

# Teoría de Lenguajes

Gianfranco Zambonni

26 de diciembre de 2022

## Índice

<b>1. Introducción</b>	<b>2</b>
1.1. Relaciones . . . . .	2
1.1.1. Operaciones . . . . .	2
1.2. Alfabetos . . . . .	3
1.2.1. Operaciones . . . . .	4
1.3. Lenguajes . . . . .	4
1.3.1. Operaciones . . . . .	4
1.4. Gramáticas . . . . .	5
1.4.1. Clasificación de gramáticas (Chomsky) . . . . .	5
<b>2. Autómatas finitos</b>	<b>7</b>
2.1. Autómatas finitos determinísticos (AFD) . . . . .	7
2.2. Autómatas finitos no deterministas (AFND) . . . . .	7
2.2.1. Equivalencia entre AFD y AFND . . . . .	8
2.3. Autómatas finitos no determinístico con transiciones $\lambda$ . . . . .	9
2.3.1. Equivalencia entre AFND y AFND- $\lambda$ . . . . .	11

## 1. Introducción

### 1.1. Relaciones

Dados dos conjuntos  $A$  y  $B$ , se llama **relación**  $R$  de  $A$  en  $B$  a todo subconjunto de  $A \times B$ . Notamos  $R : A \rightarrow B$ .

- $aRb$  denota el hecho  $(a, b) \in R$ .
- Si  $A = B$  se dice que  $R$  es una relación sobre  $A$ .

Una relación  $R : A \rightarrow A$  es

- **reflexiva** cuando  $\forall a \in A, aRa$ .
- **simétrica** cuando  $\forall a, b \in A, aRb \implies bRa$ .
- **transitiva** cuando  $a, b, c \in A, aRb \wedge bRc \implies aRc$ .
- es de **equivalencia** cuando es reflexiva, simétrica y transitiva. Este tipo de relaciones particiona a  $A$  en clases de equivalencia.

#### 1.1.1. Operaciones

**Composición de relaciones:** Si  $R : A \rightarrow B$  y  $S : B \rightarrow C$  son relaciones, entonces la composición de  $R$  y  $S$  es la relación  $S \circ R : A \rightarrow C$  definida por:

$$S \circ R = \{(a, c) \mid a \in A, c \in C : \exists b \in B, aRb \wedge bSc\}$$

.

**Relación de identidad:** La relación de identidad sobre  $A$  es la relación  $id_A : A \rightarrow A$  definida por:

$$I = \{(a, a) \mid a \in A\}$$

.

- Es el elemento neutro de la composición de relaciones.

**Relación de potencia:** Dado  $R : A \rightarrow A$  se define la relación de potencia  $R^k : A \rightarrow A$  como la composición de  $k$  copias de  $R$ :

$$R^n = \begin{cases} id_A & \text{si } n = 0 \\ R \circ R^{n-1} & \text{si } n > 0 \end{cases}$$

**Clausura transitiva/positiva:** Dada una relación  $R : A \rightarrow A$  se define la clausura transitiva de  $R$  como la relación  $R^+$  definida por:

$$R^+ = \bigcup_{n=1}^{\infty} R^n$$

1.  $R \subseteq R^+$ .
2.  $R^+$  es transitiva
3. Para toda relación  $G : A \rightarrow A$  tal que  $R \subseteq G \wedge G$  es transitiva, entonces  $R^+ \subseteq G$ , es decir  $R^+$  es la relación transitiva más pequeña que contiene a  $R$ .

#### DEMOSTRACIÓN

- 2) Si  $aR^+b$  entonces existe una secuencia de elementos  $a = a_0, a_1, \dots, a_n = b$  tales que  $a_i R a_{i+1}$  para todo  $i \in [0, n-1]$ .  
Análogamente, como  $bR^+c$  existe una secuencia de elementos  $b = b_0, b_1, \dots, b_m = c$  tales que  $b_i R b_{i+1}$  para todo  $i \in [0, m-1]$ .  
Por lo tanto,  $aR^{n+m}c$ , por lo que  $aR^+c$ .
- 3) Si  $aR^+b$  entonces existe una secuencia de elementos  $a = a_0, a_1, \dots, a_n = b$  tales que  $a_i R a_{i+1}$  para todo  $i \in [0, n-1]$ .  
Como  $R \subseteq G$  entonces  $a_i G a_{i+1}$  para todo  $i \in [0, n-1]$ . Como  $G$  es transitiva entonces la aplicación repetida de la transitividad nos lleva a que  $a_1 G a_n$ , por lo que  $aGb$ .

**Clausura transitiva reflexiva:**

$$R^* = R^+ \cup id_A = \bigcup_{n=0}^{\infty} R^n$$

Si  $A$  es un conjunto finito, entonces todas las relaciones  $R : A \rightarrow A$  son finitas.

Si  $R$  es reflexiva, entonces  $R^* = R^+$ .

## 1.2. Alfabetos

Un alfabeto es un conjunto finito de símbolos.

**Cadena:** Una cadena sobre un alfabeto  $\Sigma$  es una secuencia finita de símbolos de  $\Sigma$ . Los símbolos son notados respetando el orden de la secuencia.

### 1.2.1. Operaciones

**Concatenación:** Es una operación entre un símbolo del alfabeto  $\Sigma$  y una cadena sobre dicho alfabeto:

$$\circ : \Sigma \times \{\text{cadenas sobre } \Sigma\} \rightarrow \{\text{cadenas de } \Sigma\}$$

- La cadena nula  $\lambda$  es el elemento neutro de la concatenación.

**Clausura de Kleene de  $\Sigma$ :**  $\Sigma^*$

- $\lambda \in \Sigma^*$
- $a \in \Sigma \wedge^* \implies \forall \alpha \in \Sigma, a \circ \alpha \in \Sigma^*$

**Clausura positiva de  $\Sigma$ :**  $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$

## 1.3. Lenguajes

Un lenguaje es un conjunto de cadenas sobre un alfabeto  $\Sigma$ .

### 1.3.1. Operaciones

**Concatenación de lenguajes:** Si  $L_1$  y  $L_2$  son lenguajes definidos sobre los alfabetos  $\Sigma_1$  y  $\Sigma_2$  respectivamente, entonces la concatenación de  $L_1$  y  $L_2$  es el lenguaje  $L_1 L_2$  definido por:

$$L_1 L_2 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$$

definido sobre el alfabeto  $\Sigma_1 \cup \Sigma_2$ .

**Clausura de Kleene  $L^*$ :** Se define por:

$$\begin{aligned} L^0 &= \{\lambda\} \\ L^n &= L L^{n-1} \text{ para } n \geq 1 \\ L^* &= \bigcup_{n=0}^{\infty} L^n \end{aligned}$$

**Clausura positiva  $L^+$ :**  $L^+ = \bigcup_{n=1}^{\infty} L^n$

- $L^+ = L L^*$
- $L^* = L^+ \cup \{\lambda\}$
- Si  $L$  es un lenguaje definido sobre  $\Sigma$  entonces  $L \subseteq \Sigma^*$

## 1.4. Gramáticas

Una gramática es una 4-tupla  $(V_N, V_T, P, S)$  donde:

- $V_N$  es un conjunto finito de símbolos no terminales.
- $V_T$  es un conjunto finito de símbolos terminales.
- $P$  es un conjunto finito de reglas de producción: Son pares ordenados  $\alpha, \beta$  donde  $\alpha \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*$  y  $\beta \in (V_N \cup V_T)^*$ .

Las notamos como  $\alpha \rightarrow \beta$ .

- $S \in V_N$  es el símbolo inicial.

**Forma setencial de una gramática:** Se llama forma setencial a una derivación de la misma (es decir, una cadena formada por símbolos de  $V_N \cup V_T$  que sea el resultado de una derivación a partir de símbolos iniciales):

- $S$  es una forma setencial de  $G$
- Si  $\alpha\beta\gamma$  es una forma setencial de  $G$  y  $\beta \rightarrow \delta \in P$  entonces  $\alpha\delta\gamma$  es una forma setencial de  $G$ .

**Derivación directa en  $G$ :** Si  $\alpha\beta\lambda \in (V_N \cup V_T)^*$  y  $\beta \rightarrow \delta \in P$  entonces  $\alpha\delta\lambda$  es una derivación directa de  $G$  de  $\alpha\beta\lambda$  y se denota como  $\alpha\beta\lambda \xRightarrow{G} \alpha\delta\lambda$ .

Denotaremos con  $\xRightarrow{+}_G$  y  $\xRightarrow{*}_G$  a la clausura positiva y la clausura transitiva y reflexiva de  $\xRightarrow{G}$ , respectivamente.

Además,  $\xRightarrow{k}_G$  será la potencia  $k$ -ésima de  $\xRightarrow{G}$ .

**Lenguaje de una gramática  $\mathcal{L}(G)$ :** Es el conjunto de todas las cadenas de símbolos terminales que son formas setenciales de  $G$ .

$$\mathcal{L}(G) = \{\alpha \in V_T^* : S \xRightarrow{+}_G \alpha\}$$

### 1.4.1. Clasificación de gramáticas (Chomsky)

**Gramáticas regulares (tipo 3):** Son aquellas gramáticas que cumplen alguna de las siguientes condiciones:

- Todas sus producciones son de la forma  $A \rightarrow xB$  o  $A \rightarrow x$  donde  $A, B \in V_N$  y  $x \in V_T^*$ . En este caso se dice que es una gramática lineal a derecha.
- Todas sus producciones son de la forma  $A \rightarrow Bx$  o  $A \rightarrow x$  donde  $A, B \in V_N$  y  $x \in V_T^*$ . En este caso se dice que es una gramática lineal a izquierda.

Una forma alternativa de escribir una gramática regular es la siguiente:

- Todas las producciones son de la forma  $A \rightarrow aB$  o  $A \rightarrow a$  o  $A \rightarrow \lambda$  donde  $A, B \in V_N$  y  $a \in V_T$ , para el caso de la gramática lineal a derecha.
- Todas las producciones son de la forma  $A \rightarrow Ba$  o  $A \rightarrow a$  o  $A \rightarrow \lambda$  donde  $A, B \in V_N$  y  $a \in V_T$ , para el caso de la gramática lineal a izquierda.

**Gramáticas libres de contexto (tipo 2):** Son aquellas gramáticas en las que cada producción es de la forma  $A \rightarrow \alpha$  donde  $A \in V_N$  y  $\alpha \in (V_N \cup V_T)^*$ .

De la definición anterior puede inferirse que toda gramática regular es libre de contexto.

**Gramáticas sensibles al contexto (tipo 1):** Son aquellas gramáticas en las que cada producción es de la forma  $\alpha \rightarrow \beta$  donde  $\alpha, \beta \in (V_N \cup V_T)^*$  y  $|\alpha| \leq |\beta|$ .

Se puede inferir que toda gramática independiente del contexto que no posea regla borradora (es decir, que no posea producciones de la forma  $A \rightarrow \lambda$ ) es sensible al contexto.

**Gramáticas sin restricciones (tipo 0):** Son aquellas gramáticas que no poseen ninguna restricción como las anteriores.

El conjunto de las gramáticas tipo 0 es el conjunto de todas las gramáticas.

**Definición:** Un lenguaje generado por una gramática tipo  $t$  es llamado **lenguaje tipo  $t$** .

## 2. Autómatas finitos

### 2.1. Autómatas finitos determinísticos (AFD)

Un autómata finito determinista es una 5-tupla  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  donde:

- $Q$  es un conjunto finito de estados.
- $\Sigma$  es un conjunto finito de símbolos de entrada.
- $\delta : Q \times \Sigma \rightarrow Q$  es una función de transición.
- $q_0 \in Q$  es el estado inicial.
- $F \subseteq Q$  es el conjunto de estados finales.

**Función de transición generalizada  $\hat{\delta}$ :** La función de transición  $\delta$  está definida para que tome como parámetro un único símbolo de *Sigma*. Se puede extender para que tome como parámetro una cadena de símbolos de *Sigma*:

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

se define de manera recursiva como:

- $\hat{\delta}(q, \lambda) = q$
- $\hat{\delta}(q, \beta a) = \delta(\hat{\delta}(q, \beta), a)$  con  $\beta \in \Sigma^*$  y  $a \in \Sigma$

**Cadena aceptada por un AFD:** Una cadena  $\beta \in \Sigma^*$  es aceptada por un AFD  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  si y solo si  $\hat{\delta}(q_0, \beta) \in F$ .

**Lenguaje aceptado por un AFD:** El lenguaje aceptado por un AFD  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  es el conjunto de todas las cadenas  $\beta \in \Sigma^*$  que son aceptadas por  $\mathcal{M}$ :

$$L(\mathcal{M}) = \{\beta \in \Sigma^* : \hat{\delta}(q_0, \beta) \in F\}$$

### 2.2. Autómatas finitos no deterministas (AFND)

Un autómata finito no determinista es una 5-tupla  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  donde:

- $Q$  es un conjunto finito de estados.
- $\Sigma$  es un conjunto finito de símbolos de entrada.
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  es una función de transición.

A diferencia de los AFD, la función  $\delta$  devuelve un conjunto de estados en lugar de un solo estado.

- $q_0 \in Q$  es el estado inicial.
- $F \subseteq Q$  es el conjunto de estados finales.

**Función de transición generalizada  $\hat{\delta}$ :** Primero vamos a definir  $\delta_P : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$  de la siguiente manera:

$$\delta_P(P, a) = \bigcup_{p \in P} \delta(p, a)$$

La función  $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$  se define de manera recursiva como:

- $\hat{\delta}(q, \lambda) = \{q\}$
- $\hat{\delta}(q, \beta a) = \{p : \exists r \in \hat{\delta}(q, \beta) \text{ tal que } p \in \delta(r, a)\} = \delta_P(\hat{\delta}(q, \beta), a) \text{ con } \beta \in \Sigma^* \text{ y } a \in \Sigma$

Para generalizar a un más podemos definir  $\hat{\delta}_P : \mathcal{P}(Q) \times \Sigma^* \rightarrow \mathcal{P}(Q)$  de la siguiente manera:

$$\hat{\delta}_P(P, \beta) = \bigcup_{q \in P} \hat{\delta}(q, \beta)$$

**Cadena aceptada por un AFND:** Una cadena  $\beta \in \Sigma^*$  es aceptada por un AFND  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  si y solo si  $\hat{\delta}(q_0, \beta) \cap F \neq \emptyset$ . Es decir, si alguno de los estados alcanzados por  $\hat{\delta}(q_0, \beta)$  es un estado final.

**Lenguaje aceptado por un AFND:** El lenguaje aceptado por un AFND  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$  es el conjunto de todas las cadenas  $\beta \in \Sigma^*$  que son aceptadas por  $\mathcal{M}$ :

$$L(\mathcal{M}) = \{\beta \in \Sigma^* : \hat{\delta}(q_0, \beta) \cap F \neq \emptyset\}$$

### 2.2.1. Equivalencia entre AFD y AFND

Es trivial ver que para todo AFD existe un AFND que acepte el mismo lenguaje.

**Teorema 2.1.** *Dado una AFND  $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ , existe un AFD  $\mathcal{M}' = \langle Q', \Sigma, \delta', q'_0, F' \rangle$  tal que  $L(\mathcal{M}) = L(\mathcal{M}')$ .*

Vamos a demostrar este teorema construyendo una AFD  $\mathcal{M}'$  a partir de  $\mathcal{M}$ . Una vez constuido deberemos demostrar que  $\mathcal{M}'$  acepta el mismo lenguaje que  $\mathcal{M}$ .

**Construcción de  $\mathcal{M}'$ :**

- $Q'$  será el conjunto de partes  $\mathcal{P}(Q)$  que contenga a todos los posibles conjuntos de estados de  $\mathcal{M}$ . Vamos a denotar cada estado  $s \in Q'$  con etiquetas del estilo  $[q_1, \dots, q_k]$  donde  $q_1, \dots, q_k \in Q$ . Entonces:

$$Q' = \mathcal{P}(Q)$$

- $\delta'([q_1, \dots, q_k], a) = [p_1, \dots, p_m] \iff \delta_P(\{q_1, \dots, q_k\}, a) = \{p_1, \dots, p_m\}$
- $q'_0 = [q_0]$
- $F' = \{[q_1, \dots, q_n] \in Q' : \{q_1, \dots, q_n\} \cap F \neq \emptyset\}$



**Equivalencia de funciones de transición generalizadas:** Antes de demostrar que ambos autómatas aceptan el mismo lenguaje, vamos a demostrar que las funciones de transición generalizadas de ambos autómatas son equivalentes cuando las llamamos con el estado inicial como primer parámetro. Es decir, queremos ver que  $\hat{\delta}'(q'_0, \beta) = [p_1, \dots, p_k] \iff \hat{\delta}(q_0, \beta) = \{p_1, \dots, p_k\}$ .

Lo vamos a hacer por inducción. Recordemos que  $q'_0 = [q_0]$ :

■ Caso base:  $\beta = \lambda$ :

- $\hat{\delta}'([q_0], \lambda) = [q_0]$  por definición de  $\hat{\delta}'$ .
- $\hat{\delta}(q_0, \lambda) = \{q_0\}$  por definición de  $\hat{\delta}$ .

Luego  $\hat{\delta}'([q_0], \lambda) = [q_0] \iff \hat{\delta}(q_0, \lambda) = \{q_0\}$ .

■ Caso inductivo:  $\beta \implies \beta a$ : Por hipótesis inductiva tenemos que

$$\hat{\delta}'(q'_0, \beta) = [r_1, \dots, r_m] \iff \hat{\delta}(q_0, \beta) = \{r_1, \dots, r_m\}$$

Queremos ver que  $\hat{\delta}'(q'_0, \beta a) = [p_1, \dots, p_m] \iff \hat{\delta}(q_0, \beta a) = \{p_1, \dots, p_m\}$

$$\hat{\delta}'(q'_0, \beta a) = [p_1, \dots, p_k] \xLeftrightarrow{\text{def.}} \delta'(\hat{\delta}'(q'_0, \beta), a) = [p_1, \dots, p_k]$$

$$\xLeftrightarrow{\text{def.}} \exists [r_1, \dots, r_m] \in Q' \text{ tal que } \delta'(q'_0, \beta) = [r_1, \dots, r_m] \wedge \delta'([r_1, \dots, r_m], a) = [p_1, \dots, p_k]$$

$$\xLeftrightarrow{H.I.} \exists \{r_1, \dots, r_m\} \in Q \text{ tal que } \hat{\delta}(q_0, \beta) = \{r_1, \dots, r_m\} \wedge \delta_P(\{r_1, \dots, r_m\}, a) = \{p_1, \dots, p_k\}$$

$$\xLeftrightarrow{\text{def.}} \delta_P(\hat{\delta}(q_0, \beta), a) = \{p_1, \dots, p_k\} \xLeftrightarrow{\text{def.}} \hat{\delta}(q_0, \beta a) = \{p_1, \dots, p_k\}$$

**Demostración de la equivalencia:** Ahora que hemos demostrado que las funciones de transición generalizadas de ambos autómatas son equivalentes, vamos a demostrar que ambos autómatas aceptan el mismo lenguaje:

$$\begin{aligned} \beta \in \mathcal{L}(\mathcal{M}) &\xLeftrightarrow{\text{def.}} \hat{\delta}(q_0, \beta) = \{q_1, \dots, q_n\} \wedge \{q_1, \dots, q_n\} \cap F \neq \emptyset \\ &\iff \underbrace{\hat{\delta}(q'_0, \beta) = [q_1, \dots, q_n]}_{\text{por equivalencia de generalizaciones}} \wedge \underbrace{[q_1, \dots, q_n] \in F'}_{\text{def. } F'} \\ &\xLeftrightarrow{\text{def.}} x \in \mathcal{L}(M') \end{aligned}$$

### 2.3. Autómatas finitos no determinístico con transiciones $\lambda$

Un autómata finito no determinista con transiciones  $\lambda$  es un autómata finito no determinista que tiene transiciones  $\lambda$ . Estas transacciones nos permiten ir de un estado a otro sin consumir ningún símbolo de entrada.

Los definimos con una 5-upla  $(Q, \Sigma, \delta, q_0, F)$  donde:

- $Q$  es un conjunto finito de estados.

- $\Sigma$  es un conjunto finito de símbolos de entrada.
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$  es una función de transición.
- $q_0 \in Q$  es el estado inicial.
- $F \subseteq Q$  es el conjunto de estados finales.

**Clausura  $\lambda$  de un estado  $q$ :** Se denota  $Cl_\lambda(q)$  es el conjunto de estados que se pueden alcanzar desde  $q$  siguiendo solo transiciones  $\lambda$ . Es decir,

$$Cl_\lambda(q) = \delta(q, \lambda)$$

Además  $q \in Cl_\lambda(q)$ .

**Clausura  $\lambda$  de un conjunto de estados  $P$ :**

$$Cl_{P\lambda}(P) = \bigcup_{p \in P} Cl_\lambda(p)$$

**Generalización de la función de transición:** Podemos extender  $\delta$  a conjunto de estados:

$$\begin{aligned} \delta_P : \mathcal{P}(Q) \times (\Sigma \cup \{\lambda\}) &\rightarrow \mathcal{P}(Q) \\ \delta_P(P, a) &= \bigcup_{p \in P} \delta(p, a) \end{aligned}$$

Entonces podemos definir:

$$\begin{aligned} \hat{\delta} : Q \times \Sigma^* &\rightarrow \mathcal{P}(Q) \\ \hat{\delta}(q_0, \lambda) &= Cl_\lambda(q_0) \\ \hat{\delta}(q_0, \beta a) &= Cl_{P\lambda} \left( \delta_P(\hat{\delta}(q_0, \beta), a) \right) = Cl_{P\lambda} \left( \left\{ p : \exists q \in \hat{\delta}(q_0, \beta) \text{ tal que } p \in \delta(q, a) \right\} \right) \end{aligned}$$

También extendemos  $\hat{\delta}$  a conjuntos de estados:

$$\begin{aligned} \hat{\delta}_P : \mathcal{P}(Q) \times \Sigma^* &\rightarrow \mathcal{P}(Q) \\ \hat{\delta}_P(P, \beta a) &= \bigcup_{p \in P} \hat{\delta}(p, \beta a) \end{aligned}$$

**Cadena aceptada por un AFND- $\lambda$ :** Una cadena  $\beta$  es aceptada por un AFND- $\lambda$   $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  si y solo si  $\hat{\delta}(q_0, \beta) \cap F \neq \emptyset$ .

**Lenguaje aceptado por un AFND- $\lambda$ :** El lenguaje aceptado por un AFND- $\lambda$   $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  es el conjunto de todas las cadenas aceptadas por  $M$ :

$$\mathcal{L}(M) = \{\beta \in \Sigma^* : \hat{\delta}(q_0, \beta) \cap F \neq \emptyset\}$$

**2.3.1. Equivalencia entre AFND y AFND- $\lambda$** 

Dado un AFND- $\lambda$   $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  podemos construir un AFND  $M' = \langle Q, \Sigma, \delta', q_0, F' \rangle$  tal que  $M$  acepte el mismo lenguaje que  $M'$ .

**Construcción de  $M'$ :** Notemos que ambos autómatas tiene el mismo conjunto de estados  $Q$ , el mismo conjunto de símbolos de entrada  $\Sigma$  y el mismo estado inicial  $q_0$ . Por lo que solo debemos definir  $\delta'$  y  $F'$ .

$$\begin{aligned} \blacksquare \delta'(q, a) &= \hat{\delta}(q, a) = Cl_{P\lambda} \left( \delta_P(\hat{\delta}(q, \lambda), a) \right) \\ \blacksquare F' &= \begin{cases} F \cup \{q_0\} & \text{si } Cl_{P\lambda}(q_0) \cap F \neq \emptyset \\ F & \text{si no} \end{cases} \end{aligned}$$

**Equivalencia de funciones de transición generalizada:** Vamos a demostrar por inducción que  $\hat{\delta}'(q_0, \beta) = \hat{\delta}(q_0, \beta)$  para todo  $|\beta| \geq 1$ :

- Caso base:  $|\beta| = 1$ . Sea  $\beta = a$ , entonces  $\hat{\delta}'(q_0, \beta) = \hat{\delta}'(q_0, a) = \hat{\delta}(q_0, a)$  por como definimos  $\delta'$ .
- Caso inductivo: Supongamos que  $\hat{\delta}'(q_0, \beta) = \hat{\delta}(q_0, \beta)$  para todo  $|\beta| \leq n$ . Sea  $\omega = \beta a$ . Entonces:

$$\hat{\delta}'(q_0, \omega) = \hat{\delta}'(q_0, \beta a) \stackrel{\text{def.}}{=} \delta'_P(\hat{\delta}'(q_0, \beta), a) \stackrel{\text{H.I.}}{=} \delta'_P(\hat{\delta}(q_0, \beta), a) \quad (1)$$

Por otro lado, dado  $P \subseteq Q$  tenemos que:

$$\delta'_P(P, a) \stackrel{\text{def.}}{=} \bigcup_{p \in P} \delta'(p, a) \stackrel{\text{construcción de } M'}{=} \bigcup_{p \in P} \hat{\delta}(p, a) \stackrel{\text{def.}}{=} \hat{\delta}_P(P, a)$$

Entonces, remplazando en (1) el último término con este resultado, nos queda:

$$\delta'_P(\hat{\delta}(q_0, \beta), a) = \hat{\delta}_P(\hat{\delta}(q_0, \beta), a) \stackrel{\text{def.}}{=} \hat{\delta}(q_0, \beta a) \stackrel{\text{def.}}{=} \hat{\delta}(q_0, \omega)$$

**Demostración de equivalencia:** Veamos ahora que  $M$  acepta el mismo lenguaje que  $M'$ , vamos a separar la demostración en dos casos:  $\beta = \lambda$  y  $\beta \neq \lambda$ .

- $\beta = \lambda$ .

$$\lambda \in \mathcal{L}(M) \Leftrightarrow \hat{\delta}(q_0, \lambda) \cap F \neq \emptyset \Leftrightarrow Cl_{P\lambda}(q_0) \cap F \neq \emptyset \Leftrightarrow q_0 \in F' \Leftrightarrow \lambda \in \mathcal{L}(M')$$