

Apuntes de Lenguajes Formales

Incompleto

Leonardo H. Añez Vladimirovna¹

*Universidad Autónoma Gabriel René Moreno,
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones,
Santa Cruz de la Sierra, Bolivia*

18 de julio de 2019

¹Correo Electrónico: toborochi98@outlook.com

Agradecimiento a `marmot`

Notas del Autor

Estos apuntes fueron realizados durante mis clases en la materia INF319 (**Lenguajes Formales**), acompañados de referencias de libros, fuentes y código que use a lo largo del curso, en el período I-2019 en la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones.

Para cualquier cambio, observación y/o sugerencia pueden enviarme un mensaje al siguiente correo:

`toborochi98@outlook.com`

Índice general

1. Preliminares Formales	5
1.1. Conjuntos	5
1.1.1. Conjunto Finito e Infinito	5
1.2. Preliminares	5
1.2.1. Alfabeto	5
1.2.2. Palabra	5
1.2.3. Notaciones	6
1.2.4. Cantidad de Ocurrencias	6
1.2.5. Concatenación	6
1.2.6. Inversa	7
1.2.7. Potencia de una Palabra	7
1.2.8. Principio de Inducción para Σ^*	7
1.2.9. Lenguajes	8
1.2.10. Expresiones Regulares	8
1.2.11. Módulos	9
1.2.12. Máquinas	10
2. Autómatas	13
2.1. Autómata Finito Determinístico (AFD)	13
2.1.1. Definición	13
2.1.2. Interpretación	13
2.1.3. Representación	13
2.1.4. Configuración	13
2.2. Autómata Finito no Determinístico (AFN)	13
2.2.1. Definición	13
2.3. Equivalencia entre una AFD y un AFN	14
2.4. Propiedades de los Lenguajes Aceptados por AF's	15
2.5. Automatas Finitos y Expresiones Regulares	17
2.6. Lenguajes no Regulares	18
2.7. Gramáticas	18
2.7.1. Definición	18
2.7.2. Gramáticas Regulares	19
2.8. Autómata con Pila	19
2.8.1. Configuración	20
2.8.2. Autómatas con Pila y Gramáticas Libre de Contexto	20
2.8.3. Propiedades de los Lenguajes Libres de Contexto	20
2.9. Máquinas de Turing	21

Capítulo 1

Preliminares Formales

1.1. Conjuntos

1.1.1. Conjunto Finito e Infinito

Equivalencia

Dado A y B (conjuntos) los llamamos *equivalentes* si existe una biyección: $f : A \rightarrow B$

Conjunto Finito

Un conjunto A es finito si es equivalente a $\{1, 2, 3, \dots, n\}$ para algún $n \in \mathbb{N}$.

Conjunto Infinito

Un conjunto es infinito si no es finito. Si no es equivalente a $\{1, 2, 3, \dots, n\}$ es decir no hay biyección. Sin embargo no todos los conjuntos finitos son equivalentes.

- **Conjunto Contablemente Infinito:** Se dice que un conjunto es contablemente infinito si es equivalente con \mathbb{N} .
- **Conjunto Contable:** Es contable si es finito o contablemente infinito.
- **Conjunto Incontable:** Se dice que es incontable si no es contable.

Principio de las Casillas

Si A y B son conjuntos finitos no vacíos y $|A| > |B|$ entonces no existe una función inyectiva de: $A \rightarrow B$.

1.2. Preliminares

1.2.1. Alfabeto

Un alfabeto Σ es cualquier conjunto finito no vacío.

Ejemplo(s)

$$\begin{aligned}\Sigma_1 &= \{Leo, Martha\} \\ \Sigma_2 &= \{0, 1, 2, 3, \dots, 13\} \\ \Sigma_3 &= \{a, b\} \\ \Sigma_4 &= \{R, G, B, A\}\end{aligned}$$

1.2.2. Palabra

Una palabra sobre Σ es una sucesión finita de símbolos de Σ . Es decir:

$$(\sigma_1, \sigma_2, \dots, \sigma_n); \sigma \in \Sigma \quad \text{ó} \quad \sigma_1 \sigma_2 \sigma_3 \dots \sigma_n; \sigma \in \Sigma$$

Ejemplo(s)

Sobre Σ_1	Sobre Σ_2	Sobre Σ_3	Sobre Σ_4
$w_1 = \text{LeoLeo}$	$w_1 = 1111110$	$w_1 = \text{bababababa}$	$w_1 = \text{ABGR}$
$w_2 = \text{MarthaLeoMartha}$	$w_2 = 11235813$	$w_2 = \text{abba}$	$w_2 = \text{RRRA}$

Denotamos por Σ^* el conjunto de todas las palabras sobre Σ .

Longitud de una Palabra

Sea w una palabra sobre Σ , es decir $w = \sigma_1\sigma_2 \dots \sigma_n; \sigma \in \Sigma$. La longitud de w es n y se denota por: $|w| = n$.

Palabra vacía

Es la sucesión vacía de símbolos de Σ y se denota por: λ .

1.2.3. Notaciones

- $\Sigma^+ = \{w \in \Sigma^* / |w| > 0\}$
- $\Sigma^k = \{w \in \Sigma^* / |w| = k\}$
- $\Sigma^0 = \{w \in \Sigma^* / |w| = 0\} = \{\lambda\}$
- $\Sigma^1 = \{w \in \Sigma^* / |w| = 1\} = \Sigma$
- $\Sigma^* = \Sigma^+ \cup \{\lambda\}$
- $\Sigma^+ = \Sigma^* - \{\lambda\}$

1.2.4. Cantidad de Ocurrencias

Sea $w \in \Sigma^*$, denotamos por $|w|_\sigma$ al número de ocurrencias del símbolo σ en la palabra w .

Ejemplo(s)

$$\Sigma = \{a, b\}$$

- $\Sigma^* = \{\lambda, a, b, aa, bb, ab, ba, aaa, \dots\}$
- $\Sigma^0 = \{\lambda\}$
- $\Sigma_1 = \Sigma = \{a, b\}$
- $\Sigma^2 = \{ab, aa, ba, bb\}$

1.2.5. Concatenación

Sea $u, v \in \Sigma^*$ tal que $u = \sigma_1\sigma_2 \dots \sigma_n, v = \epsilon_1\epsilon_2 \dots \epsilon_n$. La concatenación de u y v se define por:

$$uv = \sigma_1\sigma_2 \dots \sigma_n\epsilon_1\epsilon_2 \dots \epsilon_n$$

Definición de Recurrencia

$$| \cdot | : \Sigma^* \rightarrow \mathbb{N}$$

$$\begin{cases} |\lambda| = 0 \\ |wa| = |w| + 1 \end{cases}$$

Ejemplo(s)

$$\begin{aligned} u &= abab \\ v &= bba \end{aligned}$$

$$\begin{aligned} uv &= ababbba \\ vu &= bbaabab \end{aligned}$$

Propiedades

- $uv \neq vu$
- $(uv)w = u(vw)$
- $u\lambda = \lambda u = u$
- $|uv| = |u| + |v|$
- $|uv|_a = |u|_a + |v|_a$

1.2.6. Inversa

Si $w = \sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma^n$ entonces $w' = \sigma_n, \sigma_{n-1}, \dots, \sigma_1$ se llama inversa o transpuesta de w .

Definición de Recurrencia

$$\begin{aligned} ' : \Sigma^* &\rightarrow \Sigma^* \\ \begin{cases} \lambda' = \lambda \\ (wa)' = aw' \end{cases} \end{aligned}$$

1.2.7. Potencia de una Palabra

$$w^n = \underbrace{ww \dots w}_{n\text{-veces}}$$

Definición de Recurrencia

$$\begin{aligned} ' : \Sigma^* &\rightarrow \Sigma^* \\ \begin{cases} w^0 = \lambda \\ w^{n+1} = ww^n \end{cases} \end{aligned}$$

Propiedades

- $|w^n| = n|w|$
- $w^m w^n = w^{m+n}$
- $(w^n)^m = w^{mn}$
- $\lambda^n = \lambda$

1.2.8. Principio de Inducción para Σ^*

Sea L un conjunto de palabras sobre Σ con las propiedades:

- i.) $\lambda \in L$
- ii.) $w \in L \wedge a \in \Sigma \Rightarrow wa \in L$

Entonces

$L = \Sigma^*$, (es decir, todas las palabras sobre Σ están en L .)

1.2.9. Lenguajes

Un lenguaje sobre Σ es un subconjunto de Σ^*

Operaciones

Recordemos que ya conocemos otras operaciones (Unión, Intersección, Diferencia y Complemento), para esta materia tenemos las siguientes:

- Concatenación

Sea $A, B \subseteq \Sigma^*$

$$AB = \{w \in \Sigma^* / w = xy, x \in A, y \in B\}$$

- Transposición

Sea $A \subseteq \Sigma^*$

$$A' = \{w' \in \Sigma^* / w \in A\}$$

- Estrella de Kleene

Sea $A \subseteq \Sigma^*$

$$A^* = \{w \in \Sigma^* / w = w_1 w_2 \dots w_n \text{ para algún } k \in \mathbb{N} \text{ y para algunas } w_1, w_2, \dots, w_k \in A\}$$

1.2.10. Expresiones Regulares

Las expresiones regulares (ER) sobre un alfabeto (Σ) son las palabras sobre el alfabeto $\Sigma \cup \{\}, (, \emptyset, \cup, *\}$ tal que cumple lo siguiente:

- 1.) \emptyset y cada símbolo de Σ es una ER.
- 2.) Si α y β son ER entonces $(\alpha\beta)$ es una ER.
- 3.) Si α y β son ER entonces $(\alpha \cup \beta)$ es una ER.
- 4.) Si α es una ER entonces α^* es una ER.
- 5.) Nada mas es una ER a menos que provenga de (1.) a (4.)

Ejemplo(s)

Para $\Sigma = \{a, b\}$ podemos formar:

$$\begin{aligned} &\emptyset \\ &a \\ &b \\ &(ab) \\ &(a \cup b) \\ &((ab) \cup b) \\ &(ba)^* \\ &((ba)^* \cup (a \cup b))^* \end{aligned}$$

Lenguaje Regular

Un lenguaje es regular ssi es generado por una expresión regular.

1.2.11. Módulos

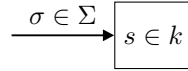
Definición

Un módulo es una tripleta $D = (k, \Sigma, f)$ donde:

- k es un conjunto finito no vacío, llamado *conjunto de estados*
- Σ es un conjunto finito no vacío, llamado *alfabeto*
- $f : k \times \Sigma \rightarrow k$, llamado *función de transición*

Interpretación

Un módulo se puede interpretar como un dispositivo que en determinados instantes de tiempo recibe señales (símbolos del alfabeto), que producen cambios en su configuración interna.



Representación

- **Tabla de Transición**

$\Sigma \backslash k$	σ_1	σ_1	\cdots	σ_j	\cdots	σ_m
s_1				\vdots		
s_2				\vdots		
\vdots				\vdots		
s_i	\cdots	\cdots	\cdots	s_k		
\vdots						
s_n						

Figura 1.1: $s_k = f(s_i, \sigma_j)$

- **Grafo**

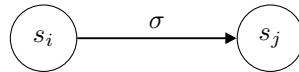
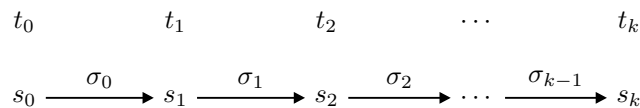


Figura 1.2: $s_j = f(s_i, \sigma)$

Comportamiento Dinámico

Sea $D = (k, \sigma, f)$ un módulo:



Función Estado Terminal

Sea $D = (k, \sigma, f)$ un módulo:

Una función de Estado Terminal del módulo D es una única función:

$$\hat{f} : k \times \Sigma \rightarrow k \text{ tal que } \forall s \in k, w \in \Sigma^*, \sigma \in \Sigma$$

$$\begin{cases} \widehat{f}(s, \lambda) = s \\ \widehat{f}(s, \sigma w) = \widehat{f}[f(s, \sigma), w] \end{cases}$$

◊ **Notas**

- $w = \lambda$

$$\widehat{f}(s, \sigma) = \widehat{f}(s, \sigma \lambda) = \widehat{f}[f(s, \sigma), \lambda] = \widehat{f}(s, \sigma)$$

- $\forall w \in \Sigma^*$

$$f : k \rightarrow k \text{ tal que: } f_w(s) = \widehat{f}(s, w)s$$

1.2.12. Máquinas

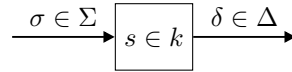
Definición

Una máquina es una quintupla $M = (k, \Sigma, \Delta, f, g)$ donde:

- k es un conjunto finito no vacío, llamado *conjunto de estados*
- Σ es un conjunto finito no vacío, llamado *alfabeto de entrada*
- Δ es un conjunto finito no vacío, llamado *alfabeto de salida*
- $f : k \times \Sigma \rightarrow k$, llamado *función de transición*
- $g : k \times \Sigma \rightarrow \Delta$, llamado *función de salida*

Interpretación

Una máquina se puede interpretar como un dispositivo que en determinados instantes de tiempo recibe señales (símbolos de entrada) que producen cambios en su configuración interna y emiten señales (símbolos de salida).



Representación

- **Tabla de Transición**

Σ	σ_1	σ_1	\cdots	σ_j	\cdots	σ_m
k						
s_1				\vdots		
s_2				\vdots		
\vdots				\vdots		
s_i	\cdots	\cdots	\cdots	s_k/δ_k		
\vdots						
s_n						

Figura 1.3: $s_k = f(s_i, \sigma_j)$ $g(s_i, \sigma_j) = \delta_k$

- **Grafo**

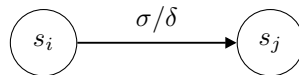


Figura 1.4: $s_j = f(s_i, \sigma)$ $g(s_i, \sigma) = \delta$

Comportamiento Dinámico

Sea $M = (k, \Sigma, \Delta, f, g)$ una máquina:

$$\begin{array}{ccccccc} t_0 & & t_1 & & t_2 & & \cdots & & t_k \\ s_0 & \xrightarrow{\sigma_0/\delta_0} & s_1 & \xrightarrow{\sigma_1/\delta_1} & s_2 & \xrightarrow{\sigma_2/\delta_2} & \cdots & \xrightarrow{\sigma_{k-1}/\delta_{k-1}} & s_k \end{array}$$

Función Estado Terminal

Sea $M = (k, \Sigma, \Delta, f, g)$ una máquina.

- 1.) Una función de Estado Terminal de M es la función de estado terminal:

$$\widehat{f} : k \times \Sigma^* \rightarrow k \text{ del módulo } (k, \Sigma, f)$$

- 2.) Una función palabra de salida de M es una única función:

$$\bar{g} : k \times \Sigma^* \rightarrow \Delta^* \text{ tal que } \forall s \in k, \sigma \in \Sigma, w \in \Sigma^*$$

$$\begin{cases} \bar{g}(s, \lambda) &= \lambda \\ \bar{g}(s, \sigma w) &= g(s, \sigma) \bar{g}[f(s, \sigma), w] \end{cases}$$

◊ Notas

- $w = \lambda$

$$\bar{g}(s, \sigma) = \bar{g}(s, \sigma \lambda) = g(s, \sigma) \underbrace{\bar{g}[f(s, \sigma), \lambda]}_{\lambda} = g(s, \sigma) \lambda = \bar{g}(s, \sigma)$$

- $\forall s \in k$

$$g_s : \Sigma^* \rightarrow \Delta^* \text{ tal que } g_s(w) = \bar{g}(s, w)$$

Capítulo 2

Autómatas

2.1. Autómata Finito Determinístico (AFD)

2.1.1. Definición

Un Autómata Finito Determinístico (AFD) es una quintupla $M = (k, \Sigma, f, s_0, F)$ donde:

- k conjunto finito no vacío, *conjunto de estados*
- Σ conjunto finito no vacío, *Alfabeto*
- $f : k \times \Sigma \rightarrow k$, *Funcion de transicion*
- $s_0 \in k$, *Estado inicial*
- $F \subseteq k$, *Conjunto de estados finales*

2.1.2. Intepretación

2.1.3. Representación

2.1.4. Configuración

Sea $M = (k, \Sigma, \delta, s_0, F)$ un AFD.

Una configuración de M es un elemento de $k \times \Sigma^*$

Relación \vdash_M

Sea (q, w) y (q', w') dos configuraciones¹:

$$(q, w) \vdash_M (q', w') \Leftrightarrow w = \sigma w' \text{ para algun } \sigma \in \Sigma \text{ y } \delta(q, \sigma) = q'$$

Lenguaje Aceptado por M

$$L(M) = \{w \in \Sigma^* / M \text{ acepta } w\}$$

$$L(M) = \{w \in \Sigma^* / (s, w) \vdash_M^* (q, \lambda) \wedge q \in F\}$$

2.2. Autómata Finito no Determinístico (AFN)

2.2.1. Definición

Un autómata Finito no Deterministico (AFN) es una quintupla $M = (k, \Sigma, \Delta, s, F)$ donde:

- k : conjunto finito no vacío
- Σ : conjunto finito no vacío

¹ \vdash_M se lee "conduce a" en un paso.

- Δ : es un subconjunto finito de $k \times \Sigma^* \times k$
- $s \in k$
- $F \subseteq k$

2.3. Equivalencia entre una AFD y un AFN

Teorema

Para cada AFN existe un AFD equivalente.

★ Prueba

Sea $M = (k, \Sigma, \Delta, s, F)$ un AFN

i.) Construimos $M' = (k', \Sigma, \Delta', s', F')$ eliminando todas las aristas de M que:

$$(q, u, q') \in \Delta \quad \wedge \quad |u| > 1$$

Si $u = \sigma_1 \sigma_2 \dots \sigma_k, k > 1$ entonces añadimos p_1, p_2, \dots, p_{k-1} estados y las nuevas transiciones:

$$(q, \sigma_1, p_1), (p_1, \sigma_2, p_2), \dots, (p_{k-1}, \sigma_k, q')$$

a Δ para u tal que $|u| > 1$.

ii.) Construimos $M'' = (k'', \Sigma, \delta'', s'', F'')$

La idea clave es considerar que un AFN en un determinado instante se encuentra en un conjunto de estados:

- $k'' = \Sigma^{k'}$
- $F'' = \{Q \subseteq k' / Q \cap F' \neq \emptyset\}$

Formalmente:

$$E(q) = \{p \in k' / (q, \lambda) \vdash_{M'}^* (p, \lambda)\}$$

Equivalentemente:

$$E(q) = \{p \in k' / (q, w) \vdash_{M'}^* (p, w)\}$$

Donde:

- $s'' = E(s')$
- $\forall Q \subseteq k' \wedge$ para cada símbolo $\sigma \in \Sigma$

Ademas:

$$\delta''(Q, \sigma) = \cup \{E(p) : p \in k' \wedge (q, \sigma, p) \in \Delta', \exists q \in Q\}$$

Afirmamos que $\forall w \in \Sigma^*$ y $\forall p, q \in k'$:

$$(q, w) \vdash_{M'}^* (p, \lambda) \Leftrightarrow (E(q), w) \vdash_{M''}^* (P, \lambda)$$

p.d. $M' \approx M''$

p.d. $L(M') = L(M'')$

$$\begin{aligned} w \in L(M') &\Leftrightarrow (s', w) \vdash_{M'}^* (q, \lambda), q \in F' \\ &\Leftrightarrow (E(s'), w) \vdash_{M''}^* (Q, \lambda) \\ &\Leftrightarrow (s'', w) \vdash_{M''}^* (Q, \lambda), Q \in F'' \\ &\Leftrightarrow w \in L(M'') \end{aligned}$$

$$\therefore L(M') = L(M'')$$

2.4. Propiedades de los Lenguajes Aceptados por AF's

Teorema

La clase de Lenguajes aceptados por AF's es cerrada bajo la:

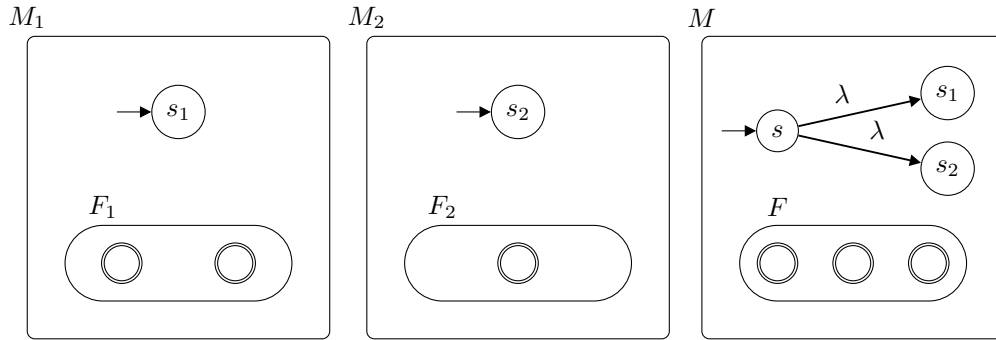
- Unión
- Concatenación
- Estrella de Kleene
- Complementación
- Intersección

Prueba

Sean $L(M_1)$ y $L(M_2)$ lenguajes aceptados por $M_1 = (k_1, \Sigma, \Delta_1, s_1, F_1)$ y $M_2 = (k_2, \Sigma, \Delta_2, s_2, F_2)$:

a) Unión

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1) \cup L(M_2)$

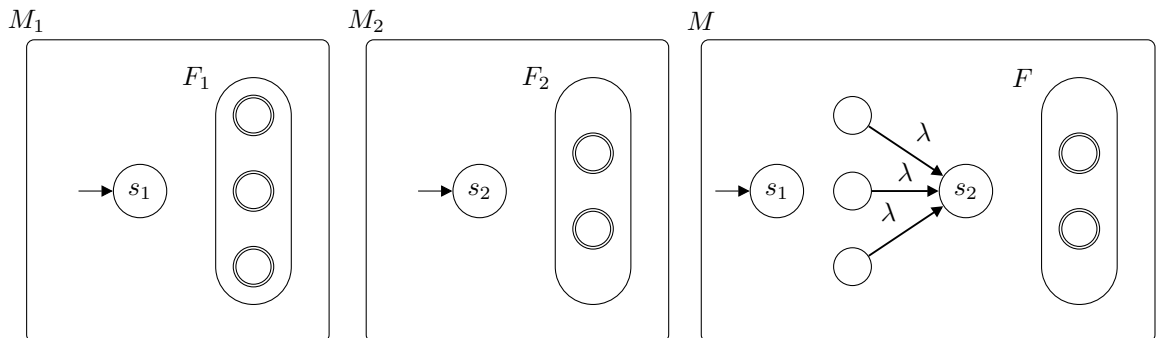


Donde:

- $k = k_1 \cup k_2 \cup \{s\}$
donde s es un nuevo estado (inicial).
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \lambda, s_1), (s, \lambda, s_2)\}$
- s : nuevo estado añadido
- $F = F_1 \cup F_2$

b) Concatenación

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1)L(M_2)$



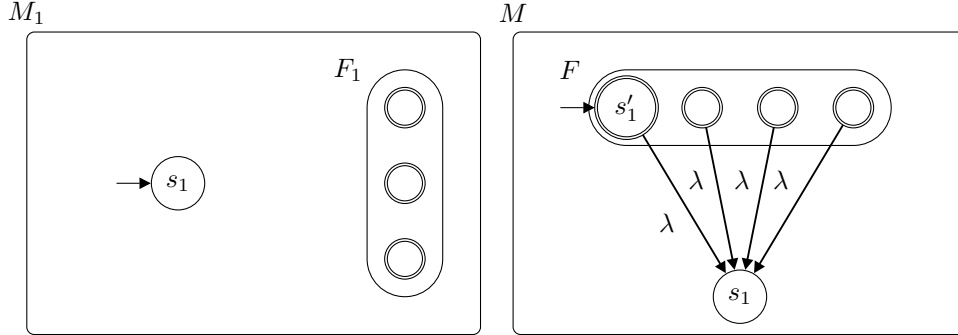
Donde:

- $k = k_1 \cup k_2$
- $\Delta = \Delta_1 \cup \Delta_2 \cup (F_1 \times \{\lambda\} \times \{s_2\})$

- $s = s_1$
- $F = F_2$

c) **Estrella de Kleene**

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1)^*$



Donde:

- $k = k_1 \cup \{s'_1\}$
donde s'_1 es un nuevo estado (*inicial y terminal*).
- $\Delta = \Delta_1 \cup (F \times \{\lambda\} \times \{s_1\})$
- $s = s'_1$
- $F = F_1 \cup \{s'_1\}$

d) **Complementación**

Sea $M = (k, \Sigma, \delta, s, F)$ un **AFD**.

Donde:

$\Sigma^* - L(M)$ es aceptado por $\overline{M} = (k, \Sigma, \delta, s, k - F)$

e) **Intersección**

Sea:

$$\begin{aligned}
 L_1 \cap L_2 &= \overline{\overline{L_1} \cap \overline{L_2}} \\
 &= \overline{\overline{L_1} \cup \overline{L_2}} \\
 &= \Sigma^* - [\underbrace{(\Sigma^* - L_1)}_1 \cup \underbrace{(\Sigma^* - L_2)}_2] \\
 &= \underbrace{\Sigma^* - [\underbrace{(\Sigma^* - L_1) \cup (\Sigma^* - L_2)}_3]}_4
 \end{aligned}$$

Notese que en 1, 2, 3, 4 se puede ver que en cada operación al aplicarla se obtiene otro AF que puede ser nuevamente usada en la siguiente operación y así sucesivamente.

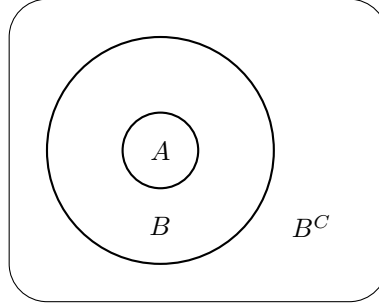
Teorema

Existen algoritmos para responder las siguientes preguntas acerca de Automatas Finitos:

- a) Dado un AF M y una palabra w , ¿ $w \in L(M)$?
- b) Dado un AF M , es ¿ $L(M) = \emptyset$?
- c) Dado un AF M , es ¿ $L(M) = \Sigma^*$?
- d) Dado 2 AF's M_1 y M_2 , es ¿ $L(M_1) \subseteq L(M_2)$?
- e) Dado 2 AF's M_1 y M_2 , es ¿ $L(M_1) = L(M_2)$?

Prueba

- a) Esto es lo que ya hemos ido haciendo.
- b) Simplemente nos basta analizar si el automata no acepta ninguna palabra.
- c) $M \rightarrow L(\bar{M}) = \emptyset$ (Basta realizar el complemento del AF y ver que el lenguaje aceptado sea \emptyset). **(b)**
- d) $A \subseteq B \Leftrightarrow A \cap B^C = \emptyset$



$$L(M_1) \subseteq L(M_2) \Leftrightarrow L(M_1) \cap [\Sigma^* - L(M_2)] = \emptyset$$

- e) Realizamos la doble inclusión:

$$L(M_1) \subseteq L(M_2) \wedge L(M_2) \subseteq L(M_1) \text{ (d)}$$

2.5. Automatas Finitos y Expresiones Regulares**Teorema**

Un Lenguaje es regular ssi es aceptado por un Automata Finito.

Prueba

- i.) Supongamos que la propiedad se cumple para γ tal que $|\gamma| < n$.

Para $|\gamma| = n$

- $\gamma = \alpha \cup \beta$

$$\begin{aligned} L(\gamma) &= L(\alpha \cup \beta) = L(\alpha) \cup L(\beta) \\ &= L(M_1) \cup L(M_2) = L(M) \end{aligned}$$

- $\gamma = \alpha\beta$

$$\begin{aligned} L(\gamma) &= L(\alpha\beta) = L(\alpha)L(\beta) \\ &= L(M_1)L(M_2) = L(M) \end{aligned}$$

- $\gamma = \alpha^*$

$$\begin{aligned} L(\gamma) &= L(\alpha^*) = L(\alpha)^* \\ &= L(M_1)^* = L(M) \end{aligned}$$

- ii.) Si un lenguaje es aceptado por un AF entonces es un lenguaje regular.

Prueba

$$R = L(M)$$

Representaremos $L(M)$ como la unión de muchos (pero en número finito) lenguajes simples.

Sea $k = \{q_1, q_2, q_3, \dots, q_n\}$, $s = q_1$ para $i, j = 1, 2, \dots, n$; $k = 1, 2, \dots, n + 1$

Definimos $R(i, j, k)$ como el conjunto de todas las palabras que nos llevan de q_i a q_j sin pasar por estados con subíndice k o mayores.

Formalmente

$$R(i, j, k) = \{x \in \Sigma^* / (q_i, x) \vdash_M^* (q_j, \lambda) \wedge (q_i, x) \vdash_M^* (q_l, y) \text{ entonces } l < k \vee (y = \lambda \wedge l = j) \vee (y = x \wedge l = i)\}$$

Si $k = n + 1$:²

$$R(i, j, n + 1) = \{x \in \Sigma^* / (q_i, x) \vdash_M^* (q_j, \lambda)\}$$

Esto es:

$$L(M) = \cup \{R(1, j, n + 1), q_j \in F\}$$

p.d. $R(i, j, k)$ son regulares (inducción) $k = 1$

$$R(i, j, 1) = \begin{cases} \{\sigma \in \Sigma / \delta(q_i, \sigma) = q_j\}, i \neq j \\ \{\lambda\} \cup \{\sigma \in \Sigma / \delta(q_i, \sigma) = q_j\}, i = j \end{cases}$$

Son regulares ya que $R(i, j, 1)$ son **finitos**.

HI

Para $k = 1, 2, \dots, n$ $R(i, j, k)$ son Regulares.

p.d. Para $k + 1$ es decir $R(i, j, k + 1)$ es regular:

2.6. Lenguajes no Regulares**2.7. Gramáticas****2.7.1. Definición**

Una gramática Libre de Contexto es una cuádrupla $G = (V, \Sigma, R, S)$ donde:

- G es un alfabeto.
- Σ es un subconjunto de V (Conjunto de Símbolos Terminales)
- $S \in (V - \Sigma)$ (Símbolo Inicial)
- R es un conjunto finito de $(V - \Sigma) \times V^*$ (Reglas)

Notación

Para cada $A \in V - \Sigma$, $w \in \Sigma^*$

$$(A, w) \in R \Leftrightarrow A \xrightarrow[G]{} w$$

Donde G se refiere a la gramática.

Derivación

Para cada par de palabras $u, v \in V^*$ escribimos $u \xRightarrow[G]{} v$ ssi existen palabras $x, y, v' \in V^*$, $A \in (V - \Sigma)$ tales que:

$$u = xAy, \quad v = xv'y, \quad A \xrightarrow[G]{} v'$$

Se denota por $u \xRightarrow[G]^* v$ la clausura reflexiva transitiva de $\xRightarrow[G]{} v$

Lenguaje Generado por G

$$L(G) = \left\{ w \in \Sigma^* / S \xRightarrow[G]^* w \right\}$$

Un lenguaje es libre de contexto ssi existe una gramática libre de contexto que genera el Lenguaje.

²Notese que no es posible llegar a $n + 1$ ya que k tiene n estados

2.7.2. Gramáticas Regulares

Definición

Una Gramática Libre de Contexto $G = (V, \Sigma, R, S)$ es regular ssi:

$$R \subseteq (V - \Sigma) \times \Sigma^*((V - \Sigma) \cup \{\lambda\})$$

Teorema

Un lenguaje es Regular ssi es generada por una Gramática Regular:

Prueba

\Rightarrow

Sea $M = (K, \Sigma, \sigma, s, F)$ construimos $G = (V, \Sigma, R, S)$ donde:

- $V = K \cup \Sigma$
- $S = s$
- $R = \{q \rightarrow ap : \delta(p, a) = p\} \cup \{q \rightarrow \lambda : q \in F\}$

\Leftarrow

Sea $G = (V, \Sigma, R, S)$ construimos $M = (K, \Sigma, \Delta, s, F)$ donde:

- $K = (V - \Sigma) \cup \{f\}$ donde f es un nuevo elemento.
- $s = S$
- $F = \{f\}$
- $\Delta = \{(A, w, B); A \rightarrow wB, A, B \in (V - \Sigma), w \in \Sigma^*\} \cup \{(A, w, f); A \rightarrow w, A \in (V - \Sigma), w \in \Sigma^*\}$

2.8. Autómata con Pila

Definición

Un autómata con Pila es una sextupla:

$$M = (K, \Sigma, \Gamma, \Delta, s, F)$$

donde :

- $k :$
- Σ : Símbolos de Entrada
- Γ : es un conjunto finito no vacío (*llamado alfabeto de la Pila*).
- $s \in K$
- $F \subseteq K$
- Δ es un subconjunto finito de $(K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$ (Relación de Transición)

Significado de una Transición

$$((p, u, \beta), (q, \gamma)) \in \Delta$$

"Estando en p leyendo w pasa al estado q y reemplaza β por γ ."

Empujar a

$$((p, u, \lambda), (q, \alpha)) \in \Delta$$

Sacar de

$$((p, u, \alpha), (q, \lambda)) \in \Delta$$

2.8.1. Configuración

Una configuración es un elemento de:

$$K \times \Sigma^* \times \Gamma^*$$

La relación \vdash_M

Para cada $((p, u, \beta), (q, \gamma)) \in \Delta$ y para cada $x \in \Sigma^*, \alpha \in \Gamma^*$ definimos:

$$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha)$$

Para ilustrar un poco mejor la idea tenemos la siguiente gráfica:

$$\begin{array}{ccc} p : \boxed{\text{u} \mid x} & & q : \boxed{\phantom{\text{u}} \mid x} \\ \\ \begin{array}{|c|} \hline \\ \hline \beta \\ \hline \alpha \\ \hline \end{array} & \vdash_M & \begin{array}{|c|} \hline \\ \hline \gamma \\ \hline \alpha \\ \hline \end{array} \\ (p, ux, \beta\alpha) & & (q, x, \gamma\alpha) \end{array}$$

Palabra aceptada por M

Sea $w \in \Sigma^*$:

$$M \text{ acepta } w \Leftrightarrow (s, w, \lambda) \vdash_M^* (q, \lambda, \lambda); q \in F$$

Lenguaje aceptado por M

$$L(M) = \{w \in \Sigma^* / (s, w, \lambda) \vdash_M^* (q, \lambda, \lambda); q \in F\}$$

2.8.2. Autómatas con Pila y Gramáticas Libre de Contexto

Definición

La clase de lenguajes aceptados por autómatas con pila es exactamente la clase de lenguajes libres de contexto:

Prueba \Leftarrow

Sea $G = (V, \Sigma, R, S)$ construimos $M = (K, \Sigma, \Gamma, \Delta, s, F)$ tal que $L(M) = L(G)$ donde:

$$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$$

Transiciones

$$((p, \lambda, \lambda), (q, S)) \tag{2.1}$$

$$((q, \lambda, A), (q, x)); \forall A \rightarrow x \in R \tag{2.2}$$

$$((q, a, a), (q, \lambda)); \forall a \in \Sigma \tag{2.3}$$

2.8.3. Propiedades de los Lenguajes Libres de Contexto

Teorema

Los lenguajes libres de contexto son cerrados bajo la unión, concatenación y estrella de Kleene.

Prueba

Sean $G_i = (V_i, \Sigma_i, R_i, S_i)$ $i = 1, 2$ las Gramáticas Libres de Contexto y sin pérdida de generalidad asumimos que $(V_1 - \Sigma_1)$ y $(V_2 - \Sigma_2)$ son disjuntos.

■ **Unión**

S es un nuevo símbolo

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$$

■ **Concatenación**

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}, S)$$

■ **Estrella de Kleene**

$$G = (V_1, \Sigma_1, R_1 \cup \{S_1 \rightarrow \lambda, S_1 \rightarrow S_1 S_1\}, S_1)$$

Teorema

La intersección de un L.L.C con un lenguaje regular es un lenguaje libre de Contexto.

Teorema de Pumping

Sea G una G.L.C. entonces existe un número k que depende de G tal que cualquier palabra $w \in L(G)$ de longitud $> k$, se puede reescribir $w = uvxyz$ de tal manera que ya sea v o y es no vacía y $uv^nxy^n z \in L(G)$. Para cada $n \geq 0$.

2.9. Máquinas de Turing

Definición

Una máquina de Turing es una cuádrupla $M = (K, \Sigma, \delta, s)$ donde:

- K : Conjunto Finito no Vacío , $h \notin K$
- Σ : Conjunto Finito no Vacío , $\# \in \Sigma, L, R \notin \Sigma$
- $s \in K$
- $\sigma : K \times \Sigma \rightarrow (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$

- i.) Si $b \in \Sigma$ la máquina reescribe b por a .
- ii.) Si $b = L$ la cabeza lectora se mueve a la izquierda.
- iii.) Si $b = R$ la cabeza lectora se mueve a la derecha.

Si la máquina esta en h se dice que la máquina se *detiene*.

Teorema

Los siguientes modelos de máquinas de Turing son igualmente poderosos:

- La Máquina de Turing Estándar.
- La Máquina de Turing de varias cabezas.
- La Máquina de Turing de varias pistas.
- La Máquina de Turing de varias cintas.
- La Máquina de Turing Bidimensional.
- Automata con Pila con 2 Pilas
- IBM Mainframe