Universidad Nacional de Ingeniería

Facultad de Ciencias

Teoría de la Computación

El presente cuaderno de apuntes fue desarrollado por **CJ** y toma como referencia las clases del profesor Victor Melchor, respetándose las definiciones que se plantearon en clase.

2018

Ciencias de la Computación

Índice general

Capítulo 1

Fundamentos Matemáticos de la Teoría de la Computación

Muchos problemas en fundamentos de computación pueden verse como problemas concernientes a conjuntos.

1.1. Conjuntos

Un conjunto es una colección de objetos distinguibles. **Notación:** $b \in L$ $x \notin L$. En un conjunto no se repiten los elementos, el orden es irrelevante.

1.1.1. Conjunto Unitario

Es aquel formado por un único elemento.

$$U = \{1\}$$

1.1.2. Conjunto Vacío

Es el conjunto sin elementos. **Notación:** ϕ o $\{\}$.

1.1.3. Conjuntos Importantes

- Enteros no Negativos $\mathbb{N}$

- Enteros Positivos $\mathbb{N}^+$
- Enteros $\mathbb{Z}$
- Racionales $\mathbb{Q}$
- Reales $\mathbb{R}$
- Complejos $\mathbb{C}$

Podemos expresar algunos conjuntos mediante referencia a otros conjuntos y a las propiedades que los elementos pueden o no tener.

Si $F = \{1, 3, 9\} \quad G = \{3, 9\}$
$G = \{x/x \in F \wedge x > 2\}$
$B = \{x/x \in A \wedge x \text{ tiene la propiedad P}\}$

Podemos expresar el conjunto de los números naturales impares.

$$I = \{x/x \in \mathbb{N} \wedge \text{x no es divisible por 2}\}$$

1.1.4. Subconjuntos

A es un subconjunto de B si cada elemento de A también está en B. **Notación:** $A \subseteq B$. Ejemplo: $I \subseteq \mathbb{N} \quad A \subseteq A$

1.1.5. Subconjunto Propio

Si A es subconjunto de B pero es diferente a B se le llamará subconjunto propio de B. **Notación:** $A \subset B$.

1.1.6. Conjunto Universal

También conocido como Universo del Discurso. **Notación:** $\mathfrak{U}$.

1.1.7. Complemento de un Conjunto

Dado un conjunto $A \subseteq \mathfrak{U}$ el conjunto de A se denota por $\bar{A}$ y está definido por.

$$\bar{A} = \{x/x \in \mathfrak{U} \wedge x \in A\}$$

1.1.8. Igualdad de Conjuntos

Dos conjuntos A y B son iguales si $A \subseteq B \wedge B \subseteq A$. **Notacion:** $A = B$.

1.1.9. Conjuntos Disjuntos

A y B son disjuntos si $A \cap B = \phi$

1.2. Operaciones Binarias

Unión: $A \cup B = \{x/x \in A \vee x \in B\}$

Intersección: $A \cap B = \{x/x \in A \wedge x \in B\}$

Diferencia: $A - B = \{x/x \in A \wedge x \notin B\}$

Diferencia Simétrica: $A \Delta B = (A - B) \cup (B - A)$

Producto Cartesiano: $A \times B = \{(a, b)/a \in A \wedge b \in B\}$

1.3. Leyes de la Teoría de la Computación

Sea A,B y C conjuntos, se cumplen las siguientes leyes:

1. **Idempotencia:** $A \cup A = A \quad A \cap A = A$

2. **Conmutatividad:** $A \cup B = B \cup A \quad A \cap B = B \cap A$

3. **Asociatividad:**
   $(A \cup B) \cup C = A \cup (B \cup C)$
   $(A \cap B) \cap C = A \cap (B \cap C)$

4. **Distributividad:**
   $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
   $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

5. **Absorción:**
   $(A \cup B) \cap A = A$
   $(A \cap B) \cup A = A$

6. **Leyes de Morgan:**
   $A - (B \cup C) = (A - B) \cap (A - C)$
   $A - (B \cap C) = (A - B) \cup (A - C)$

Prueba del item 6:
Sea: $I = A - (B \cup C) \quad , D = (A - B) \cap (A - C)$
Se debe cumplir que.$I \subseteq D \wedge D \subseteq I$

- PP $I \subseteq D$
  Sea $x \in I \rightarrow x \in A \quad pero\ x \notin (B \cup C)$
  $\rightarrow x \in A\ pero\ x \notin B \wedge x \notin C$
  $\rightarrow (x \in A\ pero\ x \notin B) \wedge (x \in A\ pero\ x \notin C)$
  $x \in (A - B) \wedge x \in (A - C)$
  $x \in (A - B) \cap (A - C) \quad \rightarrow x \in D$

- PP $D \subseteq I$
  Sea $z \in (A - B) \cap (A - C)$
  $z \in (A - B) \wedge z \in (A - C)$
  $\rightarrow (z \in A\ pero\ z \notin B) \wedge (z \in A\ pero\ z \notin C)$
  $\rightarrow z \in A\ pero\ z \notin (B \cup C)$
  $\rightarrow z \in (A - (B \cup C))$
  $\rightarrow z \in I$
  $\therefore I = D$

1.4. Operaciones Generalizadas

1. Unión Generalizada:
   $$\bigcup_{i \in I} A_i = \{x / \exists_i \in I \quad x \in A_i\}$$

2. Intersección Generalizada:
   $$\bigcap_{i \in I} A_i = \{x / x \in A_i \quad \forall_i \in I\}$$

3. Producto Cartesiano Generalizado:
   $\otimes_{i=1}^{n} A_i = \{x/x = (x_1, x_2, ..., x_n), x_i \in A_i\}$
   $A^i = \otimes_{i=1}^{n} A$

1.5. Conjunto Potencia

Dado un conjunto A, la colección de todos los subconjuntos de A y ella misma se llama conjunto potencia. **Notación:** 2^A.

Ejemplo:
$A = \{c, d\}$
$2^A = \{\phi, \{c\}, \{d\}, A\}$

1.6. Cardinalidad

$|A|$ denota la cantidad de elementos del conjunto A.
Ejemplo:
$|A| = 2$
$|2^A| = 4 \qquad |2^A| = 2^{|A|}$

1.7. Partición de un Conjunto

Sea $A \neq \phi$, una partición de A es un subconjunto π de A tal que:

1. Cada elemento de π es no vacío.
   $B_i \neq \phi \quad \forall_i \in I$
2. Miembros distintos de π deben ser disjuntos.
   $B_i \cap B_j = \phi \quad i \neq j$
3. $\bigcup \pi = A \qquad \bigcup B_i = A$

Ejemplo: Sea $A = \{a, b, c, d\}$

- $\{\{a, b\}, \{c\}, \{d\}\}$ (1)si (2)si (3)si, por tanto es una partición.

- $\{\{b, c\}, \{c, d\}\}$ (1)si (2)no (3)no, por tanto no es una partición.

Sea m un entero positivo fijo, se define $\mathbb{Z}_i$ como: $\mathbb{Z}_i = \{x/x \in \mathbb{Z} \wedge x - i = k.m,$ para algún entero k$\}$

Ejemplo: Sea m=3 entonces:

1. Obtener $\mathbb{Z}_0, \mathbb{Z}_1, \mathbb{Z}_2$
2. Verifique si $\mathbb{Z}_0, \mathbb{Z}_1, \mathbb{Z}_2$ es una partición de $\mathbb{Z}$.

Solución:

- $\mathbb{Z}_0 = \{..., -6, -3, 0, 3, 6, ...\}$
  $\mathbb{Z}_1 = \{..., -5, -2, 1, 4, 7, ...\}$
  $\mathbb{Z}_2 = \{..., -4, -1, 2, 5, 8, ...\}$
- (1)$\mathbb{Z}_i \neq \phi$: si cumple.
  (2)$\mathbb{Z}_0 \cap \mathbb{Z}_1 = \phi, \mathbb{Z}_0 \cap \mathbb{Z}_2 = \phi, \mathbb{Z}_1 \cap \mathbb{Z}_2 = \phi$: si cumple.
  (3)$\bigcup \mathbb{Z}_i = \mathbb{Z}$: si cumple. Por lo tanto si es una partición.

1.8. Representación en Computador de Conjuntos

Supongamos que $\mathfrak{U}$ es finito. Establecemos un orden arbitrario de los elementos de $\mathfrak{U}$. Ejemplo: $a_1, a_2, ..., a_n$. Representamos un conjunto A de $\mathfrak{U}$ con la cadena de bits de longitud n, en la cual el i-ésimo bit es "1" si a_i pertenece a A, y "0" si $a_i \notin A$.
Ejemplo: $\mathfrak{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, supongamos que los elementos están en orden creciente,$a_i = i$. Escriba representaciones en cadena de bits para:

1. Enteros impares en $\mathfrak{U}$: **I**
2. Enteros pares en $\mathfrak{U}$: **P**
3. Subconjuntos de enteros no mayores que 5.

Solución:

1. La cadena de bits que representa a **I**:
   1010101010

2. La cadena de bits que representa a **P**:
   0101010101

3. No mayores a 5:
   1111100000

Obs:

- Cadena de bits para el complemento a **I**: 0101010101

- Cadena para la unión de **I** y **P**:

$$\begin{array}{c} 1010101010 \\ 0101010101 \\ \hline 1111111111 \end{array}$$

Capítulo 2

Relación en TC

Utilizaremos un dispositivo llamado par ordenado (p.o).
Notación: (a, b)
$(a, b) \neq \{a, b\}$
El orden es relevante $(a, b) \neq (b, a)$
(a, b) es igual a (c,d) si $a = c \wedge b = d$
Producto cartesiano $A \times B = \{(a, b)/a \in A \wedge b \in B\}$

2.1. Relación Binaria

Ejemplo:

1. Si $A = \{1, 3, 9\} \quad B = \{b, c, d\}$
   $R_1 = \{(1, b), (1, c), (3, d), (9, d)\} \subset A \times B$

2. $R_2 = \{(i, j)/i, j \in \mathbb{N} \wedge i < j\} \subset \mathbb{N} \times \mathbb{N}$

En una relación binaria $R \subseteq A \times B$ definimos: $dom(R) = \{a/\exists b \in B, \quad (a, b) \in R\}$
$ran(R) = \{b/\exists a \in A, \quad (a, b) \in R\}$
Hay dos relaciones unarias basicas
$R^{-1} = \{(b, a)/(a, b) \in R\}$
$R^C = A \times B = R$
Una operación importante es la composición $R_2 \circ R_1$ definida para el caso en que $ran(R_1) \subset dom(R_2)$

$R_2oR_1 = \{(x,z)/\exists y\ (x,y) \in R_1 \wedge (y,z) \in R_2\}$
Si $R \subseteq S \times S$ para un subconjunto S entonces diremos que "R es una relación en S".
La relación identidad.
$I_s = \{(a,a)/a \in S\}$
Sea S una relación en S definimos.

Las potencias R^i por:
$R^o = I_s \quad R^{i+1} = RoR^i \quad i \geq 0$

La clausura transitiva.
$R^+ = \bigcup_{i=1}^{\infty} R^i$

La clausura transitiva y reflexiva.

$R^* = \bigcup_{i=0}^{\infty} R^i$

2.2. Propiedades Básicas de las relaciones binarias

Una relación binaria $R \subseteq S \times S$ se llama:
Reflexiva: Si $a \in S \rightarrow (a,a) \in R$
Simétrica: Si $(a,b) \in R \rightarrow (b,a) \in R$
Antisimétrica: Si $(a,b) \in R \wedge a \neq b \rightarrow (b,a) \notin R$
Transitiva: Si $(a,b) \in R \wedge (b,c) \in R \rightarrow (a,c) \in R$
Definición:
Una relación R se dirá:

- De equivalencia, si es reflexiva,simétrica y transitiva.
- Un orden parcial si es reflexiva, antisimétrica y transitiva.
- Un orden total si R es un orden parcial $\forall a,b \in S\ \vee\ (a,b) \in R \vee (b,a) \in R$

Definición: Si R es una relación de equivalencia en S y $a \in S$ entonces el conjunto.

$[a] = \{b/(a,b) \in R\}$
Se le llama la clase de equivalencia de a respecto a la relación R.

Lema: Si R es una relación de equivalencia en S y $a, b \in S$ las proposiciones son equivalentes:

1. $(a,b) \in R$

2. $[a]_R = [b]_R$

3. $[a]_R \cap [b]_R \neq \phi$

2.3. Representación de las Relaciones

Dos de las mas importantes representaciones para relaciones son las matrices booleanas y los grafos dirigidos.
Sea $R \subseteq S \times S$ una relación binaria, donde $|S| = n$. Esta puede representarse mediante una matriz booleana $n \times n$. M_R que estará formada por "1" si $(a,b) \in R$ y "0" si $(a,b) \neq R$.

Ejemplo: Sea R la relación.
$R = \{(0,0),(0,1),(1,2),(1,3),(2,4),(2,5),(3,6),(3,7),$
$(4,0),(4,1),(5,2),(5,3),(6,4),(6,5),(7,6),(7,7)\}$
Su matriz booleana es:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

También podemos usar un grafo dirigido $G_R = (V,E)$ donde $V = dom(R) \cup ran(R)$ y $E = \{(a,b)/(a,b) \in R\}$

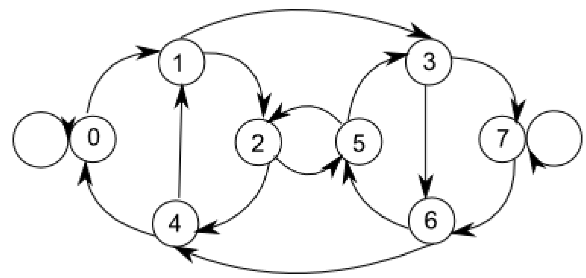

Figura 2.1: Grafo dirigido

Si $M_{R_i} \quad i = 1, 2$ es una representación en matriz booleana de la relación R_i, entonces tenemos:

$$M_{R_1 \cup R_2} = M_{R_1} \vee M_{R_2} \quad ; \quad M_{R_1 \cap R_2} = M_{R_1} \wedge M_{R_2}$$

2.4. Posets

Un conjunto S junto con una relación de orden parcial R se denomina un conjunto parcialmente ordenado o Poset y se le denota por (S, R). En el caso de los Posets se usa la notación aRb para indicar que $(a, b) \in R$.

Ejemplo: Si $\backslash$ denota la relación de divisibilidad entre enteros, entonces el par $(\mathbb{N}, \backslash)$ es un poset.

Ejemplo:

$$\begin{array}{rl} 7 \in \mathbb{N}, (7,7) \in \backslash & reflexiva. \\ (6,2) \in \backslash \wedge 6 \neq 2 \; pero \; (2,6) \notin \backslash & antisimetrica. \\ (16,8) \in R \wedge (8,2) \in R \rightarrow (16,2) \in R & transitiva. \end{array}$$

Si A es cualquier conjunto, entonces $(2^A, \subseteq)$ es también un poset. Los posets pueden ser representados gráficamente mediante los diagramas Hasse. Estos están basados en el hecho que si $aRb \wedge bRc$ en un Poset (S,R) entonces aRc.

Sea (S,R) un poset. Denotamos $R_H \subseteq R$ la relación definida como sigue: aR_Hb si y solo si aRb y no hay $a \neq c \neq b$ tal que aRc y cRb.

En otras palabras R_H es el conjunto mas pequeño con $R_H^* = R$.

Ejemplo: Sea $S = \{0, 1, 3\}$ y $R =\subseteq$, graficaremos el Poset $(2^S, \subseteq)$

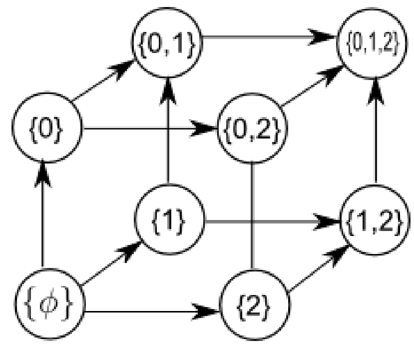

Figura 2.2: Gráfico del Poset

$2^S = \{\phi, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, S\}$

2.5. Funciones

El conjunto $f \subset A \times B$ es una función si:

- $Dom(f) = A$
- Si $(x, y), (x, z) \in f$ *entonces* $y = z$

Esto significa que para todo elemento x en A, existe un único y tal que $(x, y) \in f$.
Notación: $f : A \to B$
$y = f(x) \quad (x, y) \in f$

A una función de éstas características la denominaremos función total.

Teorema: Sean las funciones $f : A \to B$ y $g : A \to B$ Entonces $f = g$ si $f(x) = g(x) \forall x \in A$.

2.5.1. Función Parcial

Una relación $f \subseteq A \times B$ se denomina función parcial si:

- $Dom(f) \subseteq A$
- Si $(x, y), (x, z) \in f$ *entonces* $y = z$

2.5.2. Imagen

Sea $f : A \to B$ una función. Si $X \subseteq A$ se dice la imagen de X bajo f a : $f(x) = \{y \in B/y = f(x); para\ x \in X\}$

2.5.3. Imagen Inversa

Sea $Y \subseteq B$, la imagen inversa de Y bajo f es: $f^{-1}(Y) = \{x \in A/f(x) = y$ para algún $y \in Y\}$

Definición: Sea $f : A \to B$ una función. Su rango es: $\{f(x)/x \in A\}$ que es un subconjunto del codominio B.

2.5.4. Tipos Especiales de Funciones

Sea $f : A \to B$ una función.

1. **Función Inyectiva:** f es inyectiva ó 1 a 1 si cumple:
   Para cualquier $(x, z) \in f \wedge (y, z) \in f \Rightarrow x = y$ o equivalente: si $f(x) = f(y) \Rightarrow x = y$

2. **Función Sobreyectiva:** f es sobreyectiva si su rango es todo el conjunto B o equivalente. $\forall y \in B, \exists x \in A/y = f(x)$

3. **Función Biyectiva:** Si f es a la vez inyectiva y sobreyectiva se dirá biyectiva. Cuando f es una biyección, $f^{-1} : B \to A$ es una función. La función inversa $f^{-1} : B \to A$ satisface:
   $f^{-1}(f(x)) = x \wedge f(f(f^{-1}(x)) = y$

Ejemplo: Para las siguientes funciones indique si son inyectivas y/o sobreyectivas.

1. $f : \mathbb{N} \to \mathbb{N}/f(n) = n$

2. $g : \mathbb{N} \to \mathbb{Z}/g(n) = n + 1$

3. $h : \mathbb{R} \to \mathbb{R}/h(x) = x^2$

Solución: Analizando cada uno de los casos.

- f es inyectiva, f es sobreyectiva.
- g es inyectiva pero no sobreyectiva.
- Si $h(x) = h(y)$
  $x^2 = y^2$
  $\rightarrow x = y \vee x = -y$, entonces h no es inyectiva.
- Si $y = -8$, $\nexists x \in \mathbb{R}/h(x) = y$, entonces h no es sobreyectiva.

2.6. Composición de Funciones

Sean $f : A \rightarrow B \wedge g : C \rightarrow D$ definimos $g \circ f : A \rightarrow D$

$$g \circ f(x) = \begin{cases} g(f(x)) & x \in Dom\, f \wedge f(x) \in Dom\, g \\ indefinido & \text{para otros casos} \end{cases}$$

En general $f \circ g \neq g \circ f$

Ejemplo: Sean:

$$f : \mathbb{Z} \rightarrow \mathbb{N}/f(n) = |n| + 1$$
$$g : \mathbb{N} \rightarrow \mathbb{Z}/g(n) = 1 - n$$

Halle $f \circ g, g \circ f$.

Capítulo 3

Operaciones Binarias

Sea A un conjunto, una operación binaria en A es una función $f : A \times A \rightarrow A$.

- $Dom(f) = A \times A$
- Solo un elemento de A se asigna a cada par (a,b)

Convención: Usaremos * en vez de f. $a * b$ en vez de f(a,b).
Ejemplo:

1. Si $A = \mathbb{Z}$ Definiendo a*b = a+b.
   * es na operación binaria.

2. Sea $A = \mathbb{R}$ y definimos $a * b = a/b$
   * no es una operación binaria pues para (a,0) no está definida a*0.

3. Sea $A = \mathbb{Z}^+$ y definimos $a * b = a - b$
   * no es una operación binaria (ejem: 3*7 = -4, -4 no pertenece a A)

4. Sea $A = \mathbb{Z}$ y sea $a * b = max(a, b)$
   * es una operación binaria.

5. Dado un conjunto S y su respectivo P(S). Si V y W son subconjuntos de S, definimos $V * W = V \cup W$
   Ejemplo: $S = \{1, 3, 5\}$
   $P(S) = \{\phi, \{1\}, \{3\}, \{5\}, \{1, 3\}, \{1, 5\}, \{3, 5\}, S\}$

$\{3\} * \{1,5\} = \{1,3,5\}$
* es una operación binaria.

Para un conjunto finito $A = \{a_1, a_2, ..., a_n\}$ podemos definir una operación binaria mediante la tabla.

$$\begin{array}{c|ccccc}
* & a_1 & a_2 & \dots\ a_j\ \dots & a_n \\
\hline
a_1 & & & \vdots & \\
a_2 & & & \vdots & \\
\vdots & & & \vdots & \\
a_i & \cdots\cdots & & a_i * a_j & \\
a_n & & & &
\end{array}$$

Ejemplo: Sea $A = \{0, 1\}$ podemos representar las operaciones binarias $\vee, \wedge$ mediante.

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

3.1. Propiedades de las Operaciones Binarias

P1: Una operación binaria * es conmutativa si $a * b = b * a \quad \forall a, b \in A$
Ejemplo: De los ejemplos anteriores.

1. $A = \mathbb{Z} \quad * = +$
   * es conmutativa: a+b = b+a.

2. $A = \mathbb{Z}^+ \qquad * = -$
   * no es conmutativa: $\begin{aligned} 3 * 5 &\neq 5 * 3 \\ -2 &\neq 2 \end{aligned}$

3. Sea $A = \{a, b, c, d\}$ Cuál de las siguientes representaciones es conmutativa?

a)

*	a	b	c	d
a	a	c	b	d
b	b	c	b	a
c	c	d	b	c
d	a	a	b	b

b)

*	a	b	c	d
a	a	c	b	d
b	c	d	b	a
c	b	b	a	c
d	d	a	c	b

Entonces:

- $\left.\begin{aligned} a*b &= c \\ b*a &= b \end{aligned}\right\}$ * no es conmutativa.
- $a*b = c = b*a$
  $c*d = c = d*c$
  $d*a = d = a*d$
  Si la matriz de resultados M verifica que $M = M^T$, la operación es conmutativa. * es conmutativa.

P2 : Una operación binaria * es asociativa si: $(a*b)*c = a*(b*c) \quad \forall a,b,c \in A$
Ejemplo:

1. Sea $A = \mathbb{Z}$ y * = +
   * es asociativa

2. Si $A = \mathbb{Z}$ y * = -
   * no es asociativa
   $$\begin{aligned} (3*5)*2 &\neq 3*(5*2) \\ -2*2 &\neq 3*3 \\ -4 &\neq 0 \end{aligned}$$

3.2. Semigrupo

Un semigrupo es un conjunto no vacío S, junto a una operación binaria asociativa * definida sobre S.
Notación: Denotamos al semigrupo por (S,*). Adicionalmente si * es conmutativa se dirá semigrupo conmutativo.
Ejemplo:

1. $(\mathbb{Z}, +)$ es semigrupo conmutativo.

2. $(\mathbb{Z}, -)$ no es semigrupo.

3. Para un conjunto S, $(P(S), \cup)$ es semigrupo conmutativa.

4. Sea S un conjunto no vacío, denotamos S^S como el conjunto de todas las funciones $f : S \to S$
   Sean $f, g \in S^S$ $definimos:$ $f * g = f \circ g$
   $(S^S, *)$ es un semigrupo. Sin embargo no es conmutativa, porque $f \circ g \neq g \circ f$

Definición A un elemento "e" en el semigrupo (S,*) se le llama identidad si:

$$a * e = a = e * a \qquad \forall a \in S$$

Ejemplo:

1. Si $(\mathbb{Z}, +)$ se tiene el elemento identidad, es e = 0.

2. En el semigrupo $(\mathbb{Z}^+, -)$ no hay elemento identidad.

Teorema: S un semigrupo $(S, *)$ tiene elemento identidad, éste debe ser único.
Prueba: Supongamos que e_1 y e_2 son elementos identidad. Como e_1 es identidad:
$e_2 * e_1 = e_2 = e_1 * e_2$
e_2 es identidad: $e_1 * e_2 = e_1 = e_2 * e_1$
$e_2 = e_1 * e_2 = e_1$, luego sólo hay un elemento identidad.

Definición: Un semigrupo es un monoide si tiene elemento identidad.
Ejemplo:

1. El semigrupo $(P(S), \cup)$ donde S es un conjunto no vacío, tiene elemento identidad $e = \phi$, pues $\phi \cup V = V$. El semigrupo es un monoide.

2. En el semigrupo $(S^S, \circ)$ se tiene elemento identidad $e = id_S$
   $id_S \circ f = f = f \circ id_S \qquad \forall f \in S^S$
   $(S^S, \circ)$ es un monoide.

Definición: Sea (S,*) un semigrupo y sea $T \subset S$. Si T es cerrado bajo la operación * [si $a, b \in T$ entonces $a * b \in T$] entonces $(T, *)$ es un subsemigrupo de $(S, *)$.

Definición: Sea (S,*) un semigrupo y sea $\phi \neq T \subset S$. Si T es cerrado bajo * y $e \in T$, entonces a (T,*) se le llama submonoide de (S,*).

3.3. Isomorfismos

Sean (S,*),(T,*') dos semigrupos. A la función $f : S \to T$, se le llama isomorfismo de (S,*) en (T,*') si f es inyectiva, sobreyectiva y además $f(a * b) = f(a) *' f(b) \qquad \forall a, b \in S$.
Notación: Si los semigrupos (S,*) y (T,*') son isomorfos se le denota por $S \simeq T$.

Método: Para demostrar que $S \simeq T$, se seguirá el procedimiento:

1. Defínase una función $f : S \to T$
2. Verifique que f sea inyectiva
3. Verifique que f sea sobreyectiva
4. Demuestre que $f(a * b) = f(a) *' f(b)$

Ejemplo: Sea T el conjunto de los pares. Demuéstrese que los semigrupos $(\mathbb{Z}, +)$ y $(T, +)$ son isomorfos.
Solución:

1. Definimos la función $f : S \to T$. $f(x) = 2x \qquad x \in \mathbb{Z}$
2. Sean x_1, x_2:

$$\begin{gathered} f(x_1) = f(x_2) \\ 2x_1 = 2x_2 \\ x_1 = x_2 \\ \text{f es inyectiva.} \end{gathered}$$

3. Sea $b \in T$ cualquiera, luego $b = 2a \quad a \in \mathbb{Z}$
   $f(a) = f(\frac{b}{2}) = \cancel{2}.\frac{b}{\cancel{2}} = b$
   Luego f es sobreyectiva.
4. P.P que: $f(a * b) = f(a) *' f(b)$
   $f(a * b) = 2(a * b) = 2a + 2b = f(a) + f(b)$
   $f(a * b) = f(a) *' f(b)$
   Luego $\mathbb{Z} \simeq T$.

Sean (S,*),(T,*') dos semigrupos finitos y sus operaciones las expresamos mediante tablas. Entonces $S \simeq T$ si es posible reordenar y etiquetar los elementos de S para que su tabla sea idéntica a la tabla de T.

Ejemplo: Sean $S = \{a, b, c\}$ y $T = \{x, y, z\}$ con tablas respectivas.

*	a b c
a	a b c
b	b c a
c	c a b

*'	x y z
x	z x y
y	x y z
z	y z x

Sea:

$$f(a) = y$$
$$f(b) = x$$
$$f(c) = z$$

	y x z
y	y x z
x	x z y
z	z y x

$\rightarrow$

	x y z
x	z x y
y	x y z
z	y z x

Luego $S \simeq T$.

Teorema: Sean los semigrupos (S,*) y (T,*') monoides y con identidades e y e'. Sea f un isomorfismo, entonces $e' = f(e)$.

Demostración: Sea $b \in T$. Al ser f un isomorfismo, es sobreyectiva, existe $a \in S / f(a) = b$.

Como e es identidad:

- $$\begin{aligned} a &= a * e \\ f(a) &= f(a * e) \\ b &= f(a) *' f(e) \\ b &= b *' f(e) \end{aligned}$$

- $$\begin{aligned} a &= e * a \\ f(a) &= f(e * a) \\ b &= f(e) *' f(a) \\ b &= f(e) *' b \end{aligned}$$

Del 1er y 2do item, $b *' f(e) = b = f(e) *' b$, luego $f(e)$ es identidad en T, $\therefore e' = f(e)$.

Definición: Sean (S,*) y (T,*') dos semigrupos. A una función $f : S \to T$. Se le llama un homomorfismo si:

$$f(a * b) = f(a) *' f(b) \qquad \forall a, b \in S$$

Capítulo 4

Grupo

Un grupo $(G, *)$ es un monoide tal que satisface las siguientes propiedades.

P1 Propiedad asociativa: $(a * b) * c = a * (b * c) \quad \forall a, b, c \in G$.

P2 Existencia de una identidad: Existe un elemento único $\in$ G tal que : $a * e = a = e * a \; a \in G$.

P3 Existencia del elemento inverso: $\forall$ G, existe el elemento inverso denotado por $a' \in G$ tal que: $a * a' = e = a' * a$.

En un grupo $(G, *)$, * operación binaria, G deberá ser cerrada bajo *, es decir. $a * b \in G \quad \forall a, b \in G$.

Definición: Si $(G, *)$ es conmutativo, es decir: $a * b = b * a$, se le llamará **Abeliana**.

Ejemplo: $(\mathbb{Z}, +)$

- P1 se cumple.
- P2 e=0.
- P3 $y \in \mathbb{Z} \quad \exists y \in \mathbb{Z} \qquad y * y' = e$

Ejemplo: $(\mathbb{Z}^{+}, \cdot)$

- P1 si.

- P2 si.
- P3 no.

Ejemplo: $(\mathbb{R} - \{0\}, \cdot)$

- P1 si.
- P2 e=1.
- P3 si.

Ejemplo: Sea G el conjunto de los reales sin el "cero" y sea $a * b = \frac{ab}{2}; a, b \in G$
Pruebe que $(G, *)$ es Grupo.
* es una operación binaria.

- $Dom = G \times G$; $\quad *G \times G \to G$.
- $a * b = \frac{ab}{2}$ es único $\forall a, b \in G$.

1. Veamos que $(G, *)$ es asociativa.
   $(a * b) * c = (\frac{ab}{2}) * c = \frac{abc}{4}$
   $a * (b * c) = a * (\frac{bc}{2}) = \frac{abc}{4}$
   $(a * b) * c = a * (b * c)$
2. Afirmamos que $e = 2$ es la identidad.
   $a * e = \frac{a(2)}{2} = a$
   $e * a = \frac{2(a)}{2} = a$
3. Afirmamos que $a' = \frac{4}{a} \in G$ es la inversa de a.
   $a * a' = \frac{a(\frac{4}{a})}{2} = 2 = e$
   $a' * a = \frac{(\frac{4}{a})(a)}{2} = 2 = e$
   $\therefore$ G es un grupo.

Teorema 1: Sea G n grupo . Cada elemento $a \in G$ tiene un inverso único en G.

Teorema 2: Sea G un grupo y sean a,b,c elementos en G. Entonces.

- $a * b = a * c \Rightarrow b = c$ Propiedad cancelativa izquierda.

- $b * a = c * a \Rightarrow b = c$ Propiedad cancelativa derecha.

Teorema 3: Sea $(G, *)$ un grupo y $a, b \in G$. Entonces.

- $(a')' = a$
- $(a * b)' = b' * a'$

Si G es un conjunto finito, su operación binaria * se puede obtener mediante una tabla.
Sea $G = \{a_1, a_2, ...a_n\}$. La tabla de multiplicación bajo * cumple:

- La fila etiquetada por e deberá contener a: $a_1, a_2, \cdots a_n$
- La columna etiquetada por e debe contener a: $\begin{matrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{matrix}$
- Cada elemento "b" en el grupo deberá aparecer exactamente una vez en cada fila y en cada columna.

Definición: Sea $(G, *)$ un grupo que tiene un número finito de elementos.

- Se dice que G es finito.
- El orden de G es el número de elementos y se denota por $|G|$.

Ejemplo:

- Si G es unitario, $G = \{e\}$, $e * e = e$.
- Si $|G| = 2, G = \{e, a\}$.Su tabla es:

	e	a
e	e	a
a	a	e

$a' = a$

$e' = e$

- Si $|G| = 3, G = \{e, a, b\}$. Su tabla es:

	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

$a * b = e \quad a' = b$

$b * a = e \quad b' = a$

$e * e = e \quad e' = e$

Ejemplo: Sea $B = \{0, 1\}$ y $* = +$

+	0	1
0	0	1
1	1	0

$$0 * 0 = 0 \quad 0' = 0$$
$$1 * 1 = 0 \quad 1' = 1$$

4.1. Cadenas

Un tipo de problema a revisar es el de decisión. Un problema de decidibilidad es una función que produce como resultado uno de dos valores: ”si” o ”no”.

Definición: Un símbolo es un objeto indivisible. Utilizaremos como símbolos las letras iniciales del alfabeto y los dígitos.

Definición: Un alfabeto es un conjunto finito o infinito de símbolos.

Notación: $\sum, V, \Gamma$

Ejemplo:

1. Alfabeto Binario $\Sigma = \{0, 1\}$

2. Alfabeto de letras mayúsculas $\Sigma = \{A, B, C, ..., Z\}$

Definición: Una cadena ó palabra es una secuencia finita de símbolos, tomados a partir de un alfabeto.

Notación: $W(x, y, ...)$.
Sea $W = a_1, a_2, ...a_k$ con $a_i \in \Sigma$, $i = 1, ...k$
Longitud:
$|W| = k \Leftrightarrow W = a_1...a_k$
Sea $\Sigma = \{0, 1\}$ y $W = 001101 \Rightarrow |W| = 6$

4.2. Cadena Vacía

Es la cadena de longitud "cero".
Notación: ε, λ. **Notación:** Denotamos por W_k al conjunto de todas las cadenas de longitud k sobre un alfabeto Σ.

$$W_k = \{W/|W| = k \text{ y } W = a_1...a_k; a_i \in \Sigma\}$$

$W_0 = \{\varepsilon\}$ y lo denotamos por ε
Al conjunto de todas las cadenas finitas posibles.
Sobre Σ, lo denotamos por $W = \bigcup_{k=0}^{\infty} W_k$

4.3. Operación con Cadenas

4.3.1. Concatenación

El producto de cadenas es una operación binaria en W.
$m : W \times W \to W$
Si $x = a_1, ...a_i; y = b_1...b_j$ entonces $m(x, y) = a_1...a_i, b_1...b_j$. Denotamos $m(x, y)$ por $x.y$ ó simplemente xy.

Definición: Sea $W = xz$. Entonces:
"x" es un prefijo de W, es un prefijo propio si $z \neq \varepsilon$
"z" es un sufijo de W, es un sufijo propio si $x \neq \varepsilon$

4.3.2. Propiedades de la Concatenación

Sea $W = xyz$

1. **Cerradura:** $\forall x, y \in W \quad xy \in W$
2. **Asociatividad:** $(x.y).z = x.(y.z) \quad x, y, z \in W$
3. **Identidad:** $w.\varepsilon = w = \varepsilon.w \quad \forall w \in W$
4. **Longitud:** Para $wx \Rightarrow \quad |wx| = |w| + |x| \quad w, x \in W$

La concatenación no necesariamente es conmutativa. $wx \neq xw$ En general.
Notación: $w^k = \underbrace{ww...w}_{k\ veces}$

Si $w = a_1...a_k$ entonces $w^R = a_k...a_1$

4.4. Lenguajes

Un lenguaje L es un conjunto de cadenas definidas sobre Σ.
$L \subset W$

Ejemplo:

1. Sea $\Sigma = \{a_0, a_1\}$. Entonces:
   $L = \{a_0 a_1 ... a_{i_k} / a_{i_j} \in \Sigma\}$ es un lenguaje.
2. Sea $\Sigma = \{a\}$
   $L = \{a^k / k > 0\}$ es el lenguaje de todas las cadenas de "a" de longitud finita.
3. Sea $\Sigma = \{0, 1\}$
   $L = \{ww^R / w = a_1...a_k \quad a_i \in \Sigma\}$ L es el lenguaje de los palíndromos formados con ceros y unos.

4.5. Operaciones con Lenguajes

Sean L_1, L_2 dos lenguajes. Definimos las siguientes operaciones:

$$\begin{aligned}
L_1 \cup L_2 &= \{w/w \in L_1 \vee w \in L_2\} \\
L_1 \cap L_2 &= \{w/w \in L_1 \wedge w \in L_2\} \\
I &= \{w/w \notin L \wedge w = a_1...a_k; \quad a_i \in \Sigma\} \\
L_1.L_2 &= \{vw/v \in L_1 \wedge w \in L_2\}
\end{aligned}$$

Propiedad: La cardinalidad de la concatenación de dos lenguajes es menor o igual que el producto de las cardinalidades de cada uno de ellos.

Ejemplo: Sean $L_1 = \{a, ab\} \quad L_2 = \{c, bc\}$ sobre $\Sigma = \{a, b, c\}$

$$\begin{aligned}
L_1 L_2 &= \{ac, abc, ..., abbc\} \\
|L_1 L_2| &\leq |L_1||L_2| \\
3 &\leq 2 \times 2
\end{aligned}$$

Propiedad (Identidad): El conjunto $L_\varepsilon = \{\varepsilon\}$ es la identidad en la concatenación de lenguajes.

$L.L_\varepsilon = \{w.x/w \in L, x \in L_\varepsilon\} = \{w/w \in L\}$ como $w.\varepsilon = w = \varepsilon.w$

$L_\varepsilon.L = L.L_\varepsilon = L$

Elemento Nulo: Es el conjunto vacío ϕ.

Propiedad Asociativa $(L_1 L_2)L_3 = L_1(L_2 L_3)$

Propiedad Distributiva $L_1(L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$

¿Es la concatenación de lenguajes distributiva respecto a la intersección?

Definición: Sea Σ un alfabeto y $L \subseteq W$ un lenguaje sobre Σ. Definimos las siguientes potencias de L:

$$\begin{aligned}
L^0 &= \{\varepsilon\} \\
L^1 &= L \\
L^k &= L^{k-1}.L
\end{aligned}$$

Definición: La cerradura del lenguaje L es $L^* = \bigcup_{k=0}^{\infty} L^k$. Al operador * se le llama "Estrella de Kleene" o "Cerradura de Kleene".

Ejemplo: Sea $\Sigma = \{a, b, c\}$ y $L = \{a, ab, ac\}$. Halle: L^2, L^3.

$L^2 = L.L = \{aa, aab, aac, aba, abab, abac, aca, acab, acac\}$
$L^3 = L^2.L = \{aaa, aaab, aaac, aaba, aabab, aabac,$
$aaca, aacab, aacac, ...acaca, acacab, acacac\}$

Ejemplo: Sea $L = \{a\} \quad \Sigma = \{a\}$
$L^* = \{\varepsilon, a, aa, aaa, ...\}$

Ejemplo: Sea $L = \{0, 1, 00, 01\} \quad \Sigma = \{0, 1\}$
$L^* = \{\varepsilon, 0, 1, 00, 01, 00, 01, 000, 001, ...\}$

Ejemplo: Sea L un lenguaje. $L^R = \{w^R / w \in L\}$
Sea ahora $L = \{ww^R / w = a_1 ... a_k; a_i \in \Sigma\}$
a este lenguaje se le conoce como "Lenguaje espejeado".

Capítulo 5

Lenguajes Formales

En oposición al lenguaje natural, un lenguaje formal es tal que:

1. Tiene una sintaxis bien definida. De tal modo que dada una sentencia, es posible saber si pertenece o no al lenguaje.

2. Tiene una semántica precisa. No es posible encontrar sentencias ambiguas o sin significado.

Ejemplo: C, Java, HTML.

5.1. Lenguajes Regulares

Son un tipo de Lenguaje Formal. Reciben este nombre porque sus palabras contienen regularidades o repeticiones de los mismos componentes.

Ejemplo:

- $L_1 = \{cd, cdcd, cdcdcd, ...\}$

  Las cadenas contienen las subcadenas un número dado de veces.

- $L_2 = \{abc, cc, abab, abccc, ababc, ...\}$

  La regularidad consiste encadenas que empiezan con repeticiones de ”ab” seguidas de repeticiones de c.

Se considerará a los lenguajes finitos también regulares.
$L_3 = \{hoy, es, sabado\}$ L_3 es regular.

5.2. Aplicación

Se pueden utilizar para especificar la generación de analizadores léxicos.

5.2.1. Analizadores Léxicos

Es un programa que recibe como entrada el código fuente de otro programa y produce como salida Lexemas.

Las palabras están compuestas por lexemas y morfemas.

Lexema

Es la raíz o parte de la palabra que no varía.

Ejemplo: **deport**-e, **deport**-ivo, **deport**-ista.

Morfema

Es la parte que se le añade al lexema para completar su significado y así generar nuevas palabras.

Ejemplo: moder-**o**, modern-**as**, modern-**ísimo**.

5.2.2. Definición Recursiva

Sea un alfabeto Σ, Un lenguaje $L \subseteq \Sigma^*$ regular se define recursivamente como sigue.

- ϕ, el lenguaje vacío, es un lenguaje regular.
- $\{\varepsilon\}$ es un lenguaje regular.
- $\{a\}$ es un lenguaje regular $\forall a \in \Sigma$.
- Si L_1 y L_2 son lenguaje regulares, entonces $L_1 \cup L_2, L_1 \cdot L_2, L_1^*$ son lenguajes regulares.
- Ningún otro lenguaje sobre Σ (definida en las 4 anteriores) es un lenguaje regular.

Ejemplo: Sea $\Sigma = \{a, b\}$. Encuentre 6 lenguajes regulares.

$L_1 \;= \phi$

$L_2 \;= \{a\} \quad L_4\{b\}$

$L_5 \;= \{a, b\} \quad L_6 = \{ab\} \quad L_7 = \{a, ab, b\}$

Ejemplo: Dado $\Sigma = \{a, b\}$. Describa 3 L.R. infinitas y representarlos de forma abreviada.

$L_1 = \{w \in \Sigma^*/w$ empieza con $b\} = \{b\}\Sigma^*$

$L_2 = \{w \in \Sigma^*/w$ contiene exactamente una $a\} = \{b\}^* \cdot \{a\} \cdot \{b\}^*$

$L_3 = \{w \in \Sigma^*/w$ contiene a la subcadena $ba\}$

$= \{ba, aaba, bbabba, abbaaab, ...\}$

$= \Sigma^*\{ba\}\Sigma^*$

$L_4 = \{w \in \Sigma^*/w$ contiene exactamente dos símbolos $b\} = \{a\}^*\{b\}\{a\}^*\{b\}\{a\}^*$

$L_5 = \{a^k/k \geq 0, a \in \Sigma\}$

5.3. Características

Un L.R tiene estas características.

1. Puede ser escrito mediante una "Expresión Regular".
2. Puede ser reconocido mediante un "Autómata Finito".
3. Puede ser generado mediante una "Gramática Regular".

5.4. Expresiones Regulares (ER)

Una ER es una secuencia de caracteres que forma un patrón de búsqueda. Se usa para especificar un conjunto de cadenas requeridas para un propósito particular.

Aplicaciones:

- Búsqueda de patrones de cadenas de caracteres.
- Operaciones de sustitución.

Ejemplo: Dadas las cadenas Handel, Händel, Haendel. Describa el patrón que los especifique.

H$\{a|$ä$|ae\}$ndel

5.4.1. Definición Recursiva

Dado un alfabeto Σ, una ER se define recursivamente como sigue:

- El símbolo ϕ es una ER.
- ε es una ER.
- $a \in \Sigma$, es una ER.
- Si p y q son ER, entonces p.q, p$\cup$q, p* son ER.

Se cumple que toda ER sobre Σ describe a un LR sobre Σ.

Abreviatura:

Para los lenguajes siguientes, a continuación se dan las abreviaturas correspondientes.

	Exp. Regulares
$\{v\} \cup \{w\}$	$v + w$
$\{v.w\}$	vw
$\{w\}^*$	w^*
$\{w\}^+$	w^+

5.4.2. Nivel de Prioridad

Operador	Prioridad
$*$	1
$\cdot$	2
$+$	3

Ejemplo: Reducir la expresión. $E = (\{a\}^*\{b\}) \cup \{c\}$.

Queda: $a^*b + c$, mediante las ER.

Definición: Dada una expresión regular α, denotado por $L(\alpha)$ como:

- $L(\phi) = \phi$
- $L(\varepsilon) = \{\varepsilon\} = L_\varepsilon$
- $L(a) = \{a\}$, donde $a \in \Sigma$
- $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$, donde α, β son ER.

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$, donde α, β son ER.
- $L(\alpha^*) = (L(\alpha))^*$

5.5. Equivalencia de ER

Definición: Dadas las ER α, β definidas sobre Σ, diremos que son equivalentes si ambas generan el mismo lenguaje.

$L(\alpha) = L(\beta)$

Notación: Si α y β son equivalentes se denotará $\alpha = \beta$.

5.6. Propiedades de las ER

Sea Σ un alfabeto α, β y γ ER. Se cumple:

1. $\alpha + \beta = \beta + \alpha$
2. $\alpha + \phi = \alpha = \phi + \alpha$
3. $\alpha + \alpha = \alpha$
4. $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
5. $\varepsilon\alpha = \alpha = \alpha\varepsilon$
6. $\phi\alpha = \alpha = \alpha\phi$
7. $(\alpha\beta)\gamma = \alpha(\beta\gamma)$
8. $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$
   $(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$
9. $\phi^* = \varepsilon$

Pruebas

P1:Debemos probar que $\alpha + \beta \subseteq \beta + \alpha; \quad \beta + \alpha \subseteq \alpha + \beta$.

PPQ $\alpha + \beta \subseteq \beta + \alpha$

$$\begin{aligned} \text{Sea } w \in \alpha + \beta &\Rightarrow w \in \alpha \vee w \in \beta \\ &\Rightarrow w \in \beta \vee w \in \alpha \\ &\Rightarrow w \in \beta + \alpha \end{aligned}$$

PPQ $\beta + \alpha \subseteq \alpha + \beta$

$$\begin{aligned} \text{Sea } z \in \beta + \alpha &\Rightarrow z \in \beta \vee z \in \alpha \\ &\Rightarrow z \in \alpha \vee z \in \beta \\ &\Rightarrow z \in \alpha + \beta \end{aligned}$$

$$\therefore \alpha + \beta = \beta + \alpha$$

P9: Tenemos que $\alpha = \beta$ si $L(\alpha) = L(\beta)$. Bastará probar que $L(\phi^*) = L(\varepsilon)$.

$L(\phi^*) = (L(\phi))^* = \phi^*$

$$\begin{aligned} \phi^* = \textstyle\bigcup_{k=0}^{\infty} \phi^k &= \phi^0 \cup \phi^1 \cup \phi^2 ... \\ &= \phi^0 \\ &= \{\varepsilon\} \\ &= L(\varepsilon) \end{aligned}$$

$$\therefore L(\phi^*) = L(\varepsilon)$$

OBS:

$$\begin{aligned} L(\phi) &= \phi \\ L^* &= \textstyle\bigcup_{k=0}^{\infty} L^k \\ L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^{k+1} &= L^k L \\ L(\varepsilon) &= \{\varepsilon\} \end{aligned}$$

Ejemplo: Sea $\Sigma = \{0, 1\}$ y la ER $\alpha = 0^*10^*$. Describa a las cadenas de $L(\alpha)$, usando las propiedades.

Solución:

$$\begin{aligned} L(\alpha) = L(0^*10^*) &= L(0^*)L(1)L(0^*) \\ &= (L(0))^*L(1)(L(0))^* \\ &= \{0\}^*\{1\}\{0\}^* \\ &= \{0^j,1,0^k / j \geq 0 \quad k \geq 0\} \end{aligned}$$

Capítulo 6

Derivada de una ER

Sea α una ER sobre cierto alfabeto Σ y sea $a \in \Sigma$. La derivada de la ER α respecto de **a** se denota por $D_a(\alpha)$; es una ER que describe al Lenguaje:

$$L(D_a(\alpha)) = \{w \in \Sigma^* / aw \in L(\alpha)\}$$

6.1. Reglas de Derivación

Se aplica de forma recursiva las siguiente reglas de derivación.

1. $D_a(\phi) = \phi$

2. $D_a(\varepsilon) = \phi$

3. $D_a(a) = \varepsilon \quad D_a(b) = \phi \quad \forall b \neq a, b \in \Sigma$

4. $D_a(\alpha + \beta) = D_a(\alpha) + D_a(\beta)$

5. $D_a(\alpha \cdot \beta) = D_a(\alpha).\beta + \delta(\alpha).D_a(\beta)$

   $$\delta(\alpha) = \begin{cases} \varepsilon \text{ si } \varepsilon \in L(\alpha) \\ \phi \text{ si } \varepsilon \notin L(\alpha) \end{cases}$$

6. $D_a(\alpha^*) = D_a(\alpha).\alpha^*$

Ejemplo: Sea $\Sigma = \{a, b\}$ y $\alpha = a^*ab$. Derive α respecto a los símbolos a y b.

$$D_a(\alpha) = D_a(\underset{\alpha}{\underset{\downarrow}{a^*}}ab) = D_a(a^*).ab + \delta(a^*)D_a(ab)$$

$$= D_a(a).a^*.ab + \delta(a^*)(D_a(a).b + \delta(a).D_a(b))$$

$$\underset{\varepsilon \notin L(a)}{\underset{\downarrow}{L(a) = \{a\}}} \qquad \underset{\varepsilon \in L(a^*)}{\underset{\downarrow}{L(a^*) = \{\varepsilon, a, aa, aaa, ...\}}}$$

$$= \varepsilon.a^*.ab + \varepsilon(\varepsilon b + \phi.\phi)$$

$$= a^*ab + b$$

Nota: Podemos derivar una ER α respecto de una cadena de símbolos x.

$$L(D_x(\alpha)) = \{w \in \Sigma^* / x.w \in L(\alpha)\}$$

6.2. Ecuaciones de ER

Definimos una ecuación de ER con variables $x_1, x_2, x_3, ...x_n$ a una ecuación del tipo:

$$x_i = \alpha_{i0} + \alpha_{i1}x_1 + ...\alpha_{in}x_n$$

Donde cada α_{ij} es una ER; α_{i0} es el término independiente. Una solución para x_i es una ER.

Definición: A una ecuación de la forma $x = \alpha x + \beta$ donde α, β son ER, se le conoce como ecuación fundamental de ER.

Lema de Arden: Se prueba que $x = \alpha^*.\beta$ es una solución para la ecuación fundamental. Se cumple que:

- Es única si $\varepsilon \in L(\alpha)$.

- Tiene infinitas soluciones, de la forma $\alpha^*(\beta + \gamma)$ (γ es una ER) en otro caso.

6.3. Algoritmo de Solución de Sistemas de ecuaciones de ER

Entrada: n ecuaciones de ER con variables $x_1, ..., x_n$.

Salida: Una solución para cada x_i.

Algoritmo 1: Funcion1()

$i \leftarrow n$
while $i \geq 2$ **do**
    Expresar ecuación para x_i como: $x_i = \alpha x_i + R$
    Obtener $x_i \leftarrow \alpha^*.R$
    for $j = i - 1$ **to** *1* **do**
        Sustituir en la ecuación para x_j la variable x_i por $\alpha^* R$
    $i \leftarrow i - 1$

Algoritmo 2: Funcion2()

$i \leftarrow 1$
while $i \leq n$ **do**
    Obtener la solución $x_i \leftarrow \alpha^* \beta$
    for $j = i + 1$ **to** n **do**
        Sustituir en la ecuación para x_j la variable x_i por $\alpha^*.\beta$
    $i \leftarrow i + 1$

Ejemplo: Sea $\Sigma = \{0, 1\}$. Resolver el sistema de ecuaciones de ER sobre Σ.

$$\begin{array}{llllll} x_1 & = \varepsilon & +1.x_1 & +0.x_2 & \\ x_2 & = & & 1.x_2 & +0.x_3 \\ x_3 & = & 0.x_1 & & +1.x_3 \end{array}$$

Diagonalizando, usando el algoritmo:

$$x_3 = 1.x_3 + \overbrace{0.x_1}^{R}$$

$$x_3 = 1^* 0 x_1$$

$$x_2 = 1.x_2 + 0x_3 = 1x_2 + 01^*0x_1$$
$$x_2 = 1^*01^*0x_1$$

Luego:

$$x_1 = 1x_1 + \varepsilon + 0x_2 = 1x_1 + \varepsilon + 01^*01^*0x_1$$
$$x_1 = (1 + 01^*01^*0)x_1 + \varepsilon$$
$$x_1 = (1 + 01^*01^*0)^*$$
$$x_2 = 1^*01^*0(1 + 01^*01^*0)^*$$
$$x_3 = 1^*0(1 + 01^*01^*0)^*$$

6.4. Máquina de Estados Finitos

	t_0	t_1	t_2	t_3	t_4
Estados	s_0	$s_1(5c)$	$t_2(10c)$	$s_3(20c)$	$\cancel{s}_4 s_0$
Entradas	$5c$	$5c$	$10c$	D	
Salidas	$-$	$-$	$-$	*Cerveza*	

	t_0	t_1
Estados	s_0	$s_1(20c)$
Entrada	$25c$	N
Salida	$5c$	*Gaseosa*

Se reconocen:

- Un conjunto de estados: S
- Un alfabeto de entrada: I
- Un alfabeto de salida: O
- Un estado inicial: s^*

Se definen 2 funciones:

- Función de estado siguiente: $f : S \times I \to S$
- Función de salida $g : S \times I \to O$

Una máquina de estados finitos **MEF** se representa M, es una séxtupla formada por:

$$M = (S, I, O, f, g, s^*)$$

Capítulo 7

Máquinas de Estado Finito(MEF)

Una MEF M es una séxtupla $M = (S, I, O, f, g, s^*)$ donde:

- S: Segmento de estado, $s \neq \phi$.
- I: Alfabeto de entrada.
- O: Alfabeto de salida.
- s^*: Estado inicial , $s^* \in S$.
- f: Función de estado siguiente. $f : S \times I \to S$.
- g: Función de salida. $g : S \times I \to O$.

$$f(\overset{\text{estado actual}}{\overset{\uparrow}{s}}, \underset{\text{simbolo}}{\underset{\downarrow}{x}}) = \overset{\text{siguiente estado}}{\overset{\uparrow}{s_2}}$$

7.0.1. Tipo de Representación

Existen 2 tipos de representaciones que son las siguientes:

1. **Tablas de Estado:** Conocidas como tablas de transmisión. Podemos representar a las MEF M por medio de una tabla que liste.

   $f(s, x); g(s, x) \qquad \forall s \in S, \forall x \in I$

Ejemplo: Sea la MEF M donde:

$S = \{s_0, s_1, s_2\}$

$I = \{0, 1\} = Q$

f y g están dados en la siguiente tabla:

S/I	f		g	
	0	1	0	1
s_0	s_0	s_1	0	0
s_1	s_2	s_1	0	0
s_2	s_0	s_1	0	1

Se pide:

- Obtener $f(s_1, 1)$ y $g(s_1, 1)$.
- Si tenemos como entrada la cadena $u = 1010$, determine la cadena de salida w.

  Solución:

- $f(s_1, 1) = s_1 \qquad g(s_1, 1) = 0$
- Tenemos: $s^* = s_0$, $u = 1010$

Estado	s_0	s_1	s_2	s_1	s_2
Entrada	1	0	1	0	
Salida	0	0	1	0	

  $g(s_0, 1) = 0 \quad f(s_0, 1) = s_1 \quad f(s_2, 1) = s_1$

  La cadena de salida es $w = 0010$.

2. **Diagramas de Estados:**

- Cada estado interno de S es representado con un nodo.
- Dados los estados s_i, s_j

  si: $\begin{array}{ll} f(s_i, x) = s_j & \text{para } x \in I \\ g(s_i, x) = y & y \in O \end{array}$

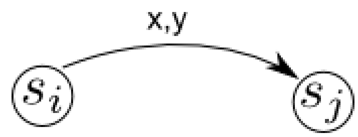

Figura 7.1: Representación Diagrama de Estados

Los representamos mediante una arista dirigida desde s_i a s_j y rotulando el arco con la entrada x y la salida y

Usando el ejemplo anterior, dibujamos el diagrama de estados para M.

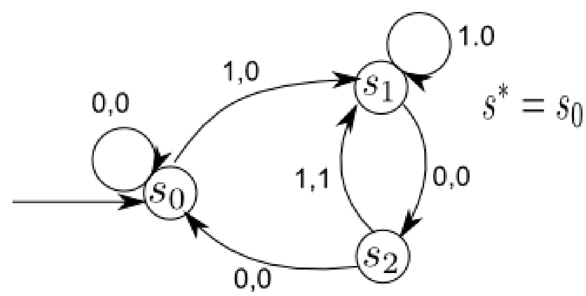

Figura 7.2: Diagrama de Estado de M

7.1. Sumador Binario

Sean las cadenas x, y

$x = x_5x_4x_3x_2x_1 = 00111$

$y = x_5y_4y_3y_2y_1 = 01101$

Un sumador binario en serie es una MEF que nos sirve para hallar x+y.

Representación

Para la adición $z = x + y$ tenemos:

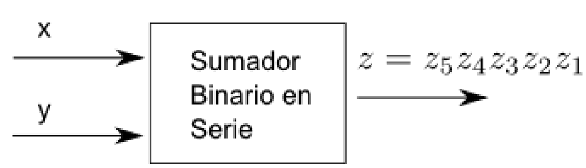

Figura 7.3: Sumador Binario

$$\begin{array}{rccccc} x = & 0 & 0 & 1 & 1 & 1 \\ y = & 0 & 1 & 1 & 0 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 \end{array}$$

Podemos modelar este sumador binario mediante una MEF M, en la cual.

$S = \{s_0, s_1\}$ — s_0 indica un caracter de 0; s_1 indica un caracter de 1

$I = \{00, 01, 10, 11\}$ $\quad s^* = s_0$

$O = \{0, 1\}$

Las funciones f y g quedan definidas en la tabla.

S/I	f				g			
	00	01	10	11	00	01	10	11
s_0	s_0	s_0	s_0	s_1	0	1	1	0
s_1	s_0	s_1	s_1	s_1	1	0	0	1

$$f(s_0, 11) = s_1 \qquad g(s_0, 11) = 0$$

El diagrama de estados para M será (Figura 7.4):

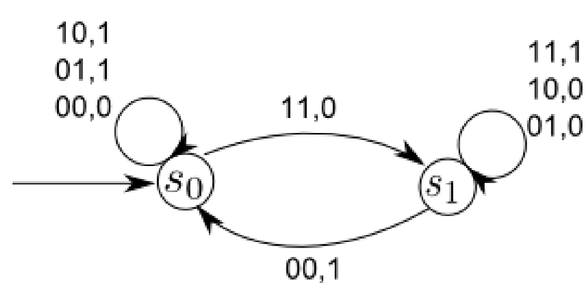

Figura 7.4: Diagrama de Estados de M

Ejercicio: Sea $M = (S, I, O, f, g, s^*)$ donde:

$S = \{s_0, s_1, s_2, s_3\}$

$I = \{a, b, c\}$

$O = \{0, 1\}$

S/I	f			g		
	a	b	c	a	b	c
s_0	s_0	s_3	s_2	0	1	1
s_1	s_1	s_1	s_3	0	0	1
s_2	s_1	s_1	s_3	1	1	0
s_3	s_2	s_3	s_0	1	0	1

Si $s^* = s_0$

- Obtener la cadena de salida si la entrada es $u = abbccc$.
- Dibuje el diagrama de estados.

Solución:

$Estado$	s_0	s_0	s_3	s_3	s_0	s_2
$Entrada$	a	b	b	c	c	c
$Salida$	0	1	0	1	1	1

$$u = abbccc$$

$$v = 101010$$

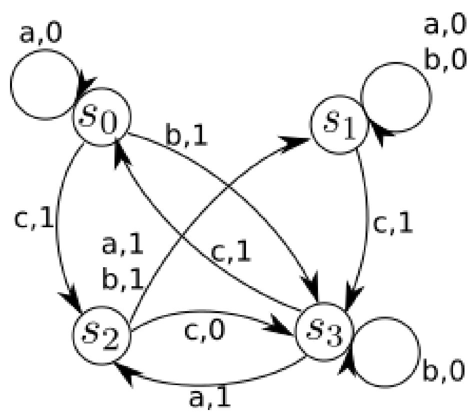

Figura 7.5: Diagrama de Estados

7.2. Extensión de las funciones f y g

Para una MEF $M = (S, I, O, f, g, s^*)$, la entrada puede ser realizada con un elemento de I^*, con la salida O^*. Extendemos los dominios de f y g de $S \times I$ a $S \times I^*$.

Para g, ampliamos el codominio de O a O^*. Con estas extensiones, si $x_1, x_2, x_3 ... x_k \in I^*$ entonces empezando en cualquier estado $s_1 \in S$ tenemos:

$$\begin{aligned}
f(s_1, x_1) &= s_2 \\
f(s_1, x_1x_2) &= f(f(s_1, s_1), x_2) = f(s_2, x_2) \\
f(s_1, x_1x_2x_3) &= f(f(\underbrace{f(s_1, x_1)}_{s_2}, x_2), x_3) \\
&= f(\underbrace{f(s_2, x_2)}_{s_3}, x_3) = f(s_3, x_3) \\
&\vdots \\
f(s_1, x_1 ... x_k) &= f(s_k, x_k) = s_{k+1}
\end{aligned}$$

función de estado siguiente extendida.

$$\begin{aligned}
g(s_1, x_1) &= y_1 \\
g(s_1, x_1x_2) &= g(s_1, x_1).g(f(s_1, x_1), x_2) \\
&= y_1.g(s_2, x_2) = y_1.y_2 \\
g(s_1, x_1x_2x_3) &= g(s_1, x_1).g(s_2, x_2).g(s_3, x_3) \\
&= y_1.y_2.y_3 \\
&\vdots \\
g(s_1, x_1 ... x_k) &= g(s_1, x_1)g(s_2, x_2)...g(s_k, x_k) \\
&= y_1y_2...y_k
\end{aligned}$$

También $f(s_i, \varepsilon) = s_i \qquad \forall s_i \in S$

Capítulo 8

MEF parte II

Definición: Sea $M = \{S, I, D, f, g, s^*\}$ una MEF.

1. Para $s_i, s_j \in S$ se dice que s_j es **alcanzable** desde s_i si $s_i = s_j$ o si hay cadenas de entrada $w \in I^+$ tal que $f(s_i, w) = s$.

2. Un estado $s \in S$ se dice **transitorio** si $f(s, w) = s$ para $w \in I^* \Rightarrow w = \varepsilon$, es decir, no hay $w \in I^+ / f(s, w) = s$.

   Ejemplo: Sea la MEF con diagrama de transición(Figura 8.1).

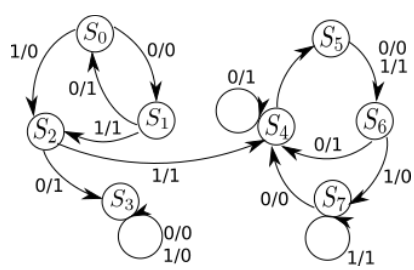

Figura 8.1: Diagrama de Transición

- ¿ Desde qué estados es alcanzable s_3?

  s_0, s_1, s_2

- ¿ Desde qué estados no es alcanzable s_3?

  s_4, s_5, s_6, s_7

- Encuentre un estado transitorio en M.

  s_2 es un estado transitorio.

3. Un estado $s \in S$ es un estado **sumidero** si: $f(s, w) = s \qquad \forall w \in I^*$

4. Sea $S_1 \in S, \quad I_1 \subseteq I$. Si $f_1 = f|_{S_1 \times I_1}$

   $f_1 : S_1 \times I_1 \to S$ tiene su rango dentro de S_1 entonces $g_1 = g|_{S_1 \times I_1}$

   $g_1 : S_1 \times I_1 \to O$, entonces $M_1 = (S_1, I_1, O_1, f_1, g_1, s^*)$ se llama **submáquina** de la MEF M.

5. Una máquina se denomina **fuertemente conexa** si para cualquier estado s_i, s_j es accesible desde s_i.

   Ejemplo: Usando la MEF anterior.

   - Identifique un estado sumidero.

     s_3

   - Encuentre una submáquina.

     - Sea $s_1 = \{s_0, s_1, s_2\} \subseteq S$ y $I_1 = \{0, 1\} \subseteq I$, f_1 tiene algunos estados que caen fuera de S_1. (✗)
     - Sea $S_1 = \{s_4, s_5, s_6, s_7\} \subseteq S$ y $I = \{0, 1\}$ luego $M_1 = (S_1, I_1, O, f_1, g_1, s^*)$ es una submáquina de M. (✓)
     - Encuentre una máquina fuertemente conexa.

       M no es fuertemente conexa, sin embargo, M_1 si es fuertemente conexa.

Definición: Para una MEF $M = \{S, I, O, f, g, s^*)$. Sean $s_i, s_j \in S$ dos estados distintos. Una cadena de entrada $w \in I^+$ es llamada **secuencia de transferencia o transición** desde s_i a s_j si:

- $f(s_i, w) = s_j$

- $v \in I^+$ con $f(s_i, v) = s_j \Rightarrow |v| \geq |w|$

Ejemplo: Obtener una secuencia de transferencia desde el estado S_0 hacia el estado S_2 para la MEF M definida sobre $I = \{0, 1\} = O$

$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$

M está representada por la siguiente tabla.

	f		g	
	0	1	0	1
s_0	s_6	s_1	0	1
s_1	s_5	s_0	0	1
s_2	s_1	s_2	0	1
s_3	s_4	s_0	0	1
s_4	s_2	s_1	0	1
s_5	s_3	s_5	1	1
s_6	s_3	s_6	1	1

Representamos las transiciones de estado como un árbol(Figura 8.2).

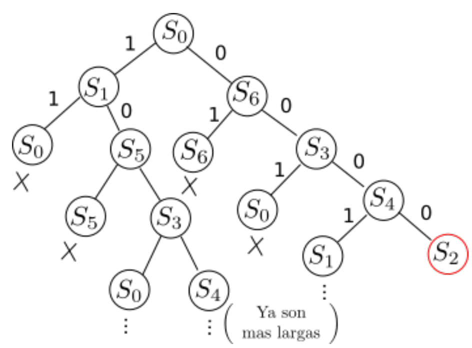

Figura 8.2: Representación como árbol

La cadena de transferencia es $w = 0000$ con $f(s_0, w) = s_2$ y $g(s_0, w) = 0100$. Otra cadena es $v = 1000$ pero $|v| > |w|$.

8.1. Tipos de MEF

Para el modelamiento de las calculadoras se ha desarrollado muchos tipos de MEF.

1. **Máquina de Moore** Una máquina de Moore $M_0 = (S, I, O, f, g, s^*)$ es tal que :

$$\begin{aligned} S &: \text{Conjunto de estados} \\ I &: \text{Alfabeto de entrada} \\ O &: \text{Alfabeto de salida} \\ f &: \text{Función de estado siguiente.} f : S \times I \to S \\ g &: \text{Función de salida, } g : S \to O \end{aligned}$$

Representación

Ejemplo: A continuación se presenta la tabla de transición de una máquina de Moore.

	f		g
S	a	b	
s_0	s_3	s_2	0
s_1	s_1	s_0	0
s_2	s_2	s_3	1
s_3	s_0	s_1	0

Se pide:

- Dibuje su diagrama de estado.
- Obtener la cadena de salida para $w = bababbb$

Solución:

s_0	s_2	s_2	s_3	s_0	s_2	s_3	s_1
b	a	b	a	b	b	b	
0	1	1	0	0	1	0	

La cadena de salida es $s = 0110010$

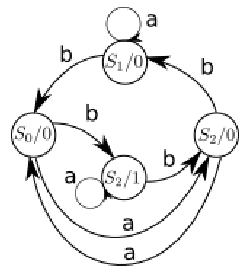

Figura 8.3: Diagrama de Transición

2. **Máquina de Mealy:** Fueron estudiados por G.H. Mealy en 1955. En este tipo de máquina las salidas corresponden a transiciones entre estados. Aquí:

$$f : S \times I \to S$$
$$g : S \times I \to O$$

Ejemplo: Dibuje una máquina de Mealy que imprime el complemento de una cadena de bits de entrada.

Solución:

$S = \{s_0\}$
$I = \{0, 1\} = O$

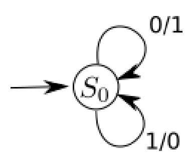

Figura 8.4: Máquina de Mealy

	f		g	
	0	1	0	1
s_0	s_0	s_0	1	0

Teorema: Si M_0 es una máquina de Moore, entonces existe una máquina de Mealy equivalente(Figura 8.5).

Método:

1. Considere cualquier estado q_i en M_0.

2. Suponga que M_0 imprime el carácter t al ingresar q_i, luego se tiene el rótulo q_i/t en el estado q_i.

3. Suponga que hay n arcos de entrada q_i con etiquetas $a_1 ... a_n$

4. Creamos la máquina de Mealy M_0 combinando las etiquetas de los arcos entrantes a q_i a a_m/t, $m = 1, 2, ...n$ y cambiamos la etiqueta del estado q_i/t a q_i.

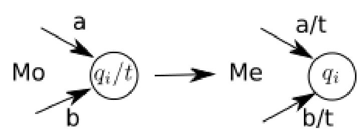

Figura 8.5: Equivalencia entre Mealy y Moore

Ejemplo: Convierta la máquina de Moore en la equivalente máquina de Mealy.

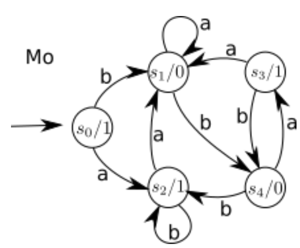

Figura 8.6: Diagrama de Transición

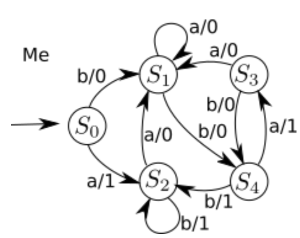

Figura 8.7: Diagrama de Transición

8.2. MEF sin Salida

Se usan para el reconocimiento de lenguajes.

8.2.1. Autómatas Finitos

Este tipo de máquinas tienen estados finales y una cadena será reconocida por la máquina si produce una transición desde el estado inicial a una de sus estados finales.

A.F. Deterministas

Su salida está limitada a los valores aceptados o rechazados. Formalmente, un $AFD = (S, I, \delta, s^*, F)$

$$\begin{aligned} S &: \text{conjunto finito de estados}(s \neq \phi) \\ I &: \text{alfabeto de entrada} \\ \delta &: S \times I \to S \\ F &: \text{conjunto de estados de aceptacion } (F \subseteq S) \\ s^* &: \text{estado inicial} \end{aligned}$$

Representación:
Sea el AFD D tal que:

$$\begin{aligned} S =& \{s_0, s_1, s_2\} \\ I =& \{a, b\} \\ s^* =& s_0 \\ F =& \{s_0\} \end{aligned}$$

$$\begin{aligned} \delta(s_0, a) &= s_1 & \delta(s_0, b) &= s_2 \\ \delta(s_1, a) &= s_1 & \delta(s_1, b) &= s_2 \\ \delta(s_2, a) &= s_2 & \delta(s_2, b) &= s_2 \end{aligned}$$

Tabla de Transición:

	a	b
$\sharp s_0$	s_1	s_2
s_1	s_1	s_2
s_2	s_2	s_2

Diagrama de Transición:

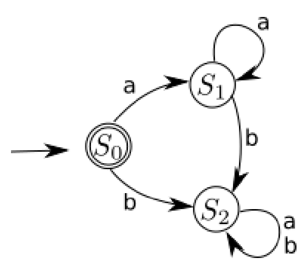

Figura 8.8: Diagrama de Transición de D

$w = babb$ no es aceptada por D.
$F = \{s_1\}$

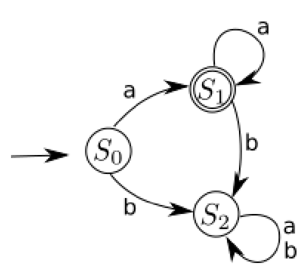

Figura 8.9: Diagrama de Transición

$$\left.\begin{array}{r} u_1 = a \\ u_2 = aa \\ \vdots \\ u_n = a^n \end{array}\right\} \text{Son aceptados por D}$$

En el autómata Fig 8.9. Este autómata acepta cadenas formadas solo por a. El estado s_2 es un estado **muerto**.

Estado Muerto: Es aquel estado que no es de aceptación y no parte de él ninguna transición hacia otro estado.

Nota: El AF D se llama determinístico porque el (s_2) estado siguiente queda bien definido (es único) si son conocidos el estado (s_1) y el símbolo (a):

$$\delta(s_1, a) = s_2$$

8.3. Autómatas Incompletos

En algunas ocasiones podemos encontrar autómatas donde no estén definidas todas las transiciones.

Si una cadena hace llegar al AFD a una situación no definida, se asumirá que la cadena no ha sido reconocido por el autómata.

Capítulo 9

AFD parte II

Ejemplo: Del ejemplo de la anterior clase.

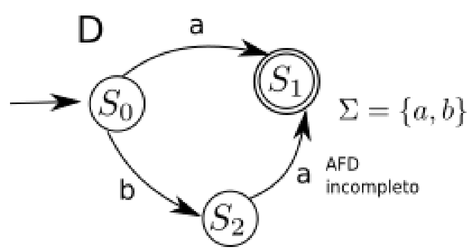

Figura 9.1: AFD incompleto

Para que el autómata D se vuelva completo se debe añadir un estado muerto.

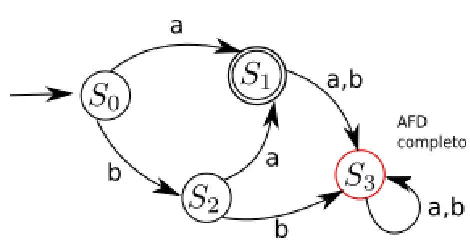

Figura 9.2: AFD completo

Definición: Un AFD es conexo si todos sus estaso son accesibles desde el estado inicial.

Definición: En un autómata que no es conexo, se llamará la **parte conexa** del autómata al conjuto de estados accesibles desde el estado inicial.

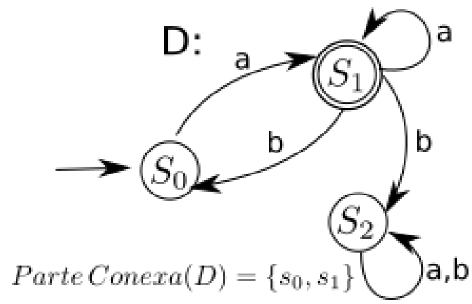

Figura 9.3: Diagrama AFD D

Ejemplo: Dibuje un AFD no conexo y determine su parte conexa.

Definición: Sea un AFD $D = (S, I, \delta, s^*, F)$, llamaremos función de transición extendida a $\widehat{\delta}$ que actúa no solo sobre símbolos sino sobre cadena de símbolos. $\widehat{\delta}$ la definimos recursivamente como sigue:

1. $\widehat{\delta}(s;\varepsilon) = S \qquad \forall s \in S$

2. (parte recursiva) $\widehat{\delta}(s, au) = \widehat{\delta}(\delta(s,a), u) \qquad a \in \Sigma^* \qquad u \in I^*$

   OBS: Notar que $\widehat{\delta}(s,a) = \delta(s,a)$

$$\begin{aligned}
\widehat{\delta}(s,a) &= \widehat{\delta}(s, a.\varepsilon)\\
&= \widehat{\delta}(\underset{\substack{\downarrow\\ \text{Estado sgte}}}{\delta(s,a)}\ ,\varepsilon)\\
&= \delta(s,a)
\end{aligned}$$

Ejemplo: Para el AFD D

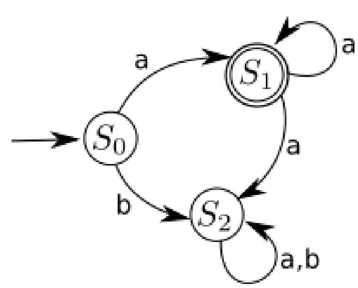

Figura 9.4: Diagrama de Transición de D

Obtener:

- $\widehat{\delta}(s_0, a)$
- $\widehat{\delta}(s_0, ab)$
- $\widehat{\delta}(s_0, baba)$

Usaremos la evaluación rápida.

Solución:

- $\widehat{\delta}(s_0, a) = \delta(s_0, a) = s_1$

  $s_0 \xrightarrow{a} \underline{s_1}$

- $\widehat{\delta}(s_0, ab) = s_2$

  $s_0 \xrightarrow{a} s_1 \xrightarrow{b} \underline{s_2}$

- $\widehat{\delta}(s_0, baba) = s_2$

  camino: $s_0 \xrightarrow{b} s_2 \xrightarrow{a} s_2 \xrightarrow{b} s_2 \xrightarrow{a} \underline{s_2}$

Ejemplo: Sea el AFD D, con $\Sigma = \{0, 1\}$.

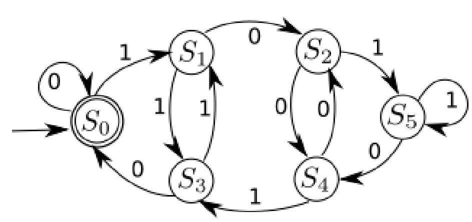

Figura 9.5: Diagrama de Transición de D

Donde $s^* = s_0$, $F = \{s_0\} \rightarrow$ estado de aceptación

Obtener: $\widehat{\delta}(s_0, w)$ para $w = 1010$, usando la definición recursiva.

$$
\begin{aligned}
\widehat{\delta}(s_0, \underset{\text{a}}{\underset{\downarrow}{1}}\underbrace{010}_{u}) &= \widehat{\delta}(\delta(s_0,1), 010) \\
&= \widehat{\delta}(s_1, \underset{\text{a}}{\underset{\downarrow}{0}}\underbrace{10}_{u}) \\
&= \widehat{\delta}(\delta(s_1,0), 10) \\
&= \widehat{\delta}(s_2, \underset{\text{a}}{\underset{\downarrow}{1}}\underset{\text{u}}{\underset{\downarrow}{0}}) \\
&= \widehat{\delta}(\delta(s_2,1), 0) \\
&= \widehat{\delta}(s_5, 0) \\
&= \delta(s_5, 0) \\
&= s_4
\end{aligned}
$$

w no es aceptada por D, para ser aceptada debió acabar en s_0.

Definición: El lenguaje reconocido por un autómata $D = (S, I, \delta, s^*, F)$ es el conjunto:

$$L(D) = \{w \in I^* / \widehat{\delta}(s^*, w) \in F\}$$

Son todas las cadenas aceptadas por el autómata.

Ejemplo: Identifique los elementos del AFD D cuyo lenguaje está dado por el siguiente conjunto:

$$L(D) = \{w \in I^* / w \text{ tiene un número par de símbolos}\}$$

$$I = \{0, 1\}$$

Para el autómata $D = (S, I, \delta, s^*, F)$ se tiene:

$$
\begin{aligned}
I =& \{0, 1\} \\
S =& \{S_{par}, S_{impar}\} \\
F =& \{S_{par}\} \qquad \rightarrow \text{Estado de aceptación} \\
s^* =& S_{par}
\end{aligned}
$$

Su tabla de transición será:

$+$	δ 0	1
$(*) \rightarrow S_{par}$	S_{impar}	S_{impar}
S_{impar}	S_{par}	S_{par}

(*):Tenemos longitud de cadena par + 0 $=S_{impar}$

Su diagrama de estados:

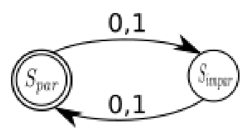

Figura 9.6: Diagrama de Transición de D

Ejemplo: Identifique los elementos del AFD D que reconoce el lenguaje:

$$L(D) = \{w \in I^*/w \text{ tiene un número } \overbrace{\text{par de 0's}}^{P} \text{ y un número } \underbrace{\text{impar de 1's}}_{P}\}$$

Dibuje su tabla y diagrama de transición, $I = \{0, 1\}$.

$$s^* = PP$$

$$S = \{\overset{(1)}{\overset{\uparrow}{P}}\overset{(2)}{\overset{\uparrow}{P}}, \underset{(3)}{\underset{\downarrow}{P}}\underset{(4)}{\underset{\downarrow}{I}}, IP, II\}$$

$$F = \{PP\}$$

(1):par de 0's; (2):par de 1's; (3):par de 0's; (4):impar de 1's.

Donde:

- PP: cantidad par de ceros y cantidad par de unos.
- PI: cantidad par de ceros y cantidad impar de unos.
- IP: cantidad impar de ceros y cantidad par de unos.

- II: cantidad impar de ceros y cantidad impar de unos.

	δ	
+	0	1
$(*)$ →PP	IP	PI
PI	II	PP
IP	PP	II
II	PI	IP

(*): PP le aumentamos 0 y quedaría cantidad impar de ceros, 1's no varían.

Diagrama:

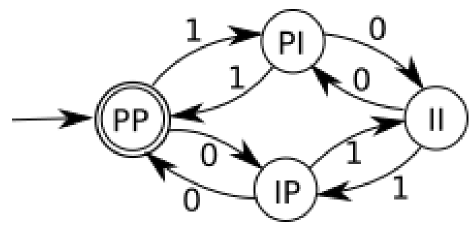

Figura 9.7: Diagrama de Transición de D

9.1. Minimización de un AFD

Definición: Dos estados $s_1, s_2 \in S$ se dirán equivalentes, denotándolos por $s_1 \simeq s_2$, si $\forall w \in I^*$:

$$\widehat{\delta}(s_1, w) \in F \Leftrightarrow \widehat{\delta}(s_2, w) \in F$$

Es decir las transiciones que parten de ellos para cada uno de los símbolos del alfabeto llevan al mismo estado 0 a estados que son equivalentes entre si:

Definición: Un AFD D se llamará autómata mínimo para el lenguaje $L(D)$ si ningún AFD M para $L(M)$ contiene un menor número de estados que D.

9.2. Algoritmo de Minimización de Estados de un AFD

Objetivo: Agrupar estados equivalentes para conseguir un AFD equivalente al original.

Pasos:

1. Vamos a construir una partición π de S formada por dos componentes: los estados de aceptación y los que no son.
2. Refinar la partición separando en diferentes componentes a los estados que no son equivalentes.
3. Terminar cuando cada componente de la partición, agrupa estados equivalentes.

Ejemplo: Minimizar el AFD D siguiente(Figura 9.8):

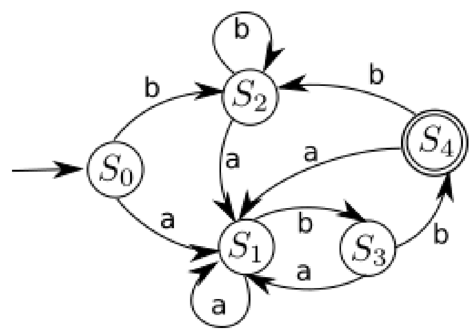

Figura 9.8: Diagrama de Transición de D

$$\Sigma = \{a, b\} \qquad , s^* = s_0$$

Solución: Tenemos $I = \{a, b\}; S = \{s_0, s_1, s_2, s_3, s_4\}; F = \{s_4\}$.

1. Construimos $\pi = \{P_1, P_2\}$

$$\begin{aligned} P_1 =& \{s_4\} \\ P_2 =& \{s_0, s_1, s_2, s_3\} \end{aligned}$$

Evaluamos P_2 en I

	δ	
	a	b
s_0	$s_1 \in P_2$	$s_2 \in P_2$
s_1	$s_1 \in P_2$	$s_3 \in P_2$
s_2	$s_1 \in P_2$	$s_2 \in P_2$
s_3	$s_1 \in P_2$	$s_4 \notin P_2, \in P_1 \rightarrow (*)$

(*): No es estado equivalente. s_3 no es equivalente a los estados s_0, s_1, s_2.

2. Refinamos la partición $\pi = \{P_1, P_2, P_3\}$

$$P_1 = \{s_4\}, \qquad P_2 = \{3\}, \qquad P_3 = \{s_0, s_1, s_2\}$$

Analizamos P_3 en I.

	a	b
s_0	$s_1 \in P_3$	$s_2 \in P_3$
s_1	$s_1 \in P_3$	$s_3 \in P_2 \rightarrow (1)$
s_2	$s_1 \in P_3$	$s_2 \in P_3$

(1): No es estado equivalente. $s_1 \not\simeq s_0, s_2$.

3. Refinamos la partición $\pi = \{P_1, P_2, P_3, P_4\}$.

$$P_1 = \{s_4\}, \qquad P_2 = \{s_3\}, \qquad P_3 = \{s_1\}, \qquad P_4 = \{s_0, s_2\}$$

Analizamos P_4 en I.

	a	b
s_0	$s_1 \in P_3$	$s_2 \in P_4$
s_2	$\underbrace{s_1 \in P_3}_{Equivalencia}$	$\underbrace{s_2 \in P_4}_{Equivalencia}$

Ya no es posible hacer mas refinamientos.

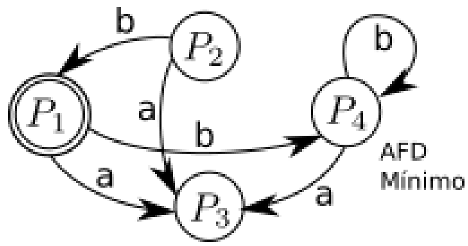

Figura 9.9: Diagrama de Transición del AFD D mínimo

Capítulo 10

Minimización de AFD

10.1. Minimización de un AFD. AFD equivalentes

Definición: Dos AFD definidos sobre un alfabeto I, se dice equivalentes si ambos reconocen el mismo lenguaje.

Formalmente:

D_1 es equivalente a D_2 sobre I si $L(D_1) = L(D_2)$

10.2. Método de Comprobación

Sean los autómatas $D_1 = (S_1, I, \delta_1, s_1^*, F_1)$ y $D_2 = (S_2, I, \delta_2, s_2^*, F_2)$

1. Unirlos en un único autómata determinista de modo que no sea conexo, como sigue:

   $D = D_1 \cup D_2 = (S_1 \cup S_2, I, \delta, s^*, F_1 \cup F_2)$

   donde :

   $\delta(s, a) = \begin{cases} \delta_1(s, a) & s \in S_1 \\ \delta_2(s, a) & s \in S_2 \end{cases}$

   Para s^*, podemos elegir s_1^* o s_2^*

2. Minimizar el autómata AFD D.

3. Si se verifica que los estados iniciales s_1^* y s_2^* son equivalentes diremos que D_1 y D_2 lo son.

Ejemplo: Sean los AFD D_1 y D_2 representados por su diagrama de transición(Figura 10.1 y 10.2):

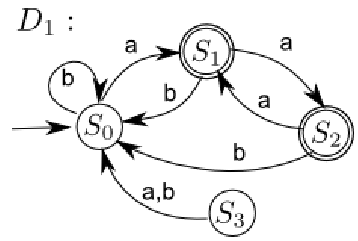

Figura 10.1: Diagrama de transición D_1

$I = \{a, b\}$

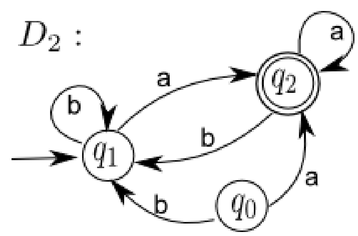

Figura 10.2: Diagrama de transición D_2

Para D_1 :	Para D_2 :
$S_1 = \{s_0, s_1, s_2, s_3\}$	$S_2 = \{q_0, q_1, q_2\}$
$s_1^* = s_0$	$s_2^* = q_1$
$F_1 = \{s_1, s_2\}$	$F_2 = \{q_2\}$

Para D:

$$S = \{s_0, s_1, s_2, s_3, q_0, q_1, q_2\}$$
$$F = \{s_1, s_2, q_2\}$$

Minimizamos D:

Se tiene $\pi = \{P_1, P_2\}$, donde:

$$P_1 = \{s_1, s_2, q_2\} \text{ y}$$
$$P_2 = \{s_0, s_3, q_0, q_1\}$$

Evaluamos P_1 para cada símbolo de I.

	δ	
$S \backslash I$	a	b
s_1	$s_2 \in P_1$	$s_0 \in P_2$
s_2	$s_1 \in P_1$	$s_0 \in P_2$
q_2	$q_2 \in P_1$	$q_1 \in P_2$
	$\downarrow$ estados equivalentes	

Evaluamos P_2 para cada símbolo de I.

	δ		
$S \backslash I$	a	b	
s_0	$s_1 \in P_1$	$s_0 \in P_2$	
s_3	$s_0 \in P_2$	$s_0 \in P_2$	$\rightarrow$ **
q_0	$q_2 \in P_1$	$q_1 \in P_2$	
q_1	$q_2 \in P_1$	$q_1 \in P_2$	

**: No es equivalente al resto.
Refinamos la partición $\pi = \{P_1, P_2, P_3\}$

$$\alpha \begin{cases} P_1 = & \{s_1, s_2, q_2\} \\ P_2 = & \{s_3\} \rightarrow \text{este es equivalente a si mismo} \\ P_3 = & \{s_0, q_0, q_1\} \end{cases}$$

Evaluamos P_3 para cada símbolo de I(miramos D aún), comparamos con α.

	δ	
$S \backslash I$	a	b
s_0	$s_1 \in P_1$	$s_0 \in P_3$
q_0	$q_2 \in P_1$	$q_1 \in P_3$
q_1	$q_2 \in P_1$	$q_1 \in P_3$

Todos son equivalentes.

Así que ya no refinamos. Además en P_3, s_0, q_1 son equivalentes, luego D_1 y D_2 son equivalentes. El autómata equivalente a D_1 y D_2 es(Figura 10.3): Para el estado

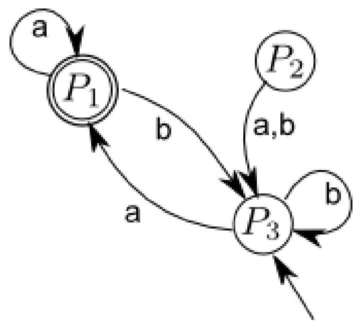

Figura 10.3: Diagrama de transición D

inicial, veo en que conjunto está incluido y ese será mi estado inicial. Veo elementos de P_2, y veo con que símbolo está en la primera gráfica, luego veo donde cae o (estado siguiente) y ese estado veo en que conjunto final está.

$$\left.\begin{array}{l} aba \checkmark \\ bba \checkmark \end{array}\right\} \text{acepta} \rightarrow \text{pero no pasa por } P_2$$

Se identifica que P_2 es un estado inaccesible, luego podemos retirarlo(Figura 10.4).

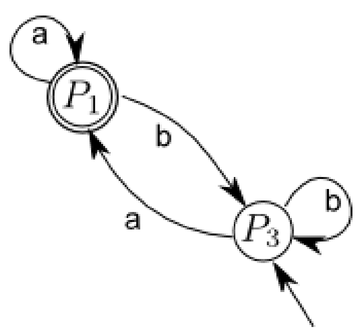

Figura 10.4: Autómata Finito Simplificado

En un AFD: $\delta(\underset{\substack{\downarrow \\ (1)}}{s_1},\ \underset{\substack{\downarrow \\ (2)}}{a}) = \underset{\substack{\downarrow \\ (3)}}{s_2}$

	(1)	Estado actual
donde:	(2)	Símbolo
	(3)	Estado siguiente

10.3. Autómatas Finitos No Deterministas

Si la condición que existe una única transición para el par (s, a) se elimina, es decir, si para algún par (s, a) hubiera transiciones hacia dos o más estados se tiene lo que se denomina, un autómata no determinístico **(AFND)**.

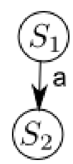

Figura 10.5: AFD

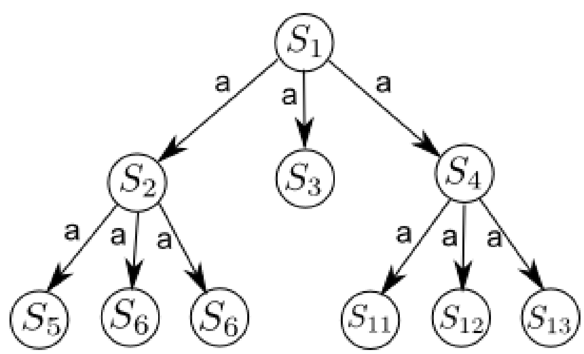

Figura 10.6: AFND

Definición: Un autómata FND, $N = (S, I, \delta, s^*, F)$. S, I, s^* y F se definen del mismo modo que en los AFD.

$$\begin{array}{ll} \delta : & \text{regla de transición} \\ \delta : & S \times I \to P(s) \\ P(s) & \text{incluye a } \phi \end{array}$$

Como δ asigna a cada par de estado y símbolo, un conjunto de estados, se le denomina relación de transición.

10.3.1. Representaciones para un AFND

1. Tabla de transición: Sea el AFND N, donde:

$$\begin{aligned} S = &\quad \{s_0, s_1, s_2\} \\ I = &\quad \{a, b\} \to ** \\ s^* = &\quad s_0 \\ F = &\quad \{s_0\} \end{aligned}$$

**: Cada estado debe tener dos salidas(normalmente). en este caso(AFND) no se cumple.

Ejemplo: Si:

$$\begin{aligned} \delta(s_0, a) = &\quad \{s_1\} \\ \delta(s_1, b) = &\quad \{s_0, s_2\} \\ \delta(s_2, a) = &\quad \{s_0\} \end{aligned}$$

Dibuje la tabla de transición:

	a	b
$(1) \to \#s_0$	$\{s_1\}$	ϕ
s_1	ϕ	$\{s_0, s_2\}$
s_2	$\{s_0\}$	$\phi \to (2)$

(1): Estado de aceptación; (2):Si no me dan información se pone ϕ.

2. Diagrama de Transición:

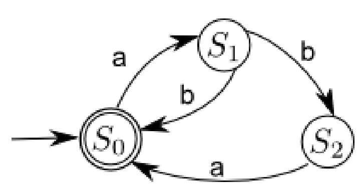

Figura 10.7: Diagrama de Transición

Ejercicio: Halle el D.T del AFND siguiente:

	0	1
$\to s_0$	$\{s_0, s_1\}$	$\{s_3\}$
s_1	$\{s_0\}$	$\{s_1, s_3\}$
$\#s_2$	$\{\phi\}$	$\{s_0, s_2\}$
$\#s_3$	$\{s_0, s_1, s_2\}$	$\{s_1\}$

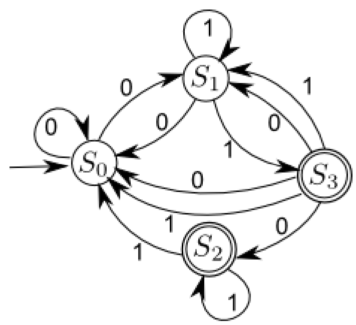

Figura 10.8: Diagrama de Transición

Ejemplo: Considere AFND: $N = (S, I, \delta, s^*, F)$. Procesemos la cadena $w = aba$, $s^* = s_0$

$$\begin{aligned} \delta(s_0, a) = &\quad s_1 \\ \delta(s_1, b) = &\quad s_2 \to \text{Elijo } \{s_0, s_2\} \\ \delta(s_2, a) = &\quad s_0 \to \text{estado de aceptación} \end{aligned}$$

Al ser $\delta(s_1, b) = s_0$ y $\delta(s_0, a) = s_1 \to w$ no es aceptada.

Pero si hay un camino que w si es aceptada entonces w es aceptada por AFND. w es aceptada por AFND N.

En un modelo de computación no determinista se asume que siempre se hace la elección correcta.

Definición: Sea un AFND $N = (S, I, \delta, s^*, F)$. La regla de transición extendida para N es una relación de $P(s) \times I^*$ hacia $P(s)$, la cual se define de forma recursiva como sigue:

- $\widehat{\delta}(\phi, w) = \phi$, $\forall w \in I^*$
- $\widehat{\delta}(A, \varepsilon) = A$, $\forall A \subseteq S$

- $\widehat{\delta}(A, au) = \widehat{\delta}\left(\bigcup_{s\in A} \delta(s,a), u\right),\ A \subseteq S, a \in I, u \in I^*$

En particular cuando $w = a$ y $A \neq \phi$.

$$
\begin{aligned}
\widehat{\delta}(A,a) &= \widehat{\delta}(A, a.\varepsilon) \\
&= \widehat{\delta}\left(\bigcup_{s\in A} \delta(s,a)\varepsilon\right), \\
&\qquad\quad \underset{B=\{s_1 \cup s_2 \cup s_3\}}{\downarrow} \\
&= \bigcup_{s\in A} \delta(s,a)
\end{aligned}
$$

Ejemplo: Dado el AFND, N donde:

$$
\begin{aligned}
S =& \{s_0, s_1\} \\
I =& \{0, 1\} \\
s^* =& s_0 \\
F =& \{s_1\}
\end{aligned}
$$

	0	1
$\rightarrow s_0$	$\{s_0, s_1\}$	$\{s_0\}$
$\# s_1$	ϕ	ϕ

Use la función recursiva y determine si $w = 1010$ es aceptada o no por N en $I = \{0,1\}$.

$$
\begin{aligned}
\widehat{\delta}(\{\underbrace{s_0}_{A}\},\overset{\mathrm{a}}{\overset{\uparrow}{1}}\overbrace{1010}^{u}) &= \widehat{\delta}(\delta(s_0,1),010)\\
&= \widehat{\delta}(\{s_0\},\overset{\mathrm{a}}{\overset{\uparrow}{0}}\overbrace{10}^{u})\\
&= \widehat{\delta}(\delta(s_0,0),10)\\
&= \widehat{\delta}(\{s_0\},10)\\
&= \widehat{\delta}(\delta(s_0,1),0)\\
&= \widehat{\delta}(\{s_0\},0)\\
&= \delta(\{s_0\},0)\\
&= \{s_1,s_0\}
\end{aligned}
$$

Si es estado de aceptación, entonces w es aceptada por N.

Definición: El lenguaje reconocido por un AFND N es el conjunto:

$$L(N) = \{w \in I^*/\widehat{\delta}(\{s^*\},w) \cap F \neq \phi\}$$

Para afirmar que una cadena no está en $L(N)$ debemos agotar todas las posibilidades (formas posibles) de recorrer el diagrama de transición para dicha cadena.

Capítulo 11

Autómatas con Transiciones Epsilon

Extenderemos la definición de los AFND para incluir transiciones de un estado a otro que no dependen de ninguna entrada.

Notación: AFND-ε

Definición: Formalmente un AFND-ε es una quíntupla $N = (S, I, \delta, s^*, F)$ donde S, I, s^*, F se definen como en los AFND. δ es una función total de $(S \times I \cup \{\varepsilon\} \to P(s))$

Ejemplo: El siguiente AFND-ε (Figura 11.1), $I = \{0, 1, 2\}$

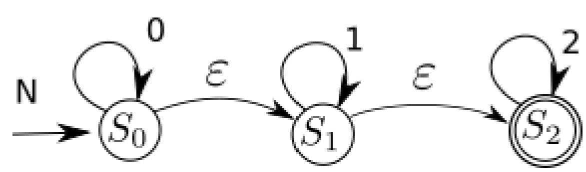

Figura 11.1: Diagrama de Transición

$$L(N) = \{0^m 1^n 2^p / m \geq 0, n \geq 0, p \geq 0\}$$

Ejemplo: Averiguar si la cadena $w = 002$ es aceptada por el AFND-ε N (Figura 11.1), usando la definición recursiva.

$$
\begin{aligned}
\widehat{\delta}(s_0, \underset{\substack{\downarrow\\ \mathrm{a}}}{0}\underset{\substack{\downarrow\\ \mathrm{u}}}{0}2) &= \widehat{\delta}(\delta(s_0,0), \underset{\substack{\downarrow\\ \mathrm{a}}}{0}\underset{\substack{\downarrow\\ \mathrm{u}}}{2}) = \widehat{\delta}(s_0, 02) = \widehat{\delta}(\delta(s_0,0),2)\\
&= \widehat{\delta}(s_0, \underset{\substack{\downarrow\\ \varepsilon 2}}{2}) = \widehat{\delta}(\delta(s_0,\varepsilon),2) = \widehat{\delta}(s_1, \underset{\substack{\downarrow\\ \varepsilon 2}}{2}) = \widehat{\delta}(\delta(s_1,\varepsilon),2)\\
&= \widehat{\delta}(s_2,2) = \delta(s_2,2) = s_2
\end{aligned}
$$

w es aceptado por N, pues $s_2 \in F$.

El camino que va de s_0 hacia s_2 es:

$$s_0 \to \underset{\substack{\downarrow\\ 0}}{s_0} \to \underset{\substack{\downarrow\\ 0}}{s_0} \to \underset{\substack{\downarrow\\ \varepsilon}}{s_1} \to \underset{\substack{\downarrow\\ \varepsilon}}{s_2} \to \underset{\substack{\downarrow\\ 2}}{s_2}$$

Representación de un AFND-ε: δ toma un estado de S y un elemento de $I \cup \{\varepsilon\}$.

Ejemplo: Dibuje la tabla de transición para N

s	0	1	2	ε
s_0	s_0	ϕ	ϕ	s_1
s_1	ϕ	s_1	ϕ	s_2
$\sharp s_2$	ϕ	ϕ	s_2	ϕ

Definición: Para todo estado $s \in S$ definimos la ε-clausura de s como:

$$clausura_{\epsilon}(s) = \{q/q \text{ es accesible desde } s \text{ sin consumir nada en la entrada}\}$$

Nota: El estado s pertenece a $clausura_{\epsilon}(s)$.

Ejemplo: En el AFND-ε N (Figura 11.1) determine la clausura ε para:

- $s = s_0$
- $s = s_1$
- $s = s_2$

Solución:

- $clausura_\varepsilon(s_0) = \{s_0, s_1, s_2\}$
- $clausura_\varepsilon(s_1) = \{s_1, s_2\}$
- $clausura_\varepsilon(s_2) = \{s_2\}$

Podemos extender la ε-clausura a un conjunto de estados Q como sigue:

$$clausura_\varepsilon(Q) = \bigcup_{q\in Q} clausura_\varepsilon(q)$$

$$clausura_\varepsilon(\{q_1, q_2, ..., q_{i_N}\}) = \bigcup_{k=1}^{N} clausura_\varepsilon(q_{i_k})$$

$$Q \subseteq S$$

Ejemplo: Dado el siguiente diagrama (Figura 11.2):

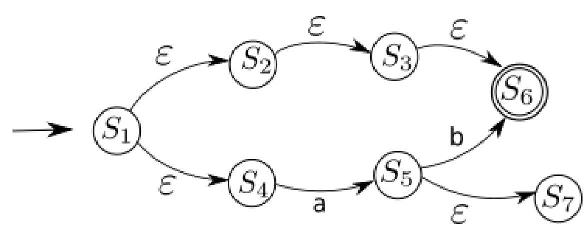

Figura 11.2: Diagrama de Transición

Determine:

- $clausura_\varepsilon(s_1)$
- El camino seguido para llegar a cada estado

Solución:

- $clausura_\varepsilon(s_1) = \{s_1, s_2, s_3, s_4, s_6\}$
-

s	camino
s_1	s_1
s_2	$s_1 \to s_2$
s_3	$s_1 \to s_2 \to s_3$
s_4	$s_1 \to s_4$
s_6	$s_1 \to s_2 \to s_3 \to s_6$

Definición: La función de transición extendida a cadenas $\widehat{\delta}$ se define recursivamente:

- $\widehat{\delta}(s, \varepsilon) = clausura_{\varepsilon}(s)$
- $\widehat{\delta}(s, ua) = clausura_{\varepsilon}(Q)$ Donde:

  $Q = \{q / \exists r \in \widehat{\delta}(s, u) \wedge q \in \delta(r, a)\}$

  $a \in I; u \in I^*; r, s \in S$

Definición: El lenguaje aceptado por un AFND-ε $N = (S, I, \delta, s^*, F)$ es el conjunto.

$$L(N) = \{w / \widehat{\delta}(s^*, w) \wedge F \neq \phi\}$$

Ejemplo: Determine si N acepta a la cadena $w = 01$ en (Figura 11.1).
Solución: Hallamos primero $\widehat{\delta}(s_0, \varepsilon)$ y $\widehat{\delta}(s_0, 0)$, $F = \{s_2\}$.

- $\widehat{\delta}(s_0, \varepsilon) = clausura_{\varepsilon}(s_0) = \{s_0, s_1, s_2\}$
-

$$\begin{aligned}
\widehat{\delta}(s_0, \underbrace{0}_{\varepsilon 0}) &= clausura_{\varepsilon}(\delta(\widehat{\delta}(s_0, \underbrace{\varepsilon}_{u}), \underbrace{0}_{a})) \\
&= clausura_{\varepsilon}(\delta(\{\underbrace{s_0, s_1, s_2}_{Q}\}, 0)) \\
&= clausura_{\varepsilon}(\delta(s_0, 0) \cup \delta(s_1, 0) \cup \delta(s_2, 0)) \\
&= clausura_{\varepsilon}(s_0) = \{s_0, s_1, s_2\}
\end{aligned}$$

Así:

$$\begin{aligned}
\widehat{\delta}(s_0, \underbrace{0}_{u}\underbrace{1}_{a}) &= clausura_{\varepsilon}(\delta(\widehat{\delta}(s_0, 0), 1)) \\
&= clausura_{\varepsilon}(\delta(\{\underbrace{s_0, s_1, s_2}_{Q}\}, 1)) \\
&= clausura_{\varepsilon}(\underbrace{\delta(s_0, 1)}_{\phi} \cup \underbrace{\delta(s_1, 1)}_{s_1} \cup \underbrace{\delta(s_2, 1)}_{\phi}) \\
&= clausura_{\varepsilon}(s_1) = \{s_1, s_2\}
\end{aligned}$$

Como $\{s_1, s_2\} \cap F = \{s_2\}$, se acepta w.

Definición: Se define el conjunto de estado que siguen a p, pasando por σ, mediante:

$$d(p, \sigma) = \{q/\text{hay una transición de } p \text{ a } q \text{ etiquetada por } \sigma\} \qquad \sigma \in I$$

Extensión

$$d(\underbrace{\{p_{i_1}, p_{i_2}, ..., p_{i_k}, a}_{Q}\}) = \bigcup_{j=1}^{k} d(p_{i_j}, a)$$

Ejemplo: Sea el AFND-ε dado el DT (Figura 11.3):

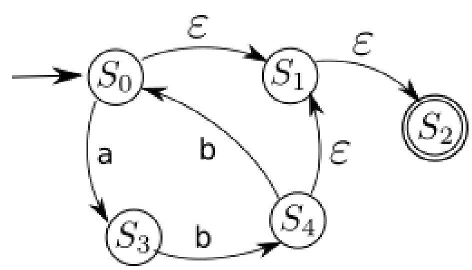

Figura 11.3: Diagrama de Transición

$I = \{a, b\}$ $F = \{s_2\}$ Obtener:

- clausura$_\varepsilon(s_3)$ $\quad d(s_0, a)$
- clausura$_\varepsilon(s_0)$ $\quad d(s_0, b)$
- clausura$_\varepsilon(s_4)$ $\quad d(\{s_3, s_4\}, b)$

Solución:

$$\text{clausura}_\varepsilon(s_3) = \{s_3\}$$
$$\text{clausura}_\varepsilon(s_0) = \{s_0, s_1, s_2\}$$
$$\text{clausura}_\varepsilon(s_4) = \{s_4, s_1, s_2\}$$
$$d(s_0, a) = \{s_3\}$$
$$d(s_0, b) = \phi$$
$$d(\{s_3, s_4\}, b) = d(s_3, b) \cup d(s_4, b) = \{s_4, s_0\}$$

Capítulo 12

Repaso

12.1. AFD - Función de transición extendida

Sea un AFD $D = (S, I, \delta, s^*, F)$, la función de transición extendida:

$$\widehat{\delta} : S \times I^* \to S$$

Queda definida:

- $\widehat{\delta}(s, \varepsilon) = s \qquad \forall s \in S$
- $\widehat{\delta}(s, au) = \widehat{\delta}(\delta(s, a), u) \qquad a \in I, u \in I^*, s \in S$

12.2. AFND - Función de transición extendida

Sea un AFND $N = (S, I, \delta, s^*, F)$ la función de transición extendida $\widehat{\delta}$:

$$\widehat{\delta} : P(s) \times I^* \to P(s)$$

- $\widehat{\delta}(\phi, w) = \phi \qquad w \in I^*$
- $\widehat{\delta}(A, \varepsilon) = A \qquad A \subseteq S$
- $\widehat{\delta}(A, au) = \widehat{\delta}(\bigcup_{s \in A} \delta(s, a), u) \qquad a \in I, A \subseteq S, s \in A, u \in I^*$

12.3. Equivalencia entre AFND y AFD

Para cualquier AFND se puede construir un AFD que reconozca el mismo lenguaje, sea el AFND $N = (S, I, \delta, s^*, F)$. El AFD equivalente será $D = (S^d, I, \delta^d, s^{*d}, F^d)$ en la que:

$S^d = P(S)$, donde:

- Para cada $X \subseteq S \wedge a \in I$

$$\delta^d(X, a) = \bigcup_{s \in X} \delta^d(s, a)$$

- Para cada $a \in I$, $\quad \delta^d(\phi, a) = \phi$
- $F^d = \{X \subseteq S / X \cap F \neq \phi\}$

Utilizamos el siguiente lema para probar la equivalencia.

Lema: Sea D y N los autómatas definidos anteriormente. Entonces para cada $X \subseteq S$ y $w \in I^*$

$$\widehat{\delta}^d(X, w) = \widehat{\delta}(X, w)$$

Prueba: Usaremos inducción sobre |w|. Para $w = \varepsilon$ tiene $\widehat{\delta}^d(X, \varepsilon) = \underset{(1)}{\underset{\downarrow}{X}} = \widehat{\delta}(X, \varepsilon)$

(1): por concepto de AFD.

Por hipótesis de inducción, para una cadena de longitud n, w se cumple:

$$\widehat{\delta}^d(X, w) = \widehat{\delta}(X, w)$$

Bastará probar que $\widehat{\delta}^d(X, aw) = \widehat{\delta}(X, aw)$

CASO I: $X = \phi$

$$\begin{aligned}
\widehat{\delta}^d(\phi, aw) &= \widehat{\delta}^d(\delta^d(\phi, a), w) \\
&= \widehat{\delta}^d(\phi, w) \overset{hi}{=} \widehat{\delta}(\phi, w) \\
&= \phi \\
&\overset{AFND(a)}{=} \widehat{\delta}(\phi, aw)
\end{aligned}$$

CASO II: $X \neq \phi$

$$
\begin{aligned}
\widehat{\delta}(X, aw) &\overset{b.AFD}{=} \widehat{\delta}^d(\delta^d(X,a), w) \\
&= \widehat{\delta}^d\left(\underbrace{\bigcup_{s \in X} \delta^d(s,a), w}_{B}\right) \\
&\text{por la construcción} \\
&= \widehat{\delta}^d(B, w) \\
&= \widehat{\delta}(B, w) \\
&= \widehat{\delta}\left(\bigcup_{s \in X} \delta^d(s,a), w\right) \\
&= \widehat{\delta}(X, aw)
\end{aligned}
$$

Teorema: Sea $N = (S, I, \delta, s^*, F)$ un AFND, entonces existe un AFD $D = (S^d, I, \delta^d, s^{*d}, F^d)$ que es equivalente a N.

Prueba: Sea N un AFND $N = (S, I, \delta, s^*, F)$ y un AFD $D = (P(S), I, \delta^d, s^{*d}, F^d)$ donde δ^d y F^d siguen las definiciones anteriores.

Para probar que $L(D) = L(N)$ basta probar que :

$$w \in L(D) \Leftrightarrow w \in L(N) \quad \forall w \in I^*$$

Tomemos una cadena $w \in I^*$ arbitraria, se cumple:

$$
\begin{aligned}
w \in L(D) &\Leftrightarrow \widehat{\delta}^d(s^*, w) \in F^d \\
&\Leftrightarrow \widehat{\delta}^d(s^*, w) \cap F^d \neq \phi \\
&\Leftrightarrow \widehat{\delta}(s^*, w) \cap F^d \neq \phi \\
&\Leftrightarrow w \in L(N)
\end{aligned}
$$

Ejemplo: Sea el AFND N sobre $I = \{a, b\}$ dado por el D.T: Obtener el AFD D equivalente.

Solución: Para $N : S = \{s_0, s_1, s_2, s_3\}, F = \{s_1, s_3\}, s^* = s_0$

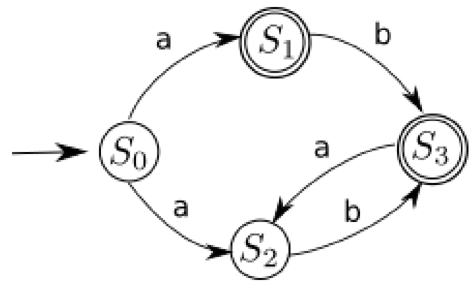

Figura 12.1: Diagrama de Transición

La tabla de transición de N:

	δ	
	a	b
ϕ	ϕ	ϕ
$\rightarrow s_0$	$\{s_1, s_2\}$	ϕ
$\# s_1$	ϕ	s_3
s_2	ϕ	s_3
$\# s_3$	s_2	ϕ

$$A = \{s_1, s_2\}$$
$$\delta(s_1, d) \cup \delta(s_2, d)$$
$$I = \{a, b\}$$

	a	b
$\{s_0\}$	$\{s_1, s_2\}\checkmark$	$\phi\checkmark$
$\{s_1, s_2\}$	ϕ	$\{s_3\}\checkmark$
ϕ	ϕ	ϕ
$\{s_3\}$	$\{s_2\}$	ϕ
$\{s_2\}$	$\{\phi\}$	$\{s_3\}$

12.4. Conversión de un AFND -ε a un AFD

Definiremos operaciones básicas que describen computaciones en los estados de un AFND.

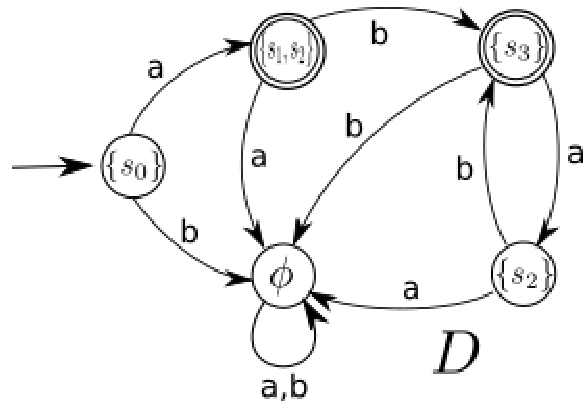

Figura 12.2: Diagrama de Transición

Operación	Descripción
$clausura_{\varepsilon}(s)$	Conjunto de estados alcanzables desde s usando solo transiciones ε.
$clausura_{\varepsilon}(Q)$	$\bigcup_{q\in Q} clausura_{\varepsilon}(q)$
$mover(Q, a)$	Conjuntos de estados del AFND, en los cuales hay una transición al leer el símbolo a desde algún estado $q \in Q$.

12.5. Técnica de Construcción de Subconjuntos

Entrada: Un AFND $N = (S, I, \delta, s^*, F)$
Salida: Un AFD D que acepta el mismo lenguaje $D = (S^d, I, \delta^d, s^{*d}, F^d)$

El estado inicial de D es $s^{*d} = clausura_{\varepsilon}(s^*)$.

D_{tran}, es la tabla de transición para D. F^d, son los estados de aceptación de D, son todos aquellos estados de N que incluyen al menos un estado de aceptación de N.

El conjunto de estados de D, se denotará por $D_{estados}$.
Inicialmente, $clausura_{\varepsilon}(s^*)$ es el único estado de $D_{estados}$ y está sin marcar.

---

while *Exista un estado sin marcar Q en $D_{estados}$* **do**
  for *cada símbolo de entrada* **do**
    $U = clausura_{\varepsilon}(mover(Q, a))$
    if *U no esta en $D_{estados}$* **then**
      añadir U como un estado no marcado a $D_{estados}$
    $D_{tran}(Q, a) = U$

Sea el AFND-ε N sobre $I = \{a, b\}$ convertido a un AFD D.

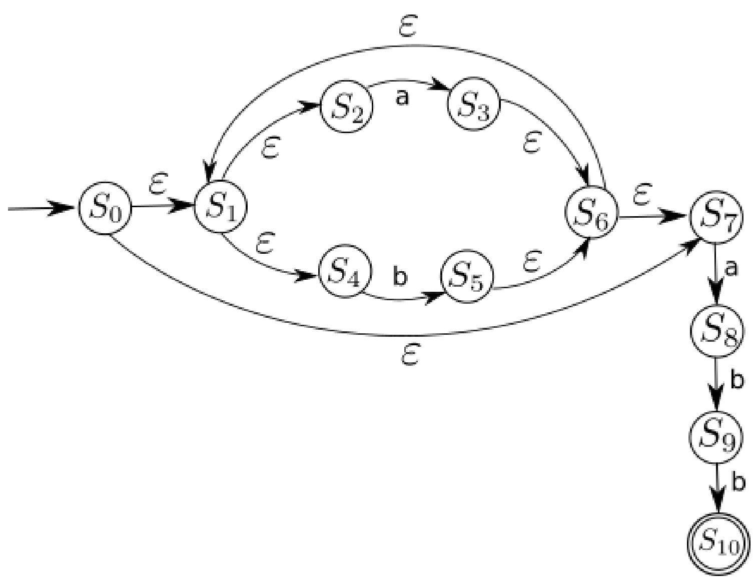

Figura 12.3: Diagrama de Transición

Se tiene $F = \{s_{10}\}, s^* = s_0$. Hallamos $s^{*d} = clausura_{\varepsilon}(s_0) = \{s_0, s_1, s_2, s_4, s_7\} = A$ (sin marcar).

1. Marcamos A y calcular:

- $D_{tran}(A, a) = clausura_{\varepsilon}(mover(A, a))$

$$mover(A, a) = \{s_3, s_8\}$$

$$\begin{array}{ll} \delta(s_0, a) = \phi & \delta(s_2, a) = s_3 \\ \delta(s_1, a) = \phi & \delta(s_4, a) = \phi \\ & \delta(s_7, a) = s_8 \end{array}$$

$$\begin{aligned} clausura_{\varepsilon}(\{s_3, s_8\}) &= clausura_{\varepsilon}(s_3) \cup clausura_{\varepsilon}(s_8) \\ &= \{s_3, s_6, s_7, s_1, s_2, s_4\} \cup \{s_8\} \\ &= \{s_6, s_7, s_1, s_2, s_3, s_4, s_8\} \neq A \end{aligned}$$

Por eso lo llamaremos B(no marcado).

- $D_{tran}(A, B) = clausura_{\varepsilon}(mover(A, b))$

$$\begin{aligned} mover(A, b) &= \{s_5\} \\ \delta(s_4, b) &= s_5 \\ clausura_{\varepsilon}(\{s_5\}) &= \{s_5, s_6, s_7, s_1, s_2, s_4\} = C(no\ marcado) \end{aligned}$$

2. Marcamos B y calcular.

- $D_{tran}(B, a) = clausura_{\varepsilon}(mover(B, a))$. Vemos el conjunto B.

$$mover(B, a) = \{s_3, s_8\}$$

$$\begin{aligned} clausura_{\varepsilon}(\{s_5, s_9\}) &= clausura_{\varepsilon}(\{s_5\}) \cup clausura_{\varepsilon}(\{s_9\}) \\ ignorar :&= \{s_5, s_6, s_7, s_1, s_2, s_4, s_9\} = D(es \neq de\ B) \\ clausura_{\varepsilon}(\{s_3, s_8\}) &= B(\text{ya lo teniamos}) \end{aligned}$$

- $D_{tran}(B, b) = clausura_{\varepsilon}(mover(B, b))$

$$\begin{aligned} mover(B, b) &= \{s_5, s_9\} \\ &= clausura_{\varepsilon}(\{s_5, s_9\} = clausura_{\varepsilon}(\{s_5\}) \cup clausura_{\varepsilon}(\{s_9\}) \\ &= \{s_1, s_2, s_4, s_5, s_6, s_7, s_9\} \\ &= D \end{aligned}$$

3. Marcamos C y calcular:

- $D_{tran}(C, a) = clausura_{\varepsilon}(mover(C, a))$

$$\begin{aligned} mover(C, a) &= \{s_3, s_8\} \\ &= clausura_{\varepsilon}(\{s_3, s_8\}) \\ &= B \end{aligned}$$

- $D_{tran}(C, b) = clausura_{\varepsilon}(mover(C, b))$

$$\begin{aligned} mover(C, b) &= \{s_5\} \\ &= clausura_{\varepsilon}(\{s_5\}) = C \end{aligned}$$

4. Marcamos D y calcular:

- $D_{tran}(D, a) = clausura_{\varepsilon}(mover(D, a))$

$$\begin{aligned} mover(D, a) &= \{s_3, s_8\} \\ &= clausura_{\varepsilon}(\{s_3, s_8\}) = B \end{aligned}$$

- $D_{tran}(D, b) = clausura_{\varepsilon}(mover(D, b))$

$$\begin{aligned} mover(D, b) &= \{s_5, s_{10}\} \\ &= clausura_{\varepsilon}(\{s_5, s_{10}\}) \\ &= clausura_{\varepsilon}(\{s_5\}) \cup clausura_{\varepsilon}(\{s_{10}\}) \\ clausura_{\varepsilon}(\{s_{10}\}) &= \{s_{10}\} \\ &= clausura_{\varepsilon}(\{s_5, s_{10}\}) \\ &= \{s_1, s_2, s_4, s_5, s_6, s_7, s_{10}\} = E(s_{10} \in E) \end{aligned}$$

5. Marcamos E y calcular:

- $D_{tran}(E, a) = clausura_{\varepsilon}(mover(E, a))$

$$\begin{aligned} mover(E, a) &= \{s_3, s_8\} \\ &= clausura_{\varepsilon}(\{s_3, s_8\}) = B \end{aligned}$$

- $D_{tran}(E, b) = clausura_{\varepsilon}(mover(E, b))$

$$\begin{aligned} mover(E, b) &= \{s_5\} \\ &= clausura_{\varepsilon}(\{s_5\}) = C \end{aligned}$$

	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

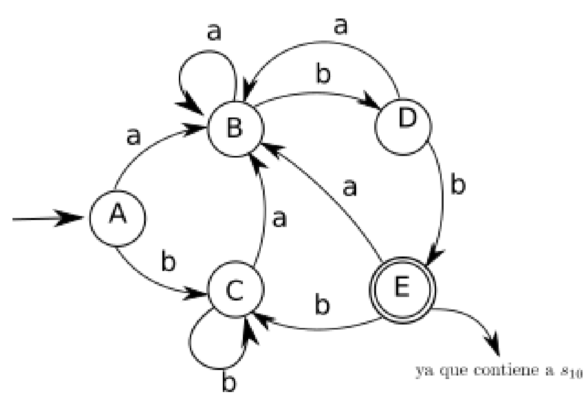

Figura 12.4: Diagrama de Transición

Capítulo 13

Gramáticas

Mientras que los autómatas nos sirven para reconocer cadenas de un lenguaje, las gramáticas son un formalismo diseñado para la definición de lenguajes. Representan un medio para la generación de cadenas que pertenecen a un lenguaje. El elemento fundamental en una gramática es la regla.

13.1. Regla

Dado un alfabeto Σ, una regla(o producción) definida sobre Σ, es un par ordenado (x, y) donde $x, y \in \Sigma^*$. Aquí se dirá a:

- x: Parte izquierda de la regla.
- y: Parte derecha de la regla.

Notación: $x ::= y$

13.1.1. Regla Compresora

Una regla es compresora cuando la longitud de la parte izquierda es mayor o igual a la longitud de la parte derecha.

$$|x| \geq |y|$$

13.2. Tipos de Derivación

13.2.1. Derivación Directa

Dado un alfabeto Σ, un conjunto de producciones P sobre Σ, donde:

$$P = \begin{cases} x_1 ::= y_1 \\ x_2 ::= y_2 \\ \quad\vdots \\ x_n ::= y_n \end{cases}$$

y sea $v, w \in \Sigma^*$. Diremos que w deriva directamente de v, si $\exists t, u \in \Sigma^*$ y una regla en P; $x_i ::= y_i$. tal que $v = tx_iu$ y $w = ty_iu$

Notación: $v \rightarrow w$; se lee "v produce directamente w".

Definición: Dado un alfabeto Σ, un conjunto de producciones sobre Σ y $v, w \in \Sigma^*$, se dice que w deriva de v si $\exists w_0, w_1, ...w_m \in \Sigma^*$ tales que:

$$\begin{aligned} v =& w_0 \rightarrow w_1 \\ & w_1 \rightarrow w_2 \\ & \vdots \\ & w_{m-1} \rightarrow w_m = w \end{aligned}$$

Notación: $v \xrightarrow{*} w$ se lee v produce w.

Ejemplo: Defina un conjunto de reglas que permitan la correcta formación de frases.

Solución: Sea el conjunto de producciones P:

$$\begin{array}{lrl}
R_1 & < oracion > & ::= < sujeto > < predicado > \\
R_2 & < sujeto > & ::= < articulo > < sustantivo > \\
R_3 & < predicado > & ::= < verbo > < complemento > \\
R_4 & < predicado > & ::= < verbo > \\
R_5 & < articulo > & ::= el \\
R_6 & < articulo > & ::= la \\
R_7 & < sustantivo > & ::= atleta \\
R_8 & < sustantivo > & ::= voleibolista \\
R_9 & < verbo > & ::= salta \\
R_{10} & < verbo > & ::= corre \\
R_{11} & < complemento > & ::= alto \\
R_{12} & < complemento > & ::= velozmente
\end{array}$$

A partir de P y comenzando en el item $< oracion >$ podemos obtener:

"el atleta salta alto"
"la voleibolista corre velozmente"
"la voleibolista salta"

Pero hay frases que no podríamos generar como: "velozmente salta voleibolista".

Ejemplo: Describa como se construye la frase "la voleibolista salta" a partir de $< oracion >$

Solución:

$$\begin{aligned}
< oracion > &\to < sujeto > < predicado > && (R_1)\\
&\xrightarrow{*} < articulo > < sustantivo > < predicado > && (R_2)\\
&\xrightarrow{*} < articulo > < sustantivo > < verbo > && (R_4)\\
&\xrightarrow{*} la < sustantivo > < verbo > && (R_6)\\
&\xrightarrow{*} \text{la voleibolista } < verbo > && (R_8)\\
&\xrightarrow{*} \text{la voleibolista salta} && (R_9)
\end{aligned}$$

Ejemplo: Defina un conjunto de reglas que describan cómo son las las instrucciones que permiten asignar el valor de una expresión a una variable en un lenguaje de programación.'m'

Solución: Sea el conjunto P.

$$
\begin{array}{lrl}
R_1 & < asignacion > & ::=< variable >' = ' < expresion > \\
R_2 & < expresion > & ::=< variable > \\
R_3 & < expresion > & ::=< numero > \\
R_4 & < expresion > & ::=< expresion >' + ' < expresion > \\
R_5 & < expresion > & ::=< expresion >' * ' < expresion > \\
R_6 & < variable > & ::=' A ' \\
R_7 & < variable > & ::=' B ' \\
R_8 & < variable > & ::=' C ' \\
R_9 & < numero > & ::=' 3 ' \\
R_{10} & < numero > & ::=' 6 '
\end{array}
$$

Si asumimos que A,B,C son $< variables >$; 3, 6 son $< numeros >$ podremos generar:

$$
\begin{aligned}
A &= B + C \\
B &= B * 3 \\
C &= B + 6
\end{aligned}
$$

No podemos obtener las siguientes instrucciones:

$$
\begin{aligned}
A &= +2 * +4 \\
4 &= B
\end{aligned}
$$

En resumen:

- El objeto es llegar a tener una secuencia correcta de símbolos.
- Los símbolos son: el, la, atleta, voleibolista, corre, etc. A, B, C, 3, 6, *, +
- Se usan algunas producciones.

Definición: Una gramática formal definida sobre Σ es una cuádrupla $G =$

$(\Sigma_N, \Sigma_T, P, s)$ donde:

$$\begin{aligned}
\Sigma_N &: \text{Alfabeto de símbolos no terminales}\\
\Sigma_T &: \text{Alfabeto de símbolos terminales}\\
P &: \text{Conjunto de producciones}\\
s &: \text{símbolo inicial}
\end{aligned}$$

Además se cumple:

- $s \in \Sigma_N$
- $\Sigma_N \cap \Sigma_T = \phi$
- $\Sigma = \Sigma_N \cup \Sigma_T$

Ejemplo: Defina formalmente una gramática para obtener cualquier número natural precedido de un signo.

Solución: Los elementos de G son:

$$\begin{aligned}
\Sigma_T &= \{+, -, 0, 1, ..., 9\}\\
\Sigma_N &= \{< signo >, < digito >, < numero >, < caracter >\}\\
s &= < numero >
\end{aligned}$$

$$P \begin{cases} <numero> & ::=<signo><digito> \\ <signo> & ::= + \\ <signo> & ::= - \\ <digito> & ::=<caracter><digito> \\ <digito> & ::=<caracter> \\ <caracter> & ::= 0 \\ <caracter> & ::= 1 \\ & \vdots \\ <caracter> & ::= 9 \end{cases}$$

Con esta gramática generamos números como: $+371, -4692, +7$.

13.3. Forma Normal de Backus-Naur (BNF)

Es una forma estandarizada de enunciar la composición de una gramática. En la BNF:

- Los símbolos no terminales inician en ”<” y terminan en ”>”.
- La regla (S, T) se expresa $S ::= T$
- Las reglas de la forma: $s ::= T_1, s ::= T_2, ..., s ::= T_n$

  pueden combinarse en: $s ::= T_1|T_2|...|T_n$. La barra se lee como ”ó”

Ejemplo: Defina el conjunto P usando la notación BNF.

$$P \begin{cases} <numero> & ::=<signo><digito> \\ <signo> & ::= +|- \\ <digito> & ::=<digito><caracter> \mid <caracter> \\ <caracteer> & ::= 0|1|2|...|9 \end{cases}$$

Definición: La relación $\xrightarrow{n}$ se define recursivamente mediante:

- $x \xrightarrow{0} x \qquad \forall x \in \Sigma^*$

- $w \xrightarrow{n} xuy \cap u \rightarrow v$ entonces $w \xrightarrow{n+1} xvy; \quad x, y, u, v, w \in \Sigma^*$

  Cuando se tiene $x \xrightarrow{n} y$ se lee y deriva de x en n pasos.

Observación: Se usarán las notaciones:

- $x \xrightarrow{*} y$ si existe $n \geq 0$ tal que $x \xrightarrow{n} y$

- $x \xrightarrow{+} y$ si existe $n > 0$ tal que $x \xrightarrow{n} y$

Ejemplo: Sea la gramática G, donde P está dado por:

$$\begin{array}{lrl} R_1 & s & ::= aAbc \\ R_2 & A & ::= aAbC \\ R_3 & A & ::= \varepsilon \\ R_4 & Cb & ::= bC \\ R_5 & Cc & ::= cc \end{array}$$

Verifique:

- Si.

$$\begin{array}{lll} & s & \xrightarrow{2} abc \\ (R_1) & s & ::= aAbc \\ (R_3) & & ::= abc \end{array}$$

- $s \xrightarrow{n} a^3b^3c^3 \qquad n =?$

Ejemplo: La siguiente gramática se usa para generar cualquier numero natural precedido de un signo. Los elementos para G seran:

$$\Sigma_T = \{0, 1, 2, ..., 9, +, -, (,)\}$$
$$\Sigma_N = \{< signo >, < digito, < numero >, < caracter >\}$$

Donde S está formado por:

$$S \begin{cases} < numero > & ::=< signo >< digito > \\ < signo > & ::= + \\ < signo > & ::= - \\ < digito > & ::=< caracter >< digito > \\ < digito > & ::=< caracter > \\ < caracter > & ::= 0 \\ < caracter > & ::= 1 \\ \vdots & \\ < caracter > & ::= 9 \end{cases}$$

Definición: Dada una gramática $G = (\Sigma_N, \Sigma_T, P, S)$, llamaremos lenguage generado a partir de una gramatica G al conjunto:

$$L(G) = \{w \in \Sigma_T^* / S \xrightarrow{*} w\}$$

Son las palabras formadas por simbolos terminales derivables a partir del simbolo inicial S.

Definición(Recursividad): Una gramática es recursiva si tiene alguna derivación recursiva, es decir si $A \xrightarrow{*} xAy$ donde $A \in \Sigma_N, x, y \in \Sigma^*$

- Si $x = \varepsilon$ la gramática es recursiva por la izquierda.

- Si $y = \varepsilon$ la gramática es recursiva por la derecha.

Ejemplo: Sea la gramática que genera expresiones aritméticas conteniendo operadores de suma, resta y paréntesis.

$$\Sigma_T = \{0, 1, 2, ..., 9, +, -, (,)\}$$
$$\Sigma_N = \{E, T, N, D\}$$

donde P está formado por:

$$P \begin{cases} E & ::= E+T|E-T|T \\ T & ::= (E)|N \\ N & ::= DN|D \\ D & ::= 0|1|...|9 \end{cases}$$

Se pide:

- A partir de T obtenga una secuencia de sumas y/o restas de T usando recursividad a la izquierda.

- A partir de N genere un número de 4 cifras usando recursividad a la derecha.

Solución:

- $$\begin{aligned} E &\rightarrow E+T && (R_1) \\ &\xrightarrow{*} E-T+T && (R_2) \\ &\xrightarrow{*} E-T-T+T && (R_2) \\ &\xrightarrow{*} T-T-T+T && (R_3) \end{aligned}$$

- $$\begin{aligned} N &\rightarrow DN && (R_6) \\ &\xrightarrow{*} DDN && (R_6) \\ &\xrightarrow{*} DDDN && (R_6) \\ &\xrightarrow{*} DDDD && (R_7) \end{aligned}$$

13.4. Clasificación de las Gramáticas

13.4.1. Gramáticas Regulares (GR)

Se llaman también lineales y es el grupo más restringido de gramática. En la parte derecha de la regla tenemos como máximo dos símbolos.

Hay dos tipos de gramáticas regulares:

1. Gramática Lineal por la Derecha (GLD). Sus reglas son de la forma:

$$\begin{array}{lll} A & ::= a & \qquad A, B \in \Sigma_N \\ A & ::= aB & \qquad a \in \Sigma_T \\ A & ::= \varepsilon & \end{array}$$

2. Gramática Lineal por la Izquierda (GLI). Sus reglas son de la forma:

$$A ::= a$$

$$A ::= Ba$$

$$A ::= \varepsilon$$

Hay una equivalencia entre la GLD y la GLI.

Ejemplo: Sea la gramática regular.

$$P \left\{ \begin{array}{ll} S & ::= aA \\ S & ::= bA \\ A & ::= bB \\ A & ::= a \\ B & ::= aA \\ B & ::= bA \end{array} \right.$$

Entonces:

- Determine si $ba \in L(G)$
- Muestre una derivación de la cadena $w = bababa$

Capítulo 14

Autómatas Finitos y Gramáticas Regulares

Teorema: La clase de los lenguajes generados por una gramática regular es precisamente la de los lenguajes regulares.

14.1. Procedimiento de Conversión de una GR a un AF

1. Para cada símbolo no terminal de la gramática asociar un estado en el autómata.

2. Para cada regla $A ::= bC$ en la gramática origina una transición (A, b, C) en el autómata.

3. Para cada regla $A ::= b$ se tendrá transiciones (A, b, Z) donde Z será el único estado final del autómata.

Ejemplo: Sea la GR $G = (\Sigma_N, \Sigma_T, S, P)$ donde P esta dado por:

Del ejemplo anterior:

$$S ::= aA$$
$$S ::= bA$$
$$A$$

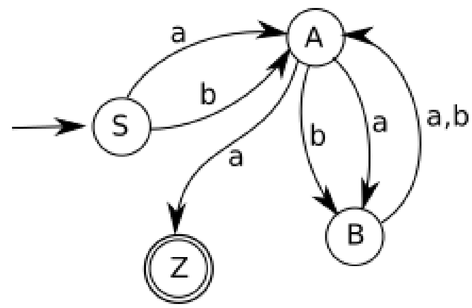

Figura 14.1: Diagrama de Transición

14.2. Procedimiento de Conversión de un AFD a un GR

1. Para cada transición de la forma $((\underset{(1)}{\underset{\downarrow}{p}}, a), \underset{(2)}{\underset{\downarrow}{q}})$ en el AFD((1): estado anterior, (2): estado siguiente), donde:

   $p, q \in S, a \in I$ habrá una regla $X_p ::= aX_q$ en la gramática. En la que X_i es la variable en que corresponde al estado i del AFD.

2. Para cada transición $((p, a), q)$ donde $q \in F$ incorpora ademas la regla.

$$X_p ::= a \qquad (\text{también } X_p ::= aX_q)$$

Ejemplo: Dado el AFD. Obtener la GR equivalente.

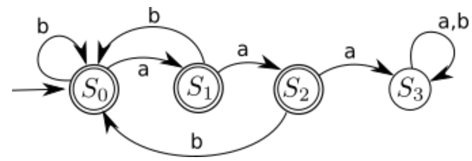

Figura 14.2: Diagrama de Transición

Solución: Se obtiene las siguientes reglas:

$$P\begin{cases} s_0 ::= bs_0 \\ s_0 ::= as_1 \\ s_1 ::= as_2 \\ s_1 ::= bs_0 \\ s_2 ::= as_3 \\ s_2 ::= as_0 \\ s_3 ::= as_3 \\ s_3 ::= bs_3 \\ s_0 ::= b \\ s_0 ::= a \\ s_1 ::= a \\ s_1 ::= b \\ s_2 ::= b \end{cases}$$

Dado un AFD $D = (S, I, \delta, s^*, F)$ se desea encontrar un GLD G, tal que $L(D) = L(G)$. Si q no es un estado final, la gramática buscada es $G = (\Sigma_N, \Sigma_T, P, q)$ en la que P consiste de reglas de la forma:

$$p ::= aq \qquad \text{Si } \delta(p,a) = q$$

$$p ::= a \qquad \text{Si } \delta(p,a) = q \in F$$

Ejemplo: Sea el AFD D dado por Fig 14.3:

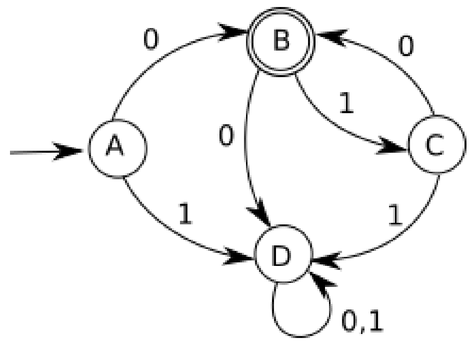

Figura 14.3: Diagrama de Transición

$$
\begin{array}{ll}
A ::= 0B & \\
A ::= 1D & ✗ \\
B ::= 0D & ✗ \\
B ::= 1C & \\
C ::= 0B & \\
C ::= 1D & ✗ \\
D ::= 0D & ✗ \\
D ::= 1D & ✗ \\
A ::= 0 & \\
C ::= 0 &
\end{array}
$$

D es un estado muerto, quitamos las reglas de la forma $p :: aD$

Quedará:

$$
P \begin{cases} A ::= 0B \\ B ::= 1C \\ C ::= 0B \\ A ::= 0 \\ C ::= 0 \end{cases}
$$

0 y 1 son Σ_T

14.3. Conversión de una Gramática Regular GR a AFND-ε

Sea la GLD $G = (\Sigma_N, \Sigma_T, s, P)$ encontraremos un AFND-ε.

$$N = (S^{nd}, I^{nd}, \delta^{nd}, s^*, F^{nd})$$

tal que $L(G) = L(N)$

Sea:

1. $I^{nd} = \Sigma_T$
2. Los estados S^{nd} de N serán S además de los sufijos de la parte derecha de las reglas.
3. $s^* = S$
4. Para definir δ, se tiene:
   - Si $\alpha ::= \beta \quad \in P$ entonces se define $\delta(\alpha, \varepsilon) = \beta$
   - Si $a\alpha \in S^{nd}, a \in I^{nd}$ se hace $\delta(a\alpha, a) = \alpha$

Los estados finales F^{nd} del autómata N serán los estados rotulados como ε.

Ejemplo: Sea G la GLD definida por:

$$\begin{aligned} \Sigma_T &= \{0, 1\} \\ S &::= 0A \\ A &::= |A|\varepsilon \end{aligned}$$

Obtener el AF equivalente.

Solución:

- $I^{nd} = \{0, 1\} \cup \{\varepsilon\}$
- $S^{nd} = \{S, 0A, 1A, \overset{\substack{\text{sufijo} \\ \uparrow}}{A}, \varepsilon\}$

- $s^* = S$
- Diagrama en Fig 14.4.

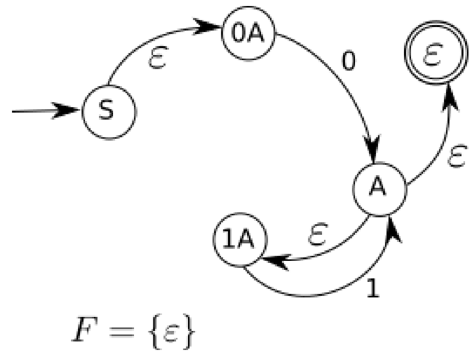

Figura 14.4: Diagrama de Transición

14.4. Jerarquía de las Gramáticas

De acuerdo a Noam Chomsky se tiene la siguiente clasificación para las gramáticas.

1. G. Regulares.
2. G. Sensibles al contexto.
3. G. sin Restricciones.
4. G. Libres de contexto.

14.5. Gramáticas Sensibles al Contexto

También se les conoce como gramáticas del tipo I y sus producciones son de la forma.

$$xAy \to xvy$$

$$A \in \Sigma_N, \quad x, y \in \Sigma^*, \qquad\qquad v \in \Sigma^*$$

En una gramática S. Contexto no hay reglas compresoras.

Ejemplo: Sea G la gramática.

$$S ::= \overbrace{abc}^{R_1} \mid \overbrace{aAbc}^{R_2}$$
$$A ::= \overbrace{abc}^{R_3} \mid \overbrace{aAbc}^{R_4}$$
$$Cb ::= \overbrace{bC}^{R_5}$$
$$Cc ::= \overbrace{cc}^{R_6}$$

$$\begin{aligned} S &::= aAbc && (R_2) \\ &::= aabCbc && (R_3) \\ &::= aabbCc && (R_5) \\ &::= aabbccc && (R_6) \end{aligned}$$

$L(G) = \{abc, \overbrace{aa}^{a^2}\overbrace{bb}^{b^2}\overbrace{ccc}^{c^3}, a^3b^3c^3, ...\}$

$L(G) = \{a^n b^n c^n / n \geq 1\}$

14.6. G. Sin Restricciones

Se llaman también recursivamente enumeradas o gramáticas del tipo ”O”. Sus reglas son de la forma:

$xAy ::= v \qquad A \in \Sigma_N; x, y, v \in \Sigma^*$

Ejemplo: La gramática G:

$$\begin{cases} aS ::= bSb \\ aSb ::= \varepsilon \\ SbS ::= bcS \end{cases}$$

Es una gramática sin restricciones.

14.7. G. Libres de Contexto

También conocidos como gramáticas del tipo 2. Se caracterizan porque la parte izquierda de la regla está formado por un único símbolo no terminal.

$$A ::= v; \qquad A \in \Sigma_N; \qquad v \in \Sigma^* = \Sigma_N \cup \Sigma_T; \cup : combinacion$$

Estas gramáticas son especialmente adecuadas para representar los aspectos sintácticos de un lenguaje de programación. Se observa de la definición que podemos incluir $A \to \varepsilon$.

Ejemplo:

1. Sea $\Sigma_T = \{a, b\}$ y P.

$$\begin{aligned} S ::= \overbrace{\varepsilon}^{R_1} \mid \overbrace{aSb}^{R_2} \\ S ::= aSb \qquad (R_2) \\ S :: aaSbb \qquad (R_2) \\ S ::= aaaSbbb \qquad (R_2) \\ S ::= a^n S b^n \qquad (R_2) \\ S ::= a^n \varepsilon b^n = a^n b^n \qquad (R_1) \\ L(G) = \{a^n b^n / n \geq 0\} \\ \text{ya que} \\ S ::= \varepsilon \\ S \to \varepsilon \end{aligned}$$

2. Sea $\Sigma_T = \{a, b\}$ y P.

$$S ::= \underbrace{aSa}_{R_1} \mid \underbrace{bSb}_{R_2} \mid \underbrace{a}_{R_3} \mid \underbrace{a}_{R_4} \mid \underbrace{\varepsilon}_{R_5}$$

$$\begin{aligned} L &= \{a, b, \varepsilon, abaaaaba, ...\} \\ S &::= aba \qquad (R_1) \\ S &::= abSba \qquad (R_2) \\ S &::= abaSaba \qquad (R_1) \\ S &::= abaaSaaba \qquad (R_2) \\ S &::= abaa|aaba \qquad (R_5) \\ L &= \{uu^R/u \in \Sigma^*\} \\ L &= \{w/w = w^R, w \in \Sigma^*\} \end{aligned}$$

3. Sea $\Sigma = \{a, b\}$ y P dado por:

$$\begin{aligned} S &::= aS|aB \\ B &::= bC \\ B &::= bC \\ c &::= aC|a \end{aligned}$$

Capítulo 15

Árbol de Derivación

Definición: Sea $G = \{\Sigma_N, \Sigma_T, S, P\}$ una gramática libre de contexto(GLC). Un AD es un árbol ordenado, construido recursivamente como sigue:

1. Un árbol sin aristas cuyo único vértice tiene etiqueta S, es un AD de S.

2. Si $x \in \Sigma_N$ es etiqueta de una hoja h de un AD A, entonces:

   - Si $x \to \varepsilon \in P$, entonces el árbol obtenido incrementando a A un vértice v con etiqueta ε y una arista $\{h, v\}$ es un AD.
   - Si $x \to x_1x_2...x_n \in P$, donde $x_1, x_2, ...x_n \in \Sigma_T \cup \Sigma_N$, entonces el árbol obtenido incrementando a A n vértices $v_1, v_2, ..., v_n$ con etiquetas $x_1, x_2, ..., x_n$ en ese orden, y con n aristas $\{h, v_1\}, \{h, v_2\}, ...\{h, v_n\}$ es un AD.

Ejemplo: Sea G un GLC donde P está dado por:

$$E \to E + T|T$$
$$T \to T + F|F$$
$$F \to (E)|t$$

Obtener el AD para la cadena $w = t + (t + t)$, partiendo de E.

Solución:

$$
\begin{aligned}
E &\rightarrow T && (R_2)\\
&\rightarrow T + F && (R_3)\\
&\rightarrow T + (E) && (R_5)\\
&\rightarrow T + (E + T) && (R_1)\\
&\rightarrow T + (T + T) && (R_2)
\end{aligned}
$$

Luego se aplica las reglas reiteradas veces, uno a la vez

$$
\begin{aligned}
&\rightarrow F + (T + T) && (R_4)\\
&\rightarrow F + (F + T) && (R_4)\\
&\rightarrow F + (F + F) && (R_4)\\
&\rightarrow t + (F + F) && (R_6)\\
&\rightarrow t + (t + F) && (R_6)\\
&\rightarrow t + (t + t) && (R_6)
\end{aligned}
$$

Se ha requerido 11 pasos para derivar w (Fig 15.1).

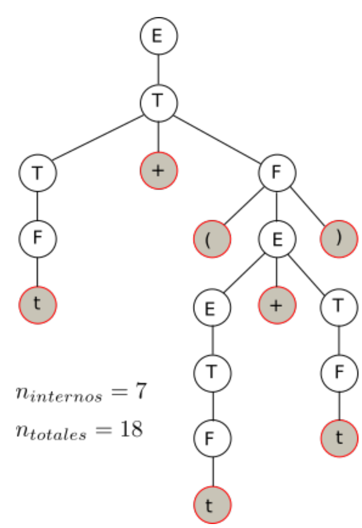

Figura 15.1: Arbol de derivación

Reglas

1. La raiz se etiqueta con el simbolo inicial
2. Los hijos de la raiz son aquellos simbolos que aparecen al lado derecho de la composicion usado para reemplazar el simbolo inicial.
3. Todo nodo etiquetado con un no terminal tiene unos nodos hijos etiquetados en los simbolos del lado derecho de la produccion usada para sustituir ese no terminal.
4. Los nodos que no tienene hijos deben ser etiquetados con sumbolos terminales.

Ejemplo: Dada la la GLC dada por:

$$P = \begin{cases} S & \rightarrow AB \\ A & \rightarrow aA|a \\ B & \rightarrow bB|b \end{cases}$$

1. Verifique si w=aabbb se deriva a partir de S

2. En caso afirmativo, presente el arbol de derivacion

Solución:

$$\begin{aligned} S &\rightarrow AB && (R_1) \\ &\rightarrow aAB && (R_2) \\ &\rightarrow aaB && (R_3) \\ &\rightarrow aabB && (R_4) \\ &\rightarrow aabbB && (R_4) \\ &\rightarrow aabbb && (R_5) \end{aligned}$$

La cantidad de pesos para derviar w es 6.

Cada nodo interno del árbol será un símbolo no terminal, mientras que las hojas serán los símbolos terminales. Una regla $A := x_1 ... x_n$ se representará como un símbolo cuyo nodo padre es A, siendo sus nodos hijos $x_1, x_2, ..., x_n$.

Ejemplo: Sea G la GLC dada por:

$$\begin{aligned} S &\to AB \\ &\to aA|a \\ &\to bB|b \end{aligned}$$

Obtener el Árbol de Derivación para $w = aabbb$ e indicar la cantidad de pasos

Solución:

$$\begin{aligned} S &\to AB && (R_1) \\ &\to aAB && (R_2) \\ &\to aaB && (R_3) \\ &\to aabB && (R_4) \\ &\to aabb && (R_5) \end{aligned}$$

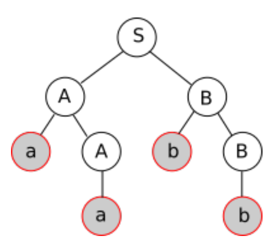

Figura 15.2: Arbol de derivación

$$\left.\begin{array}{r} n_i = 4 \\ n_T = 9 \end{array}\right\} \text{Pasos } = 9 - 4 = 5 \quad \checkmark$$

Definición: Una GLC G se llama ambigua, cuando es posible obtener dos o mas ADs diferentes para alguna sentencia que genere. Puede haber otras gramáticas GLC equivalente a una GLC ambigua, y que éstas gramáticas no sean ambiguas.

Ejemplo: Sea G una GLC donde P esta dado por:

$$\begin{aligned} S &\to S + S && (R_1) \\ &\to S * S && (R_2) \\ &\to (S) && (R_3) \\ &\to t && (R_4) \end{aligned}$$

Obtener el AD para $w = t + t + t$ partiendo de S.

Solución:

$$\begin{aligned} S &\to S + S && (R_1) \\ S &\to S + S + S && (R_1) \\ S &\to t + S + S && (R_4) \\ S &\to t + t + S && (R_4) \\ S &\to t + t + t && (R_4) \end{aligned}$$

1. $n_i = 5$, $n_t = 10$.

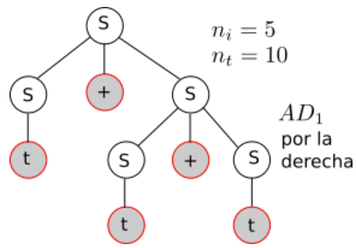

Figura 15.3: Arbol de derivación

2.

$$
\begin{aligned}
&S \to S + S && (R_1)\\
&S \to t + S && (R_4)\\
&S \to t + S + S && \text{ERROR } (R_1)\\
&S \to t + t + S && (R_4)\\
&S \to t + t + t && (R_4)
\end{aligned}
$$

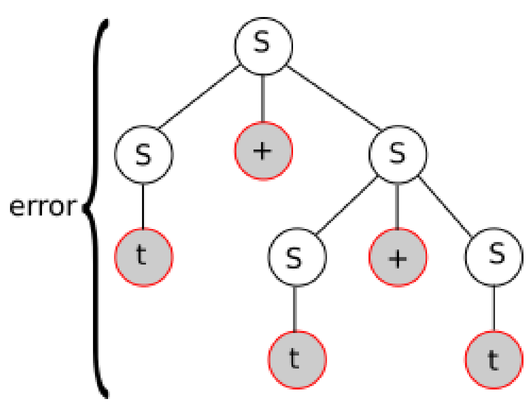

Figura 15.4: Arbol de derivación

$$S \rightarrow S + S \qquad (R_1)$$
$$S \rightarrow S + S + S \qquad (R_1)$$
$$S \rightarrow S + S + t \qquad (R_4)$$
$$S \rightarrow t + S + t \qquad (R_4)$$
$$S \rightarrow t + t + t \qquad (R_4)$$

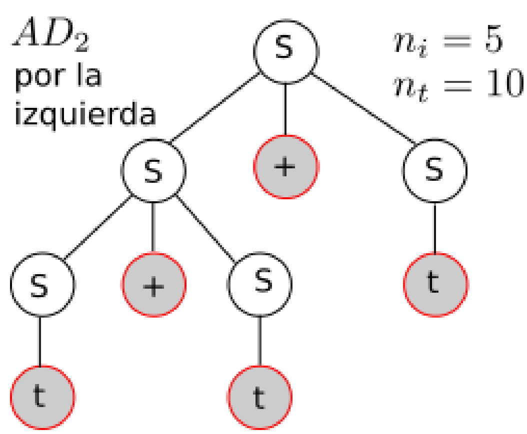

Figura 15.5: Arbol de derivación

El (1) se puede interpretar como

$$\underset{\text{segundo}}{\overset{t}{\downarrow}} + (\underbrace{t+t}_{primero})$$

El (2) se puede interpretar como

$$(\underbrace{t+t}_{primero}) + \underset{\text{segundo}}{\overset{t}{\downarrow}}$$

Definición: Un lenguaje libre de contexto L se dice inherentemente ambiguo si todas las GLC para L son ambiguas.

Definición: Una derivación se denomina derivación a la izquierda si en cada paso, se expande la variable más a la izquierda.

Una derivación se dirá derivación a la derecha si en cada paso se expande la variable más a la derecha.

Ejemplo: Sea G una GLC donde P está dado por:

$$S \to SbS \qquad (R_1)$$
$$S \to ScS \qquad (R_2)$$
$$S \to a \qquad (R_3)$$

Para la cadena $w = abaca$.

1. Obtener una derivación por la izquierda.

2. Obtener una derivación por la derecha.

3. Obtener su AD.

Solución:

1.

$$\begin{aligned} S &\to \underline{S}bS && (R_1) \\ &\to ab\underline{S} && (R_3) \\ &\to ab\underline{S}cS && (R_2) \\ &\to abacS && (R_3) \\ &\to abaca && (R_3) \end{aligned}$$

2.

$$
\begin{aligned}
S &\rightarrow Sc\underline{S} && (R_2)\\
&\rightarrow \underline{S}ca && (R_3)\\
&\rightarrow Sb\underline{S}ca && (R_1)\\
&\rightarrow \underline{S}baca && (R_3)\\
&\rightarrow abaca && (R_3)
\end{aligned}
$$

3.

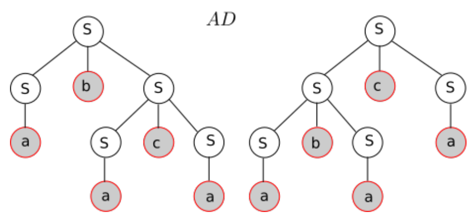

Figura 15.6: Arbol de derivación

15.1. Equivalencia en Gramáticas

Un mismo lenguaje puede ser generado por mas de una gramática.

Definición: Dos gramáticas G^1 y G^2 se llaman equivalentes, si ambas generan el mismo lenguaje sobre Σ_T. Es decir:

$$L(G^1) = L(G^2)$$

Esto es, si generan el mismo lenguaje.

En muchas ocasiones es recomendable simplificar ciertas gramaticas eliminando simbolos o reglas no deseadas.

Elementos Indeseables en Gramáticas

Definición: Una regla innecesaria es una producción de la forma $A := A$.

Definición: Sea $G = (\Sigma_N, \Sigma_T, S, P)$ una GLC. Una variable $X \in \Sigma_N$ se llama útil si y solo si existen dos cadenas $u, v \in \Sigma^*$ tales que:

$$S \xrightarrow{*} uXv \text{ y existe } w \in {\Sigma_T}^* \text{ tal que } uXv \xrightarrow{*} w$$

Definición: Un símbolo inaccesible o inútil es aquel símbolo no terminal que no aparece en ninguna cadena de símbolos que pueda derivarse a partir del símbolo inicial de la gramática.

Ejemplo: Sea G la GLC dado por:

$$\begin{aligned} S &\to AB|a \\ B &\to b \\ C &\to c \end{aligned}$$

Identifique las variables inútiles

Solución:

- C es una variable inútil. No existen subcadenas u, v tales que $S \xrightarrow{*} uCv$.
- A es inútil?. Vemos subcadenas u, v tales que $S \to AB$, $u = \varepsilon, v = B$ pero $AB \xrightarrow{*}?$. No existe $w \in {\Sigma_T}^*$, luego A es inútil.
- B es inútil, a pesar que.

$$\begin{aligned} S &\to AB \\ u &= A \\ v &= \varepsilon \end{aligned}$$

No existe w tal que

$$AB \xrightarrow{*} w$$

Definición: Un símbolo no generativo es aquel símbolo no terminal a partir del cual no puede derivarse ninguna cadena de símbolos terminales.

Sea $G^1 = (\Sigma_N^1, \Sigma_T, S, P^1)$ una GLC. Transformaremos G^1 en $G^2 = (\Sigma_N^2, \Sigma_T, S, P^2)$ de modo que $L(G^1) = L(G^2)$ y para todo $A \in \Sigma_N^2$ se obtenga $A \xrightarrow{*} w$ para algún $w \in \Sigma_T^*$.

Algoritmo

1. Inicializar Σ_N^2 con las variables A tales que $A \rightarrow w$ es una regla de G^1 donde $w \in \Sigma_T^*$

2. Inicializar P^2 con todas las reglas $A \rightarrow w$ para los cuales $A \in \Sigma_N^2$ y $w \in \Sigma_T^*$.

3. | | |
   |---|---|
   | Repetir: | |
   | | Añadir a Σ_N^2 todas las variables A para los cuales $A \rightarrow w$ para algún $w \in (\Sigma_N^2 \cup \Sigma_T)^*$ que sea una producción de P^1 y añadirla a P^2. |
   | Hasta que no se puedan añadir mas variables a Σ_N^2 | |

Ejemplo: Sea la gramática G^1:

$$\begin{aligned} & S \rightarrow Aa|B|D \\ \checkmark\ & B \rightarrow b \\ & A \rightarrow aA|bA|B \\ \checkmark\ & C \rightarrow abd \end{aligned}$$

Use el algoritmo anterior, para obtener una gramática sin símbolos no generativos.

Solución:

$$\begin{aligned} \Sigma_T &= \{a, b, d\} \\ \Sigma_N^1 &= \{S, A, B, C, D\} \end{aligned}$$

1. $\Sigma_N^2 = \{B, C\}$

2. $P^2 : \begin{cases} B \to b \\ C \to abd \end{cases}$

3. $\Sigma_N^2 = \{B, C, S, A\}$

   a)

$$S \to B$$
$$A \to B$$
$$P^2 : \begin{cases} B \to b \\ C \to abd \\ S \to B \\ A \to B \end{cases}$$

   b)

$$S \to Aa$$
$$A \to aA$$
$$A \to bA$$
$$P^2 : \begin{cases} B \to b \\ C \to abd \\ S \to B|Aa \end{cases}$$

   c) $S \to D$. Donde D no está en Σ_N^2

Capítulo 16

Eliminación de Producciones ε

Definición: Denominamos producción ε a una regla de la forma:

$$A \rightarrow \varepsilon$$

Definición: Una variable $A \in \Sigma_N$ en una GLC se dirá anulable si $A \xrightarrow{*} \varepsilon$.

Notación: El conjunto de todos los símbolos no terminales anulables se denotará por N.

Algoritmo:

1. Inicializar N con todas las $A \in \Sigma_N | A \rightarrow \varepsilon$.

2. Repetir:

   Si $B \rightarrow w$; con $w \in \Sigma^*$ y todos los símbolos w están en N, añadir B a N.

   Hasta que no se pueda añadir más variables a N

16.1. Algoritmo para eliminación de Reglas ε

Sea $G^1 = (\Sigma_N, \Sigma_T, S, P^1)$ una GLC, se obtendrá una gramática $G^2 = (\Sigma_N, \Sigma_N, \Sigma_T, S, P^2)$ sin reglas ε, excepto $S \rightarrow \varepsilon$ tal que $L(G^1) = L(G^2)$.

1. Obtener N.
2. Inicializamos $P^2 \leftarrow \phi$.
3. Para cada regla $X \rightarrow w \in P^1$, hacer:
   - Para cada representación de w como $X_1Y_1X_2, ..., X_nY_nX_{n+1}$ con $Y_1, ..., Y_n \in N$ $P^2 \leftarrow P^2 \cup \{X \rightarrow X_1X_2...X_{n+1}\}$
4. Devolver $G^2 = (\Sigma_N, \Sigma_T, S, P^2 - \{X \rightarrow \varepsilon / X\})$

Ejemplo: Sea la gramática G^1 con producciones ε:

$$S \rightarrow aA$$
$$A \rightarrow aA|\varepsilon$$

Solución:

1. $N = \{A\}$, todos los $A \rightarrow \varepsilon$
2. $P^2 \leftarrow \phi$
3.

$$S \rightarrow aA|a$$
$$A \rightarrow aA|a$$

Luego $G^2 = (\Sigma_N, \Sigma_T, S, P^2)$ donde:

$$\Sigma_N = \{A, S\}$$
$$\Sigma_T = \{a\}$$
$$P^2 = \begin{cases} S \rightarrow aA|a \\ A \rightarrow aA|a \end{cases}$$

Regla: A partir de una producción de P^1.

$$B \rightarrow X_1X_2...X_n \qquad X_i \text{ son anulables}$$

Se pueden conseguir nuevas producciones en P^2.

Ejemplo: Sea la regla $B \to X_1X_2$ con X_1, X_2 anulables. Se pueden obtener las producciones $B \to X_2|X_1|X_1X_2$.

Ejemplo: Dada la gramática:

$$P^1 \begin{cases} S \to ASB|C \\ A \to AaaA|\varepsilon \\ B \to BBb|C \\ C \to cC|\varepsilon \end{cases}$$

Elimine las producciones ε.

Solución: Las variables anulables $N = \{A, C, S, B\}$, el conjunto P^2 está determinado por:

$$\begin{cases} S \to SB|AB|AS|B|A|S|ASB|C \\ A \to aaA|Aaa|aa|AaaA \\ B \to Bb|Bb|b|C \quad \text{(nos quedamos con 1)} \\ C \to c|cC \end{cases}$$

El conjunto P^2 está dado por:

$$\begin{cases} S \to SB|AB|AS|B|A|ASB|C|\varepsilon \\ A \to aaA|Aaa|AaaA|aa \\ B \to Bb|b|b \\ C \to c|cC \end{cases}$$

16.2. Eliminación de Producciones Unitarias

Definición: Una regla se llama producción unitaria o no generativa si es de la forma:

$$A \to B \qquad \text{con } A, B \in \Sigma_N$$

Las p.u. hacen que la GLC se vuelva compleja.

Ejemplo: En la GLC G dada por:

$$A \to B$$
$$B \to w|C \qquad w \in \Sigma^*$$

Podemos eliminar la producción

$$A \to w|C$$

Aquí se añadió la p.u $A \to C$

Definición: Sea G una GLC. Para cada símbolo $A \in \Sigma_N$, se define el siguiente conjunto:

$$Unit(A) = \{B \in \Sigma_N / A \xrightarrow{*} B\} \qquad \text{(usando solo p.u)}$$

OBS: $A \in Unit(A)$, ya que $A \xrightarrow{*} A$ en ϕ p.u

Definición: Sean $A, B \in \Sigma_N$. Nos referiremos (A, B) como el par unitario, si verifica que $A \xrightarrow{*} B$ empleando solo producciones unitarias.

Ejemplo: Dada la gramática G:

$$A \to B$$
$$B \to C|w_1$$
$$C \to D$$
$$D \to w_2$$

Obtener Unit(A), Unit(B), Unit(C), Unit(D)

$$Unit(A) = \{A, B, C, D\}$$
$$Unit(B) = \{B, C, D\}$$
$$Unit(C) = \{C, D\}$$
$$Unit(D) = \{D\}$$

16.2.1. Método de eliminación de p.u

Sea $G^1 = (\Sigma_N, \Sigma_T, S, P^1)$ una GLC con p.u construiremos una GLC equivalente $G^2 = (\Sigma_N, \Sigma_T, S, P^2)$ sin p.u mediante:

1. Hállese Σ_N.
2. Para cada $A \in \Sigma_N$ determine Unit(A).
3. Para cada p.u (A, B) añadir a P^2 todas las producciones $A \to w$ donde $B \to w$ es una producción unitaria de P^1.
4. Retire todas las producciones unitarias de P^1.

Ejemplo: Sea G la GLC con producciones unitarias:

$$P^1 \begin{cases} S \to A|Aa \\ A \to B \\ B \to C|b \\ C \to D|ab \\ D \to b \end{cases}$$

Obtener la gramática equivalente G^2 sin p.u tal que $L(G^1) = L(G^2)$

Solución:

1. $\Sigma_N = \{S, A, B, C, D\}$

2.

$$Unit(S) = \{S, A, B, C, D\} \qquad Unit(A) = \{A, B, C, D\}$$
$$Unit(B) = \{B, C, D\} \qquad Unit(C) = \{C, D\}$$
$$Unit(D) = \{D\}$$

3.

Pares Unitarios	Producciones no unitarias
(S,S)	$S \to Aa$
(S,A)	——
(S,B)	$S \to b$
(S,C)	$S \to ab$
(S,D)	$S \to b$
(A,A)	——
(A,B)	$A \to b$
(A,C)	$A \to ab$
(A,D)	$A \to b$
(B,B)	$B \to b$
(B,D)	$B \to ab$
(B,D)	$B \to b$
(C,C)	$C \to ab$
(C,D)	$C \to b$
(D,D)	$D \to b$

Las reglas de P^2 son:

$$\begin{cases} S \to Aa|b|ab \\ A \to b|ab \\ B \to b|ab \\ C \to ab|b \\ D \to b \end{cases}$$

16.2.2. Estandarizacion de gramaticas

Existen formas estandares de gramáticas,a saber:

1. Forma Normal de Chomsk
2. Forma normal de Greibach

Capítulo 17

Forma Normal de Chomsky

Definición: Una $G = (\Sigma_N, \Sigma_T, s, P)$ está en la FNCh si:

- G no contine variables inútiles.
- G no contiene producciones ε
- G no contiene producciones unitarias.

Todas las producciones son de la forma:

$$\begin{array}{ll} A \to \sigma & A \in \Sigma_N, \sigma \in \Sigma_T \\ A \to BC & A, B, C \in \Sigma_N \\ S \to \varepsilon & si\ \varepsilon \in L(G) \end{array}$$

Teorema: Sea G una GLC. Existe una GLC en la FNCh que es equivalente a G.

17.1. Método de Conversión

Debemos hacer la siguientes simplificaciones:

- Identificas las variables anulables
- Eliminamos las producciones ε (salvo $s \to \varepsilon$)

- Eliminamos las producciones unitarias.

Toda producción de tal gramática tendrá la forma:

$$A \to a \qquad\qquad a \in \Sigma_T$$

ó

$$A \to w \qquad\qquad |w| \geq 2$$

Luego, la tarea será:

1. Disponer que todas las cadenas de longitud mayor o igual a 2 consistan sólo de variables.

2. Descomponer todas las cadenas w de longitud mayor o igual que 3 en una cascada de producciones tal que cada regla tenga el cuerpo formada por 2 variables.

Para tal efecto, la construcción es como sigue: Debemos descomponer todas las producciones de la forma:

$A \to B_1B_2...B_k$(con $k \geq 3$) en un grupo de reglas con 2 variables en el cuerpo.

Agregaremos $(k-2)$ nuevas variables: $Z_1, ..., Z_{k-2}$

Con esto, la producción original se reemplaza por las $(k-1)$ reglas.

$$\begin{aligned} A &\to B_1Z_1 \\ Z_1 &\to B_2Z_2 \\ Z_2 &\to B_3Z_3 \\ &\vdots \\ Z_{k-2} &\to B_{k-1}B_k \end{aligned}$$

Ejemplo: Reemplace las producción $A \to abBaC$ con producciones simples y binarias.

Solución: Incorporamos las variables T_a, T_b:

$$T_a \to a \qquad T_b \to b$$
$$A \to T_a T_b B T_a C$$

Añadimos 3 nuevas variables: Z_1, Z_2, Z_3

$$P = \begin{cases} A \to T_a Z_1 \\ Z_1 \to T_b Z_2 \\ Z_2 \to B Z_3 \\ Z_3 \to T_a C \\ T_a \to a \\ T_b \to b \end{cases}$$

Ejercicio: Llevar a producciones simples y binarias la regla:

$$A \to BAaCbb$$

Ejemplo: Dada la GLC G

$$S \to AB|aBC|SBS$$
$$A \to aA|C$$
$$B \to bbB|b$$
$$C \to cC|\varepsilon$$

Obtener la GLC equivalente a G que esté en su FNCh.
Solución:

- Eliminaremos las producciones ε. Hallamos los anulables $\Gamma = \{C, A\}$ Agregamos las producciones:

$$
\begin{aligned}
S &\to B|aB \\
A &\to a \\
C &\to c
\end{aligned}
$$

Nos queda:

$$
G^1 = \begin{cases}
S \to AB|aBC|SBS|B|aB \\
A \to aA|C|a \\
B \to bbB|b \\
C \to cC|c
\end{cases}
$$

- Eliminamos las producciones unitarias. Para cada variables hallamos su unitario.

$$
\begin{aligned}
Unit(A) &= \{A, C\} \\
Unit(B) &= \{B\} \\
Unit(C) &= \{C\} \\
Unit(S) &= \{S, B\}
\end{aligned}
$$

Incluimos los pares unitarios.

Par no unitario - Prod. No Unitarias	
(A,A)	$A \to aA\|a$
(A,C)	$A \to cC\|c$
(B,B)	$B \to bbB\|b$
(C,C)	$C \to cC\|c$
(S,S)	$S \to AB\|aBC\|SBS\|aB$
(S,B)	$S \to bbB\|b$

$$G^2 = \begin{cases} S \to AB|aBC|SBS|aB|bbB|b \\ A \to aA|a|cC|c \\ B \to bbB|b \\ C \to cC|c \end{cases}$$

- Reemplazamos los símbolos terminales por variables en cada regla que no es simple.

  Incorporamos las variables: T_a, T_b, T_c tales que: $T_a \to a, T_b \to b, T_c \to c$

$$G^3 = \begin{cases} S \to AB|T_aBC|SBS|T_aB|T_bT_bB|b \\ A \to T_aA|a|T_cC|c \\ B \to T_bT_bB|b \\ C \to T_cC|c \end{cases}$$

  Llevamos a producciones binarias incorporando Z_1, Z_2, Z_3

$S \to T_aBC$ se reemplaza por: $S \to T_aZ_1$

$Z_1 \to BC$

$S \to SBS$ se reemplaza por: $S \to SZ_2$

$Z_2 \to BS$

$S \to T_bT_bB$ se reemplaza por: $S_{\to}T_bZ_3$

$Z_3 \to T_bB$

$B \to T_bT_bB$ se reemplaza por: $B \to T_bZ_3$

$Z_3 \to T_bB$

$$G^4 = \begin{cases} S \to AB|T_aZ_1|SZ_2|T_aB|T_bZ_3|b \\ A \to T_aA|a|T_cC|c \\ B \to T_bZ_3|b \\ C \to T_cC|c \\ Z_1 \to BC \\ Z_2 \to BS \\ Z_3 \to T_bB \\ T_a \to a \\ T_b \to b \\ T_c \to c \end{cases}$$

G^4 está en FNCh

Capítulo 18

Simplificación de GLC

En una GLC podemos identificar tres defectos que debemos eliminar.

1. Los factores comunes izquierdos.

2. La recursividad por la izquierda.

3. La ambigüedad.

18.1. Factores Comunes Izquierdos

Una GLC tiene factores comunes izquierdos si hay por lo menos 2 producciones con el mismo símbolo en la parte izquierda y puede tener algunos símbolos coincidentes en la parte derecha.

Se tendrá formalmente:

$$A ::= \delta\alpha_1|\delta\alpha_2|...|\delta\alpha_n|\beta_1|...|\beta_n \quad n \geq 2, |\delta| > 0$$

Para eliminar los FCI realice la siguiente sustitución. Agregar una variable C de modo que:

$$A ::= \delta C|\beta_1|...|\beta_m$$
$$C ::= \alpha_1|...|\alpha_n$$

18.2. La recursividad por la Izquierda

Un símbolo no terminal A es denominado recursivo por la izquierda si:

$$A ::= Aw \qquad w \in \Sigma^*$$

Eliminación de Recursividad Izquierda.

Las reglas de un símbolo no terminal A se pueden descomponer como:

$$A ::= A\alpha_1|...|A\alpha$$
$$A ::= \beta_1|...|\beta_m \qquad \text{donde } \alpha_i, \beta_i \in \Sigma^*$$

El primer símbolo de cada β_i es diferente de A. Podemos eliminar la recursividad por la izquierda si introducimos una variable Z de tal modo que:

$$A ::= \beta_1|...|\beta_m|\beta_1 Z|...|\beta_m Z$$
$$Z ::= \alpha_1|...|\alpha_n|\alpha_1 Z|...|\alpha_n Z$$

Lema: En una GLC cualquiera, una producción de la forma $A \to uBv$ se puede reemplazar por:

$$A \to uw_1v|...|uw_nv$$
$$B \to w_1|...|w_n$$

18.3. Ambigüedad

No hay algún algoritmo que permita eliminar la ambigüedad. En caso de LLC que solo tienen GLC ambiguas es imposible eliminar dicha ambigüedad.

Sin embargo en algunos casos es posible resolver este problema, analizando cuales son sus causas.

Ejemplo: Sea la gramática G, que sirve para la definición de expresiones aritméticas, donde:

$$\Sigma_T = \{id, cte, (,), +, -, *, /\}$$
$$\Sigma_N = \{E, O\} \qquad S = E$$
$$E ::= E\ O\ E|(E)|id|cte$$
$$O ::= +|-|*|/$$

Se pide obtener el árbol de derivación para $w = id + cte * id$

Solución:

$$\begin{aligned} E \ &::= EOE \\ &::= EOEOE \\ &::= idOEOE \\ &::= id + EOE \\ &::= id + cteOE \\ &::= id + cte * E \\ &::= id + cte * id \end{aligned}$$

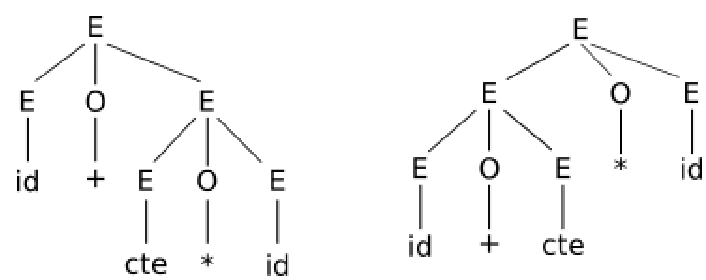

Figura 18.1: Arbol de derivación

Luego G es ambigua.

Para resolver esta ambigüedad que se debe a que no se ha definido un orden de prioridad entre los operadores , se considerará lo siguiente:

1. Los símbolos * y / tiene una prioridad más alta que + y -.

2. Si hay dos operaciones con igual prioridad realizar la evaluación de izquierda a derecha.

Introducimos las variables:

T: término

A: + ó -

F: factor

M: multiplicación y división

Definimos las gramáticas equivalentes G^2.

$$\Sigma_N = \{E, T, F, A, M\}$$
$$S = E$$
$$E ::= EAT|T$$
$$T ::= TMF|F$$
$$F ::= (E)|id|cte$$
$$A ::= +|-$$
$$M ::= *|div$$

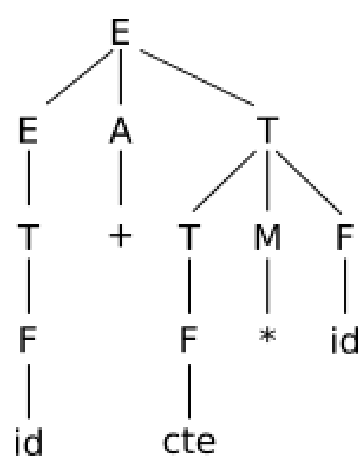

Figura 18.2: Arbol de derivación

18.4. Forma Normal de Greibach

Una GLC está en la FNG si:

1. La variable inicial no es recursiva.

2. G no tiene variables inútiles.

3. G no tiene producciones ε(salvo $S \to \varepsilon$)

4. Todas las producciones son de la forma:

$$A \to \sigma \qquad \text{(regla simple)}$$
$$A \to aB_1 ... B_k \qquad \sigma, a \in \Sigma_T; B_i \in \Sigma_N$$

18.4.1. Características

1. En cada paso de la derivación aparece un único símbolo terminal.

2. La derivación de una cadena de longitud n ($n \geq 1$) tiene exactamente n pasos.

Método: Para convertir una gramática a su FNG. realice estos pasos:

1. Enumere las variables en un orden arbitrario pero fijo en la que el símbolo S sea la variable de orden.

2. Para cada variable de la gramática, A, de acuerdo al orden elegido, modifique las producciones de tal modo que el primer símbolo a la derecha de la flecha sea un terminal.

3. Utilice el **Lema** para modificar las variables originales de tal modo que el primer símbolo a la derecha de la flecha sea un terminal. Se debe seguir el orden inverso de enumeración de las variables: última, penúltima,etc.

4. Utilice de nuevo el **Lema**, para modificar las producciones de las variables nuevas, de tal modo que el primer símbolo a la derecha de la flecha sea un terminal.

Ejemplo: Sea G la GLC dada por:

$$S ::= AA|a$$
$$A ::= AA|b$$

Llevarla a su FNG **Solución:**

1. El orden es S,A

2. Eliminamos la recursividad a la izquierda de la variable A. Introducimos la variable Z.

$$A ::= AA$$
$$A ::= b$$
$$A ::= b|bZ$$
$$Z ::= A|AZ$$

   Luego, nos queda la gramática:

$$S ::= AA|a$$
$$A ::= b|bZ \qquad (*)$$
$$Z ::= A|AZ$$

   Usamos(*) para reemplazar en las derivaciones de S

$$S ::= bA|bZA|a$$
$$A ::= b|bZ$$
$$Z ::= A|AZ$$

   Descomponemos las reglas de Z por:

$$Z ::= A$$
$$Z ::= AZ$$

Usando $A ::= b|bZ$, se obtiene finalmente:

$$S ::= bA|bZA|a$$
$$A ::= b|bZ$$
$$Z ::= b|bZ|bZZ$$

Finalmente ya está en la FNG.

18.5. Automatas Push Down

Arquitectura

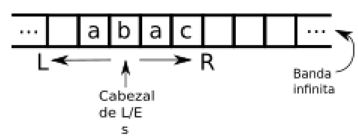

Figura 18.3: Máquina de Turing

Un automata nde pila determinista (PDA-D) es una séptupla

$$M = (S, \Sigma, s_0, \Gamma, \gamma_0, \delta, F)$$

S : Conjunto de estados
Σ : Alfabeto de la cinta
Γ : Alfabeto de pila
s_0 : Estado inicial $s_0 \in S$
γ_0 : Simbolo en la cima de la pila
Δ : Es la funcion de transicion.
Δ :S x $(\Sigma U\{\varepsilon\})x\Gamma \to Sx\Gamma^*$
F : Conjunto de estados de aceptación $(F \subseteq S)$

18.5.1. Paso computacional

La transicion $\Delta(s_1, a, z) = (s_2, v)$ se le denominará un paso computacional. Graficamente:

Casos especiales:

1. $\Delta(s_1, a, z) = (s_2, z)$. El contenido de la pila no se altera.
2. $\Delta(s_1, a, z) = (s_2, \varepsilon)$. Se borra el simbolo Z ubicado en la cima de la pila, y el control pasa a leer la nueva cima de la pila.
3. $\Delta(s_1, \varepsilon, z) = (s_2, w)$

Esta es una transicion ε. No se procesa el simbolo de la cinta de entrada, la unidad de control no se mueve a la derecha. En la cima reemplazamos Z por la cadena w.

Para garantizar el determinismo solo debe estar definido:

$$\Delta(s, a, z) \text{ o } \Delta(s, \varepsilon, z); a \in \Sigma$$

Configuracion instantanea

Es una terna de forma:

$$(s, au, zv)$$

Representa: El automata esta en el estado s. au es la parte no procesada de la cadena. El cabezal apunta al simbolo a. La cadena zv es el contenido total de la pila.

Para representar el paso computacional, escribiremos:

$$(s_1, au, zw) \to (s_2, u, tw)$$

El automata utilizo la transicion $\Delta(s_1, au, zw) = (s_2, t)$

La notación: $(s_1, u, \beta) \to^* (s_2, v, \gamma)$ significa el automata pasa de la CI (s_1, u, β) a la CI (s_2, v, γ) en cero, uno o mas pasos computacionales.

Configuracion inicial

Para una cadena $w \in \Sigma^*$, la configuración inicial es: (s_0, w, γ). El contenido inicial es γ_0 al iniciar el procesamiento de la cadena.

Configuracion de aceptación

La configuración $(s_a, \varepsilon, \beta)$ se llama de aceptacion si $s_a \in F$ y ademas se debe haber procesado toda la cadena. La cadena β que queda en la pila puede ser cualquier cadena de simbolos en Γ^*

Lenguaje aceptado por una APD

Representa el lenguaje aceptado por un PDA-D mediante:

$$L(P) = \{w \in \Sigma^*/(s_0, w, \gamma_0) \to^* (s_a, \varepsilon, \beta)\}$$

Donde $s_0 \in S$, $s_a \in F$, $\beta \in \Gamma^*$

Teorema: Todo lenguaje regular L es aceptado por algun PDA-D

Prueba:

Sea $M_1 = (S, \Sigma, \delta, s_0, F)$ un AFD que acepta a L.

El PDA-D $M_2 = (S, \Sigma, \Gamma, \Delta, s_0, \gamma_0, F)$ definido haciendo $\Gamma = \{\gamma_0\}$ y $\Delta(s, a, \gamma_0) = (\delta(s, a), \gamma_0)$ $\forall a \in \Sigma, \gamma_0 \in \gamma$

Satisfacer $L(M_1) = L(M_2)$

Ejm: Diseñar un PDA-D M que acepte el lenguaje

$$L = \{a^i b^i / i \geq 1\} \text{ sobre } \Sigma = \{a, b\}$$

Sol: Consideramos el PDA-D

$$M = (S, \Sigma)$$

...

La función de transicion Δ esta dada por:

$\Delta(s_0, a, \gamma_0)$	d
Σ	: Alfabeto de la cinta
Γ	: Alfabeto de pila
s_0	: Estado inicial $s_0 \in S$
γ_0	: Simbolo en la cima de la pila
Δ	: Es la funcion de transicion.
F	: Conjunto de estados de aceptación ($F \subseteq S$)

Capítulo 19

Máquina de Turing

Fue diseñada por el matematic ingles Alan M. Turing en 1936. consta de una cinta infinita, en los que podemos ubicar a la cadena a procesasr en cualquier celda la cual le da una capacidad ilimitada de almacenamiento.

Tinene un cabezal que es de lectura y escritura. Usa dos alfabetos, el alfabeto de entrada y el de la cinta.
En la función de transición se considera el desplazamiento.

Los autómatas finitos son menos potentes que los autómatas a pila con respecto a la capacidad de aceptar lenguajes. Las máquinas de Turing son más generales al poder aceptar lenguajes regulares como lenguajes libres de contexto u otros lenguajes.

19.1. Arquitectura de una MT

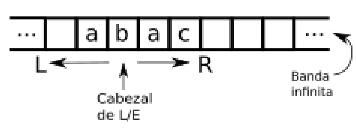

Figura 19.1: Máquina de Turing

Definición: Una máquina de Turing es una séptupla

$$M = (S, \Sigma, \Gamma, s, \not{b}, F, \delta)$$

S : Conjunto finito de estados
Σ : Alfabeto de entrada
Γ : Alfabeto de la cinto
s : Estado inicial $s \in S$
$\not b$: El símbolo blanco ($\not b \notin \Sigma$)
F : Conjunto de estados de aceptación ($F \subseteq S$)
δ : $S \times \Sigma \to S \times \Gamma \times \underset{\underset{\text{desplaza}}{\downarrow}}{D}$; $D = \{L, R\}$

1. El valor inicial de todas las celda es $\not b$.

2. Permitimos que $\Sigma \in \Gamma - \{\not b\}$.

3. δ transforma pares en ternas

$$(\underset{\underset{\text{(a)}}{\downarrow}}{p}, \underset{\underset{\text{(b)}}{\downarrow}}{\sigma}) \to (\underset{\underset{\text{(c)}}{\downarrow}}{q}, \underset{\underset{\text{(d)}}{\downarrow}}{t}, \underset{\underset{\text{(e)}}{\downarrow}}{X})$$

   (a)estado actual, (b)símbolo actual, (c)estado siguiente, (d)símbolo de la cinta, (e)desplazamiento

Ejemplo: La transición $\delta(q_i, a) = (q_s, b, R)$, provoca que la máquina de Turing pase de la configuración(Fig 19.2):

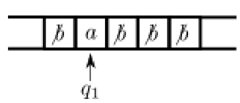

Figura 19.2: Configuración inicial

A la siguiente configuración(Fig 19.3):

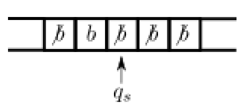

Figura 19.3: Configuración final

OBS: δ no necesariamente está definido para algún par de $S \times \Sigma$.

19.2. Etapas en el Proceso de una Cadena

Ejemplo: Sea la MT definida mediante:

$$
\begin{aligned}
S &= \{q_1, q_2\} \\
\Sigma &= \{a, b\} \\
\Gamma &= \{a, b, \not{b}\} \\
F &= \{q_2\} \\
s &= q_1
\end{aligned}
$$

δ está dada por

$$
\begin{aligned}
\delta(q_1, a) &= (q_1, a, R) \\
\delta(q_1, b) &= (q_1, a, R) \\
\delta(q_1, \not{b}) &= (q_2, \not{b}, L)
\end{aligned}
$$

Supongamos que se tiene $w = abba$. La configuracion seria(Fig 19.4):

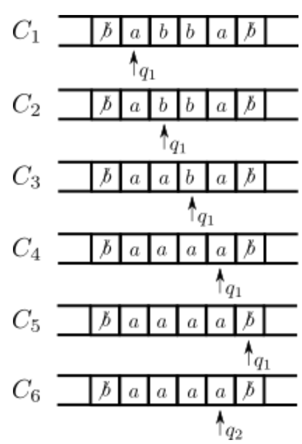

Figura 19.4: Llega a un estado de aceptación, por tanto se acepta w

19.3. Representación para la configuración de una MT

Podemos utilizar dos notaciones:

Notación 1: $(q_1, w_1\sigma w_2)$

q_1 : Estado actual

w_1 : Cadena que precede al símbolo apuntado por la cabeza de L/E

σ : Símbolo al que apunta el cabezal.

w_2 : Subcadena que está por procesar.

Configuración: Se tiene lo siguiente:

$$
\begin{array}{ll}
C_1 & (q_1, \not{b}\underline{a}bba) \\
C_2 & (q_1, a\underline{b}ba) \\
C_3 & (q_1, aa\underline{b}a) \\
C_4 & (q_1, aaa\underline{a}\ \not{b}) \\
C_5 & (q_1, aaaa\underline{\not{b}}\ \not{b}) \\
C_6 & (q_2, aaa\underline{a}\ \not{b})
\end{array}
$$

Notación 2: $a_1...a_{k-1}q_1a_ka_{k+1}...a_n$ es la representación de $(q_1, a_1...a_{k-1}a_k...a_n)$ donde q_1 es el estado actual.

Configuración:

$$
\begin{array}{ll}
C_1 & \not{b}q_1abba \\
C_2 & aq_1bba \\
C_3 & aaq_1ba \\
C_4 & aaaq_1a\ \not{b} \\
C_5 & aaaaq_1\ \not{b}\ \not{b} \\
C_6 & aaaq_2a\ \not{b}
\end{array}
$$

OBS: Se utiliza el símbolo $\vdash$ para representar el paso de una configuración a otra.

$$(q_1, \not{b}\underline{a}bba) \vdash (q_1, a\underline{b}ba) \vdash ... \vdash (q_2, aaa\underline{a}\ \not{b})$$

Recordar que:

$\vdash^*$	significa "0 a mas"
$\vdash^+$	significa "1 a mas"

Ejemplo: Dada la MT.

$$S = \{q_1, q_2, q_3\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, b, \not{b}\}$$
$$F = \{q_3\}$$
$$s = q_3$$

δ viene definida por:

$$\delta(q_1, a) = (q_1, a)L$$
$$\delta(q_1, b) = (q_1, b)L$$
$$\delta(q_1, \not{b}) = (q_2, \not{b})R$$
$$\delta(q_2, a) = (q_3, a)L$$
$$\delta(q_2, b) = (q_3, b)L$$
$$\delta(q_2, \not{b}) = (q_3, \not{b})L$$

Procesar $w = aababb$ mediante transiciones

$$(q_1, \underline{a}ababb) \vdash (q_1, \underline{\not{b}}aababb) \vdash (q_2, \not{b}\underline{a}ababb) \vdash (q_3, \underline{\not{b}}aababb)$$

Cuando $\delta(q, a)$ está indefinida y la configuración de la MT es (q, w_1qw_2) es imposible que pase a otra. Entonces se dice que la MT está parada.

La MT se parará siempre que llegue a un estado d aceptación.

Definición: La secuencia de todos los movimientos que conducen a una configuración parada se llama **computación**.

OBS: También es posible que la MT se mueva indefinidamente con la cabeza de L/E desplazándose de izquierda a derecha alternativamente.

Notación:

$$(q_1, w_1\sigma w_2) \vdash^* \infty$$
$$w_1q_1\sigma w_2 \vdash^* \infty$$

Significa que la MT nunca se detendrá.

19.4. Paso Computacional

El paso de una CI a otra a través de una transición por δ se llama paso computacional y se denota por:

$$u_1qu_2 \vdash v_1q_2v_2$$

19.5. Cómputos Especiales

Al procesar una cadena de entrada hay dos casos especiales.

1. El cómputo no termina porque entró a un bucle infinito. $u_1s_1u_2 \vdash^* \infty$.

2. El cómputo termina porque en determinado momento no hay una transición definida.

19.6. Lenguaje Aceptado por una MT

Una cadena $w \in \Sigma^*$ es aceptado por una MT si el cómputo que parte de una configuración inicia $s_0 w$ termina en una CI $\alpha_1 s \alpha_2$, $s \in F$ en la cual la MT se detiene por completo.

$$L(M) = \{w \in \Sigma^* / s_0 w \vdash^* \alpha_1 s \alpha_2, s \in F, \alpha_1 \alpha_2 \in \Gamma^*\}$$

Ejemplo: Dibuje la tabla de transición para una MT que obtenga el complemento binario de un número binario almacenado en la cinta.

Sea la MT definida por:

$$\Sigma = \{0, 1\}$$
$$\Gamma = \{0, 1, \not{b}\}$$
$$S = \{s_0, s_1\}$$
$$s = s_0$$
$$F = \{s_1\}$$

	0	1	$\not{b}$
s_0	$s_0 1 D$	$s_0 0 D$	$s_1 \ \not{b} I$

Bibliografía