

# Lenguajes Formales, Autómatas y Computabilidad

BAS

Resumen de la materia Lenguajes Formales, Autómatas y Computabilidad dictada en el segundo cuatrimestre de 2024, pensado principalmente para uso personal y orientado a practicar para el final

Fecha de compilación: 03 / 02 / 2025.

Este resumen esta basado en la bibliografía y clases de la materia dictadas en el segundo cuatrimestre de 2024, así como también clases de cuatrimestres pasados de las materias (F) de Teoría de Lenguajes y Lógica y Computabilidad

El template de este resumen es esencialmente el mismo usado por @valn y como soy un sinvergüenza lo pedí prestado sin avisar. Si quieren usar un resumen como guía de estudio, recomiendo que usen el suyo antes que el mío.

Link al resumen original:

([https://gitlab.com/valn/uba/-/tree/main/Lenguajes%20Formales%2C%20Aut%C3%B3matas%20y%20Computabilidad/Final?ref\\_type=heads](https://gitlab.com/valn/uba/-/tree/main/Lenguajes%20Formales%2C%20Aut%C3%B3matas%20y%20Computabilidad/Final?ref_type=heads))

# Índice

1. Lenguajes y Gramáticas .....	5
1.1. Por qué nos interesa la teoría de autómatas? .....	5
1.2. Pero entonces, qué son autómatas? (informalmente) .....	5
1.3. Los Conceptos Centrales de la Teoría de Autómatas .....	5
1.3.1. Alfabetos .....	6
1.3.2. Cadenas .....	6
1.3.2.1. Potencias de un alfabeto .....	6
1.3.3. Hay tantas palabras como números naturales .....	6
1.4. Lenguajes .....	7
1.5. Gramáticas .....	8
1.5.1. Lenguaje generado por una gramática .....	8
1.5.2. Forma sentencial de una gramática .....	8
1.5.3. Derivación directa de una gramática .....	9
1.5.4. La Jerarquía de Chomsky .....	10
1.5.5. Árbol de derivación de gramáticas .....	11
1.5.6. Algunos lemas relevantes .....	11
2. Autómatas Finitos Determinísticos, no Determinísticos, y Gramáticas Regulares .....	12
2.1. Autómata Finito Determinístico (AFD) .....	13
2.1.1. Función de transición generalizada $\hat{\delta}$ .....	13
2.1.2. El Lenguaje de un AFD .....	14
2.2. Autómatas Finitos no Determinísticos (AFND) .....	14
2.2.1. Función de transición generalizada $\hat{\delta}$ .....	14
2.2.2. Lenguaje aceptado por un AFND .....	15
2.2.3. Función de transición de conjuntos de estados .....	15
2.3. Equivalencia Entre AFND y AFD .....	15
2.4. AFND con transiciones $\lambda$ ( $AFND - \lambda$ ) .....	16
2.5. Gramáticas regulares y AFDs .....	19
3. Expresiones Regulares .....	21
4. Lema de Pumping y propiedades de clausura de los lenguajes regulares .....	25
4.1. Configuración instantánea de un AFD .....	26
4.2. Lema de Pumping .....	26
4.3. Propiedades de clausura de los lenguajes regulares .....	28
4.3.1. Unión .....	28
4.3.2. Concatenación y clausura de Kleene .....	29
4.3.3. Complemento .....	29
4.3.4. Intersección .....	30
4.4. Unión e Intersección finitos de lenguajes regulares .....	30
4.5. Todo lenguaje finito es regular .....	31
4.6. Problemas de decidibilidad de lenguajes regulares .....	31
4.6.1. Vacuidad .....	31
4.6.2. Pertenencia .....	31
4.6.3. Finititud .....	32

4.6.4. Equivalencia .....	32
5. Gramáticas Libres de Contexto .....	32
5.1. Derivación .....	33
5.2. Inferencia .....	33
5.3. Lenguaje de una Gramática .....	33
5.4. árbol de derivación .....	33
5.5. Equivalencia inferencia, árbol Derivación y Derivaciones .....	34
5.6. Gramáticas Ambiguas .....	36
6. Autómatas de Pila .....	36
6.1. Configuración de un AP .....	37
6.2. Lenguaje de un AP .....	38
6.3. Equivalencia $\mathcal{L}(P)$ y $\mathcal{L}_\lambda(P)$ .....	38
6.4. Equivalencia GLCs y APDs .....	40
7. Ejercicios (Después debería pasarlo a otro lado) .....	41

# 1. Lenguajes y Gramáticas

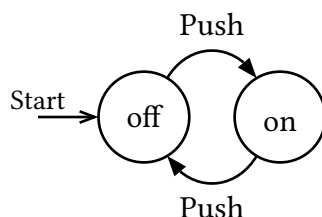
## 1.1. Por qué nos interesa la teoría de autómatas?

Los autómatas finitos son un modelo útil para varios tipos de software y hardware. Por ejemplo:

- Software para la verificación del comportamiento de circuitos digitales, así como de sistemas con una cantidad finita de estados (protocolos de comunicación, entre otros)
- Los analizadores léxicos de los compiladores
- Software para escanear grandes cuerpos de texto

## 1.2. Pero entonces, qué son autómatas? (informalmente)

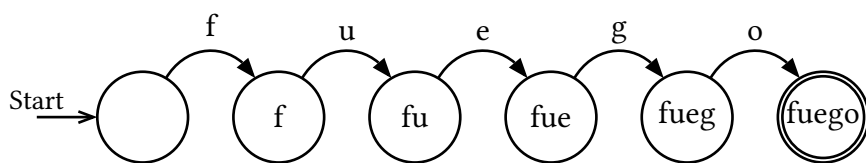
Antes de discutir la definición formal de estos modelos, consideremos primero como un autómata finito se ve y qué es lo que hace.



Acá tenemos un autómata que simula un interruptor de encendido / apagado. el dispositivo recuerda si está en estado «prendido» o «apagado», y permite presionar un botón cuyo efecto va a ser diferente dependiendo del estado en el que se encuentre en ese momento (representado por los arcos saliendo de los estados). Notar la etiqueta «Start», la cual indica el estado inicial del autómata

Hay muchos sistemas que pueden ser vistos de una manera similar, es decir, que cumplen que en cualquier momento determinado se encuentran en uno de un número finito de estados. El propósito de estos estados es recordar la porción relevante de la historia del sistema, para poder actuar de acuerdo a ésta.

Otro ejemplo de un autómata puede ser un analizador léxico. Por ejemplo, un autómata que sólo reconozca la palabra «fuego» podría estar dado por:



Como queremos que sea capaz de reconocer la palabra «fuego», el autómata precisa de 5 estados, cada uno representando una posición diferente de la palabra que ya fue alcanzada. Y los arcos representan un input de una letra. Notar ahora el uso de un estado con doble círculos, el mismo denota un estado final, un estado especial que determina que el autómata aceptó el input dado

## 1.3. Los Conceptos Centrales de la Teoría de Autómatas

Ahora vamos a introducir definiciones esenciales para el estudio de autómatas: el **alfabeto**, las **cadena**s, y el **lenguaje**

### 1.3.1. Alfabetos

**Definición 1.3.1.1:** Un *alfabeto* es conjunto finito no vacío de símbolos, que por convención denotamos  $\Sigma$ .

*Ejemplo:*  $\Sigma = \{0, 1\}$ . El alfabeto binario

*Ejemplo:*  $\Sigma = \{a, b, \dots, z\}$ . El alfabeto de todas las letras minúsculas

### 1.3.2. Cadenas

**Definición 1.3.2.1:** Una *cadena* (también conocida como palabra) es una secuencia finita de símbolos pertenecientes a un mismo alfabeto

*Ejemplo:* La cadena 0110 formada por símbolos pertenecientes al alfabeto binario

**Definición 1.3.2.2:** Nos referimos con *longitud* de una cadena a la cantidad de símbolos en la misma, y denotamos la longitud de la cadena  $w$  usando  $|w|$

#### 1.3.2.1. Potencias de un alfabeto

**Definición 1.3.2.1.1:** Si  $\Sigma$  es un alfabeto, podemos denotar al conjunto de todas las cadenas pertenecientes al alfabeto de cierta longitud usando notación exponencial. Definimos  $\Sigma^k$  como el conjunto de cadenas de longitud  $k$ , con todos sus símbolos pertenecientes a  $\Sigma$

*Ejemplo:* Para el alfabeto binario,  $\Sigma^1 = \{0, 1\}$ ,  $\Sigma^2 = \{00, 01, 10, 11\}$

*Ejemplo:* Notar que  $\Sigma^0 = \{\lambda\}$ , sin importar el alfabeto. Es la única cadena de longitud 0

**Definición 1.3.2.1.2:** Usamos la notación  $\Sigma^*$  para referirnos al conjunto de todas las cadenas de un alfabeto, y lo denominamos **clausura de Kleene**, y usamos  $\Sigma^+$  para referirnos a su clausura positiva (es decir, que no incluya la palabra vacía), se definen como:

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i \quad (1)$$

$$\Sigma^+ = \bigcup_{i \geq 1} \Sigma^i \quad (2)$$

### 1.3.3. Hay tantas palabras como números naturales

**Teorema 1.3.3.1:**  $|\Sigma^*|$  es igual a la cardinalidad de  $\mathbb{N}$

**Definición 1.3.3.1:** Definimos el orden lexicográfico  $\prec \subset \Sigma^* \times \Sigma^*$ . Asumimos el orden lexicográfico entre los elementos de un mismo alfabeto, y luego lo extendemos a un orden lexicográfico entre palabras de una misma longitud. De esta manera, las palabras de menor longitud son menores en este sentido que las de mayor longitud

*Demostración:* Definimos una biyección  $f : \mathbb{N} \rightarrow \Sigma^*$  tal que  $f(i)$  = la  $i$ -ésima palabra en el orden  $\prec$  sobre  $\Sigma^*$ . Luego, como tenemos una biyección entre los conjuntos  $\Sigma^*$  y  $\mathbb{N}$ , tenemos que necesariamente deben tener la misma cardinalidad  $\square$

## 1.4. Lenguajes

**Definición 1.4.1:** Un lenguaje  $L$  sobre un alfabeto  $\Sigma$  es un conjunto de palabras pertenecientes a  $\Sigma$ . Es decir,  $L \subseteq \Sigma^*$

*Ejemplo:*

- $\emptyset$ , el lenguaje vacío, es un lenguaje para cualquier alfabeto
- $\{\lambda\}$ , el lenguaje que consiste sólo de la cadena vacía, que también es un lenguaje sobre cualquier alfabeto. (Notar que  $\emptyset \neq \{\lambda\}$ )
- El lenguaje de todas las cadenas formadas por  $n$  1s seguidos por  $n$  0s sobre  $\sigma = \{0, 1\}$ . Algunas cadenas de este lenguaje son:  $\{\lambda, 10, 1100, 111000, \dots\}$

La única restricción importante en cuanto qué puede ser un lenguaje, es que el alfabeto del mismo siempre es finito

**Definición 1.4.2:** Sea  $A$  un conjunto,  $\mathcal{P}(A)$  es el conjunto de todos los subconjuntos de  $A$ ,  $\mathcal{P}(A) = \{B \subseteq A\}$ . Si tenemos que  $A$  es un conjunto finito, entonces  $\mathcal{P}(A) = 2^{|A|}$

**Teorema 1.4.1:**  $|\Sigma^*| < \mathcal{P}(\Sigma^*)$ , es decir, la cantidad de lenguajes sobre un alfabeto  $\Sigma$  no es numerable

*Demostración:* Supongamos que  $\mathcal{P}(\Sigma^*)$  es numerable, entonces tenemos que podemos enlistar los lenguajes y para cada lenguaje  $L_i$  (es decir cada lenguaje  $\subset \mathcal{P}(\Sigma^*)$ ), podemos ordenar las palabras pertenecientes al mismo según el orden lexicográfico definido anteriormente de la siguiente manera:

$L_1: w_{1,1}, w_{1,2}, w_{1,3}, \dots$

$L_2: w_{2,1}, w_{2,2}, w_{2,3}, \dots$

...

Ahora consideremos el lenguaje  $L = \{u_1, u_2, u_3, \dots\}$  tal que, para todo  $i$ , se cumpla que  $w_{i,i} \prec u_i$ . Pero entonces tenemos un lenguaje cuyo  $i$ ésimo elemento de  $L$  siempre es «mayor» que el  $i$ ésimo elemento de  $L_i$ , entonces no puede ser que  $L$  pertenezca a nuestra lista de lenguajes. Pero entonces tenemos un lenguaje sobre  $\Sigma^*$  que no estaba incluido en nuestra enumeración de todos los lenguajes, por lo que tenemos un absurdo. El mismo vino de suponer que  $\mathcal{P}(\Sigma^*)$  era numerable.

$\square$

## 1.5. Gramáticas

**Definición 1.5.1:** Una gramática es una 4-upla  $G = \langle V_N, V_T, P, S \rangle$  donde:

- $V_N$  es un conjunto de símbolos llamados no terminales
- $V_T$  es un conjunto de símbolos terminales
- $P$  es el conjunto de producciones, que es un conjunto finito de

$$[(V_N \cup V_T)^* V_N (V_N \cup V_T)^*] \times (V_N \cup V_T)^*$$

donde las producciones son tuplas  $(a, b)$  y usualmente las notamos  $a \rightarrow b$

- $S$  es el símbolo distinguido o inicial de  $V_N$

*Ejemplo:* Sea  $G_{\text{arithm}} = \langle V_N, V_T, P, S \rangle$  una gramática libre de contexto (esto último lo introducimos más tarde) tal que:

- $V_N = \{S\}$
- $V_T = \{+, *, a, (, )\}$
- Y producciones determinadas por:
 
$$S \rightarrow S + S,$$

$$S \rightarrow S * S,$$

$$S \rightarrow a,$$

$$S \rightarrow (S)$$

### 1.5.1. Lenguaje generado por una gramática

**Definición 1.5.1.1:** Dada una gramática  $G = \langle V_N, V_T, P, S \rangle$ , definimos a  $\mathcal{L}(G)$  como:

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

donde  $\xRightarrow{+}_G$  es la derivación en uno o más pasos, que se obtiene de la clausura transitiva de  $\Rightarrow_G$  (la derivación directa, cuya definición se da en un momento)

### 1.5.2. Forma sentencial de una gramática

**Definición 1.5.2.1:** Sea  $G = \langle V_N, V_T, P, S \rangle$ .

- $S$  es una forma sentencial de  $G$
- Si  $\alpha\beta\gamma$  es una forma sentencial de  $G$ , y tenemos que  $(\beta \rightarrow \delta) \in P$ , entonces  $\alpha\delta\gamma$  es también una forma sentencial de  $G$

Las formas sentenciales pertenecen a  $(V_N \cup V_T)^*$



### 1.5.3. Derivación directa de una gramática

**Definición 1.5.3.1:** Si  $\alpha\beta\gamma \in (V_N \cup V_T)^*$  y  $(\beta \rightarrow \delta) \in P$ , entonces  $\alpha\beta\gamma$  deriva directamente en  $G$  a  $\alpha\delta\gamma$  y lo denotamos:

$$\alpha\beta\gamma \xRightarrow[G]{} \alpha\delta\gamma$$

Observemos que como la derivación directa es una relación sobre  $(V_N \cup V_T)^*$ , podemos componerla consigo misma 0 o más veces:

- $\left(\xRightarrow[G]{}\right)^0 = \text{id}_{(V_N \cup V_T)^*}$
- $\left(\xRightarrow[G]{}\right)^+ = \bigcup_{k=1}^{\infty} \left(\xRightarrow[G]{}\right)^k$
- $\left(\xRightarrow[G]{}\right)^* = \left(\xRightarrow[G]{}\right)^+ \cup \text{id}_{(V_N \cup V_T)^*}$

Según como elijamos derivar, podemos considerar:

- Derivación más a la izquierda:  $\xRightarrow[L]{}_L$
- Derivación más a la derecha:  $\xRightarrow[R]{}_R$

Así como también sus derivaciones en uno o más pasos, y sus clausuras transitivas y de Kleene

### 1.5.4. La Jerarquía de Chomsky

**Definición 1.5.4.1:** La jerarquía de Chomsky clasifica las gramáticas en 4 tipos en una jerarquía tal que puedan generar lenguajes cada vez más complejos, y donde cada uno de las gramáticas puede generar también los lenguajes de una de complejidad inferior. Las clasificaciones son:

- **Gramáticas de tipo 0 (o sin restricciones):**

$$\alpha \rightarrow \beta$$

- **Gramáticas de tipo 1 (o sensibles al contexto):**

$$\alpha \rightarrow \beta, \text{ con } |\alpha| \leq |\beta|$$

- **Gramáticas de tipo 2 (o libres de contexto):**

$$A \rightarrow \gamma \text{ con } A \in V_N$$

- **Gramáticas de tipo 3 (o regulares):**

$$A \rightarrow a, A \rightarrow aB, A \rightarrow \lambda, \text{ con } A, B \in V_N, a \in V_T$$

La jerarquía de gramáticas da origen también a la jerarquía de lenguajes:

- **Recursivamente enumerables**
- **Sensitivos al contexto**
- **Libres de contexto**
- **Regulares**

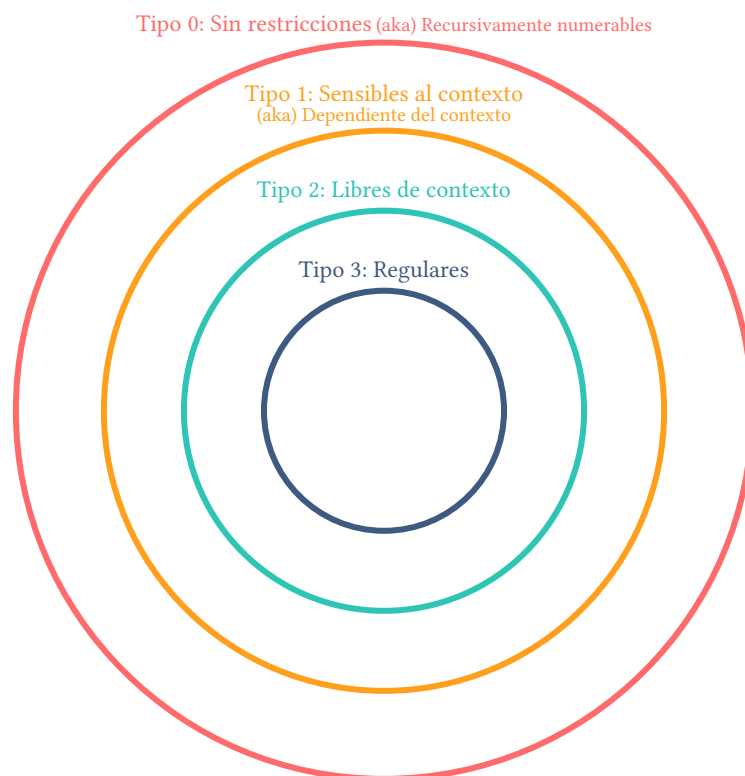


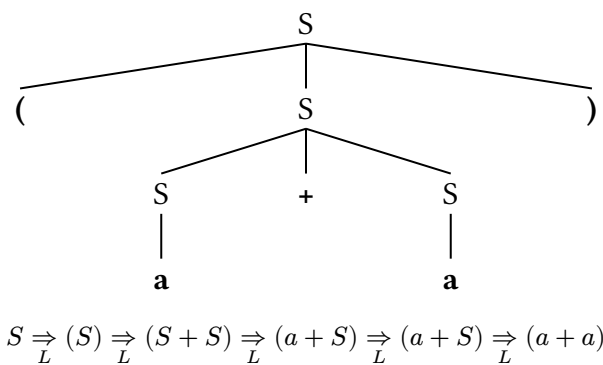
Figura 1: Imagen perteneciente al repo de @Valn (No sé que haces acá todavía a ser honesto).

### 1.5.5. Árbol de derivación de gramáticas

**Definición 1.5.5.1:** Un árbol de derivación es una representación gráfica de una derivación (look at me I'm so smart) donde:

- Las etiquetas de las hojas están en  $V_T \cup \{\lambda\}$
- Las etiquetas de los nodos internos están en  $V_N$ . Las etiquetas de sus símbolos son los símbolos del cuerpo de una producción
- Un nodo tiene etiqueta  $A$  y tiene  $n$  descendientes etiquetados  $X_1, X_2, \dots, X_N$  sii hay una derivación que usa una producción  $A \rightarrow X_1 X_2 \dots X_N$

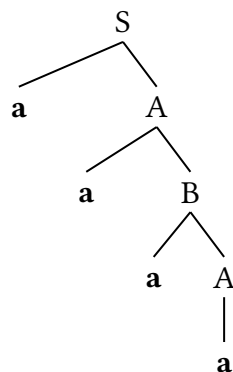
*Ejemplo:* Considerando la gramática  $G_{\text{arithm}}$  definida anteriormente, un posible árbol de derivación podría ser el siguiente:



### 1.5.6. Algunos lemas relevantes

**Lema 1.5.6.1:** Sea  $G = \langle V_N, V_T, P, S \rangle$  regular, no recursiva a izquierda. Si  $A \xRightarrow[L]{i} wB$  entonces  $i = |w|$

*Demostración:* Consideremos el árbol de derivación de la cadena  $w = a_1 a_2 a_3 \dots a_n$ . Si entonces cortamos el árbol en una altura  $h$  determinada, para  $h \geq 1$  se obtiene un subárbol de  $h$  hojas, con el único nodo que no es hoja con una etiqueta de  $V_N$ . Por la forma que las producciones de una gramática regular tiene, tenemos que cada derivación pone un símbolo no terminal a lo sumo y uno terminal, luego, tenemos que en  $n$  derivaciones (altura  $n$  del árbol) tenemos  $n$  hojas y el no terminal de la derecha  $\square$



Un posible árbol de derivación para las producciones:

$$\begin{aligned} S &\rightarrow aA \mid \lambda \\ A &\rightarrow a \mid aB \\ B &\rightarrow aA \end{aligned}$$

**Lema 1.5.6.2:** Sea  $G = \langle V_N, V_T, P, S \rangle$  libre de contexto, no recursiva a izquierda (es decir, no tiene derivaciones  $A \xRightarrow[L]{+} A\alpha$ ). Existe una constante  $c$  tal que si  $A \xRightarrow[L]{i} \omega B\alpha$  entonces  $i \leq c^{|\omega|} + 2$

*Demostración:* Pendiente (soy **vago**)

□

## 2. Autómatas Finitos Determinísticos, no Determinísticos, y Gramáticas Regulares

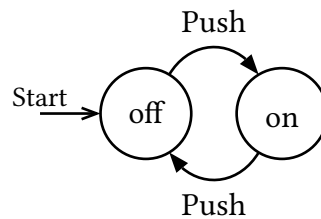
### 2.1. Autómata Finito Determinístico (AFD)

**Definición 2.1.1:** Un autómata finito determinístico es una 5-upla  $\langle Q, \Sigma, \delta, q_0, F \rangle$  donde:

- $Q$  es un conjunto finito de estados
- $\Sigma$  es el alfabeto de entrada del autómata
- $\delta$  es la función de transición del autómata que toma como argumentos un estado y un símbolo de input, y devuelve un estado, es decir:  $\delta : Q \times \Sigma \rightarrow Q$ .
- $q_0 \in Q$  es el estado inicial del autómata
- $F \subseteq Q$  es el conjunto de estados finales del autómata

Para determinar si un autómata acepta cierta cadena, se hace uso de su función de transición y se confirma si se termina llegando a un estado  $q_f \in F$  final. Es decir, suponiendo que se busca. Como es poco práctico escribir la función  $\delta$  en su totalidad, se suele hacer uso de diagramas de autómatas (como el usado al principio de este resumen) y se aceptan los arcos del mismo como una definición implícita.

*Ejemplo:* En nuestro primer autómata, teníamos (notar que «Push» es sólo un símbolo, no una cadena / palabra)



Luego, la 5-upla correspondiente sería  $\langle \{\text{off}, \text{on}\}, \{\text{Push}\}, \delta, \text{off}, \emptyset \rangle$  con  $\delta$  definido por:

$$\delta(\text{off}, \text{Push}) = \text{on}$$

$$\delta(\text{on}, \text{Push}) = \text{off}$$

#### 2.1.1. Función de transición generalizada $\hat{\delta}$

Ahora necesitamos definir precisamente qué ocurre cuando estamos en un estado determinado y recibimos como input una cadena, no un símbolo. Para esto, lo definimos por inducción en el largo de la cadena

**Definición 2.1.1.1:** Definimos  $\hat{\delta}$  como:

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

Cabe recalcar que  $\hat{\delta}(q, \alpha) = \delta(\hat{\delta}(q, \lambda), \alpha) = \delta(q, \alpha)$  y que muchas veces vamos a hacer uso de la misma notación para ambas funciones

### 2.1.2. El Lenguaje de un AFD

**Definición 2.1.2.1:** Ahora podemos definir el lenguaje aceptado por un AFD como:

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

Es decir, el lenguaje de un autómata  $M$  es el conjunto de cadenas que mediante  $\hat{\delta}$  llegan desde el estado inicial  $q_0$  a un estado final. Llamamos **lenguajes regulares** a aquellos aceptados por un AFD

## 2.2. Autómatas Finitos no Determinísticos (AFND)

**Definición 2.2.1:** Un AFND es una 5-upla  $\langle Q, \Sigma, \delta, q_0, F \rangle$  donde:

- $Q$  es un conjunto finito de estados
- $\Sigma$  es el alfabeto de entrada
- $\delta$  es la función de transición, que toma un estado en  $Q$  y un símbolo en  $\Sigma$ , pero que ahora devuelve un subconjunto de  $Q$ . Es decir:

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

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

### 2.2.1. Función de transición generalizada $\hat{\delta}$

**Definición 2.2.1.1:** Definimos  $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$  :

- $\hat{\delta}(q, \lambda) = \{q\}$
- $\hat{\delta}(q, x\alpha) = \{p \in Q : \exists r \in \hat{\delta}(q, x) \text{ y } p \in \delta(r, a)\}$
- Una definición alternativa, siendo  $w = xa$ , con  $w, x \in \Sigma^*$  y  $a \in \Sigma$  y suponiendo que  $\hat{\delta}(q, x) = \{p_1, \dots, p_k\}$ . Sea

$$\bigcup_{i=1}^k \delta(p_i, a) = \{r_1, \dots, r_k\}$$

Entonces tenemos  $\hat{\delta}(q, w) = \{r_1, \dots, r_k\}$

Notar que:

$$\begin{aligned} \hat{\delta}(q, \lambda\alpha) &= \{p \in Q : \exists r \in \hat{\delta}(q, \lambda) \text{ y } p \in \delta(r, a)\} \\ &= \{p \in Q : \exists r \in \{q\} \text{ y } p \in \delta(r, a)\} \\ &= \{p \in Q : p \in \delta(q, a)\} \\ &= \delta(q, a) \end{aligned}$$

### 2.2.2. Lenguaje aceptado por un AFND

**Definición 2.2.2.1:** El lenguaje aceptado por un AFND  $M$ ,  $\mathcal{L}(M)$ , es el conjunto de cadenas aceptadas por  $M$  y está definido como:

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

### 2.2.3. Función de transición de conjuntos de estados

**Definición 2.2.3.1:** Podemos extender la función de transición aún más, haciendo que mapee conjuntos de estados y cadenas en conjuntos de estados. Sea entonces  $\delta : \mathcal{P}(Q) \times \Sigma^* \rightarrow \mathcal{P}(Q)$  dada por:

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

## 2.3. Equivalencia Entre AFND y AFD

**Teorema 2.3.1:** Dado un AFND  $N = \langle Q_N, \Sigma, \delta_N, q_0, F_N \rangle$ , existe un AFD  $D = \langle Q_D, \Sigma, \delta_D, \{q_0\}, F_D \rangle$  tal que  $\mathcal{L}(N) = \mathcal{L}(D)$

*Demostración:* La demostración comienza mediante una construcción llamada *construcción de subconjunto*, llamada así porque construye un autómata a partir de subconjuntos del conjunto de estados de otro (es decir, subconjuntos de  $\mathcal{P}(Q)$ ).

Dado el autómata  $N$  AFND, construimos al autómata  $D$  de la siguiente manera:

- $Q_D$  es el conjunto de subconjuntos de  $Q_N$  (es decir,  $\mathcal{P}(Q_N)$ ). Notar que si  $Q_N$  tenía  $n$  estados,  $Q_D$  va a tener  $2^n$  estados (ouch)
- $F_D$  es el conjunto de subconjuntos  $S$  de  $Q_N$  tal que  $S \cap F_N \neq \emptyset$
- Para cada  $S \subseteq Q_N$  y símbolo  $\alpha \in \Sigma$ :

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Ahora, probamos primero por inducción sobre  $|\omega|$  que vale:

$$\hat{\delta}_D(\{q_0\}, \omega) = \hat{\delta}_N(q_0, \omega)$$

Notar que ambas  $\hat{\delta}$  devuelven un conjunto de estados de  $Q_N$ , pero para nuestro AFD  $D$  el resultado es interpretado como un estado, mientras que en el AFND  $N$  se trata de un subconjunto de estados de  $Q_N$ .

- Caso Base:  $|\omega| = 0$  (O sea,  $\omega = \lambda$ )  
Por la definición de ambos  $\hat{\delta}$  tenemos que ambos son  $\{q_0\}$

- Paso Inductivo: Sea  $\omega$  de longitud  $n + 1$  y asumiendo que la H.I vale para  $n$ . Separamos  $\omega = x\alpha$ , donde  $\alpha$  es el último símbolo de  $\omega$ . Por la H.I, tenemos que  $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) = \{p_1, \dots, p_k\}$ . conjuntos de estados  $\in Q_N$ .

Recordando la definición de AFND teníamos que:

$$\hat{\delta}_N(q_0, \omega) = \bigcup_{i=1}^k \delta_N(p_i, \alpha)$$

Por otro lado, al construir el AFD definimos que:

$$\delta_D(\{p_1, p_2, \dots, p_k\}, \alpha) = \bigcup_{i=1}^k \delta_N(p_i, \alpha)$$

Con esto en mente, podemos resolver:

$$\hat{\delta}_D(\{q_0\}, \omega) = \delta_D(\hat{\delta}_D(\{q_0\}, x), \alpha) \stackrel{HI}{=} \delta_D(\{p_1, \dots, p_k\}, \alpha) = \bigcup_{i=1}^k \delta_N(p_i, \alpha) = \hat{\delta}_N(q_0, \omega)$$

Sabiendo ahora que ambas funciones de transición son "equivalentes" y que ambas aceptan una cadena sii el conjunto resultante contiene un estado final, tenemos que  $\mathcal{L}(N) = \mathcal{L}(D)$   $\square$

## 2.4. AFND con transiciones $\lambda$ (AFND $-\lambda$ )

**Definición 2.4.1:** Un AFND- $\lambda$  es una 5-upla  $\langle Q, \Sigma, \delta, q_0, F \rangle$  donde todos los componentes tienen la misma interpretación que antes con la excepción de que  $\delta$  ahora tiene su dominio definido como:

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$$

**Definición 2.4.2:** Definimos como **clausura  $\lambda$**  de un estado  $q$ , denotado  $Cl_\lambda(q)$ , al conjunto de estados alcanzables desde  $q$  mediante transiciones  $\lambda$ . Es decir, sea  $R \subseteq Q \times Q$  tal que  $(q, p) \in R \iff p \in \delta(q, \lambda)$ . Definimos  $Cl_\lambda : Q \rightarrow \mathcal{P}(Q)$  como:

$$Cl_\lambda(q) = \{p : (q, p) \in R^*\}$$

**Definición 2.4.3:** Podemos también extender la definición para un conjunto de estados  $P$ :

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



**Definición 2.4.4:** La función de transición  $\delta$  puede extenderse para aceptar cadenas en  $\Sigma$ , es decir,  $\hat{\delta} : Q \times \Sigma^* \rightarrow P(Q)$ , y la definimos de la siguiente manera:

- $\hat{\delta}(q, \lambda) = Cl_\lambda(q)$ .
- $\hat{\delta}(q, x\alpha) = Cl_\lambda(\{p : \exists r \in \hat{\delta}(q, x) : p \in \delta(r, \alpha)\})$  con  $x \in \Sigma^*$  y  $\alpha \in \Sigma$  o, lo que es equivalente  

$$\hat{\delta}(q, x\alpha) = Cl_\lambda\left(\bigcup_{r \in \hat{\delta}(q, x)} \delta(r, \alpha)\right)$$

Extendiendo la definición a conjunto de estados, tenemos que:

- $\delta(P, \alpha) = \bigcup_{q \in P} \delta(q, \alpha)$
- $\hat{\delta}(P, x) = \bigcup_{q \in P} \hat{\delta}(q, x)$

Lo que nos permite reescribir  $\hat{\delta}(q, x\alpha)$  como:

$$\hat{\delta}(q, x\alpha) = Cl_\lambda(\delta(\hat{\delta}(q, x), \alpha))$$

**Definición 2.4.5:** Definimos al lenguaje aceptado por un AFND- $\lambda$  E:

$$\mathcal{L}(E) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

**Teorema 2.4.1:** Dado un AFND- $\lambda$   $E = \langle Q, \Sigma, \delta_\lambda, q_0, F_\lambda \rangle$  puede encontrarse un AFND  $N = \langle Q, \Sigma, \delta_N, q_0, F_N \rangle$  que reconoce el mismo lenguaje

*Demostración:* Tomemos

$$F_N = \begin{cases} F_\lambda \cup \{q_0\} & \text{si } Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \\ F_\lambda & \text{si no} \end{cases}$$

Y tomemos  $\hat{\delta}_N(q, \alpha) = \hat{\delta}_\lambda(q, \alpha)$ . Para empezar, vamos a probar que  $\hat{\delta}_N(q, x) = \hat{\delta}_\lambda(q, x)$  para  $|x| \geq 1$ . Lo hacemos por inducción:

- Para  $|x| = 1$ :  
 $x = \alpha$ , es decir, un símbolo del alfabeto, por lo que se cumple por definición
- Para  $|x| > 1$ :  
Tomemos  $x = \omega\alpha$  entonces:

$$\hat{\delta}_N(q_0, \omega\alpha) = \hat{\delta}_N(\hat{\delta}_N(q_0, \omega), \alpha) \stackrel{H.I.}{=} \hat{\delta}_N(\hat{\delta}_\lambda(q_0, \omega), \alpha)$$

Por otro lado, tenemos que para  $P \subseteq Q$  es cierto que  $\hat{\delta}_N(P, \alpha) = \hat{\delta}_\lambda(P, \alpha)$  ya que:

$$\hat{\delta}_N(P, \alpha) = \bigcup_{q \in P} \hat{\delta}_N(q, \alpha) = \bigcup_{q \in P} \hat{\delta}_\lambda(q, \alpha) = \hat{\delta}_\lambda(P, \alpha)$$

Luego, con  $P = \hat{\delta}_\lambda(q_0, \omega)$  tenemos que ambas funciones de transición son "equivalentes" pues:

$$\hat{\delta}_N(q_0, \omega) = \hat{\delta}_N(\hat{\delta}_\lambda(q_0, \omega), \alpha) = \hat{\delta}_\lambda(\hat{\delta}_\lambda(q_0, \omega), \alpha) = \hat{\delta}_\lambda(q_0, \omega\alpha)$$

Ahora queda ver que  $\mathcal{L}(N) = \mathcal{L}(E)$ . Para  $x = \lambda$  tenemos

$$\begin{aligned} \blacktriangleright \lambda \in \mathcal{L}(E) &\iff \hat{\delta}_\lambda(q_0, \lambda) \cap F_\lambda \neq \emptyset \iff_{\hat{\delta}_\lambda(q_0, \lambda) = Cl_\lambda(q_0)} Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \\ &\implies q_0 \in F_N \iff \lambda \in \mathcal{L}(N) \end{aligned}$$

$$\begin{aligned} \blacktriangleright \lambda \in \mathcal{L}(N) &\iff q_0 \in F_N \implies q_0 \in F_\lambda \vee Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \iff \\ &Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \vee \lambda \in \mathcal{L}(E) \iff \lambda \in \mathcal{L}(E) \vee \lambda \in \mathcal{L}(E) \iff \lambda \in \mathcal{L}(E) \end{aligned}$$

Para  $|x| \neq \lambda$ :

$$\begin{aligned} \blacktriangleright x \in \mathcal{L}(E) &\stackrel{\text{def cadena aceptada AFND-}\lambda}{\iff} \hat{\delta}_\lambda(q_0, x) \cap F_\lambda \neq \emptyset \stackrel{\text{porque } F_\lambda \subset F_N}{\implies} \hat{\delta}_\lambda(q_0, x) \cap F_N \neq \emptyset \\ &\implies x \in \mathcal{L}(N) \end{aligned}$$

Por el otro lado:

$$\begin{aligned} \blacktriangleright x \in \mathcal{L}(N) &\iff \hat{\delta}_N(q_0, x) \cap F_N \neq \emptyset \\ &\implies \hat{\delta}_\lambda(q_0, x) \cap F_\lambda \neq \emptyset \vee (\hat{\delta}_\lambda(q_0, x) \cap \{q_0\} \neq \emptyset \wedge Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset) \end{aligned}$$

$\blacktriangleright$  Del lado izquierdo de la disyunción tenemos:

$$\hat{\delta}_\lambda(q_0, x) \cap F_\lambda \neq \emptyset \implies x \in \mathcal{L}(E)$$

$\blacktriangleright$  Del lado derecho tenemos que:

$$- \hat{\delta}_\lambda(q_0, x) \cap \{q_0\} \neq \emptyset \implies \text{existe un loop } x \text{ de } q_0 \text{ sobre sí mismo}$$

$$- Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \implies \text{existe un camino } \lambda \text{ desde } q_0 \text{ hasta } F_\lambda, \\ \text{por lo que existe un camino } x \text{ desde } q_0 \text{ hasta } F_\lambda$$

$\blacktriangleright$  Juntando los dos, tenemos que:

$$(\hat{\delta}_\lambda(q_0, x) \cap \{q_0\} \neq \emptyset \wedge Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset) \implies x \in \mathcal{L}(E)$$

Finalmente, podemos concluir que:

$$x \in \mathcal{L}(E) \iff x \in \mathcal{L}(N)$$

□

## 2.5. Gramáticas regulares y AFDs

Recordemos que una gramática  $G = \langle V_N, V_T, P, S \rangle$  es regular si todas sus producciones son de la forma  $A \rightarrow \lambda$ ,  $A \rightarrow a$  o  $A \rightarrow aB$

**Teorema 2.5.1:** Dada una gramática regular  $G = \langle V_N, V_T, P, S \rangle$ , existe un AFND  $N = \langle Q, \Sigma, \delta, q_0, F \rangle$  tal que  $\mathcal{L}(G) = \mathcal{L}(N)$

*Demostración:* Definamos  $N$  de la siguiente manera:

- $Q = V_N \cup \{q_f\}$ . A partir de ahora, usamos  $q_A$  para referirnos al estado correspondiente al no terminal  $A$
- $\Sigma = V_T$
- $q_0 = q_S$
- $q_B \in \delta(q_A, a) \iff A \rightarrow aB \in P$
- $q_f \in \delta(q_A, a) \iff A \rightarrow a \in P$
- $q_A \in F \iff A \rightarrow \lambda \in P$
- $q_f \in F$

Como paso intermedio, ahora probamos el siguiente lema:

**Lema 2.5.2:** Para todo  $\omega \in V_T^*$ , si  $A \xrightarrow{*} \omega B$  entonces  $q_B \in \hat{\delta}(q_A, \omega)$

*Demostración:* Por inducción en la longitud de  $\omega$

- Caso base  $|\omega| = 0$ , ( $\omega = \lambda$ )

Como  $A \xrightarrow{*} A$  y  $q_A \in \hat{\delta}(q_A, \lambda)$  tenemos por definición de  $N$  que  $A \xrightarrow{*} A \iff q_A \in \hat{\delta}(q_A, \lambda)$

- Caso  $|\omega| = n + 1$ ,  $n \geq 0$ , con,  $\omega = x\alpha$ :

$$\begin{aligned}
 A \xrightarrow{*} x\alpha B &\iff \exists C \in V_N : A \xrightarrow{*} xC \wedge C \rightarrow \alpha B \in P \\
 &\stackrel{HI}{\implies} \exists q_C \in Q : q_C \in \hat{\delta}(q_A, x) \wedge q_B \in \delta(q_C, \alpha) \\
 &\iff q_B \in \delta(\hat{\delta}(q_A, x), \alpha) \\
 &\iff q_B \in \hat{\delta}(q_A, x\alpha)
 \end{aligned}$$

□

Volviendo ahora con el teorema,

$$\begin{aligned}
\omega\alpha \in \mathcal{L}(G) &\stackrel{\text{def lenguaje generado por una gramática}}{\iff} S \xrightarrow{*} \omega\alpha \\
&\iff (\exists A \in V_N, S \xrightarrow{*} \omega A \wedge A \rightarrow \alpha \in P) \vee (\exists B \in V_N, S \xrightarrow{*} \omega\alpha B \wedge B \rightarrow \lambda) \\
&\iff (\exists A \in V_N, S \xrightarrow{*} \omega A \wedge q_f \in \delta(q_A, \alpha)) \vee (\exists B \in V_N, S \xrightarrow{*} \omega\alpha B \wedge q_B \in F) \\
&\stackrel{\text{por lema}}{\implies} (\exists q_A \in Q : q_A \in \hat{\delta}(q_S, \omega) \wedge q_f \in \delta(q_A, \alpha)) \vee (\exists q_B \in Q : q_B \in \hat{\delta}(q_S, \omega\alpha) \wedge q_B \in F) \\
&\iff (q_f \in \hat{\delta}(q_S, \omega\alpha)) \vee (\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset) \\
&\iff (\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset) \vee (\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset) \\
&\iff (\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset) \\
&\iff \omega\alpha \in \mathcal{L}(N)
\end{aligned}$$

Luego, queda demostrado que  $\mathcal{L}(G) = \mathcal{L}(N)$

**Teorema 2.5.3:** Dado un AFD  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  existe una gramática regular  $G = \langle V_N, V_T, P, S \rangle$  tal que  $\mathcal{L}(M) = \mathcal{L}(G)$

*Demostración:* Para comenzar definimos una gramática  $G = \langle V_N, V_T, P, S \rangle$ , donde  $V_N = Q$  y llamaremos  $A_p$  al no terminal correspondiente a  $p \in Q$ ;  $S = A_{q_0}$ ;  $V_T = \Sigma$  y el conjunto  $P$  es:

$$\begin{aligned}
A_p \rightarrow aA_q \in P &\iff \delta(p, a) = q \\
A_p \rightarrow a \in P &\iff \delta(p, a) = q \in F \\
S \rightarrow \lambda \in P &\iff q_0 \in F
\end{aligned}$$

Ahora vamos a probar el siguiente lema como paso intermedio:

**Lema 2.5.4:**  $\delta(p, \omega) = q \iff A_p \xrightarrow{*} \omega A_q$

Lo probamos por induccion en la longitud de  $\omega$ :

Para  $\omega = \lambda$  tenemos que  $\delta(p, \lambda) = p$  y que  $A_p \xrightarrow{*} A_p$ , por lo que  $\delta(p, \lambda) = p \iff A_p \xrightarrow{*} A_p$

Veamos que vale ahora para  $\omega = x\alpha$ :

$$\begin{aligned}
\delta(p, x\alpha) = q &\iff \exists r \in Q, \hat{\delta}(p, x) = r \wedge \delta(r, \alpha) = q \\
&\stackrel{HI}{\iff} \exists A_r, A_p \xrightarrow{*} xA_r \wedge A_r \rightarrow \alpha A_q \in P \\
&\iff A_P \xrightarrow{*} x\alpha A_q
\end{aligned}$$

Luego, probamos el lema. Volviendo ahora a la demo del teorema:

$$\begin{aligned}
\lambda \in \mathcal{L}(M) &\iff q_0 \in F \\
&\iff (S \rightarrow \lambda) \in P \\
&\iff S \xRightarrow{*} \lambda \\
&\iff \lambda \in \mathcal{L}(G).
\end{aligned}$$

Caso cadena no vacía:

$$\begin{aligned}
wa \in \mathcal{L}(M) &\iff \delta(q_0, wa) \in F \\
&\iff \exists p \in Q : \delta(q_0, w) = p \wedge \delta(p, a) \in F \\
&\stackrel{\text{Lema}}{\iff} \exists A_p \in V_N : A_{q_0} \xRightarrow{*} wA_p \wedge (A_p \rightarrow a) \in P \\
&\iff A_{q_0} \xRightarrow{*} wa \\
&\iff wa \in \mathcal{L}(G).
\end{aligned}$$

Con esto queda demostrado que  $\mathcal{L}(M) = \mathcal{L}(G)$ .

□

### 3. Expresiones Regulares

**Definición 3.1:** Una expresión regular es una cadena de símbolos de un alfabeto que denotan un lenguaje sobre el alfabeto. Las expresiones regulares se construyen a partir de los siguientes elementos:

- $\emptyset$  es una expresión regular que representa el lenguaje vacío  $\emptyset$
- $\lambda$  es una expresión regular que representa el lenguaje que contiene sólo la cadena vacía  $\{\lambda\}$
- $a$  es una expresión regular que representa el lenguaje que contiene la cadena  $a$  para cada  $a \in \Sigma(\{a\})$
- Si  $r$  y  $s$  son expresiones regulares que denotan los lenguajes  $R$  y  $S$ , entonces:
  - $(r + s) = (r \mid s)$  es una expresión regular que representa la unión de los lenguajes  $R$  y  $S$ .
  - $(rs)$  es una expresión regular que representa la concatenación de los lenguajes  $R$  y  $S$ .
  - $(r^*)$  es una expresión regular que representa la clausura de Kleene del lenguaje representado por  $r$ .
  - $r^+$  representa la clausura positiva del lenguaje representado por  $r$ .

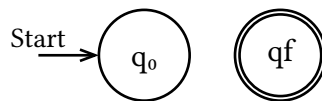
*Ejemplo:* Algunos ejemplos de expresiones regulares son:

- $a + b$  representa el lenguaje que contiene las cadenas  $a$  y  $b$
- $a^*b$  representa el lenguaje que contiene las cadenas que comienzan con una cantidad arbitraria de  $a$  seguida de una  $b$
- $(a + b)^*$  representa el lenguaje que contiene todas las cadenas que contienen únicamente  $a$  y  $b$

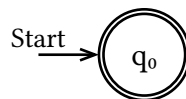
**Teorema 3.1:** Dada una expresión regular  $r$ , existe un AFND- $\lambda$  con un solo estado final y sin transiciones a partir del mismo  $E = \langle Q, \Sigma, \delta, q_0, F \rangle$  tal que  $\mathcal{L}(r) = \mathcal{L}(E)$

*Demostración:* Por inducción sobre la estructura de la expresión regular  $r$ :

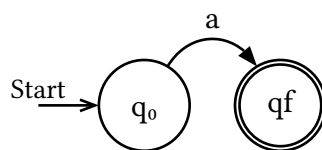
- Caso base:  $r = \emptyset$



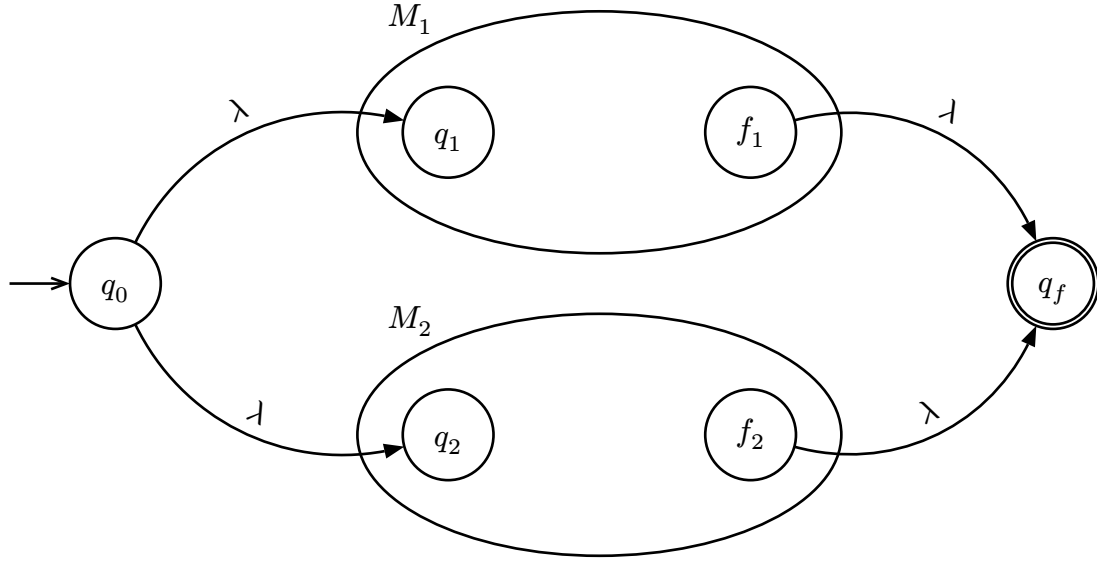
- Caso base:  $r = \lambda$



- Caso base:  $r = a$



- Caso inductivo:  $r = r_1 + r_2$  Por H.I. existen  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\} \rangle$  y  $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, \{f_2\} \rangle$  tales que  $\mathcal{L}(M_1) = \mathcal{L}(r_1)$  y  $\mathcal{L}(M_2) = \mathcal{L}(r_2)$ . Sea  $M = \langle Q_1 \cup Q_2 \cup \{q_0, q_f\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{q_f\} \rangle$ :



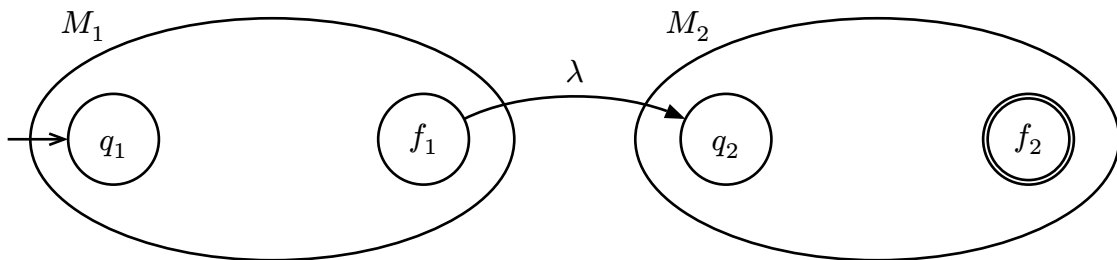
- $\delta(q_0, \lambda) = \{q_1, q_2\}$ .
- $\delta(q, a) = \delta_1(q, a)$  para  $q \in Q_1 - \{f_1\}$  y  $a \in \Sigma_1 \cup \{\lambda\}$ .
- $\delta(q, a) = \delta_2(q, a)$  para  $q \in Q_2 - \{f_2\}$  y  $a \in \Sigma_2 \cup \{\lambda\}$ .
- $\delta(f_1, \lambda) = \delta(f_2, \lambda) = \{q_f\}$ .

De manera informal, podemos alcanzar, a partir del nuevo estado inicial, cualquiera de los estados iniciales correspondientes a los autómatas para  $r_1$  y  $r_2$  mediante transiciones  $\lambda$ . Luego, podemos seguir las transiciones de los autómatas hasta llegar a sus estados previamente finales correspondientes, y luego tomar una última transición  $\lambda$  al nuevo estado final.

- Caso inductivo:  $r = r_1 r_2$

Por HI existen  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\} \rangle$  y  $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, \{f_2\} \rangle$  tales que  $\mathcal{L}(M_1) = \mathcal{L}(r_1)$  y  $\mathcal{L}(M_2) = \mathcal{L}(r_2)$ .

Entonces sea  $M = \langle Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, \{f_2\} \rangle$ :

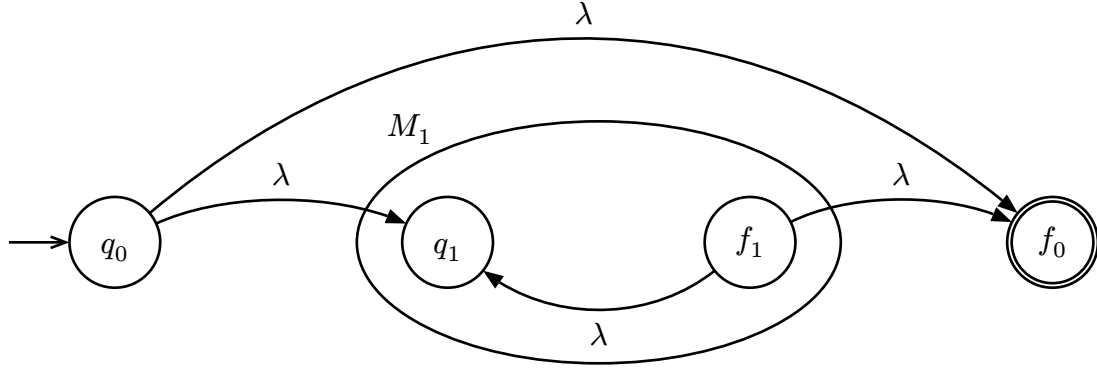


- $\delta(q, a) = \delta_1(q, a)$  para  $q \in Q_1 - \{f_1\}$  y  $a \in \Sigma_1 \cup \{\lambda\}$ .
- $\delta(f_1, \lambda) = \{q_2\}$ .
- $\delta(q, a) = \delta_2(q, a)$  para  $q \in Q_2 - \{f_2\}$  y  $a \in \Sigma_2 \cup \{\lambda\}$ .

- Caso inductivo:  $r = r_1^*$

Por HI existe  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\} \rangle$  tal que  $\mathcal{L}(M_1) = \mathcal{L}(r_1)$ .

Entonces, podemos construir el autómata  $M = \langle Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\} \rangle$ .



- $\delta(q, a) = \delta_1(q, a)$  para  $q \in Q_1 - \{f_1\}$  y  $a \in \Sigma_1 \cup \{\lambda\}$ .
- $\delta(q_0, \lambda) = \delta(f_1, \lambda) = \{q_1, f_0\}$ .

- Caso infuctivo  $r = r_1^+$ :

Dado que  $r_1^+ = r_1 r_1^*$ , queda demostrado por los casos anteriores.

Con esto, queda demostrado que  $\mathcal{L}(r) = \mathcal{L}(M)$  para todo  $r$  expresión regular.

□

**Teorema 3.2:** Dado un AFD  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ , existe una expresión regular  $r$  tal que  $\mathcal{L}(r) = \mathcal{L}(M)$

*Demostración:* Como tenemos que los estados de un autómata son finitos, podemos renombrar los mismos como  $\{q_1, q_2, \dots, q_n\}$  con  $n = |Q|$ . Con esto en cuenta, denotamos con  $R_{i,j}^k$  al conjunto de cadenas que llevan al autómata de  $q_i$  a  $q_j$  sin pasar por ningún estado intermedio con índice mayor que  $k$ . Luego, definimos  $R_{i,j}^k$  inductivamente como:

- $R_{i,j}^0 = \begin{cases} \{a: \delta(q_i, a) = q_j\} & a \in \Sigma \text{ si } i \neq j \\ \{a: \delta(q_i, a) = q_j\} \cup \{\lambda\} & \text{si } i = j \end{cases}$
- $R_{i,j}^k = R_{i,j}^{k-1} \cup R_{i,k}^{k-1} (R_{k,k}^{k-1})^* R_{k,j}^{k-1}$

Como paso intermedio, queremos demostrar que para todo  $R_{i,j}^k$  existe una e.r  $r_{i,j}^k$  tal que  $\mathcal{L}(r_{i,j}^k) = R_{i,j}^k$ . Haciendo inducción sobre  $k$ :

- Caso base:  $k = 0$

$R_{i,j}^0$  es el conjunto de cadenas de un solo caracter o  $\lambda$ . Por lo que la e.r  $r_{i,j}^k$  que lo denota será:

- $\emptyset$  si no existe ningún  $a_i$  que una  $q_i$  y  $q_j$  y  $i \neq j$
- $\lambda$  si no existe ningún  $a_i$  que una  $q_i$  y  $q_j$ , pero con  $i = j$
- $a_1 \mid \dots \mid a_p$  con  $a_l$  simbolos del alfabeto, si  $\delta(q_i, a_l) = q_j$  y  $j \neq i$
- $a_1 \mid \dots \mid a_p \mid \lambda$  con  $a_l$  simbolos del alfabeto, si  $\delta(q_i, a_l) = q_j$  y  $j = i$

- Paso inductivo: Por H.I tenemos que:

$$\mathcal{L}(r_{i,k}^{k-1}) = R_{i,k}^{k-1} \quad \mathcal{L}(r_{k,k}^{k-1}) = R_{k,k}^{k-1} \quad \mathcal{L}(r_{k,j}^{k-1}) = R_{k,j}^{k-1} \quad \mathcal{L}(r_{i,j}^{k-1}) = R_{i,j}^{k-1}$$

Si definimos  $r_{i,j}^k = r_{i,k}^{k-1} (r_{k,k}^{k-1})^* r_{k,j}^{k-1} \mid r_{i,j}^{k-1}$  tenemos que:



$$\begin{aligned}
\mathcal{L}(r_{i,j}^k) &= \\
\mathcal{L}(r_{i,k}^{k-1} (r_{k,k}^{k-1})^* r_{k,j}^{k-1} \mid r_{i,j}^{k-1}) &= \\
\mathcal{L}(r_{i,k}^{k-1} (r_{k,k}^{k-1})^* r_{k,j}^{k-1}) \cup \mathcal{L}(r_{i,j}^{k-1}) &= \\
\mathcal{L}(r_{i,k}^{k-1}) \mathcal{L}((r_{k,k}^{k-1})^*) \mathcal{L}(r_{k,j}^{k-1}) \cup \mathcal{L}(r_{i,j}^{k-1}) &= \\
\mathcal{L}(r_{i,k}^{k-1}) (\mathcal{L}(r_{k,k}^{k-1})^*) \mathcal{L}(r_{k,j}^{k-1}) \cup \mathcal{L}(r_{i,j}^{k-1}) &= \\
R_{i,k}^{k-1} (R_{k,k}^{k-1})^* R_{k,j}^{k-1} \cup R_{i,j}^{k-1} &= \\
R_{i,j}^k &
\end{aligned}$$

Con esto en mente, notemos primero que el lenguaje denotado por el autómata M está dado por todas las cadenas que llevan al autómata de  $q_0$  a un estado final. Es decir:

$$\mathcal{L}(M) = \bigcup_{q_j \in F} R_{1,j}^n$$

Luego, tenemos que:

$$\mathcal{L}(M) = \bigcup_{q_j \in F} \mathcal{L}(R_{1,j}^n) = \bigcup_{q_j \in F} \mathcal{L}(r_{1,j}^n) = \mathcal{L}(r_{1,j_1}^n \mid \dots \mid r_{1,j_m}^n)$$

Por lo que concluimos que  $\mathcal{L}(M)$  es denotado por la e.r  $r_{1,j_1}^n \mid \dots \mid r_{1,j_m}^n$  □

## 4. Lema de Pumping y propiedades de clausura de los lenguajes regulares

### 4.1. Configuración instantánea de un AFD

Para comenzar, vamos a introducir una nueva notación que nos facilitará un par de pruebas

**Definición 4.1.1:** Sea AFD  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ . Una configuración instantánea es un par  $(q, w) \in Q \times \Sigma^*$  donde  $q$  es el estado en el que está el autómata y  $w$  es la cadena que resta consumir

**Definición 4.1.2:** Llamamos transición a la siguiente relación sobre  $Q \times \Sigma^*$ :

$$(q, w) \vdash (p, \beta) \text{ si } (\delta(q, a) = p \text{ y } w = a\beta)$$

De esto tenemos que  $(q, \alpha\beta) \vdash^* (p, \beta) \iff \hat{\delta}(q, \alpha) = p$ . Es decir, se puede llegar al estado  $p$  consumiendo la cadena  $\alpha$

**Lema 4.1.1:** Sea  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  un AFD. Para todo  $q \in Q$  y  $\alpha, \beta \in \Sigma^*$  se tiene que:

$$\text{si } (q, \alpha\beta) \vdash^* (q, \beta) \text{ entonces } \forall i \geq 0, (q, \alpha^i\beta) \vdash^* (q, \beta)$$

*Demostración:* Por inducción sobre  $i$ :

- Caso base:  $i = 0$

$$(q, \alpha^0\beta) \vdash^* (q, \beta)$$

- Paso inductivo:

Supongamos que vale para  $i$ , es decir si  $(q, \alpha\beta) \vdash^* (q, \beta)$  entonces  $(q, \alpha^i\beta) \vdash^* (q, \beta)$ . Veamos que vale para  $i + 1$ :

$$(q, \alpha^{i+1}\beta) = (q, \alpha\alpha^i\beta) \xRightarrow[\text{por el antecedente q asumimos}]{\text{HI}} (q, \alpha\alpha^i\beta) \vdash^* (q, \alpha^i\beta) \vdash^* (q, \beta)$$

□

### 4.2. Lema de Pumping

**Teorema 4.2.1:** Sea  $L$  un lenguaje regular, entonces existe un número  $n$  tal que para toda cadena  $z$  en  $L$  con  $|z| \geq n$ , existen cadenas  $u, v, w$  tales que:

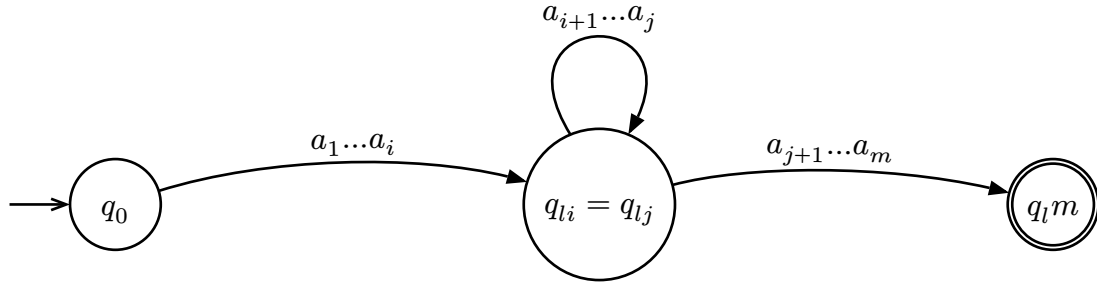
- $z = uvw$ ,
- $|uv| \leq n$ ,
- $|v| \geq 1$

$$\forall i \geq 0, uv^i w \in L$$

*Demostración:* Sea  $M$  un AFD tal que  $\mathcal{L}(M) = L$ . Sea  $n$  su cantidad de estados. Sea  $z$  una cadena de longitud  $m \geq n$ ,  $z = a_1 \dots a_m$  los símbolos que forman la cadena.

Para aceptar  $z$  usamos  $m$  transiciones, por lo tanto pasamos a través de  $m + 1$  estados. Como  $m + 1 > n$ , tenemos que necesariamente debemos pasar al menos dos veces por un mismo estado para aceptar la cadena (pigeonhole principle).

Sea entonces  $q_{l_0} \dots q_{l_m}$  la sucesión de estados desde  $q_0 (q_{l_0})$  hasta un estado final ( $q_{l_m}$ )



Existen entonces  $j$  y  $k$  mínimos tales que  $q_{l_j} = q_{l_k}$  con  $0 \leq j < k \leq n$ . Esto separa a  $z$  en tres subcadenas:

- $u = \begin{cases} a_1 \dots a_j & \text{si } j > 0 \\ \lambda & \text{si } j = 0 \end{cases}$
- $v = a_{j+1} \dots a_k$
- $w = \begin{cases} a_{k+1} \dots a_m & \text{si } k < m \\ \lambda & \text{si } k = m \end{cases}$

Juntando todo esto, tenemos que:

$$\begin{aligned} |uv| &\leq n \\ |v| &\geq 0 \end{aligned}$$

y que:

$$(q_0, uvw) \vdash^* (q_{l_j}, vw) \vdash^* (q_{l_k}, w) \vdash^* (q_{l_m}, \lambda)$$

Pero como  $q_{l_j} = q_{l_k}$ , y por el lema probado en la sección anterior, tenemos que:

$$\forall i \geq 0 (q_{l_j}, v^i w) \vdash^* (q_{l_j}, w) = (q_{l_k}, w) \text{ que tenemos alcanza un estado final}$$

Por lo tanto,  $uv^i w \in L$ ,  $\forall i \geq 0$

□

*Ejemplo:* Sea AFD  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ , con  $|Q| = n$ . Determinar veracidad y justificar:

1.  $\mathcal{L}(M)$  es no vacío si y solo si existe  $w \in \Sigma^*$  tal que  $\hat{\delta}(q_0, w) \in F$  y  $|w| < n$

*Demostración:*

- $\Leftarrow$ ) Es trivial ver que el lenguaje no es vacío
- $\Rightarrow$ ) Supongamos que el lenguaje no es vacío y regular. Entonces, supongamos existe una cadena  $z \in \mathcal{L}(M)$ . Hay dos posibilidades, o bien la longitud de la cadena es menor a  $n$ , o bien es mayor. En el primer caso, no hace falta demostrar nada más. En el segundo caso, por el lema de pumping, podemos descomponer la cadena en  $uvw$  tal que  $|uv| \leq n$  y  $|v| \geq 1$  y estar seguros que para todo  $i$  se cumple que  $uv^i w \in \mathcal{L}(M)$ . Luego por el lema de pumping, podemos "pumpear" a  $v$  con  $i = 0$ , reduciendo el tamaño de la cadena y repitiendo este proceso hasta llegar a una con longitud menor a  $n$ .

□

2.  $\mathcal{L}(M)$  es infinito si y solo si existe  $w \in \Sigma^*$  tal que  $\hat{\delta}(q_0, w) \in F$  y  $n \leq |w| < 2n$

*Demostración:*

- $\Leftarrow$ ) Suponemos  $z \in \mathcal{L}(M)$  y  $n \leq |z| < 2n$ . Por el lema de Pumping, tenemos  $z = uvx$  con  $|uv| \leq n$  y  $|v| \geq 1$  y  $uv^i x \in \mathcal{L}(M)$  para todo  $i$ . Luego  $\mathcal{L}(M)$  es infinito
- $\Rightarrow$ ) Supongamos  $\mathcal{L}(M)$  es infinito y regular. Supongamos también que no existe una cadena  $z$  en el lenguaje con longitud entre  $n$  y  $2n - 1$ . Primero notemos que como el lenguaje es infinito, necesariamente debe haber cadenas de longitud mayor a  $n$  (Pues lo único que puede aportar a la "infinitud" es que el tamaño de las cadenas pueda ser arbitrariamente grandes)

Sin pérdida de generalidad, supongamos que  $|z| = 2n$  (Notar que si la longitud fuese mayor, simplemente podríamos bombear hacia repetir el argumento hasta que se cumpla esta condición o se llegue al rango deseado. Notar también que no es posible "saltarse" el rango pues el mismo es de longitud  $n + 1$ , y saltarlo implicaría un ciclo que abarca más que la cantidad de estados del autómata)

Por Lema de Pumping, tenemos que existen  $u, v, x$  tales que  $z = uvx$ , con  $|uv| \leq n$ ,  $|v| \geq 1$  y  $\forall i, uv^i x \in \mathcal{L}(M)$ . En particular,  $uv^0 x = ux$  está en el lenguaje.

Como  $|uvx| = 2n$  y  $1 \leq |v| \leq n$  tenemos que  $n \leq |ux| \leq 2n - 1$ , contradiciendo nuestra suposición que dicha cadena no existía.

□

## 4.3. Propiedades de clausura de los lenguajes regulares

### 4.3.1. Unión

**Teorema 4.3.1.1:** El conjunto de lenguajes regulares incluidos en  $\Sigma^*$  es cerrado respecto de la unión

*Demostración:* Sean  $L_1$  y  $L_2$  lenguajes regulares. Sea  $M_1 = \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle$  tal que  $\mathcal{L}(M_1) = L_1$  y  $M_2 = \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle$  tal que  $\mathcal{L}(M_2) = L_2$  y  $Q_1 \cap Q_2 = \emptyset$ . definimos  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  tal que:

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$  para  $q \in Q_1, r \in Q_2, a \in \Sigma$
- $F = \{(p, q) : p \in F_1 \vee q \in F_2\}$

con  $\mathcal{L}(M) = L_1 \cup L_2$ .

Ahora queda demostrar la pertenencia

$$\begin{aligned} x \in \mathcal{L}(M) &\iff \delta((q_{1_0}, q_{2_0}), x) \in F \\ &\iff (\delta_1(q_{1_0}, x), \delta_2(q_{2_0}, x)) \in F \\ &\iff \delta_1(q_{1_0}, x) \in F_1 \vee \delta_2(q_{2_0}, x) \in F_2 \\ &\iff x \in \mathcal{L}(M_1) \vee x \in \mathcal{L}(M_2). \end{aligned}$$

□

**(Si no me equivoco, acá falta demostrar la equivalencia para  $\hat{\delta}$ , si no no se puede asumir nada sobre la aplicación de la misma a cadenas. No debería ser difícil, pero si llego debería corregir esto)**

Una demostración alternativa es hacer uso de las expresiones regulares. Dados  $r_1$  y  $r_2$  expresiones regulares que denotan los lenguajes  $L_1$  y  $L_2$ , respectivamente, podemos construir una expresión regular  $r = r_1 \mid r_2$  que denota la unión de los lenguajes, por definición.

#### 4.3.2. Concatenación y clausura de Kleene

**Teorema 4.3.2.1:** El conjunto de lenguajes regulares incluidos en  $\Sigma^*$  es cerrado respecto de la concatenación y la clausura de Kleene

**(La demo más aceptada es hacer uso nuevamente de expresiones regulares, ya que por definición trivializan la prueba, pero me gustaría demostrar también que los autómatas que armamos son válidos, considerar agregarla / hacerla como ejercicio)**

#### 4.3.3. Complemento

**Teorema 4.3.3.1:** El conjunto de lenguajes regulares incluidos en  $\Sigma^*$  es cerrado respecto del complemento

*Demostración:* Sea  $L = \mathcal{L}(M)$  para algún AFD  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  cuya función de transición  $\delta$  está completamente definida. Tenemos entonces que el autómata:

$$M' = \langle Q, \Sigma, \delta, q_0, Q - F \rangle$$

Como tenemos que ahora todas las cadenas que no eran aceptadas por  $M$  lo son, y viceversa, tenemos que  $\mathcal{L}(M') = \Sigma^* - L = \bar{L}$ .  $\square$

#### 4.3.4. Intersección

**Teorema 4.3.4.1:** El conjunto de lenguajes regulares incluidos en  $\Sigma^*$  es cerrado respecto de la intersección

*Demostración:* Sea  $L_1$  y  $L_2$  lenguajes regulares. Sea  $M_1 = \langle Q_1, \Sigma, \delta_1, q_{0_1}, F_1 \rangle$  tal que  $\mathcal{L}(M_1) = L_1$  y  $M_2 = \langle Q_2, \Sigma, \delta_2, q_{0_2}, F_2 \rangle$  tal que  $\mathcal{L}(M_2) = L_2$ . Definamos  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  tal que:

- $Q = Q_1 \times Q_2$
- $q_0 = (q_{0_1}, q_{0_2})$
- $\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$  para  $q \in Q_1, r \in Q_2, a \in \Sigma$
- $F = F_1 \times F_2 = \{(p, q) : p \in F_1 \wedge q \in F_2\} =$

Con una inducción sobre  $|w|$  es fácil ver que  $\hat{\delta}((q, r), w) = (\hat{\delta}_1(q, w), \hat{\delta}_2(r, w))$ . Con esto en cuenta, tenemos que:

$$\begin{aligned} w \in \mathcal{L}(M) &\iff \hat{\delta}((p, r), w) \in F \\ &\iff (\hat{\delta}_1(p, w), \hat{\delta}_2(r, w) \in F_1 \times F_2) \\ &\iff (\hat{\delta}_1(p, w) \in F_1) \wedge (\hat{\delta}_2(r, w) \in F_2) \\ &\iff w \in L_1 \wedge w \in L_2 \end{aligned}$$

$\square$

#### 4.4. Unión e Intersección finitos de lenguajes regulares

**Teorema 4.4.1:**  $\forall n \in \mathbb{N}, \bigcup_{i=1}^n L_i$  es regular

**Teorema 4.4.2:**  $\forall n \in \mathbb{N}, \bigcap_{i=1}^n L_i$  es regular

*Demostración:* Demostramos la unión por inducción sobre  $n$ , la intersección es similar:

- Caso base:  $n = 0$

$\bigcup_{i=1}^0 L_i = \emptyset$ , que sabemos es regular

- Paso inductivo:

Supongamos que  $\bigcup_{i=1}^n L_i$  es regular. Sea  $L_{\{n+1\}}$  un lenguaje regular.

Entonces, por la propiedad de clausura de la unión, tenemos que  $\bigcup_{i=1}^{n+1} L_i = \bigcup_{i=1}^n L_i \cup L_{\{n+1\}}$  es regular.

□

**Lema 4.4.3:** Los lenguajes regulares no están clausurados por union infinita

*Demostración:* Sea  $L_i = \{a^i b^i\}$ , si lo estuvieran, entonces  $\bigcup_{i=1}^{\infty} L_i$  sería regular, pero:

$$\bigcup_{i=1}^{\infty} L_i = \bigcup_{i=1}^{\infty} \{a^i b^i\} = \{a^k b^k : k \in \mathbb{N}\}, \text{ que sabemos no es regular}$$

□

## 4.5. Todo lenguaje finito es regular

**Teorema 4.5.1:** Todo lenguaje finito es regular

*Demostración:* Sea  $L$  un lenguaje finito, con  $n$  cadenas  $\{w_1, w_2, \dots, w_n\}$ . Para cada  $1 \leq i \leq n$ , sea  $L_i = \{w_i\}$

Entonces  $L = \bigcup_{i=1}^n L_i$ , que sabemos es regular pues cada  $L_i$  lo es

□

## 4.6. Problemas de decibilidad de lenguajes regulares

### 4.6.1. Vacuidad

**Teorema 4.6.1.1:** El problema de la vacuidad de un lenguaje regular es decible

*Demostración:* suponiendo que la representación del lenguaje está dada mediante un AFD (notar que siempre se puede convertir de una representación a otra, y en esta materia estamos ignorando complejidad, porque el plan nuevo de la carrera es **muy bueno**)

- Se determina el conjunto de estados alcanzables desde el estado inicial
- Si ninguno de los estados alcanzables es final, entonces el lenguaje es vacío

□

### 4.6.2. Pertenencia

**Teorema 4.6.2.1:** El problema de la pertenencia de un lenguaje regular es decible

*Demostración:* Suponiendo que la representación del lenguaje está dada mediante un AFD, se puede determinar si una cadena pertenece simplemente determinando si la misma es aceptada por el autómata

□

#### 4.6.3. Finititud

**Teorema 4.6.3.1:** El problema de la finitud de un lenguaje regular es decidable

*Demostración:* Anteriormente demostramos cual era la condición suficiente y necesaria para que un lenguaje fuese infinito  $\square$

#### 4.6.4. Equivalencia

**Teorema 4.6.4.1:** El problema de la equivalencia de dos lenguajes regulares es decidable

*Demostración:*  $L_1 \Delta L_2 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset \iff L_1 = L_2$   $\square$

(El libro propone una demostración más constructiva y eficiente, introduciendo el concepto de equivalencia de estados)



## 5. Gramáticas Libres de Contexto

**Definición 5.1:** Recordemos que, llamamos a una gramática  $G = \langle V_N, V_T, P, S \rangle$  libre de contexto si las producciones  $P$  son de la forma:

$$A \rightarrow \alpha \text{ con } A \in V_N \text{ y } \alpha \in (V_N \cup V_T)^*$$

### 5.1. Derivación

Si  $\alpha, \beta, \gamma_1, \gamma_2 \in (V_N \cup V_T)^*$  y  $\alpha \rightarrow \beta \in P$  entonces:

$$\gamma_1 \alpha \gamma_2 \Rightarrow \gamma_1 \beta \gamma_2$$

La relación  $\Rightarrow$  es un subconjunto de  $(V_N \cup V_T)^* \times (V_N \cup V_T)^*$  y significa derivar en un solo paso

Las relaciones  $\xRightarrow{*}$  y  $\xRightarrow{+}$  son las clausuras reflexiva y transitiva de  $\Rightarrow$ , respectivamente

Si  $\alpha \in (V_N \cup V_T)^*$  y  $S \xRightarrow{*} \alpha$  decimos que la misma es una forma sentencial de  $G$ .

### 5.2. Inferencia

**Definición 5.2.1:** Una manera alternativa de determinar la pertenencia de una cadena es partir desde el cuerpo de las producciones, identificando las cadenas que ya sabemos pertenecen al lenguaje, y concatenandolas formando así cadenas más complejas

### 5.3. Lenguaje de una Gramática

**Definición 5.3.1:** El lenguaje de una gramática  $G$ , denotado  $\mathcal{L}(G)$  es el conjunto de todas las cadenas que pueden ser derivadas desde el símbolo inicial  $S$  de  $G$ . Es decir:

$$\mathcal{L}(G) = \left\{ w \in V_T^* : S \xRightarrow{+} w \right\}$$

### 5.4. árbol de derivación

**Definición 5.4.1:** Un árbol de derivación para una gramática  $G$  es un árbol tal que:

- Sus hojas están etiquetadas con símbolos de  $V_T$ ,  $V_N$  o  $\lambda$ , caso en el cual el mismo debe ser el único hijo de su padre.
- Sus nodos internos están etiquetados con símbolos de  $V_N$ .
- La raíz está etiquetada con el símbolo inicial  $S$
- Si un nodo interno está etiquetado con  $A$  y sus hijos están etiquetados con  $X_1, X_2, \dots, X_n$  entonces  $A \rightarrow X_1 X_2 \dots X_n$  es una producción de  $G$

El árbol de derivación correspondiente a una variable  $A$  lo notaremos con  $\mathcal{T}(A)$

**Definición 5.4.2:** Llamamos producto (yield) de un árbol de derivación a la cadena que se obtiene al concatenar las etiquetas de las hojas de izquierda a derecha. En particular, tenemos que aquellos árboles que cumplen que:

- Todas las hojas están etiquetadas con símbolos de  $V_T$  o con  $\lambda$
- La raíz está etiquetada con el símbolo inicial  $S$

Son los árboles cuyo producto son las cadenas en el lenguaje de la gramática correspondiente

**Definición 5.4.3:** Llamamos camino de  $x$  en un árbol  $\mathcal{T}(A)$  a la cadena  $A, X_1, \dots, X_k, x$  tal que:

$$A \Rightarrow \dots X_1 \dots \Rightarrow \dots X_2 \dots \Rightarrow \dots \Rightarrow \dots X_k \dots \Rightarrow x$$

**Definición 5.4.4:** Llamamos altura de  $\mathcal{T}(A)$  a:

$$\{\max\{|ax| : x \text{ es una hoja de } \mathcal{T}(A) \text{ y } Aax \text{ es un camino de } x\}\}$$

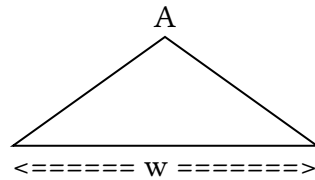
## 5.5. Equivalencia inferencia, árbol Derivación y Derivaciones

**Teorema 5.5.1:** Sea  $G$  una GLC. Si el procedimiento de inferencia recursivo determina que una cadena  $w$  con sólo terminales esta en el lenguaje de una gramática definido por una variable, entonces hay un árbol de derivación con raíz en esa variable que produce  $w$

*Demostración:* Por inducción en la cantidad de pasos utilizados para inferir que  $w$  esta en el lenguaje de la variable  $A$ :

- Caso base: Un sólo paso.

En este caso, tenemos que  $A \rightarrow w$  debe ser una producción de la gramática, y por lo tanto, el árbol de derivación es simplemente un nodo con raíz en  $A$  y hojas etiquetadas con los símbolos de  $w$ .



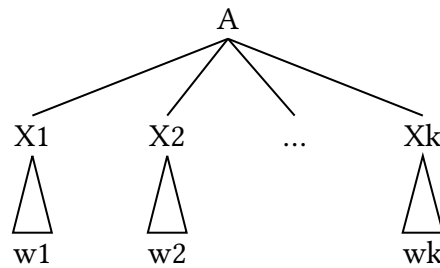
- Paso inductivo:

Supongamos que el hecho de que  $w$  esté en el lenguaje es inferido luego de  $n + 1$  pasos, y que el teorema vale para todas las cadenas  $x$  y variables  $B$  tal que su pertenencia fue determinada con  $n$  o menos pasos. Consideremos el último paso de la inferencia utilizada para determinar que  $w$  está en el lenguaje de  $A$ . Por definición, la inferencia usa alguna producción de  $A$ , supongamos que es  $A \rightarrow X_1 X_2 \dots X_k$ , donde cada  $X_i$  es o bien una variable o un terminal. Podemos entonces tomar  $w = w_1 \dots w_k$  como:

- Si  $X_i$  es un terminal, entonces  $X_i = w_i$ , es decir,  $w_i$  solo consiste de este terminal en la producción

- Si  $X_i$  es una variable, entonces  $w_i$  es una cadena que fue inferida con  $n$  pasos que está en el lenguaje de esa variable, y por hipótesis inductiva, sabemos que hay un árbol de derivación con raíz en  $X_i$  que produce  $w_i$

Con esto, podemos armar el siguiente árbol de derivación, notar que en el primer caso los subárboles correspondientes son triviales de un solo nodo:



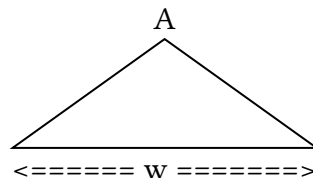
□

**Teorema 5.5.2:** Sea  $G$  una GLC. Si hay un árbol de derivación con raíz en una variable  $A$  que produce una cadena  $w$ , tal que  $w \in V_T^*$ . Entonces hay una derivación a izquierda  $A \xRightarrow{*}_L w$  en la gramática

*Demostración:* Por inducción en la altura del árbol de derivación:

- Caso base: Altura 1

En este caso, tenemos que el árbol de derivación es simplemente un nodo con raíz en  $A$  y hojas etiquetadas con los símbolos de  $w$ . Como se trata de un árbol de derivación,  $A \rightarrow w$  debe ser una producción, por lo que la derivación es simplemente  $A \xRightarrow{*}_L w$



- Paso inductivo:

Supongamos que el árbol de derivación tiene altura  $n + 1$ . Entonces, la raíz del árbol de derivación es  $A$ , y tiene hijos  $X_1, X_2, \dots, X_k$ :

- Si  $X_i$  es un terminal, definimos  $w_i$  como la cadena que consiste solo de  $X_i$ , es decir  $w_i = X_i$
- Si  $X_i$  es una variable, entonces debe ser la raíz para algún subárbol con una producción de sólo terminales (pues si no no se cumpliría el precedente), a la que identificamos como  $w_i$ . Notar que por hipótesis inductiva, sabemos que hay una derivación a izquierda  $X_i \xRightarrow{*}_L w_i$ .

Ahora podemos armar la derivación a izquierda  $A \xRightarrow{*}_L w$  de la siguiente manera:

$$A \xRightarrow{L} X_1 X_2 \dots X_k \xRightarrow{L}^* w_1 X_2 \dots X_k \xRightarrow{L}^* w_1 w_2 \dots X_k \xRightarrow{L}^* w_1 w_2 \dots w_k = w$$

□

**Teorema 5.5.3:** Sea  $G$  una GLC. Si hay una derivación  $A \xRightarrow{*} w$  en la gramática, entonces el procedimiento de inferencia recursivo determina que  $w$  está en el lenguaje de la variable  $A$

*Demostración:* Por inducción en la cantidad de pasos de la derivación:

- Caso base: Un solo paso

En este caso, tenemos que  $A \rightarrow w$  debe ser una producción de la gramática, y por lo tanto, el procedimiento de inferencia recursivo determina que  $w$  está en el lenguaje de  $A$  de manera inmediata.

- Paso inductivo:

Supongamos que la derivación tiene  $n + 1$  pasos, y que la hipótesis vale para cualquier derivación con menos de  $n$  pasos. Escribiendo la primera derivación de la forma  $A \Rightarrow X_1 X_2 \dots X_k$ , separamos a  $w = w_1 \dots w_k$  tal que:

- Si  $X_i$  es un terminal, entonces  $X_i = w_i$ , es decir,  $w_i$  solo consiste de este terminal en la producción
- Si  $X_i$  es una variable, entonces  $X_i \Rightarrow w_i$ . Como sabemos que  $A \xRightarrow{*} w$  no es parte de esta derivación, (pues la misma tiene más pasos) tenemos que la misma tiene menos de  $n$  pasos, por lo que, por H.I sabemos que se puede inferir que  $w_i$  está en  $X_i$

Por lo tanto, tenemos una producción de la forma  $A \rightarrow X_1 X_2 \dots X_k$ , con cada  $w_i$  equivalente a  $X_i$  o en el lenguaje del mismo, por lo tanto, el procedimiento de inferencia recursivo determina en el siguiente paso que  $w$  está en el lenguaje de  $A$

□

## 5.6. Gramáticas Ambiguas

**Definición 5.6.1:** Una gramática es ambigua si existe al menos una cadena  $w \in V_T^*$  para la cual haya al menos dos árboles de derivación, ambos con raíz  $S$  y que produzcan  $w$

**Teorema 5.6.1:** Para toda gramática  $G$  y cadena  $w \in T^*$ ,  $w$  tiene dos árboles de derivación distintos  $\iff w$  tiene dos derivaciones a izquierda distintas a partir de  $S$

*Demostración:*

□

**Definición 5.6.2:** Un Lenguaje libre de contexto es inherentemente ambiguo si no existe una gramática no ambigua que genere el lenguaje

## 6. Autómatas de Pila

**Definición 6.1:** Un autómata de pila es una tupla  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$  donde:

- $Q$  es un conjunto finito de estados
- $\Sigma$  es un alfabeto finito de entrada
- $\Gamma$  es un alfabeto finito de la pila
- $q_0 \in Q$  es el estado inicial
- $Z_0 \in \Gamma$  es la configuración inicial de la pila
- $F \subseteq Q$  es el conjunto de estados finales
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$  es la función de transición

La interpretación de  $\delta(q, x, Z) = \{(p_1, \gamma_1), \dots, (p_n, \gamma_n)\}$  es como sigue:

Cuando el estado del autómata es  $q$ , el símbolo que la cabeza lectora está inspeccionando en ese momento es  $x$ , y en el tope de la pila hay  $Z$ , se realizan las siguientes acciones:

- Si  $x \in \Sigma$  ( $\neq \lambda$ ), la cabeza lectora avanza una posición para inspeccionar el siguiente símbolo
- Se elimina el símbolo  $Z$  del tope de la pila
- Se selecciona un par  $(p_i, \gamma_i)$  entre los existentes
- Se apila la cadena  $\gamma_i = c_1 c_2 \dots c_k$ , con  $c_i \in \Gamma$  en la pila del autómata, con el símbolo  $c_1$  quedando en el tope
- Se cambia el control del autómata al estado  $p_i$

### 6.1. Configuración de un AP

**Definición 6.1.1:** Una configuración instánea de un AP es una terna  $(q, w, \gamma)$  donde:

- $q$  es el estado actual del autómata
- $w$  es la cadena de entrada que queda por leer
- $\gamma$  es el contenido de la pila

La configuración inicial de un AP para una cadena  $w_0$  es  $(q_0, w_0, Z_0)$

**Definición 6.1.2:** Representamos al cambio entre configuraciones instantáneas de un autómata tal que, para todo  $x \in \Sigma, w \in \Sigma^*, Z \in \Gamma, \gamma, \pi \in \Gamma^*, q, p \in Q$ :

- $(q, xw, Z\pi) \vdash (p, w, \gamma\pi)$  si  $(p, \gamma) \in \delta(q, x, Z)$
- $(q, w, Z\pi) \vdash (p, w, \gamma\pi)$  si  $(p, \gamma) \in \delta(q, \lambda, Z)$

**Teorema 6.1.1:** Información que el AP no utiliza no afecta su comportamiento. Formalmente, sea  $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$  un AP, y  $(q, x, \alpha) \vdash^* (p, v, \beta)$ , entonces, para toda cadena  $w \in \Sigma^*$  y  $\gamma \in \Gamma^*$ :

$$(q, xw, \alpha\gamma) \vdash^* (p, vw, \beta\gamma)$$

*Demostración:* Se trata de una simple inducción sobre la cantidad de pasos en los cambios de configuración □

## 6.2. Lenguaje de un AP

**Definición 6.2.1:** El lenguaje de un AP  $P$ , denotado  $\mathcal{L}(P)$  es el conjunto de todas las cadenas que consume y a la vez alcanzan un estado final, es decir:

$$\mathcal{L}(P) = \left\{ w \in \Sigma^* : \exists (p \in F, \gamma \in \Gamma^*) \ (q_0, w, Z_0) \vdash^* (p, \lambda, \gamma) \right\}$$

**Definición 6.2.2:** El lenguaje reconocido por  $P$  por pila vacía es:

$$\mathcal{L}_{\lambda(P)} = \left\{ w \in \Sigma^* : \exists (p \in Q) \ (q_0, w, Z_0) \vdash^* (p, \lambda, \lambda) \right\}$$

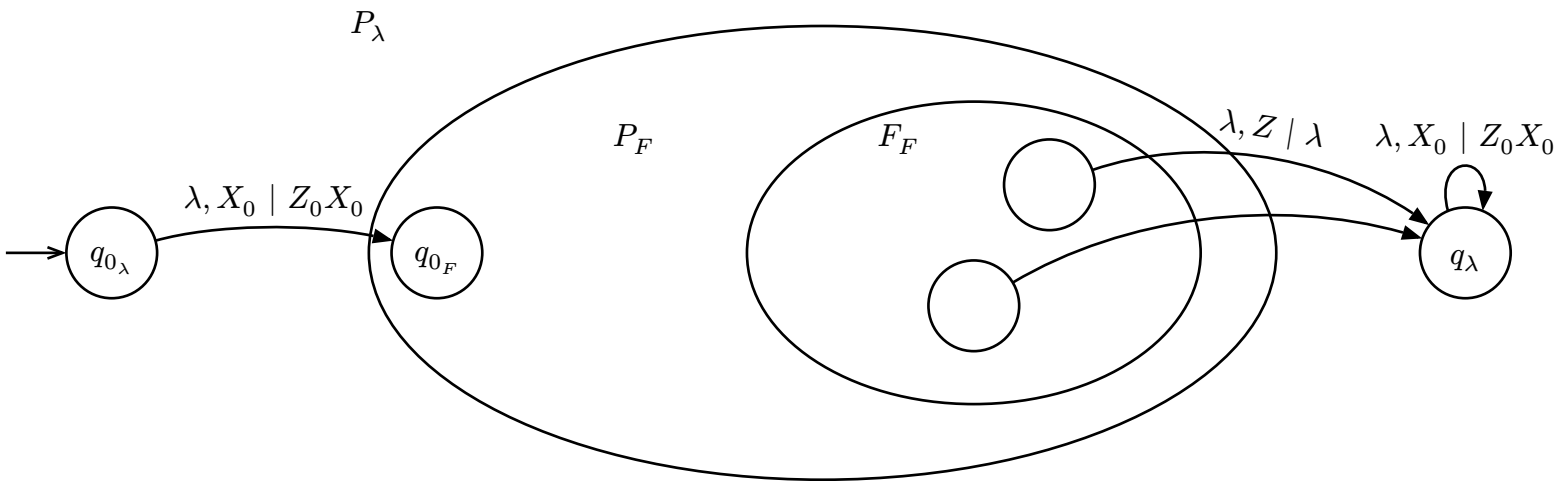
## 6.3. Equivalencia $\mathcal{L}(P)$ y $\mathcal{L}_{\lambda}(P)$

**Teorema 6.3.1:** Para todo AP  $P_F = \langle Q_F, \Sigma, \Gamma_F, \delta_F, q_{0_F}, Z_0, F_F \rangle$ , existe un AP  $P_{\lambda}$  tal que:

$$\mathcal{L}(P_F) = \mathcal{L}_{\lambda}(P_{\lambda})$$

*Demostración:* Definimos  $P_{\lambda} = \langle Q_F \cup \{q_{0_{\lambda}}, q_{\lambda}\}, \Sigma, \Gamma \cup \{X_0\}, \delta_{\lambda}, q_{0_{\lambda}}, \emptyset \rangle$  tal que:

1.  $\delta_{\lambda}(q_{0_{\lambda}}, w, X_0) = \{(q_{0_F}, Z_0 X_0)\}$
2.  $\forall (q \in Q_F, x \in \Sigma \cup \{\lambda\}, Z \in \Gamma_F), \delta_{\lambda}(q, x, Z) = \delta_F(q, x, Z)$
3.  $\forall (q \in F_F, Z \in \Gamma_F \cup \{X_0\}), (q_{\lambda}, \lambda) \in \delta_{\lambda}(q, \lambda, Z)$
4.  $\forall (Z \in \Gamma_F \cup \{X_0\}), (q_{\lambda}, \lambda) \in \delta_{\lambda}(q_{\lambda}, \lambda, Z)$



Queremos ahora ver que  $w \in \mathcal{L}(P_F) \iff w \in \mathcal{L}_{\lambda}(P_{\lambda})$ :

- $\implies$ ) Si  $w \in \mathcal{L}(P_F)$  entonces  $(q_{0_F}, w, Z_0) \vdash_{P_F}^* (q, \lambda, \gamma)$ , con  $q \in F_F, \gamma \in \Gamma^*$

Por la regla 1 de  $\delta_{\lambda}$ ,  $\delta_{\lambda}(q_{0_{\lambda}}, \lambda, X_0) = \{(q_{0_F}, Z_0 X_0)\}$ , por lo que:

$$q_{0_\lambda}, w, X_0 \vdash_{P_\lambda} (q_{0_F}, w, Z_0 X_0)$$

Por la regla 2 de  $\delta_\lambda$ , tenemos que todas las transiciones de  $P_F$  están en  $P_\lambda$  (lo "simula"), por lo que:

$$(q_{0_F}, w, Z_0) \vdash_{P_\lambda}^* (q, \lambda, \gamma)$$

Entonces:

$$(q_{0_\lambda}, w, X_0) \vdash_{P_\lambda} (q_{0_F}, w, Z_0 X_0) \vdash_{P_\lambda}^* (q, \lambda, \gamma X_0)$$

Nuevamente por las reglas 3 y 4  $\delta_\lambda$ , para todo estado final  $q \in F_F$  y  $Z \in \Gamma \cup \{X_0\}$  tenemos que una vez que se alcanza un estado final en  $P_F$ , se puede llegar a  $P_\lambda$  con la pila vacía, por lo que:

$$(q, \lambda, \gamma X_0) \vdash_{P_\lambda} (q_\lambda, \lambda, \gamma X_0)$$

Y que, una vez en  $q_\lambda$ , se puede llegar a la pila vacía sin importar el tope de la pila, por lo que:

$$(q_\lambda, \lambda, \gamma) \vdash_{P_\lambda}^* (q_\lambda, \lambda, \lambda)$$

Juntando todo, tenemos:

$$(q_{0_\lambda}, w, X_0) \vdash_{P_\lambda} (q_{0_F}, w, Z_0 X_0) \vdash_{P_\lambda}^* (q, \lambda, \gamma X_0) \vdash_{P_\lambda} (q_\lambda, \lambda, \gamma X_0) \vdash_{P_\lambda}^* (q_\lambda, \lambda, \lambda)$$

Por lo que si  $w \in \mathcal{L}(P_F)$ , entonces  $w \in \mathcal{L}(P_\lambda)$

- $\Leftarrow$ ) Tenemos que  $w \in \mathcal{L}(P_\lambda)$ . Sabemos por la definición de  $\delta_\lambda$  que la única manera de que  $P_\lambda$  vacíe su pila es mediante el estado  $p_\lambda$  (pues es el único estado que puede identificar  $X_0$ ), además, sabemos que la única manera en la que  $P_\lambda$  alcance este estado, es si el autómata simulado alcanza un estado final de  $P_F$ . Finalmente, sabemos que el primer movimiento del autómata es siempre saltar al autómata simulado, por lo que tenemos:

$$(q_{0_\lambda}, w, X_0) \vdash_{P_\lambda} \underbrace{(q_{0_F}, w, Z_0 X_0) \vdash_{P_\lambda}^* (q, \lambda, \gamma X_0) \vdash_{P_\lambda} (q_\lambda, \lambda, \gamma X_0) \vdash_{P_\lambda}^* (q_\lambda, \lambda, \lambda)}_A$$

Pero la transición en A necesariamente implica que (pues la misma no hace uso de  $X_0$ ):

$$(q_{0_F}, w, Z_0) \vdash_{P_F}^* (q, \lambda, \gamma)$$

Por lo que si  $w \in \mathcal{L}(P_\lambda)$ , entonces  $w \in \mathcal{L}(P_F)$  □

**Teorema 6.3.2:** Para todo AP  $P_\lambda = \langle Q_\lambda, \Sigma, \Gamma_\lambda, \delta_\lambda, q_{0_\lambda}, X_0, \emptyset \rangle$ , existe un AP  $P_F$  tal que:

$$\mathcal{L}_\lambda(P_\lambda) = \mathcal{L}(P_F)$$

*Demostración:* Definimos  $P_F = \langle Q_\lambda \cup \{q_{0_F}, q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, q_{0_\lambda}, F_\lambda \rangle$  tal que:

1.  $\delta(q_{0_F}, \lambda, Z_0) = \{(q_{0_\lambda}, X_0 Z_0)\}$
2.  $\forall (q \in Q_\lambda, x \in \Sigma \cup \{\lambda\}, Z \in \Gamma_\lambda), \delta_{F(q,x,Z)} = \delta_\lambda(q, x, Z)$
3.  $\forall q \in Q_\lambda, (q_f, \lambda) \in \delta_{F(q,\lambda,Z_0)}$

Ahora tenemos que demostrar que  $w \in \mathcal{L}(P_F) \iff w \in \mathcal{L}_\lambda(P_\lambda)$ :

- $\Leftarrow$ ) Si  $w \in \mathcal{L}_\lambda(P_\lambda)$  entonces tenemos que  $(q_{0_\lambda}, w, X_0) \vdash_{P_\lambda}^* (q, \lambda, \lambda)$

Por definición de delta, tenemos que

$$(q_{0_F}, w, Z_0) \vdash_{P_F} (q_{0_\lambda}, w, X_0 Z_0) \vdash_{P_F}^* (q, \lambda, \lambda) \vdash_{P_F} (q_F, \lambda, \lambda)$$

Y por lo tanto  $w \in \mathcal{L}(P_F)$  □

- $\Rightarrow$ ) Si  $w \in \mathcal{L}(P_F)$ , tenemos que:

$$(q_{0_F}, w, Z_0) \vdash_{P_F} (q_{0_\lambda}, w, X_0 Z_0) \vdash_{P_F}^* (p, \lambda, Z_0) \vdash_{P_F} (q_f, \lambda, \lambda)$$

Pero por definición de  $P_F$ :

$$(q_{0_\lambda}, w, X_0 Z_0) \vdash_{P_F} (p, \lambda, Z_0) \iff (q_{0_\lambda}, w, X_0) \vdash_{P_\lambda} (p, \lambda, \lambda)$$

Luego, tenemos que  $(q_{0_\lambda}, w, X_0) \vdash_{P_\lambda} (p, \lambda, \lambda)$ , por lo que  $w \in \mathcal{L}_\lambda(P_\lambda)$

## 6.4. Equivalencia GLCs y APDs

**Teorema 6.4.1:** Para toda GLC  $G$ , existe un AP  $M$  tal que:

$$\mathcal{L}(G) = \mathcal{L}_\lambda(M)$$

*Demostración:* Sea  $G = \langle V_N, V_T, P, S \rangle$  una GLC, definimos  $M = \langle \{q\}, V_T, V_T \cup V_N, \delta, q, S, \emptyset \rangle$  tal que:

- $\delta : Q \times (V_T \cup \lambda) \times (V_N \cup V_T) \rightarrow \mathcal{P}(Q \times (V_T \cup V_N)^*)$ 
  - Si  $A \rightarrow \alpha \in P$ , entonces  $\delta(q, \lambda, A) \ni \{(q, \alpha)\}$  (Para toda producción en  $P$ ,  $M$  lo simula desapilando la variable correspondiente y pusheando el cuerpo de la producción)
  - Para todo  $x \in V_T$ ,  $\delta(q, x, x) = \{(q, \lambda)\}$  (Si se tiene que el tope de la pila es un terminal, se desapila y se avanza si es igual al símbolo que la cabeza lectora está leyendo)

Queremos ver que  $w \in \mathcal{L}(G) \iff w \in \mathcal{L}_\lambda(M)$ :

**Lema 6.4.2:**  $\forall (A \in V_N, w \in V_T^*) A \xRightarrow{+} w$  sii  $(q, w, A) \vdash_M^* (q, \lambda, \lambda)$

Primero demostramos este lema por inducción sobre  $m$ , la cantidad de pasos en la derivación:

- Caso base:  $m = 1$

En este caso, tenemos que  $A \xRightarrow{1} w$  para  $w = x_1 x_2 \dots x_k$ . Notar que si esta es una derivación posible para  $w$ , entonces necesariamente tiene que haber una producción  $A \rightarrow x_1 x_2 \dots x_k \in P$ , luego, por la primer regla de  $\delta$ , seguida por la aplicación de la segunda (que nos permite eliminar terminales del tope siempre que matcheen con la cadena), tenemos:

$$(q, w, A) \vdash_M (q, x_1 x_2 \dots x_k, x_1 x_2 \dots x_k) \vdash_M^k (q, \lambda, \lambda)$$

- Paso inductivo:

Por H.I, tenemos que, para todo  $j < m$ ,  $A \xRightarrow{j} w$  sii  $(q, w, A) \vdash_M^j (q, \lambda, \lambda)$



Sea  $w = w_1 \dots w_k$ , por definición de derivación, tenemos que  $A \xRightarrow{m} w$  sii  $A \rightarrow X_1 X_2 \dots X_k$  es una producción de la gramática, y que  $X_1 \xRightarrow{m_1} w_1, X_2 \xRightarrow{m_2} w_2, \dots, X_k \xRightarrow{m_k} w_k$ , para  $m_i < m$ .

Por otro lado, por def. de M tenemos que  $A \rightarrow X_1 \dots X_k \in P$  sii  $(q, w, A) \vdash_M (q, w, X_1 \dots X_k)$

Si  $X_i \in V_N$ , entonces por H.I  $(q, w_i, X_i) \vdash_M^* (q, \lambda, \lambda)$

Si  $X_i \in V_T, X_i = w_i$  y entonces por la segunda regla de  $\delta, (q, w_i, X_i) \vdash_M (q, \lambda, \lambda)$

Juntando todo:

$$(q, w, A) \vdash_M (q, w_1 \dots w_k, X_1 \dots X_k) \vdash_M^* (q, \lambda, \lambda)$$

Volviendo al teorema, el lema nos dice que  $A \xRightarrow{+} w$  sii  $(q, w, A) \vdash_M^* (q, \lambda, \lambda)$ , luego, tomando  $S = A$  obtenemos la prueba del teorema

□

## 7. Ejercicios (Después debería pasarlo a otro lado)

1. Demuestra que  $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$  para cualquier estado  $q$  y cadenas  $x$  e  $y$ . **Pista:** hacer inducción sobre  $|y|$

*Demostración:* Siguiendo la sugerencia:

- Caso base:  $|y| = 0$  ( $y = \lambda$ )

$$\hat{\delta}(q, x\lambda) = \hat{\delta}(\hat{\delta}(q, x), \lambda) \stackrel{\text{def } \hat{\delta}}{=} \hat{\delta}(q, x) \stackrel{\lambda \text{ es neutro}}{=} \hat{\delta}(q, x\lambda)$$

- Paso inductivo:

$\forall q \forall \hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ . Sea entonces  $y = y'\alpha$ , tenemos, reescribiendo el lado derecho de la igualdad,  $\hat{\delta}(\hat{\delta}(q, x), y'\alpha) \stackrel{\text{def de } \hat{\delta}}{=} \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), \alpha) \stackrel{\text{H.I.}}{=} \hat{\delta}(\hat{\delta}(q, xy'), \alpha) \stackrel{\text{def de } \hat{\delta}}{=} \hat{\delta}(q, xy'\alpha) = \hat{\delta}(q, xy)$ .  $\square$

2. Demostrá que para cualquier estado  $q$ , cadena  $x$ , y símbolo  $\alpha$ ,  $\hat{\delta}(q, \alpha x) = \hat{\delta}(\delta(q, \alpha), x)$

*Demostración:* Por el ejercicio anterior, tenemos que  $\hat{\delta}(q, \alpha x) = \hat{\delta}(\hat{\delta}(q, \alpha), x)$ . Luego, solo resta probar que  $\hat{\delta}(q, \alpha) = \delta(q, \alpha)$  que sale fácil usando la definición  $\square$

3. Sea  $M$  un AFD y  $q$  un estado del mismo, tal que  $\delta(q, \alpha) = q$  para cualquier símbolo  $\alpha$ . Demostrar por inducción sobre el largo de la cadena de entrada que para toda cadena  $\omega$ ,  $\hat{\delta}(q, \omega) = q$

*Demostración:* Haciendo inducción sobre el largo de la cadena  $w$ :

- Caso  $|w| = 0$  ( $w = \lambda$ )

$$\hat{\delta}(q, \lambda) \stackrel{\text{def } \hat{\delta}}{=} q$$

- Caso  $w = xa$

$$\hat{\delta}(q, xa) \stackrel{\text{def } \hat{\delta}}{=} \hat{\delta}(\hat{\delta}(q, x), a) \stackrel{\text{H.I.}}{=} \delta(q, a) \stackrel{\text{Por enunciado}}{=} q$$

$\square$

4. Sea  $M$  un AFD y  $\alpha$  un símbolo particular de su alfabeto, tal que para todos los estados  $q$  de  $M$  tenemos que  $\delta(q, \alpha) = q$

**a)** Demostrar por inducción sobre  $n$  que para cualquier  $n \geq 0$ ,  $\hat{\delta}(q, \alpha^n) = q$

*Demostración:* Haciendo inducción sobre  $n$ :

- Caso  $n = 0$

$$\hat{\delta}(q, a^0) = \hat{\delta}(q, \lambda) \stackrel{(\text{def } \hat{\delta})}{=} q$$

- Caso  $n \geq 1$

$$\hat{\delta}(q, a^n) \stackrel{\text{def } \hat{\delta}}{=} \delta(\hat{\delta}(q, a^{n-1}), a) \stackrel{HI}{=} \delta(q, a) \stackrel{\text{Por enunciado}}{=} q$$

□

b) Demostrar o bien  $\{\alpha\}^* \subseteq \mathcal{L}(M)$  o bien  $\{\alpha\}^* \cap \mathcal{L}(M) = \emptyset$

*Demostración:* Primero, una observación. Tenemos que la única manera en la que una cadena  $a^k$  para algún  $k$  esté en el lenguaje de  $M$  es si el estado inicial es final, pues tenemos por enunciado que la transición por  $a$  siempre es desde un estado a sí mismo. Con esta observaciones, tratamos primero de probar:

$$\{a\}^* \cap \mathcal{L}(M) \neq \emptyset \iff q_0 \in F$$

Probamos los dos lados:

- $\implies$ ) Que la intersección no sea vacía implica que hay una cadena  $a^k$  aceptada por el lenguaje. Supongamos que  $q_0$  no fuese un estado final, por enunciado sabemos que por  $a$  no podemos llegar a ningún otro estado, por lo que siempre nos quedamos en  $q_0$  pero, como este no era estado final entonces nuestro autómata en realidad no acepta la cadena  $a^k$ . Llegamos a un absurdo, que vino de suponer que  $q_0$  no era estado final
- $\impliedby$ ) Sabemos por enunciado que tiene una transición por  $a$  hacia sí mismo, luego, como  $q_0 \in F$ , tenemos que  $a \in \mathcal{L}(M)$

Con esto en mente, quedaría demostrar por inducción que si acepta a la cadena  $a$ , acepta todas las cadenas  $a^k$ , y una vez probado esto, como la inclusión de  $q_0$  en  $F$  o bien pasa (y entonces todas las cadenas  $\{a\}^*$  están en el lenguaje) o bien no está (por lo que ninguna cadena está):

$$q_0 \in F \oplus q_0 \notin F \iff$$

$$\{a\} \subseteq \mathcal{L}(M) \oplus \{a\}^* \cap \mathcal{L}(M) = \emptyset \iff$$

$$\{\alpha\}^* \subseteq \mathcal{L}(M) \oplus \{a\}^* \cap \mathcal{L}(M) = \emptyset$$

**(Considerar terminar esto último, es fácil creo pero bue, lo que sí debería considerarse es emprolijar esto porque es mucho mas fácil)**

□

5. Sea  $M = \langle Q, \Sigma, \delta, q_0, \{q_f\} \rangle$  un AFD tal que para todos los símbolos  $\alpha \in \Sigma$  se cumple que  $\delta(q_0, \alpha) = \delta(q_f, \alpha)$

a) Demostrar que para todo  $\omega \neq \lambda$ ,  $\hat{\delta}(q_0, \omega) = \hat{\delta}(q_f, \omega)$

*Demostración:* Por inducción sobre  $\omega$ :

- Caso base:  $\omega = \alpha$

$$\hat{\delta}(q_0, \alpha) = \delta(q_0, \alpha) = \delta(q_f, \alpha) = \hat{\delta}(q_f, \alpha)$$

- Paso inductivo:  $\omega = x\alpha$

$$\hat{\delta}(q_0, x\alpha) = \delta(\hat{\delta}(q_0, x), \alpha) \stackrel{HI}{=} \delta(\hat{\delta}(q_f, x), \alpha) \stackrel{\text{def } \hat{\delta}}{=} \hat{\delta}(q_f, x\alpha)$$

□

b) Demostrar que si  $\omega$  es una cadena no vacía en  $\mathcal{L}(M)$ , entonces para toda  $k > 0$ ,  $\omega^k$  también pertenece a  $\mathcal{L}(M)$

*Demostración:* Por inducción sobre  $k$ :

- Caso base:  $k = 1$

$$\omega^1 = \omega \text{ y por enunciado } \omega \in \mathcal{L}(M)$$

- Paso inductivo:  $k = n + 1$

$$\omega^{n+1} \in \mathcal{L}(M) \iff \hat{\delta}(q_0, w^{n+1}) \cap F \neq \emptyset \iff \hat{\delta}(q_0, w^{n+1}) = q_f \stackrel{\text{Por 1)}}{\iff} \hat{\delta}(\hat{\delta}(q_0, w^n), w) = q_f \stackrel{HI}{\iff} \hat{\delta}(q_f, w) = q_f \text{ que sabemos es cierto por a)}$$

□

- Sea  $N$  un AFND tal que tenga a lo sumo una opción para cualquier estado y símbolo (es decir,  $|\delta(q, \alpha)| = 1$ ), entonces el AFD  $D$  construido tiene la misma cantidad de transiciones y estados más las transiciones necesarias para ir a un nuevo estado trampa cuando una transición no esté definida para un estado particular
- Demostrar que para todo AFND- $\lambda$   $E$  existe un AFD  $D$  tal que  $\mathcal{L}(E) = \mathcal{L}(D)$  (Usando la demo del libro)

*Demostración:* Sea  $E = \langle Q_\lambda, \Sigma, \delta_\lambda, q_0, F_\lambda \rangle$  un AFND- $\lambda$ . Definimos

$D = \langle Q_D, \Sigma, \delta_D, q_D, F_D \rangle$  de la siguiente manera:

- $Q_D = \mathcal{P}(Q_\lambda)$ . En particular, va a ocurrir que todos los estados accesibles de  $D$  son subconjuntos de  $Q_\lambda$  cerrados por la clausura  $\lambda$ , es decir sea  $S$  el subconjunto,  $S = Cl_\lambda(S)$
- $q_D = Cl_\lambda(q_0)$
- $F_D = \{S \subseteq Q_\lambda : S \cap F_\lambda \neq \emptyset\}$
- Para cada  $S \subseteq Q_\lambda$  y  $\alpha \in \Sigma$ , definimos:
  - $\delta_D(S, \alpha) = Cl_\lambda(\{r \in Q_\lambda : \exists p \in S \wedge r \in \delta_\lambda(p, \alpha)\})$

Ahora probamos que para una cadena  $w \in \Sigma^*$ ,  $w \in \mathcal{L}(E) \iff w \in \mathcal{L}(D)$ :

- $\Leftarrow$ ) Simplemente podemos agregar transiciones  $\lambda$  en todos los estados hacia el estado representando el estado trampa, y convertimos cada una de las transiciones en el equivalente de conjuntos (i.e en vez de  $q_i$ ,  $\{q_i\}$  en las funciones de transición)
- $\Rightarrow$ ) Primero demostramos por inducción sobre la longitud de la cadena  $w$  que  $\hat{\delta}_\lambda(q_0, w) = \hat{\delta}_D(q_D, w)$ :
  - $|w| = 0$ :
 
$$\hat{\delta}_\lambda(q_0, \lambda) = Cl_\lambda(q_0) = q_D = \hat{\delta}_D(q_D, \lambda)$$
  - $|w| > 0$ ,  $w = xa$ :
 
$$\hat{\delta}_\lambda(q_0, xa) = Cl_\lambda(\{r \in Q_\lambda : \exists p \in \hat{\delta}_\lambda(q_0, x) \wedge r \in \delta_\lambda(p, a)\}) = Cl_\lambda(\{r \in Q_\lambda : \exists p \in \hat{\delta}_D(q_D, x) \wedge r \in \delta_\lambda(p, a)\}) = \hat{\delta}_D(q_D, xa)$$

□

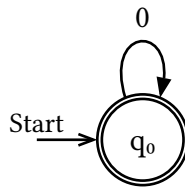
8. Indicar V o F y justificar

- Si  $D = \langle Q, \Sigma, \delta, q_0, F \rangle$  es un AFD entonces reconoce al menos  $|Q|$  palabras distintas, es decir  $\#\mathcal{L}(D) \geq |Q|$

Falso. Como contraejemplo, un AFD que no tiene estados finales

- Si  $N = \langle Q, \Sigma, \delta, q_0, F \rangle$  es un AFND entonces todas las palabras de  $\mathcal{L}(N)$  tienen longitud menor o igual a  $|Q|^2$

Falso. Como contraejemplo, para  $\Sigma = \{0, 1\}$ ,  $L = \{w \mid 1 \notin w\}$ , tenemos el AFND:



9. Cuántos AFD hay con  $|Q| = 2$  y  $|\Sigma| = 3$ ?

La fórmula general para la cantidad de AFDs posibles es  $|Q|^{|Q|} \times |\Sigma|^{|Q|} \times |Q|$ , por lo que para  $|Q| = 2$  y  $|\Sigma| = 3$  tenemos 512 AFDs posibles

10. Qué pasa al invertir los arcos en un AFD?

Noc si a esta pregunta le faltó algo más o quería darnos una pista sobre como arrancar a conseguir un AFD para el inverso de un lenguaje

11. Qué pasa al invertir los estados finales con no finales en un AFND?

A diferencia de con un AFD, invertir los estados no nos provee con un autómata que reconozca el complemento al lenguaje original

12. Demostrar que para cada AFND  $N = \langle Q, \Sigma, \delta, q_0, F \rangle$  existe otro AFND- $\lambda$   $E = \langle Q_\lambda, \Sigma, \delta_\lambda, q_0, F_\lambda \rangle$  tal que  $\mathcal{L}(N) = \mathcal{L}(E)$  y  $F_\lambda$  tiene un solo estado final

*Demostración:* Definimos  $E$  de la siguiente manera:

- $Q_\lambda = Q \cup \{q_f\}$ , donde  $q_f$  es un nuevo estado final
- $F_\lambda = \{q_f\}$
- $\delta_\lambda(q, \alpha) = \delta(q, \alpha)$  para todo  $q \in Q, \alpha \in \Sigma$
- $\delta_\lambda(q, \lambda) = \{q_f\}$  para todo  $q \in Q$  si  $q \in F$  (Esto debería definirlo así, o usar la clausura lambda?)

Queremos demostrar que  $\mathcal{L}(N) = \mathcal{L}(E)$ , para eso primero notamos que  $\hat{\delta}_N(q_0, w) = \hat{\delta}_E(q_0, w)$  para toda cadena  $w \in \Sigma^+$ . Con esto en mente tenemos que (Separamos en casos):

- $\lambda \in \mathcal{L}(N) \iff \delta(q_0, \lambda) \cap F \neq \emptyset \iff \{q_0\} \cap F \neq \emptyset \implies \delta_\lambda(q_0, \lambda) = \{q_f\} \implies \delta(q_0, \lambda) \cap F_\lambda \neq \emptyset \iff \lambda \in \mathcal{L}(E)$
- $\lambda \in \mathcal{L}(E) \iff \delta(q_0, \lambda) \cap F_\lambda \neq \emptyset \implies \delta(q_0, \lambda) \cap F \neq \emptyset \iff \lambda \in \mathcal{L}(N)$
- $w \in \mathcal{L}(N) \iff \hat{\delta}_N(q_0, w) \cap F \neq \emptyset \implies \delta_\lambda(\hat{\delta}_\lambda(q_0, w), \lambda) = \{q_f\} \iff \hat{\delta}_\lambda(q_0, w) \cap F_\lambda \neq \emptyset \iff w \in \mathcal{L}(E)$
- $w \in \mathcal{L}(E) \iff \hat{\delta}_\lambda(q_0, w) \cap F_\lambda \neq \emptyset \iff \delta_\lambda(\hat{\delta}_\lambda(q_0, w), \lambda) = \{q_f\} \implies \hat{\delta}(q_0, w) \cap F \neq \emptyset \iff w \in \mathcal{L}(N)$

□

13. Sea  $\Sigma$  un alfabeto con al menos dos símbolos, y sea  $a$  un símbolo de  $\Sigma$ . Sea  $N = \langle Q, \Sigma, \delta, q_0, F \rangle$  un AFND, considerar el AFND- $\lambda$   $E = \langle Q, \Sigma \setminus \{a\}, \delta_\lambda, q_0, F \rangle$  que se obtiene por reemplazar todas las transiciones por el símbolo  $a$  por transiciones  $\lambda$ . Es decir:

- Para todo  $q \in Q, x \in \Sigma : x \neq a, \delta(q, x) = \delta_\lambda(q, x)$
- Para todo  $q \in Q, \delta_\lambda(q, \lambda) = \delta(q, a)$

Determinar cuál es el lenguaje aceptado por  $E$

14. Es posible acotar superiormente cuántas transiciones requiere la aceptación de una palabra en un AFND- $\lambda$ ?

Noc ( $xd$ )

15. Sea  $E = \langle Q, \Sigma, \delta, q_0, \{q_f\} \rangle$  un AFND- $\lambda$  tal que no haya transiciones hacia  $q_0$  ni desde  $q_f$ . Describir los lenguajes que se obtienen a partir de las siguientes modificaciones:

- Agregar una transición  $\lambda$  desde  $q_f$  hacia  $q_0$
- Agregar una transición  $\lambda$  desde  $q_0$  hacia cada estado alcanzables desde  $q_0$  (notar que no es sólo aquellos directos)
- Agregar una transición  $\lambda$  hacia  $q_f$  desde cada estado que tiene un camino hacia  $q_f$
- El autómata obtenido haciendo b) y c)

16. Demostrar que los lenguajes regulares son cerrados respecto de la concatenación y la clausura de Kleene mediante la construcción de un autómata.
17. Demostrar que los lenguajes regulares son cerrados respecto de la diferencia de conjuntos.
18. Demostrar que los lenguajes regulares son cerrados respecto al reverso del lenguaje, donde el reverso está definido como el lenguaje formado por el reverso de todas sus cadenas.
19. Idem para homomorfismos y sus inversos (?)
20. Sea  $L$  un lenguaje regular, y  $a$  un símbolo, llamamos  $L/a$  al cociente de  $L$  y  $a$  al conjunto de cadenas  $w$  tales que  $wa \in L$ . Por ejemplo, si  $L = \{a, aab, baa\}$  entonces  $L/a = \{\lambda, ba\}$ . Demostrar que si  $L$  es un lenguaje regular, entonces  $L/a$  también.
21. De manera similar, probar que  $a \setminus L$  es un lenguaje regular, donde  $a \setminus L$  es el conjunto de cadenas  $w$  tales que  $aw \in L$ .
22. Demostrar que los lenguajes regulares son cerrados respecto de las siguientes operaciones:
  - a)  $\min(L) = \{w \mid w \in L \text{ y no existe } \alpha \text{ tal que } \alpha w \in L\}$
  - b)  $\max(L) = \{w \mid w \in L \text{ y no existe } \alpha \neq \lambda \text{ tal que } w\alpha \in L\}$
  - c)  $\text{init}(L) = \{w \mid \text{Para algún } x, wx \in L\}$
23. La mayoría de los ejes de la sección 4.2 (Me dió fiaca seguir copiando xd)
24. Sea  $L$  un lenguaje regular, y sea  $n$  la constante del Lema de Pumping para  $L$ . Determinar veracidad y demostrar:
  - a) Para cada cadena  $z$  en  $L$ , con  $|z| \geq n$ , la descomposición de  $z$  en  $uvw$ , con  $|v| \geq 1$  y  $|uv| \leq n$ , es única.
  - b) Cada cadena  $z$  en  $L$ , con  $|z| \geq n$ , es de la forma  $uv^i w$  para algún  $u, v, w$  con  $|v| \geq 1$  y  $|uv| \leq n$  y algún  $i$
  - c) Hay lenguajes no regulares que satisfacen la condición afirmada por el Lema de Pumping
  - d) Sean  $L_1, L_2$  lenguajes sobre el alfabeto  $\Sigma$  tal que  $L_1 \cup L_2$  es un lenguaje regular. Entonces  $L_1$  y  $L_2$  son regulares.
25. Sea  $\mathcal{C}$  el mínimo conjunto que contiene todos los lenguajes finitos, y está cerrado por unión finita, intersección, complemento, y concatenación ¿Cuál es la relación entre  $\mathcal{C}$  y el conjunto de todos los lenguajes regulares?
26. Dar un algoritmo de decisión que determine si el lenguaje aceptado por un autómata finito es el conjunto de todas las cadenas del alfabeto
27. Dar un algoritmo de decisión que determine si el lenguaje aceptado por un autómata finito es cofinito
28. Demostrar que todo lenguaje regular es libre de contexto. **Pista:** construir una gramática mediante inducción en la cantidad de operadores de una expresión regular
29. Una GLC es lineal a derecha si el cuerpo de cada producción tiene a lo sumo una variable, y la misma aparece más a la derecha. Es decir, todas las producciones son de la forma  $A \rightarrow wB$  o  $A \rightarrow w$ .



a) demostrar que toda GLC lineal a derecha genera un lenguaje regular. **Pista:** construir un autómata finito  $\lambda$  que simule la derivación más a izquierda de la gramática, con sus estados representando el símbolo no terminal de la forma sentencial actual.

b) Demostrar que todo lenguaje regular tiene una GLC lineal a derecha. **Pista:** Empezar un AFD y hacer que las variables de la gramática representen estados

30. Considerar la gramática G definida por las producciones:

$$S \rightarrow aS \mid Sb \mid a \mid b$$

a) Demostrar por inducción sobre la longitud de la cadena que ninguna cadena generada por G tiene  $ba$  como subcadena.

b) Identificar  $\mathcal{L}(G)$

31. Sea G la gramática con producciones:

$$S \rightarrow aSbS \mid bSaS \mid \lambda$$

Demostrar que  $\mathcal{L}(G)$  es el conjunto de cadenas con una misma cantidad de  $a$ 's que de  $b$ 's

32. Supongamos que G es una GLC sin producciones que tengan a  $\lambda$  del lado derecho. Si  $w$  está en el lenguaje de G,  $|w| = n$  y  $S \xRightarrow{m} w$ , demostrar que  $w$  tiene un árbol de derivación con  $n + m$  nodos.

33. Supongamos lo mismo que en el ej anterior, pero ahora puede haber producciones con  $\lambda$  en la derecha. Demostrar que un árbol de derivación para  $w$  puede tener a lo sumo  $n + 2m - 1$  nodos.

34. Demostrar que si  $X_1 X_2 \dots X_k \xRightarrow{*} a$  entonces todos los símbolos que provienen de derivar  $X_i$  están a la izquierda de todos los que provienen de derivar  $X_j$ , si  $i < j$ . **Pista:** Usar inducción sobre la cantidad de pasos en la derivación

35. Indicar veracidad:

- Si  $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$  es un autómata de pila entonces cada cadena  $w \in \mathcal{L}_\lambda(P)$  es reconocida por P en a lo sumo  $|w| * \#Q * \#\Gamma$  transiciones, es decir:

$$\text{Sea } n \leq |w| * \#Q * \#\Gamma, \text{ entonces existe } p \in Q \text{ tal que } (q_0, w, Z_0) \stackrel{n}{\vdash} (p, \lambda, \lambda)$$

36. Dado un autómata finito  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ , dar un autómata de pila  $M' = \langle Q', \Sigma, \delta', q_0', Z_0', \emptyset \rangle$  tal que  $\mathcal{L}(M) = \mathcal{L}_\lambda(M')$ .

37. Consideremos la demostración del teorema que afirma que para cada autómata  $M = \langle Q, \Sigma, \delta, q_0, Z_0, F \rangle$  existe un autómata  $M'$  tal que  $\mathcal{L}(M) = \mathcal{L}_\lambda(M')$ .

¿Si  $M$  es determinístico, entonces el autómata  $M'$  construido en la demostración también lo es?

38. Consideremos la demostración del teorema que afirma que dado  $M' = \langle Q', \Sigma, \delta', q_0', X_0, \emptyset \rangle$  existe un autómata  $M$  tal que  $\mathcal{L}_\lambda(M') = \mathcal{L}(M)$ .

¿Si  $M'$  es determinístico, entonces el autómata  $M$  construido en la demostración también lo es?

39. Demostrar que si P es un APDm entonces existe un APD  $P_2$  con solo dos símbolos de pila (es decir  $|\Gamma_2| = 2$ ) tal que  $\mathcal{L}_\lambda(P) = \mathcal{L}_\lambda(P_2)$ . **Pista:** Considerar una codificación binaria para la pila

40. Un APD está restringido si en toda transición puede incrementar la altura de la pila con a lo sumo un símbolo, es decir, para toda transición  $\delta(q, w, Z)$  que contiene algún  $(p, \gamma)$ , debe ocurrir que  $|\gamma| \leq 2$ . Demostrar que si P es un APD, entonces existe un APD restringido  $P_2$  tal que  $\mathcal{L}(P) = \mathcal{L}(P_2)$