# Apuntes de Lenguajes Formales

Leonardo H. Añez Vladimirovna[1]

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

12 de junio de 2019

[1]Correo 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

# 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: $f : A \to B$

**Conjunto Finito**

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

**Conjunto Infinito**

Un conjunto es infinito si no es finito. Si no es equivalente a $\{1, 2, 3, \ldots, n\}$ 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: $A \to B$.

## 1.2. Preliminares

### 1.2.1. Alfabeto

Un alfabeto $\Sigma$ es cualquier conjunto finito no vacío.

**Ejemplo(s)**

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

### 1.2.2. Palabra

Una palabra sobre $\Sigma$ es una sucesión finita de símbolos de $\Sigma$. Es decir:

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

**Ejemplo(s)**

| **Sobre** $\Sigma_1$ | **Sobre** $\Sigma_2$ | **Sobre** $\Sigma_3$ | **Sobre** $\Sigma_4$ |
|---|---|---|---|
| $w_1 = LeoLeo$ | $w_1 = 1111110$ | $w_1 = bababababa$ | $w_1 = ABGR$ |
| $w_2 = MarthaLeoMartha$ | $w_2 = 11235813$ | $w_2 = abba$ | $w_2 = RRRA$ |

Denotamos por $\Sigma^*$ el conjunto de todas las palabras sobre $\Sigma$.

**Longitud de una Palabra**

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

**Palabra vacía**

Es la sucesión vacía de símbolos de $\Sigma$ y se denota por: $\lambda$.

### 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 $\sigma$ en la palabra $w$.

**Ejemplo(s)**

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

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

### 1.2.5. Concatenación

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

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

**Definición de Recurrencia**

$$|\ | : \Sigma^* \to \mathbb{N}$$

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

**Ejemplo(s)**

$$u = abab \qquad uv = ababbba$$
$$v = bba \qquad vu = bbaabab$$

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

**Definición de Recurrencia**

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

### 1.2.7. Potencia de una Palabra

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

**Definición de Recurrencia**

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

**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 $\Sigma^*$

Sea $L$ un conjunto de palabras sobre $\Sigma$ 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 $\Sigma$ están en $L$.)

### 1.2.9. Lenguajes

Un lenguaje sobre $\Sigma$ es un subconjunto de $\Sigma^*$

#### 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 ($\Sigma$) son las palabras sobre el alfabeto $\Sigma \cup \{), (, \emptyset, \cup, *\}$ tal que cumple lo siguiente:

**1.)** $\emptyset$ y cada símbolo de $\Sigma$ es una ER.

**2.)** Si $\alpha$ y $\beta$ son ER entonces $(\alpha\beta)$ es una ER.

**3.)** Si $\alpha$ y $\beta$ son ER entonces $(\alpha \cup \beta)$ es una ER.

**4.)** Si $\alpha$ es una ER entonces $\alpha^*$ 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:

$$\emptyset$$
$$a$$
$$b$$
$$(ab)$$
$$(a \cup b)$$
$$((ab) \cup b)$$
$$(ba)^*$$
$$((ba)^* \cup (a \cup b))^*$$

#### Lenguaje Regular

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

### 1.2.11. Módulos

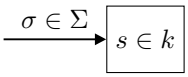
**Definición**

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

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

| $k$ \ $\Sigma$ | $\sigma_1$ | $\sigma_1$ | $\cdots$ | $\sigma_j$ | $\cdots$ | $\sigma_m$ |
|---|---|---|---|---|---|---|
| $s_1$ | | | | $\vdots$ | | |
| $s_2$ | | | | $\vdots$ | | |
| $\vdots$ | | | | $\vdots$ | | |
| $s_i$ | $\cdots$ | $\cdots$ | $\cdots$ | $s_k$ | | |
| $\vdots$ | | | | | | |
| $s_n$ | | | | | | |

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

- **Grafo**

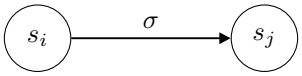
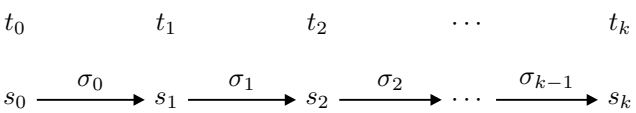



Figura 1.2: ssi: $f(s_i, \sigma) = s_j$

**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:

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

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

◇ **Notas**

- $w = \lambda$

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

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

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

### 1.2.12. Máquinas

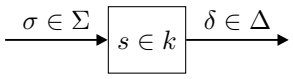
**Definición**

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

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

| $k$ \ $\Sigma$ | $\sigma_1$ | $\sigma_1$ | $\cdots$ | $\sigma_j$ | $\cdots$ | $\sigma_m$ |
|---|---|---|---|---|---|---|
| $s_1$ | | | | $\vdots$ | | |
| $s_2$ | | | | $\vdots$ | | |
| $\vdots$ | | | | $\vdots$ | | |
| $s_i$ | $\cdots$ | $\cdots$ | $\cdots$ | $s_k/\delta_k$ | | |
| $\vdots$ | | | | | | |
| $s_n$ | | | | | | |

Figura 1.3: $s_k = f(s_i, \sigma_j) \quad g(s_1, \sigma_j) = \delta_k$

- **Grafo**

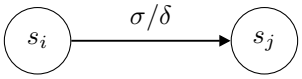



Figura 1.4: ssi: $f(s_i, \sigma) = s_j \quad g(s_i, \sigma) = \delta$

**Comportamiento Dinámico**

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

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

**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:

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

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

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

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

◇ **Notas**

- $w = \lambda$

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

- $\forall s \in k$

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

# 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 vacio, *conjunto de estados*
- $\Sigma$ conjunto finito no vacio, *Alfabeto*
- $f : k \times \Sigma \to k$ , *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 $\vdash_{\overline{M}}$**

Sea $(q, w)$ y $(q', w')$ dos configuraciones[1]:

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

**Lenguaje Aceptado por M**

$$L(M) = \{w \in \Sigma^* / M \text{ acepta } w\}$$
$$L(M) = \{w \in \Sigma^* / (s, w) \vdash^*_{\overline{M}} (q, \lambda) \wedge q \in F\}$$

## 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 vacio
- $\Sigma$ : conjunto finito no vacio

[1] $\vdash_{\overline{M}}$ se lee *"conduce a"* en un paso.

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

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

a $\Delta$ para $u$ tal que $|u| > 1$.

**ii.)** Construimos $M'' = (k'', \Sigma, \delta'', s'', F'')$
La idea clave es considerar que un AFN en un determinando instante se encuentra en un conjunto de estados:

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

Formalmente:

$$E(q) = \{p \in k'/(q, \lambda) \overset{*}{\underset{M'}{\vdash}} (p, \lambda)\}$$

Equivalentemente:

$$E(q) = \{p \in k'/(q, w) \overset{*}{\underset{M'}{\vdash}} (p, w)\}$$

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 $\forall p, q \in k'$:

$$(q, w) \overset{*}{\underset{M'}{\vdash}} (p, \lambda) \Leftrightarrow (E(q), w) \overset{*}{\underset{M''}{\vdash}} (P, \lambda)$$

**p.d.** $M' \approx M''$
**p.d.** $L(M') = L(M'')$

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

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

## 2.4. Propiedades de los Lenguajes Aceptados por AF's

**Teorema**

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

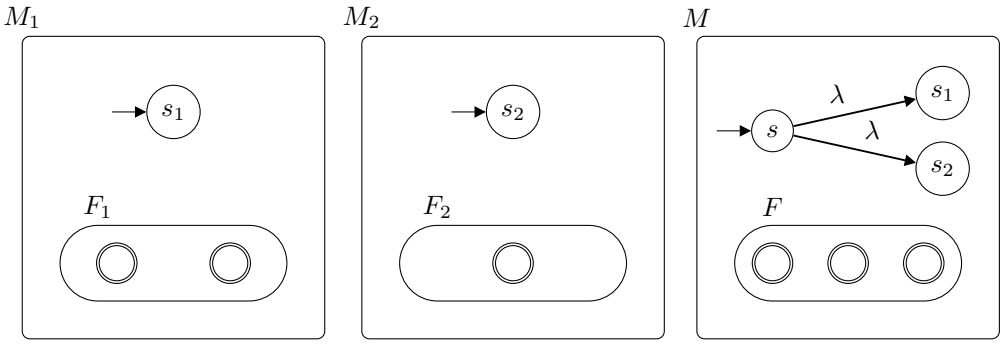
- Unión
- Concatenación
- Estrella de Kleene
- Complementación
- Intersección

**Prueba**

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

**a) Unión**

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

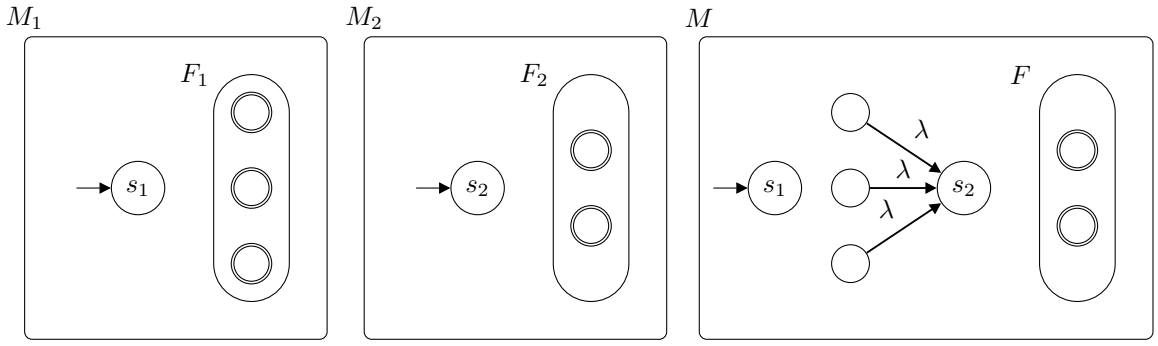



Donde:

- $k = k_1 \cup k_2 \cup \{s\}$
  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: $L(M) = L(M_1)L(M_2)$



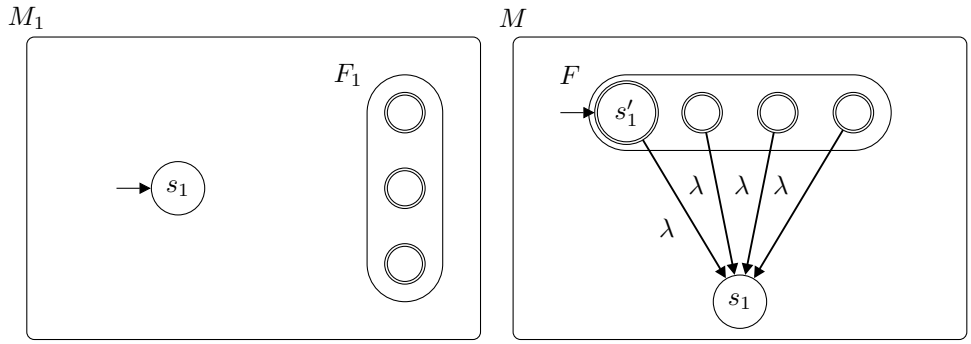

Donde:

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

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

**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:

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

**e) Intersección**

Sea:

$$\begin{aligned} L_1 \cap L_2 &= \overline{\overline{L_1 \cap L_2}} \\ &= \overline{\overline{L_1} \cup \overline{L_2}} \\ &= \Sigma^* - \underbrace{[\underbrace{\underbrace{(\Sigma^* - L_1)}_{1} \cup \underbrace{(\Sigma^* - L_2)}_{2}]}_{3}}_{4} \end{aligned}$$

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 ¿$L(M_1) = L(M_2)$?

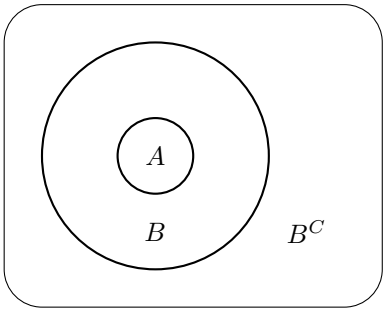
### Prueba

**a)** Esto es lo que ya hemos ido haciendo.

**b)** Simplemente nos basta analizar si el automata no acepta ninguna palabra.

**c)** $M \to L(\bar{M}) = \emptyset$ (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:

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

## 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 $\gamma$ tal que $|\gamma| < n$

## 2.6. Lenguajes no Regulares

## 2.7. Gramáticas

### 2.7.1. Definición

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

- $G$ es un alfabeto.
- $\Sigma$ 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 $A \in V - \Sigma$ , $w \in \Sigma^*$

#### Derivación

#### Lenguaje Generado por $G$

### 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: