circ \Rightarrow E$
- $\check{\bullet}\bullet\bullet\square\bullet\bullet \Rightarrow \circ\bullet\bullet\square\check{\bullet}\circ \Rightarrow \circ\check{\bullet}\bullet\square\bullet\circ \Rightarrow \circ\circ\bullet\check{\square}\circ\circ \Rightarrow \neg E$

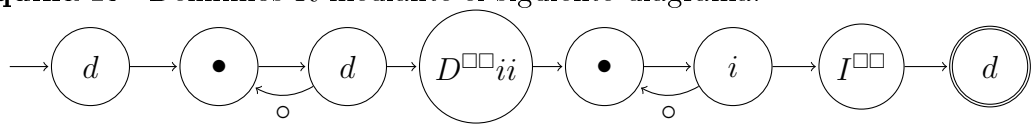
Maquina E Definimos $E := I^{\square\square}dR$. Veamos algunos ejemplos:

- $\circ\circ\check{\square}\bullet\square\circ\circ \Rightarrow \check{\square}\square\circ\circ\square\bullet\square\circ\circ \Rightarrow \check{\square}\circ\circ\square\bullet\square\circ\circ \Rightarrow \check{\square}\bullet\bullet\square\bullet\square\bullet\bullet$

Maquina ¬E Definimos $\neg E := I^{\square\square}dRd$. Veamos algunos ejemplos:

- $\circ\circ\check{\square}\bullet\square\circ\circ \Rightarrow \check{\square}\square\circ\circ\square\bullet\square\circ\circ \Rightarrow \check{\square}\circ\circ\square\bullet\square\circ\circ \Rightarrow \check{\square}\bullet\bullet\square\bullet\square\bullet\bullet \Rightarrow \check{\bullet}\bullet\square\bullet\square\bullet\bullet$

Maquina R Definimos R mediante el siguiente diagrama:



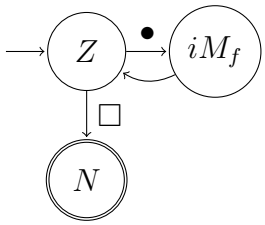
Veamos algunos ejemplos:

- $\check{\square} \circ \circ \bullet \square \circ \circ \Rightarrow \check{\circ} \circ \bullet \square \circ \circ \Rightarrow \check{\bullet} \circ \bullet \square \circ \circ \Rightarrow \bullet \check{\circ} \bullet \square \circ \circ \Rightarrow \bullet\check{\bullet} \bullet \square \circ \circ \Rightarrow$
  $\Rightarrow \bullet \bullet \check{\bullet} \square \circ \circ \Rightarrow \bullet \bullet \bullet \square \circ \check{\circ} \Rightarrow \bullet \bullet \bullet \square \circ \check{\bullet} \Rightarrow \bullet \bullet \bullet \square \check{\circ} \bullet \Rightarrow \bullet \bullet \bullet \check{\square} \bullet \bullet \Rightarrow \check{\square} \bullet \bullet \bullet \square \bullet \bullet$

Conclusion Nuestra maquina quedaria definida como $Z := PC$.

7.3.7.4. Repeticion

Dada una funcion $f \in FRL$ representada por $M_f \in MT$ la funcion $\langle f \rangle$ esta representada por la maquina:



7.3.8. Representacion de MT con FR

Veremos que para toda $M \in MT$ existe una $F_M \in FR$ que la representa. Demostraremos en principio el teorema, para las maquinas de Turing que tienen un alfabeto de diez simbolos $\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}$ donde $s_0 = \square$ y M tiene $p + 1$ estados.

7.3.8.1. Representacion de configuraciones

En cada momento la situacion de una MT esta completamente definida por los datos que contiene la cinta, la casilla observada y el estado en el 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.

Asociaremos a la parte izquierda de la cinta el numero que tiene como cifras decimales a los subindices de los simbolos contenidos en las casillas situadas a la izquierda del cabezal. Llamaremos a este numero C_i. Por ejemplo para la cinta $\ldots s_0 s_5 s_3 s_8 s_4 s_2 \check{s}_3 s_1 s_7 s_9 s_2 s_0 \ldots$ el numero C_i sera: 53842.

A la casilla observada le asociamos como numero el indice del simbolo que contiene y lo notaremos C_o. En nuestro ejemplo $C_o = 3$.

Finalmente asociamos a la parte derecha de la cinta el numero que forman los subindices de los simbolos que contienen pero leyendo de derecha a izquierda . Con lo que los que son los infinitos «ceros a la derecha» los que no cuentan en este caso. Notaremos dicho numero como C_d y en nuestro ejemplo resulta $C_d = 2971$.

Asociaremos entonces a cada configuracion de una MT el numero $2^{C_i}3^{C_o}5^{C_d}7^e$ donde e es el subindice del estado en el que se encuentra la maquina. Notemos que como hemos elegido numeros primos para las bases, a partir de un numero natural que represente una configuracion podemos recuperar los exponentes mediante una funcion recursiva primitiva G tal que $G(x,2) = C_i$, $G(x,3) = C_o$, $G(x,5) = C_d$ y $G(x,7) = e$.

7.3.8.2. Transiciones sobre representaciones

Representaremos los movimientos del cabezal con numeros: $i = 1$, $d = 2$ y $n = 3$. Observemos como se comporta sobre una cinta, una funcion de transicion para el estado actual q y el simbolo observado o:

		1	$\cdots$	j	$\cdots$
	f			q	
0				$\vdots$	
$\vdots$				$\vdots$	
i	o	$\cdots$	$\cdots$	s, m, q'	$\cdots$
$\vdots$				$\vdots$	

- Si $m = 3$ entonces la configuracion $N = 2^{C_i}3^{C_o}5^{C_d}7^q$ se transforma en $N' = 2^{C_i}3^s5^{C_d}7^{q'}$.
- Si $m = 2$ entonces se transforma en $N' = 2^{10C_i+s}3^{C_d\,\%10}5^{\lfloor C_d/10\rfloor}7^{q'}$.
- Si $m = 1$ entonces se transforma en $N' = 2^{\lfloor C_i/10\rfloor}3^{C_i\,\%10}5^{10C_d+s}7^{q'}$.

Veamos algunos ejemplos partiendo de la cinta $\underbrace{6\check{3}01}_{2}$ (representada por el numero $N = 2^6 3^3 5^{10} 7^2 = 826.875.000.000$) reemplazando el simbolo actual por 4 y pasando al estado final:

- Si $m = 3$ se transforma en $N' = 2^6 3^4 5^{10} 7^0 = 16.875.000.000$ es decir $\underbrace{6\check{4}01}_{0}$.
- Si $m = 2$ se transforma en $N' = 2^{10 \cdot 6+4} 3^{10\,\%10} 5^{\lfloor 10/10 \rfloor} 7^0 = 2^{64} 3^0 5^1 7^0$ es decir $\underbrace{64\check{0}1}_{0}$.
- Si $m = 1$ se transforma en $N' = 2^{\lfloor 6/10 \rfloor} 3^{6\,\%10} 5^{10 \cdot 10+4} 7^0 = 2^0 3^6 5^{104} 7^0$ es decir $\underbrace{0\check{6}401}_{0}$.

A partir de aqui definimos tres funciones recursivas que estaran asociadas a una funcion de transicion f de una MT:

- $S_f(i,j) = \begin{cases} \text{simbolo de la entrada } i,j & \text{si i,j es una entrada de la tabla} \\ 0 & \text{si no} \end{cases}$
- $M_f(i,j) = \begin{cases} \text{movimiento de la entrada } i,j & \text{si i,j es una entrada de la tabla} \\ 0 & \text{si no} \end{cases}$
- $E_f(i,j) = \begin{cases} \text{estado de la entrada } i,j & \text{si i,j es una entrada de la tabla} \\ 0 & \text{si no} \end{cases}$

7.3.8.3. Demostracion

Sea $M = (S, \Sigma, \Gamma, f, s_0, q_1, q_0)$ donde $\Sigma = \Gamma = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}$ y $S = \{q_0, \ldots, q_p\}$; y sea $R(c,e) = 2^{C_i} 3^{C_o} 5^{C_d} 7^e$ la funcion que representa configuraciones como numeros.

Definimos la funcion recursiva:

$$T_f(c_i, c_o, c_d, e) = \begin{cases} 2^{\lfloor C_i/10 \rfloor} 3^{C_i \%10} 5^{10C_d + S_f(c_o,e)} 7^{E_f(c_o,e)} & M_f(c_o, e) = 1 \\ 2^{10C_i + S_f(c_o,e)} 3^{C_d \%10} 5^{\lfloor C_d/10 \rfloor} 7^{E_f(c_o,e)} & M_f(c_o, e) = 2 \\ 2^{C_i} 3^{S_f(c_o,e)} 5^{C_d} 7^{E_f(c_o,e)} & M_f(c_o, e) = 3 \end{cases}$$

luego la funcion recursiva asociada a f sera $\overline{f}(x) = T_f(G[x,2], G[x,3], G[x,5], G[x,7])$.

Si partiendo de la cinta inicial C_1 la maquina M la transforma en la cinta C_t en t pasos entonces: $\overline{f}^t[R(C_1,1)] = R(C_t,q)$. Cuando resulte que $\overline{f}^t[R(C_1,1)] = R(C_t,0)$ entonces habremos terminado los calculos. Si efectivamente existe un valor de t tal que satisfaga dicha igualdad entonces este valor sera el minimo t tal que $G\left\{\overline{f}^t[R(C_1,1)], 7\right\} = 0$, que podemos construir utilizando el minimizador. Sea entonces $H(x) = \mu_t\left\{G\left[\overline{f}^t(x), 7\right] = 0\right\}$ la funcion que encuentra dicho valor.

De todo lo aqui expuesto podemos concluir que la funcion recursiva que representa a M es: $\overline{f}^{H(x)}$ donde $x = R(C_1, 1)$.

Resumen A partir de una maquina M y una cinta inicial C_1, transformamos C_1 en un numero mediante $R(C_1, 1)$. Luego imitamos las transiciones de f mediante $\overline{f}$ hasta que el estado que obtenemos a traves de G sea 0 y de esta manera obtenemos un numero que representa la configuracion final de M aplicada a C_1.

Observacion La condicion de que el alfabeto tenga diez valores no es ninguna limitacion. En el caso de un alfabeto de k elementos basta tomar el sitema numerico de base k. El valor 10 de las formulas sigue valiendo pues en cada caso es la forma de expresar el valor de la base.

Conclusion Juntando todos los teoremas de representacion de modelos de calculo (y haciendo abuso de notacion) tenemos $FR \preceq FRL \preceq MT \preceq FR$ logrando concluir que todos estos modelos son equivalentes.

7.3.9. Numerabilidad de maquinas de Turing

Sea $M_n = \{m \in MT : \text{ el alfabeto de m tiene n simbolos}\}$, observemos que $MT = \bigcup_{n=2}^{\infty} M_n$.

Asi mismo $M_n = \bigcup_{k=2}^{\infty} M_{nk}$ donde $M_{nk} = \{m \in M_n : \text{ m tiene k estados}\}$.

Notese que M_{nk} es un conjunto finito. En efecto hay un numero finito de funciones de transicion, pues tanto el dominio como el codominio tienen cardinal finito.

Como $MT = \bigcup_{n=2}^{\infty} \bigcup_{k=2}^{\infty} M_{nk}$ concluimos que $\#MT = \aleph_0$ pues es union numerabla de conjuntos numerablos.

7.3.10. Limite de lo calculable

Hemos visto que $\#\{f : \mathbb{N} \to \{0,1\}\} = \mathcal{P}(\mathbb{N}) = \aleph_1$ y claramente hay tantas (o mas) funciones numericas. Ademas por el teorema anterior podemos concluir que $\#MT = \aleph_0 \prec \aleph_1 \preceq \#\{f : \mathbb{N} \to \mathbb{N}\}$, es decir que existen funciones que ninguna maquina de Turing puede calcular.

Veremos a continuacion una funcion util y perfectamente definida que no podemos calcular con maquinas de Turing.

El problema de la parada Notemos que dependiendo de cada MT no siempre ante una configuracion inicial se llega a una configuracion final, por ejemplo la maquina $D^{\bullet\bullet}$ con una cinta inicial que no tenga dos $\bullet$ seguidos no se detendra nunca.

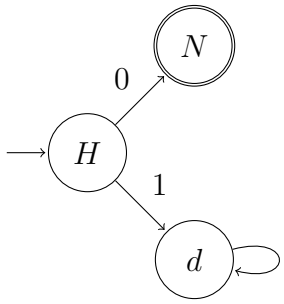
Intentaremos construir una maquina que dada cualquier maquina de Turing y cualquier entrada, nos indique si esta finaliza sus calculos o no. Sin lugar a dudas para casos particulares podemos determinar dicha respuesta, pero buscamos una maquina de proposito general.

Sabemos por el teorema anterior que es posible asignarte a cada maquina de Turing un numero natural. Consideremos entonces la siguiente funcion:

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

$$k \to h(k) = \begin{cases} 1 & M_k \text{ termina con k como entrada} \\ 0 & \text{en caso contrario} \end{cases}$$

¿Es h una funcion calculable? Es decir: ¿Existe una MT que al recibir k como entrada calcule $h(k)$? Supongamos que esta maquina existe, llamemosla H y definamos la maquina Y mediante el siguiente diagrama:



Sea y el numero asociado a la maquina Y. ¿Cuanto vale $h(y)$?

- Supongamos que Y termina al recibir y como entrada, es decir $h(y) = 1$. Analicemos el comportamiento de Y en dicha cinta:

$$\check{y} \underbrace{\Rightarrow}_{h(y)=1} y\check{\square} \Rightarrow y\square\check{\square} \Rightarrow y\square\square\check{\square} \Rightarrow \dots$$

  es decir que Y no termina. Contradiccion.

- Supongamos que Y no termina al recibir y como entrada, es decir $h(y) = 0$. Analicemos el comportamiento de Y en dicha cinta:

$$\check{y} \underbrace{\Rightarrow}_{h(y)=0} \check{y}$$

  es decir que Y termina. Contradiccion.

La contradiccion vino de suponer que H existia, luego H no existe.

7.3.11. Modificaciones equivalentes

Existen diversas modificaciones a este modelo de maquina de Turing, que no modifican el poder de calculo de estas. Entre ellas:

- Maquina de Turing con cinta acotada a izquierda.
- Maquina de Turing con multiples cintas.
- Maquina de Turing no determinista.
- Automata de multiples pilas.
- Automata de cola.
- Calculo λ.

7.3.12. Ejemplos

7.3.12.1. Izquierda hasta dos $\square$

Sean $S = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{\bullet\}$, $\Gamma = \Sigma \cup \{\square\}$ definimos $I^{\square\square} = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ con la siguiente funcion de transicion:

f	q_1	q_2	q_3
$\square$	$\square, i, q_2$	$\square, i, q_3$	$\square, n, q_0$
$\bullet$	$\bullet, i, q_2$	$\bullet, i, q_2$	$\bullet, i, q_2$

Observemos como se comporta esta maquina partiendo de la cinta vacia:

$$\underbrace{\ldots\square\square\check{\square}\ldots}_{q_1} \Rightarrow \underbrace{\ldots\square\check{\square}\square\ldots}_{q_2} \Rightarrow \underbrace{\ldots\check{\square}\square\square\ldots}_{q_3} \Rightarrow \underbrace{\ldots\check{\square}\square\square\ldots}_{q_0}$$

Observemos como se comporta esta maquina partiendo de la cinta inicial $\bullet\square\square\bullet\bullet\square\check{\bullet}$:

$$\underbrace{\bullet\square\square\bullet\bullet\square\check{\bullet}}_{q_1} \Rightarrow \underbrace{\bullet\square\square\bullet\bullet\check{\square}\bullet}_{q_2} \Rightarrow \underbrace{\bullet\square\square\bullet\check{\bullet}\square\bullet}_{q_3} \Rightarrow \underbrace{\bullet\square\square\check{\bullet}\bullet\square\bullet}_{q_2} \Rightarrow \underbrace{\bullet\square\check{\square}\bullet\bullet\square\bullet}_{q_2} \Rightarrow$$
$$\Rightarrow \underbrace{\bullet\check{\square}\square\bullet\bullet\square\bullet}_{q_3} \Rightarrow \underbrace{\bullet\check{\square}\square\bullet\bullet\square\bullet}_{q_0}$$

7.3.12.2. Sucesor decimal

Sean $S = \{q_0, q_1, q_2\}$, $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}, \Gamma = \Sigma \cup \{\square\}$ definimos $S_{10} = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ con la siguiente funcion de transicion:

f	q_1	q_2
$\square$	$\square, i, q_2$	$1, n, q_0$
0	$0, d, q_1$	$1, n, q_0$
1	$1, d, q_1$	$2, n, q_0$
2	$2, d, q_1$	$3, n, q_0$
3	$3, d, q_1$	$4, n, q_0$
4	$4, d, q_1$	$5, n, q_0$
5	$5, d, q_1$	$6, n, q_0$
6	$6, d, q_1$	$7, n, q_0$
7	$7, d, q_1$	$8, n, q_0$
8	$8, d, q_1$	$9, n, q_0$
9	$9, d, q_1$	$0, i, q_2$

Observemos como se comporta esta maquina partiendo de la cinta inicial con el numero $\check{3}99$:

$$\underbrace{\check{3}99}_{q_1} \Rightarrow \underbrace{3\check{9}9}_{q_1} \Rightarrow \underbrace{39\check{9}}_{q_1} \Rightarrow \underbrace{399\check{\square}}_{q_1} \Rightarrow \underbrace{39\check{9}}_{q_2} \Rightarrow \underbrace{3\check{9}0}_{q_2} \Rightarrow \underbrace{\check{3}00}_{q_2} \Rightarrow \underbrace{\check{4}00}_{q_0}$$

7.3.12.3. Reconocedora de $a^n b^n$

Sean $S = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}, \Gamma = \Sigma \cup \{\square\}$ definimos $M = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ con la siguiente funcion de transicion:

f	q_1	q_2	q_3	q_4
$\square$	$\square, n, q_0$	$\square, i, q_3$	b, n, q_0	$\square, d, q_1$
a	$\square, d, q_2$	a, d, q_2	b, n, q_0	a, i, q_4
b	b, n, q_0	b, d, q_2	$\square, i, q_4$	b, i, q_4

Esta maquina terminara con el cabezal apuntando a un blanco si la palabra pertenece al lenguaje, y a otro simbolo en caso contrario.

Observemos como se comporta esta maquina partiendo de la cinta inicial con la palabra $\check{a}abb$:

$$\underbrace{\check{a}abb}_{q_1} \Rightarrow \underbrace{\check{a}bb}_{q_2} \Rightarrow \underbrace{a\check{b}b}_{q_2} \Rightarrow \underbrace{ab\check{b}}_{q_2} \Rightarrow \underbrace{abb\check{\square}}_{q_2} \Rightarrow \underbrace{ab\check{b}}_{q_3} \Rightarrow \underbrace{a\check{b}}_{q_4} \Rightarrow$$
$$\Rightarrow \underbrace{\check{a}b}_{q_4} \Rightarrow \underbrace{\check{\square}ab}_{q_4} \Rightarrow \underbrace{\check{a}b}_{q_1} \Rightarrow \underbrace{\check{b}}_{q_2} \Rightarrow \underbrace{b\check{\square}}_{q_2} \Rightarrow \underbrace{\check{b}}_{q_3} \Rightarrow \underbrace{\check{\square}}_{q_4} \Rightarrow \underbrace{\check{\square}}_{q_1} \Rightarrow \underbrace{\check{\square}}_{q_0}$$

Observemos como se comporta esta maquina partiendo de la cinta inicial con la palabra $\check{a}ab$:

$$\underbrace{\check{a}ab}_{q_1} \Rightarrow \underbrace{\check{a}b}_{q_2} \Rightarrow \underbrace{a\check{b}}_{q_2} \Rightarrow \underbrace{ab\check{\square}}_{q_2} \Rightarrow \underbrace{a\check{b}}_{q_3} \Rightarrow \underbrace{\check{a}}_{q_4} \Rightarrow \underbrace{\check{\square}a}_{q_4} \Rightarrow \underbrace{\check{a}}_{q_1} \Rightarrow \underbrace{\check{\square}}_{q_2} \Rightarrow \underbrace{\check{\square}}_{q_3} \Rightarrow \underbrace{\check{b}}_{q_0}$$

7.3.12.4. Duplicar puntos

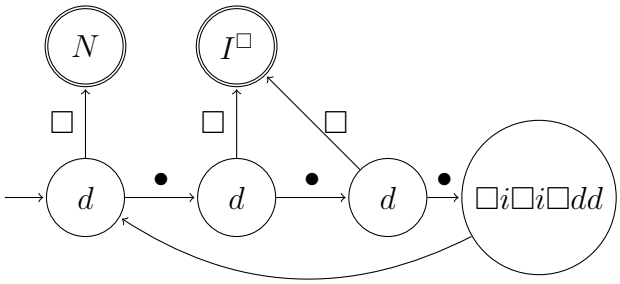
Definimos $M = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ con la siguiente funcion de transicion:

f	q_1	q_2	q_3	q_4	q_5
$\square$	$\square, n, q_0$	$\square, d, q_3$	$\bullet, i, q_4$	$\square, i, q_5$	$\bullet, d, q_1$
$\bullet$	$\square, d, q_2$	$\bullet, d, q_2$	$\bullet, d, q_3$	$\bullet, i, q_4$	$\bullet, i, q_5$

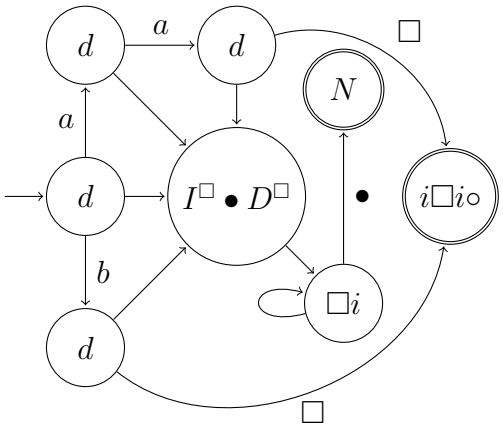
Observemos como se comporta esta maquina partiendo de la cinta inicial con la palabra $\check{\bullet}\bullet\bullet$:

$$\underbrace{\check{\bullet}\bullet\bullet}_{q_1} \Rightarrow \underbrace{\check{\bullet}\bullet}_{q_2} \Rightarrow \underbrace{\bullet\check{\bullet}}_{q_2} \Rightarrow \underbrace{\bullet\bullet\check{\square}}_{q_2} \Rightarrow \underbrace{\bullet\bullet\square\check{\square}}_{q_3} \Rightarrow \underbrace{\bullet\bullet\check{\square}\bullet}_{q_4} \Rightarrow \underbrace{\bullet\check{\bullet}\square\bullet}_{q_5} \Rightarrow \underbrace{\check{\bullet}\bullet\square\bullet}_{q_5} \Rightarrow \underbrace{\check{\square}\bullet\bullet\square\bullet}_{q_5} \Rightarrow$$
$$\Rightarrow \underbrace{\bullet\check{\bullet}\bullet\square\bullet}_{q_1} \Rightarrow \underbrace{\bullet\square\check{\bullet}\square\bullet}_{q_2} \Rightarrow \underbrace{\bullet\square\bullet\check{\square}\bullet}_{q_2} \Rightarrow \underbrace{\bullet\square\bullet\square\check{\bullet}}_{q_3} \Rightarrow \underbrace{\bullet\square\bullet\square\bullet\check{\square}}_{q_3} \Rightarrow \underbrace{\bullet\square\bullet\square\check{\bullet}\bullet}_{q_4} \Rightarrow$$
$$\Rightarrow \underbrace{\bullet\square\bullet\check{\square}\bullet\bullet}_{q_4} \Rightarrow \underbrace{\bullet\square\check{\bullet}\square\bullet\bullet}_{q_5} \Rightarrow \underbrace{\bullet\check{\square}\bullet\square\bullet\bullet}_{q_5} \Rightarrow \underbrace{\bullet\bullet\check{\bullet}\square\bullet\bullet}_{q_1} \Rightarrow \underbrace{\bullet\bullet\square\check{\square}\bullet\bullet}_{q_2} \Rightarrow \underbrace{\bullet\bullet\square\square\check{\bullet}\bullet}_{q_3} \Rightarrow$$
$$\Rightarrow \underbrace{\bullet\bullet\square\square\bullet\check{\bullet}}_{q_3} \Rightarrow \underbrace{\bullet\bullet\square\square\bullet\bullet\check{\square}}_{q_3} \Rightarrow \underbrace{\bullet\bullet\square\square\bullet\check{\bullet}\bullet}_{q_4} \Rightarrow \underbrace{\bullet\bullet\square\square\check{\bullet}\bullet\bullet}_{q_4} \Rightarrow \underbrace{\bullet\bullet\square\check{\square}\bullet\bullet\bullet}_{q_4} \Rightarrow$$
$$\Rightarrow \underbrace{\bullet\bullet\check{\square}\square\bullet\bullet\bullet}_{q_5} \Rightarrow \underbrace{\bullet\bullet\bullet\check{\square}\bullet\bullet\bullet}_{q_1} \Rightarrow \underbrace{\bullet\bullet\bullet\check{\square}\bullet\bullet\bullet}_{q_0}$$

7.3.12.5. Modulo 3



7.3.12.6. Decidibilidad de $\{aa, b\}$ sobre $\Sigma = \{a, b\}$



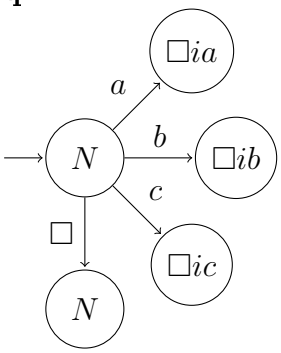
7.3.12.7. Reconocedora de $\{w_1 w_2 / w_1 \neq w_2 \wedge |w_1| = |w_2|\}$ sobre $\{a, b, c\}^*$

Descripcion Nuestra cinta inicial comienza a ser manipulada por P (par). Esta maquina detecta si es de longitud impar y en tal caso la rechada inmediatamente. De no ser asi le pasa el control a C (comparar).

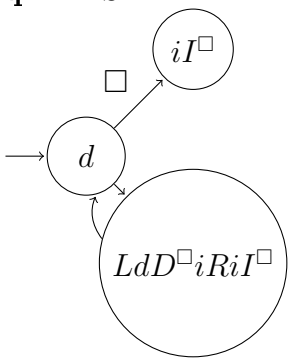
Lo primero que hace C es ejecutar S (separar) que transforma una cinta $\check{\square} w_1 w_2$ en $\check{\square} w_1 \square\square w_2$ y luego agrega un marcador quedandose con la cinta $\bullet\check{\square} w_1 \square\square w_2$.

A partir de aqui se empieza a comparar caracter con caracter (desde el final de las subcadenas) y de encontrarse una diferencia inmediatamente se pasa a una rutina de aceptacion que borra el marcador, restaura la palabra a traves de U (unir) y coloca el cabezal en la posicion indicada. De haber terminado todas las comparaciones y persistir la igualdad, se pasa a la rutina de no aceptacion que actua en forma similar.

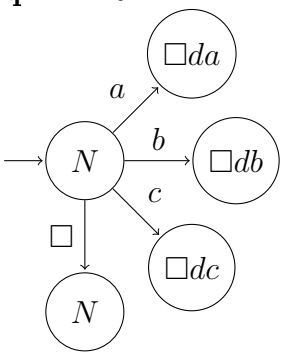
Maquina L



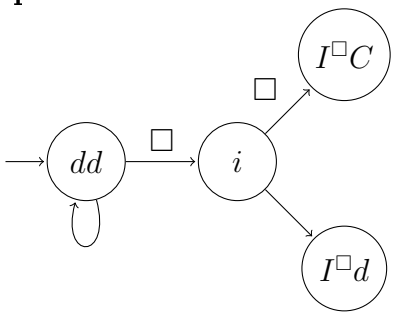
Maquina S



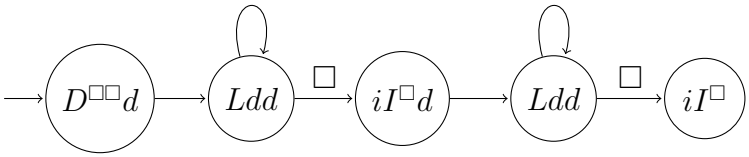
Maquina R



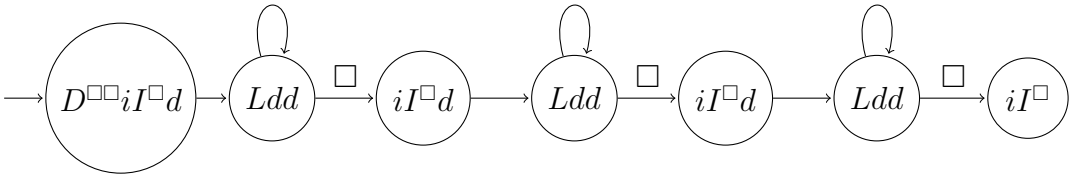
Maquina P



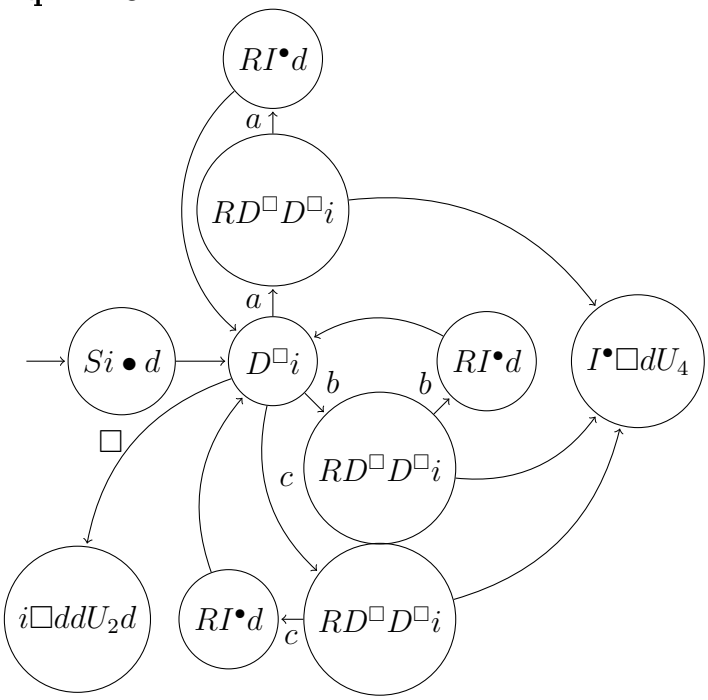
Maquina U_2



Maquina U_4



Maquina C



7.3.12.8. Representacion en FR

La maquina suma unaria esta representada por la siguiente funcion de transicion:

f	1	2	3	4	5
0	$0, d, 1$	$1, d, 3$	$0, i, 4$	$0, i, 4$	$0, n, 0$
1	$1, d, 2$	$1, d, 2$	$1, d, 3$	$0, i, 5$	$1, i, 5$

La suma de cuatro y tres partira de la cinta inicial $\check{0}11110111$ que sera representada como $2^0 3^0 5^{11101111} 7^1$ y llegara a la final como $2^0 3^0 5^{1111111} 7^0$.

Observemos una traza de la funcion recursiva que la representa:

$$\overline{f}^{19}\left[2^0 3^0 5^{11101111} 7^1\right] = \overline{f}^{18}\left[T_f\left(0, 0, 11101111, 1\right)\right] =$$
$$\underbrace{=}_{M(0,1)=2} \overline{f}^{18}\left[2^{10\cdot 0+S_f(0,1)} 3^{11101111\,\%10} 5^{\lfloor 11101111/10\rfloor} 7^{E_f(0,1)}\right] =$$
$$= \overline{f}^{18}\left[2^{10\cdot 0+0} 3^{11101111\,\%10} 5^{\lfloor 11101111/10\rfloor} 7^1\right] = \overline{f}^{18}\left[2^0 3^1 5^{1110111} 7^1\right] =$$
$$= \overline{f}^{17}\left[T_f\left(0, 1, 1110111, 1\right)\right] \underbrace{=}_{M(1,1)=2} \overline{f}^{17}\left[2^{0+S_f(1,1)} 3^{1110111\,\%10} 5^{\lfloor 1110111/10\rfloor} 7^{E_f(1,1)}\right] =$$
$$= \overline{f}^{17}\left[2^1 3^1 5^{111011} 7^2\right] \underbrace{=}_{M(1,2)=2} \overline{f}^{16}\left[2^{10\cdot 1+1} 3^{111011\,\%10} 5^{\lfloor 111011/10\rfloor} 7^2\right] = \overline{f}^{16}\left[2^{11} 3^1 5^{11101} 7^2\right] =$$
$$\underbrace{=}_{M(1,2)=2} \overline{f}^{15}\left[2^{110+1} 3^{11101\,\%10} 5^{\lfloor 11101/10\rfloor} 7^2\right] = \overline{f}^{15}\left[2^{111} 3^1 5^{1110} 7^2\right] = \overline{f}^{14}\left[2^{1111} 3^0 5^{111} 7^2\right] =$$
$$\underbrace{=}_{M(0,2)=2} \overline{f}^{13}\left[2^{11110+1} 3^1 5^{11} 7^3\right] \underbrace{=}_{M(1,3)=2} \overline{f}^{12}\left[2^{111110+1} 3^1 5^1 7^3\right] \underbrace{=}_{M(1,3)=2} \overline{f}^{11}\left[2^{1111111} 3^1 5^0 7^3\right] =$$
$$\underbrace{=}_{M(1,3)=2} \overline{f}^{10}\left[2^{11111111} 3^0 5^0 7^3\right] \underbrace{=}_{M(0,3)=1} \overline{f}^{9}\left[2^{\lfloor 11111111/10\rfloor} 3^{11111111\,\%10} 5^{10\cdot 0+S_f(0,3)} 7^{E_f(0,3)}\right] =$$
$$= \overline{f}^{9}\left[2^{1111111} 3^1 5^{0+0} 7^4\right] \underbrace{=}_{M(1,4)=1} \overline{f}^{8}\left[2^{111111} 3^1 5^{0+0} 7^5\right] \underbrace{=}_{M(1,5)=1} \overline{f}^{7}\left[2^{11111} 3^1 5^{0+1} 7^5\right] =$$
$$\underbrace{=}_{M(1,5)=1} \overline{f}^{6}\left[2^{1111} 3^1 5^{10+1} 7^5\right] \underbrace{=}_{M(1,5)=1} \overline{f}^{5}\left[2^{111} 3^1 5^{111} 7^5\right] = \overline{f}^{4}\left[2^{11} 3^1 5^{1111} 7^5\right] =$$
$$= \overline{f}^{3}\left[2^1 3^1 5^{11111} 7^5\right] = \overline{f}^{2}\left[2^0 3^1 5^{111111} 7^5\right] = \overline{f}^{1}\left[2^0 3^0 5^{1111111} 7^5\right] \underbrace{=}_{M(0,5)=3} \overline{f}^{0}\left[2^0 3^0 5^{1111111} 7^0\right]$$

Bibliografía

[1] Pablo Verdes. *Catedra de Lenguajes Formales y Computabilidad.*

[2] J. Glenn Brookshear. *Teoria de la Computacion. Lenguajes formales, automatas y complejidad.*

[3] Julio Hurtado, Raul Kantor, Carlos Luna, Luis Sierra y Dante Zanarini. *Temas de Teoria de la Computacion.*

[4] Daniel J. Velleman. *How to Prove It.*

[5] Richard Hammack. *Book of Proof.*

[6] ProofWiki. *proofwiki.org.*