Lenguajes Formales y Computabilidad

31 de diciembre de 2017

Índice general

Parte I

Principio de inducción y cardinalidad

Capítulo 1

Principio de inducción

1.1. Definiciones

1.1.1. Conjunto Inductivo

Una definición inductiva de un conjunto A comprende base, inducción y clausura:

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

inducción 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 básicos y aplicando las reglas de inducción, sin considerar elementos «*extra*».

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

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

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

- $B \subseteq A$.
- Si $f^{(n)} \in K$ y $a_1, \ldots, a_n \in A$ entonces $f(a_1, \ldots, a_n) \in A$.

1.1.2. Secuencia de formación

Sean U, B, K como en la definición anterior. Una secuencia $a_1, \ldots, a_m$ de elementos de U es una secuencia de formación 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 formación.

Diremos que B y K definen una gramática 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 formación.

Ejemplo Sea L el mínimo 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 formación: $1 \Rightarrow 111 \Rightarrow 1011101 \Rightarrow 110111011$.

1.2.2. No pertenencia

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

- Mostrar que no existe una secuencia de formación 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) podríamos 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 inducción primitiva que se detalla a continuación.

1.2.3. Principio de inducción 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 $P(x)$ $\forall x \in A$.

Demostración 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 definición del conjunto.
- Veamos que C satisface las clausulas de la definición 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 $f^{(n)} \in K, a_1, a_2, \ldots, a_n \in C$ y $f(a_1, a_2, \ldots, a_n) = a$ queremos probar que $a \in C$:
    - Por definición de C valen $P(a_1), P(a_2), \ldots, P(a_n)$.
    - Por *(2)* vale $P(a)$.

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

  Dado que A es el mínimo conjunto que cumple las clausulas de su definición 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. Función 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. Función sobreyectiva

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

2.1.3. Función 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 función 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 función 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 vacío o equipotente a $\{1, 2, \ldots, n\}$ para algún $n \in \mathbb{N}$. 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 familia de conjuntos si sus elementos son conjuntos. Diremos que F es una familia indexada de conjunto indice I (no vacío) si existe una función 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 función inyectiva de A en B y otra de B en A entonces existe una función biyectiva de A a B.

Demostración Consultar bibliografía [4] y [5] paginas 322 y 232.

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

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

Demostración Sea $f : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ dada por $f(n) = (n, 1)$. Esta función 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 aritmética nos permite asegurar que esta función es inyectiva.

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

Observación La función $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ dada por $f[(i, j)] = \frac{1}{2}(i + j - 1)(i + j - 2) + i$ es una biyeccion:

$f[(i,j)]$	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	↙					

2.2.3. Corolario

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

Demostración Lo demostraremos por inducción:

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 hipótesis 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 $\mathbb{N}^{d+1} \sim \mathbb{N} \times \mathbb{N} \sim \mathbb{N}$.

2.2.4. Unión numerable de conjuntos numerables

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

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

2.2.5. Cardinalidad infinita mas pequeña

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

Demostración 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 sucesión $(x_n)_{n \geq 1}$ 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 $\aleph_0 \preceq \#A$.

2.2.6. Cardinalidad del conjunto de partes

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

Demostración La función $f(x) = \{x\}$ es inyectiva de S en $\mathcal{P}(S)$ por lo que $\#S \preceq \#\mathcal{P}(S)$. Veamos ahora que no existe función 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 través 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 definición de B resulta $x \notin g(x) = B$. Contradicción.
- Si $x \notin B$: por definición de B resulta $x \in g(x) = B$. Contradicción.

Por lo tanto g no es sobreyectiva.

2.2.7. Innumerabilidad del continuo

Enunciado El conjunto de los números reales no es numerable.

Demostración Alcanza 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 expansión decimal infinita, por ejemplo $0,229384112598\ldots$. Supongamos que $(0, 1)$ es numerable, habrá 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 $b = 0, b_1 b_2 b_3 b_4 b_5 \ldots$ donde cada dígito b_i puede ser cualquier dígito excepto a_{ii} (es decir los números en negrita ubicados en la diagonal). Es claro que $b \in (0, 1)$ pero es distinto a todos los números del listado ya que difiere de cada numero en por lo menos un dígito. Esto constituye una contradicción, luego el intervalo $(0, 1) \sim \mathbb{R}$ no es numerable.

2.3. Ejemplos

2.3.1. Cardinalidad 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 $\mathbb{Q}$

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

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 función, es decir n y notaremos $f^{(n)}$.

3.1.2. Función característica

Dado un conjunto X, para cada subconjunto $A \subseteq X$ definimos su *función característica* $\chi_A : X \to \{0, 1\}$ como:

$$\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

3.1.3. Función numérica

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

3.1.4. Funciones base

Llamaremos funciones base a las siguientes tres funciones:

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

3.1.5. Operadores

Definiremos dos operadores que nos permitirán construir nuevas funciones:

- El operador de composición Φ que dada una función numérica $f^{(n)}$ y n funciones numéricas de aridad k, construye la función numérica 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 numéricas $g^{(k)}$ y $h^{(k+2)}$ construye una nueva función numérica $f^{(k+1)}$ 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 $f = R(g, h)$.

3.1.6. Definición 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 composición y recursion sobre elementos de FRP también pertenecen a FRP.

3.2. Ejemplos

3.2.1. Predecesor natural

La función $\widehat{Pd}^{(1)}(y) = \begin{cases} 0 & y = 0 \\ y-1 & y > 0 \end{cases}$ 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 función $\Sigma^{(2)}(y,x) = y + x$ 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. Función potencia

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

3.3. Conjuntos

3.3.1. Conjunto recursivo primitivo

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

3.3.2. Relaciones recursivas primitivas

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

Capítulo 4

Funciones recursivas

4.1. Introducción

4.1.1. Función parcial y función total

Una función numérica $f : \mathbb{N}^k \to \mathbb{N}$ 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. Función mayora

Decimos que una función $f^{(1)}$ mayora a otra función $g^{(n)}$ 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 sucesión de funciones que llamaremos sucesión 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. Función de Ackermann

Definimos una función 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 Ackermann 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 función total.

Demostración Lo demostraremos por inducción sobre el conjunto FRP.

- Caso base: Las funciones bases son totales por definición.
- Composición: Supongamos que $f^{(n)}, g_1^{(k)}, \ldots, g_n^{(k)}$ son totales. Veremos que $h = \Phi\left(f^{(n)}, g_1^{(k)}, \ldots, g_n^{(k)}\right)$ también 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 hipótesis inductiva. Ademas f también 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 $f\left(y, X^k\right) = R(g, h)$ también lo es, por inducción en y.
  1. Caso base $y = 0$: $f(0, X^k) = g\left(X^k\right)$ que es total por hipótesis.
  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. $\forall x, k \in \mathbb{N} \Rightarrow f_k(x) < f_{k+1}(x)$.

Demostración

1. Lo demostraremos por inducción 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} también lo es. En efecto $f_{h+1}(x) = f_h^{x+2}(x) = f_h^{s[s(x)]}(x)$.
2. Consultar bibliografía [3], pag. 56.
3. Consultar bibliografía [3], pag. 56.
4. Ejercicio.

4.2.3. Mayorabilidad de las FRP

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

Demostración Lo demostraremos por inducción:

1. Caso base: Todas las funciones bases son mayoradas por f_0.
2. Caso inductivo:
   a) Composición: 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 hipótesis $g(0,X) = B(X) \leq f_k[max(X)]$ *(**)* 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 inducción 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 función $ACK(x)$ no es FRP.

Demostración 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 mínimo valor de t tal que $h(t,X) = 0$.

Observación Nada garantiza que tal valor t exista, por lo que las funciones construidas con el operador M pueden ser parciales.

4.3.2. Definición 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 composición, recursion y minimizacion sobre elementos de FR también pertenecen a FR.

4.4. Ejemplos

4.4.1. División

La función numérica $div\,(x,y) = x/y$ 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:

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

- $div\,(2,0) = \mu_t\,[h\,(t,2,0) = 0]$.
  - $t = 0 : h\,(0,2,0) = \neg E\,[\Pi\,(0,0)\,,2] = \neg E\,[0,2] = 1 \neq 0$.
  - $t = 1 : h\,(1,2,0) = \neg E\,[\Pi\,(1,0)\,,2] = \neg E\,[0,2] = 1 \neq 0$.
  - $\vdots$
- $div\,(0,0) = \mu_t\,[h\,(t,0,0) = 0]$.
  - $t = 0 : h\,(0,0,0) = \neg E\,[\Pi\,(0,0)\,,0] = \neg E\,[0,0] = 0$. ERROR.

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

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

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

4.4.2. Logaritmo

La función numérica $Log\,(x,y) = log_x y$ 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 $h\,(t,x,y) = \neg E\left\{\widehat{Exp}\,[x,t] + D_0\,[x] + D_1\,[x]\,,y\right\}$.

Nótense los términos $D_0\,[x]$ y $D_1\,[x]$ que logran «indefinir» la función para las bases correspondientes.

Veamos algunos ejemplos:

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

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

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 $\mathbb{N}_0$.

Notación

- $[x_1, x_2, \ldots, x_k]$ para una lista de longitud k.
- [] para la lista vaciá.
- 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. Concatenación

Dadas dos listas $X = [x_1, \ldots, x_m]$ e $Y = [y_1, \ldots, y_n]$ llamamos concatenación 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 concatenación es una operación asociativa cuyo elemento neutro es la lista vaciá.

5.1.3. Funciones de lista

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

Observación Podemos pensar a las funciones numéricas $f : \mathbb{N}_0^k \to \mathbb{N}_0$ 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: $S_d[x_1, x_2, \ldots, x_k] = [x_1, x_2, \ldots, \boldsymbol{x_k} + \mathbf{1}]$.

Observación Nótese 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 permitirán construir nuevas funciones:

- El operador de composición que dadas dos funciones F y G construye la función $H = F \circ G$ que notaremos simplemente $H = FG$ definida como $HX = G[FX]$. Nótese que contrariamente a lo usual, primero se aplica la función F y al resultado se aplica la función G.
- El operador de repetición que dada una función F construye la función $\langle F \rangle$ 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 función $\langle F \rangle$ actúa aplicando F hasta que el primer y ultimo elemento de su argumento sean iguales.

Dominio
El dominio de la composición 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 sucesión $\{X, FX, FFX, FFFX, \ldots\}$ existe una lista cuyo primer y ultimo elemento son iguales.

5.1.6. Definición 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 composición y repetición sobre elementos de FRL también pertenecen a FRL.

5.2. Ejemplos

5.2.1. Pasar a izquierda

La función $\triangleleft = 0_i\,\langle S_i\rangle\,\square_d$ 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 función $\triangleright = 0_d\,\langle S_d\rangle\,\square_i$ pasa el elemento de la izquierda a la derecha.

5.2.3. Duplicar a izquierda

La función $D_i = 0_d\,\langle S_d\rangle\,\triangleleft$ 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 función $D_d = 0_i \langle S_i \rangle \triangleright$ duplica el elemento de la derecha (a la derecha).

5.2.5. Intercambiar extremos

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

5.3. Representación de FR mediante FRL

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

5.3.1. Representabilidad

Sea $g : \mathbb{N}_0^k \to \mathbb{N}_0$ una función recursiva entonces si existe una función 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 función recursiva g como función de listas.

5.3.2. Casos base

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

Demostración

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

5.3.3. Composición

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 composición $h = \Phi\left(f, g_1, \ldots, g_n\right)$ también tiene representación en funciones de listas.

Demostración 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 composición:

	$[X, Y]$
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	$[\boldsymbol{f}\left\{\boldsymbol{g_1}\left(\boldsymbol{X}\right), \boldsymbol{\ldots}, \boldsymbol{g_n}\left(\boldsymbol{X}\right)\right\}, g_1\left(X\right), \ldots, g_n\left(X\right), X, Y]$
$=$	$[\boldsymbol{h}\left(\boldsymbol{X}\right), g_1\left(X\right), \ldots, g_n\left(X\right), X, Y]$
$\triangleright$	$[g_1\left(X\right), \ldots, g_n\left(X\right), X, Y, \boldsymbol{h}\left(\boldsymbol{X}\right)]$
$\square_i^n$	$[X, Y, h\left(X\right)]$
$\triangleleft$	$[h\left(X\right), X, Y]$

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 función $f^{(k+1)} = R(g,h)$ también tiene representación en FRL.

Demostración 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 aquí 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	$[\mathbf{1}, X, Y, y, f\,(1, X)]$
$\leftrightarrow$	$[\boldsymbol{f\,(1, X)}, X, Y, y, \mathbf{1}]$
$\triangleleft\rangle$	$[\mathbf{1}, f\,(1, X), X, Y, y]$

A partir de aquí podemos ver que el resultado de la repetición 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 función recursiva $h^{(n+1)}$ y F_h su representación en FRL entonces la función $g^{(n)} = M\,[h]$ también tiene representación en FRL.

Demostración 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 aquí 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 repetición es transformar $[h(t,X),t,X,Y,0]$ en $[h(t+1,X),t+1,X,Y,0]$. La repetición se aplicara un numero k de veces hasta que $h(k,X)=0$ y a partir de ahí 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. Conclusión

Con todo lo anterior hemos demostrado que toda función recursiva puede ser representada por una función 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

Gramáticas y expresiones regulares

6.1. Definiciones

6.1.1. Alfabeto

Un alfabeto es un conjunto finito y no vacío, cuyos elementos reciben el nombre de símbolos, 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 símbolos de Σ. Mas precisamente se la define como la función $p : \{1, \ldots, n\} \to \Sigma$ donde el entero positivo n es la longitud de la palabra, que se denota también como $|p|$.

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

Dada la palabra $p : \{1, \ldots, n\} \to \Sigma$ por simplicidad de notación escribiremos la cadena p como $p(1)\, p(2) \ldots p(n) = l_1 l_2 \ldots l_n$.

6.1.3. Clausuras de Kleene

Dado un alfabeto Σ definimos $\Sigma^0 = \{\lambda\}$ y $\Sigma^n = \{p/p : \{1, \ldots, n\} \to \Sigma\}$ 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 vaciá.

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. Concatenación

Dado un alfabeto Σ y dos palabras $p : \{1, \ldots, m\} \to \Sigma$ y $q : \{1, \ldots, n\} \to \Sigma$ definimos la concatenación de p y q (denotada pq) como la palabra $pq : \{1, \ldots, m+n\} \to \Sigma$ 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 fácil ver que $p\,(qr) = (qp)\,r$ pero $pq \neq qr$, es decir, la concatenación es asociativa pero no conmutativa.

6.1.5. Cadena potencia

Dada una palabra $p \in \Sigma^*$ 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 palabra $p \in \Sigma^*$ 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. Definición

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, $\mathcal{L} = \mathcal{P}(\Sigma^*)$.

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

6.2.2. Unión

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

6.2.3. Intersección

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la intersección 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. Concatenación

Dados dos lenguajes L_1 y L_2 sobre Σ definimos la concatenación de los lenguajes como: $L_1L_2 = \{p \in \Sigma^* / \exists q \in L_1, \exists r \in L_2, p = qr\}$.

Observaciones

- $L\emptyset = \emptyset L = \emptyset$.
- $L\{\lambda\} = \{\lambda\}L = L$.
- En general $L_1L_2 \neq L_2L_1$.

6.2.7. Potencia

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

$$\begin{aligned} L^0 &= \{\lambda\} \\ L^{n+1} &= L^nL \end{aligned}$$

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 unión 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 unión 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 Kleene construyen siempre lenguajes infinitos.

6.3. Gramáticas

6.3.1. Definición

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

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

Observaciones

- Nótese 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 producción (α, β) la notaremos: $\alpha \to \beta$.
- Una regla del tipo $\alpha \to \lambda$ recibe el nombre de regla λ.

6.3.2. Derivación

Si $\alpha \to \beta$ es una regla de producción 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 gramática $G = (N, T, P, \sigma)$ como $L(G) = \{p \in T^* / \sigma \Rightarrow^* p\}$.

6.3.4. Gramáticas regulares

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

Las reglas de producción de una gramática 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 símbolo terminal que puede estar seguido o no, por un símbolo no terminal o la cadena vaciá.

Es decir que las producciones de una gramática 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 gramática regular izquierda las reglas de producción son de la forma: $A \to a$, $A \to Ba$ o $A \to \lambda$.

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

6.3.5. Gramáticas libres de contexto

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

Observaciones

- Estas gramáticas 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, podría ocurrir que en el aparezcan mas de uno como en $A \to XY$. En esta situación el enfoque mas común consiste en reemplazar en el paso siguiente el no terminal situado mas a la izquierda.

- Alternativamente se podría reemplazar no terminales desde la derecha o algún otro patrón pues resultara que el orden en el que se apliquen las reglas, no afecta la determinación de si una cadena puede ser generada por la gramática.

6.3.6. Gramáticas sensibles al contexto

Las reglas de producción de una gramática 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 símbolo inicial siempre que σ no aparezca en el lado derecho de otra producción.

Quizás a simple vista podría parecer que las gramáticas 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. Gramáticas estructuradas por frases

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

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

6.3.8. Relación entre gramáticas

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 gramática G del tipo i tal que $L(G) = L$.

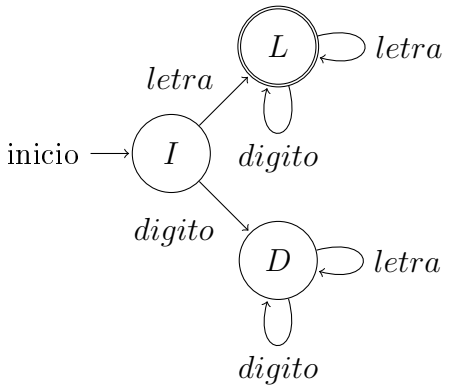
Capítulo 7

Teoría de autómatas

7.1. Autómatas finitos

7.1.1. Diagrama de transiciones

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



Diremos que una cadena es aceptada si sus símbolos 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 análisis léxico.

7.1.2. Autómata de estado finito determinista

7.1.2.1. Definición

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

- Σ es un conjunto finito de *símbolos de entradas.*
- S es un conjunto FINITO de *estados.*
- $f : S \times \Sigma \to S$ es una *función de transición.*
- $Ac \subseteq S$ es un conjunto de *estados de aceptación.*
- $\sigma \in S$ es el estado inicial.

La función de transición del diagrama anterior estaría definida de la siguiente forma:

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

Observación Nótese que cada estado del diagrama de transiciones de un autómata de estado finito DETERMINISTA solo debe tener un arco que salga para cada símbolo del alfabeto; de lo contrario, un autómata que llega a ese estado se enfrentara a una elección de cual debe ser el arco a seguir. Ademas, dicho diagrama deberá estar completamente definido, es decir, debe existir por lo menos un arco para cada símbolo del alfabeto; de lo contrario, un autómata que llega a ese estado puede enfrentarse a una situación donde no puede aplicar ninguna transición.

Estos requisitos están reflejados en la definición formal primero porque f es una función 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 autómata 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. $\sigma_n \in Ac$.

Observación La palabra vaciá es aceptada si y solo si el estado inicial es de aceptación ($\sigma \in Ac$).

7.1.2.3. Función de transición 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 autómata A como el conjunto: $\mathcal{AC}(A) = \{p \in \Sigma^* / p \text{ es aceptada por } A\}$.

7.1.2.5. Equivalencia de autómatas

Sean A_1, A_2 autómatas 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:

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

Demostración 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 producción:

- $U \to xV \iff f(U, x) = V$.
- $U \to \lambda \iff U \in Ac$

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 demostración.

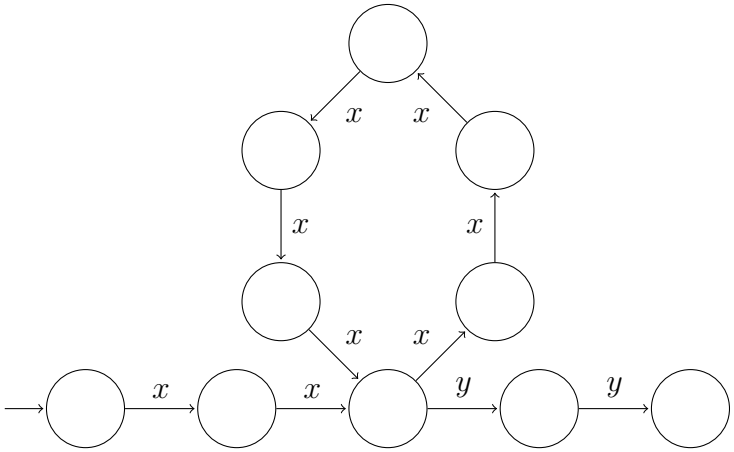
Observación Este teorema nos indica que los AEF no sirven para reconocer lenguajes independientes de contexto.

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 también contiene infinitas cadenas de la forma $x^m y^n$ con $m \neq n$.

Demostración El numero de estados de A ($|S|$) es finito. Sea $n > |S| : x^n y^n \in L$. Observemos que durante el proceso de aceptación de la palabra $x^n y^n$ (en particular, de la subcadena x^n) el autómata A deberá 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 aquí concluimos que A acepta también la palabra $x^{n+m}y^n$ (donde m denota la longitud del bucle).

7.1.2.8. Existencia de lenguajes libres de contexto

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

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

Observación Una consecuencia de este teorema es que los AEF no sirven como analizadores léxicos de expresiones aritméticas que contienen paréntesis, pues el autómata debería aceptar cadenas de la forma $({}^n)^n$ pero de ser así también aceptaría por ejemplo $(({}^n)^n$ que es incorrecta.

7.1.3. Autómata de estado finito no determinista

7.1.3.1. Definición

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

- Σ es un conjunto finito de *símbolos de entrada*.
- S es un conjunto finito de *estados*.
- $R \subseteq S \times \Sigma \times S$ es una *relación de transición*.

- $Ac \subseteq S$ es un conjunto de *estados de aceptación.*
- $\sigma \in S$ es el *estado inicial.*

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

Observación Como hemos cambiado una relación por una función, ahora al leer un nuevo símbolo se podrán seguir múltiples 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 función es a su vez una relación, resulta que todo AEF es también 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 $\mathcal{AC}(A') = \mathcal{AC}(A)$.

Veremos a continuación que en efecto, ambos conjuntos son idénticos.

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

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

- $S' = \mathcal{P}(S)$.
- $Ac' = \{s' \in \mathcal{P}(S) / s' \cap Ac \neq \emptyset\}$.
- $s'_0 = \{s_0\}$.
- $f : S' \times \Sigma \to S'$ 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 inducción sobre la cantidad de letras de p, mas precisamente mostraremos que $\forall n \in \mathbb{N}_0$ vale el siguiente enunciado E_n: «*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 $s_n \in s'_n$. Recíprocamente, 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}. 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 $s_{n+1} \in s'_{n+1}$.

  Recíprocamente 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 definición 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. Igualdad entre $\mathcal{AC}(AEF)$, $\mathcal{AC}(AEFND)$ y $\mathcal{L}_3$

Puesto que toda función es ademas relación sabíamos 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 también probamos que dado un AEF existe una gramática regular que genera el mismo lenguaje. En síntesis: $\mathcal{AC}(AEFND) = \mathcal{AC}(AEF) \subseteq \mathcal{L}_3$. A continuación veremos que estos tres conjuntos son iguales.

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

Demostración Queda como ejercicio al lector completar esta demostración.

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

7.2. Expresiones regulares

7.2.1. Definición

Una expresión 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 paréntesis, convenimos el siguiente orden de precedencia: $*, \circ, \cup$.

Observación Nótese que una expresión regular es una cadena de símbolos, no un lenguaje.

7.2.2. Lenguaje asociado

El lenguaje asociado a una expresión esta dado por la función $L : ER \to \mathcal{L}$ 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.

7.2.3. Igualdad entre lenguajes asoc. a L_{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.

Demostración Deberemos probar una doble contención:

- $L_{ER} \subseteq \mathcal{L}_3$: Lo demostraremos por inducción sobre ER
  - $e = \emptyset \Rightarrow L(e) = \emptyset \in \mathcal{L}_3$.
  - $e = x \Rightarrow L(e) = \{x\} \in \mathcal{L}_3$.
  - $e = p \cup q \Rightarrow$: Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \cup q) = L(p) \cup L(q) \in \mathcal{L}_3$ y la unión es cerrada en $\mathcal{L}_3$. (Ejercicio).
  - $e = p \circ q \Rightarrow$: Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \circ q) = L(p) \circ L(q) \in \mathcal{L}_3$ y la concatenación es cerrada en $\mathcal{L}_3$. (Ejercicio).
  - $e = p^* \Rightarrow$: Debemos probar que $L(p) \in \mathcal{L}_3 \Rightarrow L(p^*) = L(p)^* \in \mathcal{L}_3$ y la clausura de Kleene es cerrada en $\mathcal{L}_3$. (Ejercicio).
- $\mathcal{L}_3 \subseteq L_{ER}$: Consultar bibliografía [2], pagina 63.

7.3. Autómatas de pila

7.3.1. Introducción

Los autómatas de pila son autómatas que cuentan con las mismas características que un $AEFND$ pero ademas disponen de una pila de memoria en la que podrán leer o escribir símbolos, de forma de poder saber en cada momento si el autómata ya realizo alguna transición en el pasado.

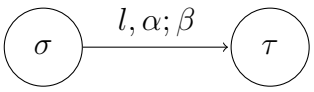
Durante cada transición el autómata ejecuta la siguiente secuencia:

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

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

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

Representaremos este proceso mediante el siguiente diagrama de transiciones:



Puesto que permitimos que l, α, β sean la cadena vaciá, podemos en cualquier transición no leer un caracter, no extraer nada de la pila o no escribir nada en la pila si así lo necesitáramos.

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

Observación Nótese 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: $\mathcal{L}_3 \subseteq \mathcal{AC}(AP)$.

7.3.2. Definición formal

Un autómata 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 *símbolos de entrada*.
- Γ es un conjunto finito de *símbolos de pila*.
- $T \subseteq S \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times S$ es una *relación de transición*.
- $\sigma \in S$ es el *estado inicial*.
- $Ac \subseteq S$ es un conjunto de estados de aceptación.

7.3.3. Configuración

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

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

7.3.4. Relación entre configuraciones

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

La relación «lleva en uno o mas pasos» ($\Rightarrow_A^*$) define recursivamente a partir de la relación $\Rightarrow_A$ de forma análoga a lo hecho para la función de transición f de los AEF.

7.3.5. Lenguaje aceptado

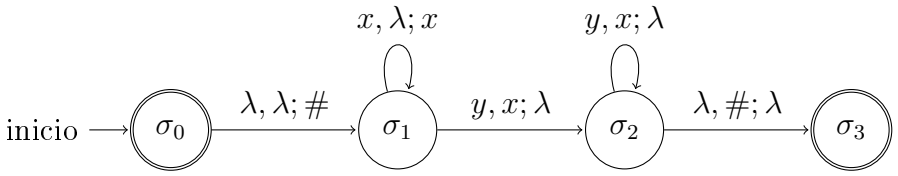
Sea $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ un autómata de pila, el lenguaje aceptado por A es el conjunto: $\mathcal{AC}(A) = \{p \in \Sigma^* / (\sigma, p, \lambda) \Rightarrow^* (\tau, \lambda, \gamma) : \gamma \in \Gamma^* \wedge \tau \in Ac\}$.

Es decir, una cadena p sera aceptada por un AP si, arrancando desde su estado inicial y con la pila vaciá, es posible que el autómata llegue a un estado de aceptación después de leer toda la cadena.

Observación No necesariamente se llegara a un estado de aceptación luego de leer el ultimo caracter de una palabra pues a continuación el autómata podría realizar transiciones de la forma $(\sigma, \lambda, \alpha; \beta, \tau)$ y llegar luego al estado de aceptación.

7.3.6. Aceptacion de lenguajes no regulares

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



7.3.7. Teorema del vaciado de pila

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

Demostración 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 transición del nuevo al anterior que lo único que hace es insertar en la pila un marcador # (suponiendo que $\# \notin \Gamma$).
- Los estados de aceptación 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 aceptación 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 único estado de aceptación Q junto a la transición $(P, \lambda, \#; \lambda, Q)$.

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 vaciá, α nos lleva en uno o mas pasos a un estado de aceptación. En términos de configuraciones: $(\sigma, \alpha, \lambda) \Rightarrow^* (\tau, \lambda, \gamma)$ donde $\tau \in Ac, \gamma \in \Gamma^*$. Luego en A' tenemos la derivación:

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

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

- $\supseteq$: Análogo.

7.3.8. Igualdad entre $\mathcal{L}_2$ y $\mathcal{AC}(AP)$

Enunciado

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

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

Demostración

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

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

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

7.3.9. Lema de bombeo para autómatas 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 $xu^n yv^n z \in L \; \forall n \in \mathbb{N}$.

Demostración Sabemos que existe una gramática independiente de contexto $G = (N, T, P, \sigma)$ tal que $L(G) = L$ y sea m el máximo numero de símbolos de $N \cup T$ que aparecen en el lado derecho de las reglas de producción de G. Observemos que como m es la mayor cantidad de símbolos por los que se puede reemplazar un no terminal, al aplicar i reglas de producción cualesquiera, la longitud de la cadena resultante es a lo sumo m^i.

Llamemos k a la cantidad de símbolos no terminales de G ($k = |N|$). Dada una palabra $p \in L$ tal que $|p| > m^k$ (que existe pues L es infinito), llamemos i a la cantidad de reglas de producción aplicadas para producir p. Por lo tanto $m^k \leq |p| \leq m^i \Rightarrow k < i$ es decir, para formar p debieron aplicarse mas de k reglas de producción (debieron expandirse mas de k no terminales).

Como no hay k no terminales, existe un no terminal X que debió expandirse 2 veces que podremos «bombear» cuantas veces sea necesario.

7.3.10. Existencia de lenguajes sensibles al contexto

Enunciado Existen lenguajes sensibles al contexto.

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

Como L es infinito 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 símbolo 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 símbolos y por lo tanto no pertenece al lenguaje.

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

7.4. Maquinas de Turing

7.4.1. Descripción

Al igual que los demás autómatas 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 cálculos. Por convención 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 cálculos. De esta manera, el estado de parada de una maquina de Turing difiere de los estados de aceptación de los autómatas de estados finito y los autómatas de pila en que estos pueden continuar sus cálculos después de llegar a un estado de aceptación, 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.

Nótese que con base en la definición 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 autómatas de los capítulos 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 símbolos en la cinta de la maquina, que es infinita a izquierda y derecha, pero se conviene que solo un conjunto finito de casillas no están vaciás. Así una maquina de Turing puede emplear su cinta como almacenamiento auxiliar tal como lo hacen los autómatas de pila. Sin embargo con este almacenamiento una maquina de Turing no se limita a las operaciones de inserción y extracción, sino que puede rastrear los datos de la cinta y modificar las celdas que desee sin alterar las demás.

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 símbolos que no aparecen en los datos de entrada; es decir, hacemos una distinción entre el conjunto (finito) de símbolos, llamado alfabeto de la maquina, en el que deben estar codificados los datos de entrada iniciales, y un conjunto, posiblemente mayor (también finito), de símbolos de la cinta, que la maquina puede leer y escribir. Esta distinción es similar a la que se establece entre el alfabeto de un autómata y sus símbolos de pila. El símbolo blanco es un símbolo de la cinta que esta en cualquier celda de la cinta que

no este ocupada. Notaremos a este símbolo como $\square$.

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

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

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

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

La acción que se ejecutara en un momento dado dependerá del símbolo apuntado por el cabezal así como del estado actual del mecanismo de control de la maquina. Si representamos con Γ al conjunto de símbolos 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 función $f : S' \times \Gamma \to \Gamma \times \{d, i, n\} \times S$. Podemos entonces interpretar $f(q_r, \alpha) = (\beta, d, q_n)$ como «si el estado actual es q_r y el símbolo actual es α: reemplazar α por β, mover el cabezal a la derecha y pasar al estado q_n».

Esta función, 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 símbolo s_i aparecerá una terna que indicara que símbolo escribir, el movimiento a ejecutar y el estado al que pasar. La primer columna de la tabla corresponderá al estado q_1.

Nótese que al describir con una función las transiciones de una maquina de Turing, la maquina es determinista. Para ser mas precisos, existe una y solo una transición asociada a cada par estado-símbolo, donde el estado no es el de detención.

Durante la operación 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 cálculos de una maquina de Turing, ya que su programa interno puede quedar atrapado en un ciclo sin fin.

7.4.2. Definición 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 símbolos llamado alfabeto de la maquina.
- Γ es un conjunto finito de símbolos llamados símbolos de la cinta, tal que $\Sigma \subseteq \Gamma$.
- $\square \in \Gamma$ es el símbolo 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 función de transición de la maquina.

7.4.3. Maquinas elementales

Es posible demostrar que con un alfabeto de un solo símbolo (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 continuación 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 podrá construir cualquier maquina de Turing sobre el alfabeto dado.

7.4.4. Composición

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 composición de M con M' a la maquina $MM' = (S \cup S' - \{q_0\}, \Sigma, \Gamma, f, \square, q_1, q'_0)$ que realiza la operación equivalente a aplicar primero la maquina M y al resultado aplicar M', donde f esta dada por:

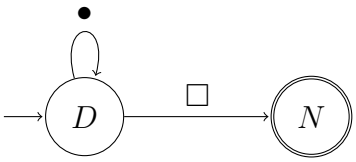
$$f(q,s) \begin{cases} f_M(q,\alpha)[q'_1/q_0] & q \in S \\ f_{M'}(q,\alpha) & q \in S' \end{cases}$$

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 aparición de q_0 por q'_1.

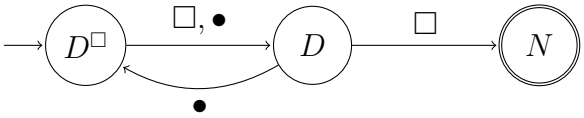
7.4.5. Diagramas de composición

Veamos una forma mas general de componer maquinas. En este caso ante el estado final de una maquina se toma en consideración 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 símbolo 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:

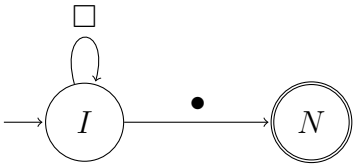
- Maquina «derecha hasta blanco» (que notaremos $D^{\square}$):



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



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



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

7.4.6. Representación de FRL con MT

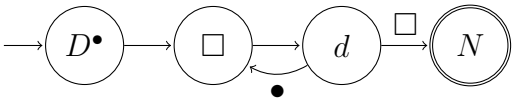
Mostraremos que dada una función de lista, existe una maquina de Turing sobre $\Sigma = \{\square, \bullet\}$ que la representa. Para ello definiremos primero una representación de una lista, en la cinta de la maquina.

Representaremos entonces las listas como grupos de $\bullet$ separados por un solo símbolo 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 representación de una lista $X \in \mathcal{L}$ en una cinta de una maquina de Turing como $\langle\langle X \rangle\rangle$.

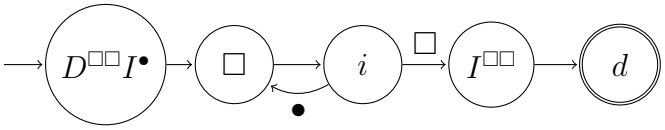
Diremos que dada una función de lista F, una maquina de Turing (que notaremos M_F) la representa si dada cualquier lista $X \in Dom\,(F)$, la maquina M_F tomando como configuración inicial $\langle\langle X \rangle\rangle$ y con el cabezal en la primer casilla vaciá antes del primer $\bullet$, luego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

7.4.6.1. Funciones base

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



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



7.4.6.2. Composición

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

7.4.6.3. Maquina Z

Introducción La maquina Z es una maquina de Turing sobre $\Sigma = \{\bullet, \square\}$ que parte en la primer celda vaciá 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 configuración final es igual a la inicial.
- Si los grupos primero y ultimo no tienen el mismo numero de elementos, entonces la configuración final es igual a la inicial, pero con el cabezal apuntando al primer $\bullet$.

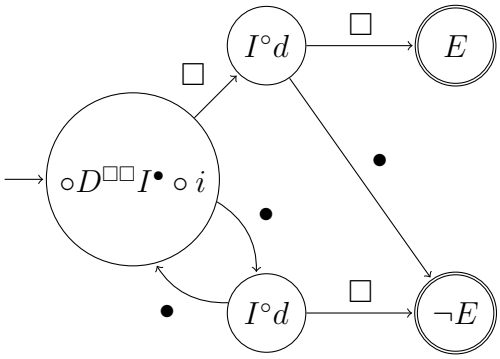
Definiciones Definiremos primero cinco maquinas:

1. La maquina preámbulo P, que no para si la entrada no es correcta.
2. La maquina contadora C, que vaciá los puntitos de a pares, mientras se pueda.
3. La maquina igualdad E, que actúa si el conteo fue igual.
4. La maquina desigualdad $\neg E$, que actúa 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:

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

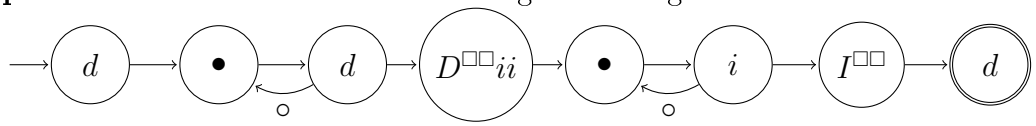
Maquina E Definimos $E := I^{\square\square}dR$. Veamos algunos ejemplos:

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

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:



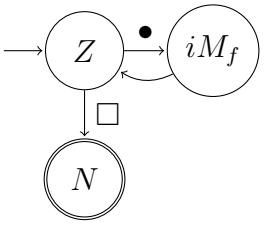
Veamos algunos ejemplos:

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

Conclusión Nuestra maquina quedaría definida como $Z := PC$.

7.4.6.4. Repetición

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



7.4.7. Representación 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 símbolos $\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}$ donde $s_0 = \square$ y M tiene $p + 1$ estados.

7.4.7.1. Representación de configuraciones

En cada momento la situación 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 símbolos 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 símbolo 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 símbolos 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 configuración 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 números primos para las bases, a partir de un numero natural que represente una configuración podemos recuperar los exponentes mediante una función recursiva primitiva G tal que $G(x,2) = C_i$, $G(x,3) = C_o$, $G(x,5) = C_d$ y $G(x,7) = e$.

7.4.7.2. Transiciones sobre representaciones

Representaremos los movimientos del cabezal con números: $i = 1$, $d = 2$ y $n = 3$. Observemos como se comporta sobre una cinta, una función de transición para el estado actual q y el símbolo 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 configuración $N = 2^{C_i}3^{C_o}5^{C_d}7^q$ 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 $N = 2^63^35^{10}7^2 = 826.875.000.000$) reemplazando el símbolo actual por 4 y pasando al estado final:

- Si $m = 3$ se transforma en $N' = 2^63^45^{10}7^0 = 16.875.000.000$ es decir $\underbrace{6\check{4}01}_{0}$.
- Si $m = 2$ se transforma en $N' = 2^{10\cdot6+4}3^{10\,\%10}5^{\lfloor 10/10\rfloor}7^0 = 2^{64}3^05^17^0$ 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 aquí definimos tres funciones recursivas que estarán asociadas a una función de transición f de una MT:

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

7.4.7.3. Demostración

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 función que representa configuraciones como números.

Definimos la función 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 función 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 cálculos. Si efectivamente existe un valor de t tal que satisfaga dicha igualdad entonces este valor sera el mínimo 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 función que encuentra dicho valor.

De todo lo aquí expuesto podemos concluir que la función 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 través de G sea 0 y de esta manera obtenemos un numero que representa la configuración final de M aplicada a C_1.

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

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

7.4.8. Numerabilidad de maquinas de Turing

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

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

Nótese que M_{nk} es un conjunto finito. En efecto hay un numero finito de funciones de transición, 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 unión numerable de conjuntos numerados.

7.4.9. 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 numéricas. 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 continuación una función útil 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 configuración inicial se llega a una configuración final, por ejemplo la maquina $D^{\bullet\bullet}$ con una cinta inicial que no tenga dos $\bullet$ seguidos no se detendrá nunca.

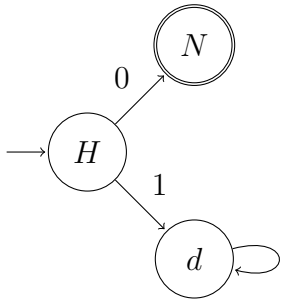
Intentaremos construir una maquina que dada cualquier maquina de Turing y cualquier entrada, nos indique si esta finaliza sus cálculos o no. Sin lugar a dudas para casos particulares podemos determinar dicha respuesta, pero buscamos una maquina de propósito general.

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

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

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

¿Es h una función 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. Contradicción.

- 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. Contradicción.

La contradicción vino de suponer que H existía, luego H no existe.

7.4.10. 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 múltiples cintas.
- Maquina de Turing no determinista.
- Autómata de múltiples pilas.
- Autómata de cola.
- Calculo λ.

7.4.11. Ejemplos

7.4.11.1. 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 función de transición:

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 vaciá:

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

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

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

7.4.11.2. 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 función de transición:

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 $\check{3}99$:

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

7.4.11.3. 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 función de transición:

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 símbolo 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 $\check{a}ab$:

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

7.4.11.4. Duplicar puntos

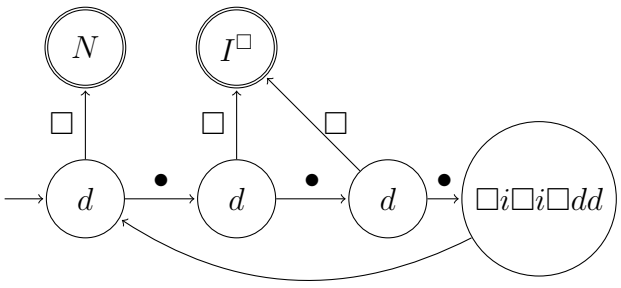
Definimos $M = (S, \Sigma, \Gamma, f, \square, q_1, q_0)$ con la siguiente función de transición:

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

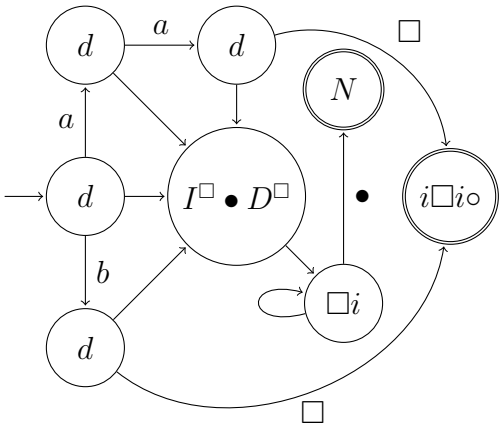
Observemos como se comporta esta maquina partiendo de la cinta inicial con la palabra $\check{\bullet}\bullet\bullet$:

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

7.4.11.5. Modulo 3



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



7.4.11.7. Representación en FR

La maquina suma unaria esta representada por la siguiente función de transición:

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 partirá 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 $2^0 3^0 5^{11111111} 7^0$.

Observemos una traza de la función 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 y Dante Zanarini. *Cátedra de Lenguajes Formales y Computabilidad.*

[2] J. Glenn Brookshear. *Teoría de la Computación. Lenguajes formales, autómatas y complejidad.*

[3] Julio Hurtado, Raúl Kantor, Carlos Luna, Luis Sierra y Dante Zanarini. *Temas de Teoría de la Computación.*

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

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

[6] ProofWiki. *proofwiki.org.*