Teoría de Lenguajes

Gianfranco Zambonni

29 de diciembre de 2022

Índice

1. Introducción

1.1. Relaciones

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

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

Una relación $R : A \to A$ es

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

1.1.1. Operaciones

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

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

.

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

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

.

- Es el elemento neutro de la composición de relaciones.

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

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

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

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

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

DEMOSTRACIÓN

2) Si aR^+b entonces existe una secuencia de elementos $a = a_0, a_1, \ldots, a_n = b$ tales que a_iRa_{i+1} para todo $i \in [0, n-1]$.

   Análogamente, como bR^+c existe una secuencia de elementos $b = b_0, b_1, \ldots, b_m = c$ tales que b_iRb_{i+1} para todo $i \in [0, m-1]$.

   Por lo tanto, $aR^{n+m}c$, por lo que aR^+c.

3) Si aR^+b entonces existe una secuencia de elementos $a = a_0, a_1, \ldots, a_n = b$ tales que a_iRa_{i+1} para todo $i \in [0, n-1]$.

   Como $R \subseteq G$ entonces a_iGa_{i+1} para todo $i \in [0, n-1]$. Como G es transitiva entonces la aplicación repetida de la transitividad nos lleva a que a_1Ga_n, por lo que aGb.

Clausura transitiva reflexiva:

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

Si A es un conjunto finito, entonces todas las relaciones $R : A \to A$ son finitas.
Si R es reflexiva, entonces $R^* = R^+$.

1.2. Alfabetos

Un alfabeto es un conjunto finito de símbolos.

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

1.2.1. Operaciones

Concatenación: Es una operación entre un símbolo del alfabeto Σ y una cadena sobre dicho alfabeto:

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

- La cadena nula λ es el elemento neutro de la concatenación.

Clausura de Kleene de Σ: Σ^*

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

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

1.3. Lenguajes

Un lenguaje es un conjunto de cadenas sobre un alfabeto Σ.

1.3.1. Operaciones

Concatenación de lenguajes: Si L_1 y L_2 son lenguajes definidos sobre los alfabetos Σ_1 y Σ_2 respectivamente, entonces la concatenación de L_1 y L_2 es el lenguaje L_1L_2 definido por:

$$L_1L_2 = \{\alpha\beta \mid \alpha \in L_1,\ \beta \in L_2\}$$

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

Clausura de Kleene L^*: Se define por:

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

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

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

1.4. Gramáticas

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

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

  Las notamos como $\alpha \to \beta$.
- $S \in V_N$ es el símbolo inicial.

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

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

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

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

Además, $\overset{k}{\underset{G}{\Longrightarrow}}$ será la potencia k-ésima de $\underset{G}{\Longrightarrow}$.

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

$$\mathcal{L}(G) = \{\alpha \in V_T^* : \ S \overset{+}{\underset{G}{\Longrightarrow}} \alpha\}$$

1.4.1. Clasificación de grámaticas (Chomsky)

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

- Todas sus producciones son de la forma $A \to aB$ ó $A \to a$ ó $A \to \lambda$ donde $A, B \in V_N$ y $a \in V_T$. En este caso se dice que es una gramática lineal a derecha.
- Todas sus producciones son de la forma $A \to Ba$ ó $A \to a$ ó $A \to \lambda$ donde $A, B \in V_N$ y $a \in V_T$. En este caso se dice que es una gramática lineal a izquierda.

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

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

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

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

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

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

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

2. Autómatas finitos

2.1. Autómatas finitos deterministicos (AFD)

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

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

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

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

se define de manera recursica como:

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

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

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

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

2.2. Autómatas finitos no deterministas (AFND)

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

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

  A diferencia de los AFD, la función δ devuelve un conjunto de estados en lugar de un solo estado.
- $q_0 \in Q$ es el estado inicial.
- $F \subseteq Q$ es el conjunto de estados finales.

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

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

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

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

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

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

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

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

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

2.2.1. Equivalencia entre AFD y AFND

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

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

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

Construcción de $\mathcal{M}'$:

- Q' será el conjunto de partes $\mathcal{P}(Q)$ que contenga a todos los posibles conjuntos de estados de $\mathcal{M}$. Vamos a denotar cada estado $s \in Q'$ con etiquetas del estilo $[q_1, \ldots, q_k]$ donde $q_1, \ldots, q_k \in Q$. Entonces:
$$Q' = \mathcal{P}(Q)$$
- $\delta'([q_1, \ldots, q_k], a) = [p_1, \ldots, p_m] \iff \delta_P(\{q_1, \ldots, q_k\}, a) = \{p_1, \ldots, p_m\}$
- $q_0' = [q_0]$
- $F' = \{[q_1, \ldots, q_n] \in Q' : \{q_1, \ldots, q_n\} \cap F \neq \emptyset\}$

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

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

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

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

- Caso inductivo: $\beta \implies \beta a$: Por hipotesis inductiva tenemos que

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

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

$$\begin{aligned}
&\hat{\delta}'(q'_0, \beta a) = [p_1, \ldots, p_k] \underset{\text{def.}}{\iff} \delta'(\hat{\delta}'(q'_0, \beta), a) = [p_1, \ldots, p_k] \\
&\underset{\text{def.}}{\iff} \exists [r_1, \ldots, r_m] \in Q' \text{ tal que } \delta'(q'_0, \beta) = [r_1, \ldots, r_m] \wedge \delta'([r_1, \ldots, r_m], a) = [p_1, \ldots, p_k] \\
&\underset{H.I.}{\iff} \exists \{r_1, \ldots, r_m\} \in Q \text{ tal que } \hat{\delta}(q_0, \beta) = \{r_1, \ldots, r_m\} \wedge \delta_P(\{r_1, \ldots, r_m\}, a) = \{p_1, \ldots, p_k\} \\
&\underset{\text{def.}}{\iff} \delta_P(\hat{\delta}(q_0, \beta), a) = \{p_1, \ldots, p_k\} \underset{\text{def.}}{\iff} \hat{\delta}(q_0, \beta a) = \{p_1, \ldots, p_k\}
\end{aligned}$$

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

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

2.3. Autómatas finitos no deterministico con transiciones λ

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

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

- Q es un conjunto finito de estados.

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

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

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

Además $q \in Cl_\lambda(q)$.

Clausura λ de un conjunto de estados P:

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

Generalización de la función de transición: Podemos extender δ a conjunto de estados:

$$\delta_P : \mathcal{P}(Q) \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$$
$$\delta_P(P, a) = \bigcup_{p \in P} \delta(p, a)$$

Entonces podemos definir:

$$\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$$
$$\hat{\delta}(q_0, \lambda) = Cl_\lambda(q_0)$$
$$\hat{\delta}(q_0, \beta a) = Cl_{P\lambda}\Big(\delta_P(\hat{\delta}(q_0, \beta), a)\Big) = Cl_{P\lambda}\Big(\Big\{p : \exists q \in \hat{\delta}(q_0, \beta) \text{ tal que } p \in \delta(q, a)\Big\}\Big)$$

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

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

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

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

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

2.3.1. Equivalencia entre AFND y AFND-λ

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

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

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

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

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

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

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

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

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

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

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

- $\beta = \lambda$
  - $\lambda \in \mathcal{L}(M) \implies \lambda \in \mathcal{L}(M')$

$$\lambda \in \mathcal{L}(M) \underset{def.}{\Longrightarrow} \hat{\delta}(q_0, \lambda) \cap F \neq \emptyset \underset{def.}{\Longrightarrow} Cl_\lambda(q_0) \cap F \neq \emptyset$$
$$\underset{\text{por construcción } M'}{\Longrightarrow} q_0 \in F' \underset{def.}{\Longrightarrow} \lambda \in \mathcal{L}(M')$$

- $\lambda \in \mathcal{L}(M') \implies \lambda \in \mathcal{L}(M)$.

$$\lambda \in \mathcal{L}(M') \underset{\text{def.}}{\implies} q_0 \in F' \underset{\text{construcción } M'}{\implies} q_0 \in F \vee Cl_\lambda(q_0) \cap F \neq \emptyset$$

Como $q_0 \in F \implies Cl_\lambda(q_0) \cap F \neq \emptyset$, entonces:

$$q_0 \in F \vee Cl_\lambda(q_0) \cap F \neq \emptyset \implies Cl_\lambda(q_0) \cap F \neq \emptyset \underset{\text{def.}}{\implies} \lambda \in \mathcal{L}(M)$$

- $\beta \neq \lambda$
  - $\beta \in \mathcal{L}(M) \implies \beta \in \mathcal{L}(M')$

$$\beta \in \mathcal{L}(M) \underset{\text{def.}}{\implies} \hat{\delta}(q_0, \beta) \cap F \neq \emptyset \underset{\text{por equivalencia transiciones}}{\implies} \delta'(q_0, \beta) \cap F$$
$$\underset{\text{construcción } M'}{\implies} \delta'(q_0, \beta) \cap F' \neq \emptyset \underset{\text{def.}}{\implies} \beta \in \mathcal{L}(M')$$

  - $\beta \in \mathcal{L}(M') \implies \beta \in \mathcal{L}(M)$

$$\beta \in \mathcal{L}(M') \underset{\text{def.}}{\implies} \delta'(q_0, \beta) \cap F' \neq \emptyset \underset{\text{equiv. transiciones}}{\implies} \hat{\delta}(q_0, \beta) \cap F' \neq \emptyset$$
$$\underset{\text{constr.} M'}{\implies} \hat{\delta}(q_0, \beta) \cap F \neq \emptyset \vee \hat{\delta}(q_0, \beta) \cap (F \cup \{q_0\}) \neq \emptyset$$

Si vale la primera parte de la última expresión $\delta(q_0, \beta) \cap F \neq \emptyset$ entonces $\beta \in \mathcal{L}(M)$ por definición.
Veamos que pasa si vale $\hat{\delta}(q_0, \beta) \cap (F \cup \{q_0\}) \neq \emptyset$:

$$\hat{\delta}(q_0, \beta) \cap (F \cup \{q_0\}) \neq \emptyset \implies \hat{\delta}(q_0, \beta) \cap F \neq \emptyset \vee \hat{\delta}(q_0, \beta) \cap \{q_0\} \neq \emptyset$$

La primer parte es lo mismo que arriba, analizemos la segunda:

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

Queda demostrada la equivalencia de lenguajes.

3. Expresiones regulares

Una expresión regular es una expresión que describe un lenguaje de forma compacta y sencilla:

- $\varnothing$ es una expresión regular que describe el lenguaje vacío $\emptyset$.
- λ es una expresión regular que describe el lenguaje unitario $\{\lambda\}$.
- Para cada $a \in \sigma$, a es una expresión regular que describe el lenguaje $\{a\}$.
- Si r y s son dos expresiones que denotan los lenguajes R y S entonces:
  - $r|s$ ó $r + s$ es una expresión regular que describe el lenguaje $R \cup S$.
  - rs es una expresión regular que describe el lenguaje RS.
  - r^* es una expresión regular que describe el lenguaje R^*.
  - r^+ es una expresión regular que describe el lenguaje R^+.

Expresiones regulares recursivas: Si $r = \alpha r + \beta$, entonces $r = \alpha^* \beta$. Además, si $\alpha^* = \alpha^+$, entonces $r = \alpha^*(\beta + \gamma)$ para culquier expresión regular γ.

3.1. Expresiones regulares a AFND-λ

Dada una expresión regular r, existe una AFND-λ M con un solo estado final y sin trancisiones a partir del mismo tal que $\mathcal{L}(M) = \mathcal{L}(r)$.

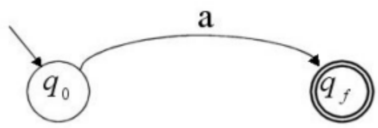
Vamosa a demostrar por inducción sobre los operadores de las expresiones regulares.

3.1.1. Casos base

$$r = \varnothing$$

$$r = \lambda$$

$$r = a$$



3.1.2. Pasos inductivos

Sean r_1 y r_2 dos expresiones regulares. Supongamos que existen AFND-λ $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\}\rangle$ y $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, \{f_{\}}\rangle$ tal que $\mathcal{L}(M_1) = \mathcal{L}(r_1)$ y $\mathcal{L}(M_2) = \mathcal{L}(r_2)$. Vamos a armar a partir de estos autómata uno nuevo que acepte los lenguajes generados por las expresiones $r_1|r_2$, r_1r_2, r_1^* y r^+.

$\boldsymbol{r_1|r_2}$**:** Podemos construir un automata $M_0 = \langle Q_0, \Sigma_0, \delta_0, q_0, \{f_0\}\rangle$ tal que $\mathcal{L}(M_0) = \mathcal{L}(r_1|r_2)$ de la siguiente forma:

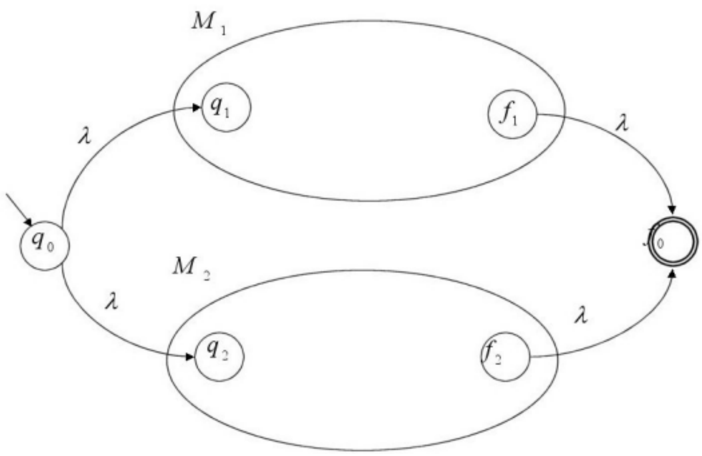
- $Q_0 = Q_1 \cup Q_2 \cup \{q_0, f_0\}$
- $\Sigma_0 = \Sigma_1 \cup \Sigma_2$
- $\delta_0 : Q_0 \times \Sigma_0 \to \mathcal{P}(Q_0)$

$\delta(q_0, \lambda) = \{q_1, q_2\}$

$\delta(q, a) = \delta_1(q, a)$ 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 $a \in \Sigma_2 \cup \{\lambda\}$

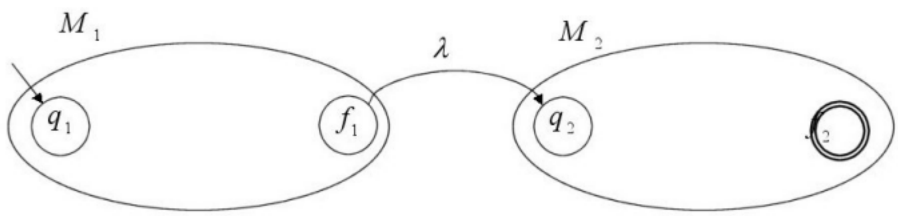
$d(f_1, \lambda) = d(f_2, \lambda) = \{f_0\}$



$\boldsymbol{r_1 r_2}$**:** $M_0 = \langle Q_0, \Sigma_0, \delta_0, q_1, \{f_2\} \rangle$:

- $Q_0 = Q_1 \cup Q_2$
- $\Sigma_0 = \Sigma_1 \cup \Sigma_2$
- $\delta_0 : Q_0 \times \Sigma_0 \to \mathcal{P}(Q_0)$

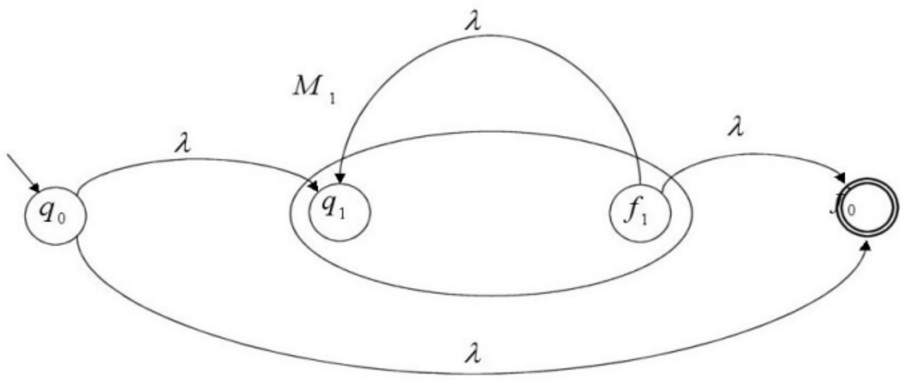
$$\delta(q, a) = \delta_1(q, a) \text{ para } q \in Q_1 - \{f_1\} \text{ y } a \in \Sigma_1 \cup \{\lambda\}$$
$$\delta(q, a) = \delta_2(q, a) \text{ para } q \in Q_2 - \{f_2\} \text{ y } a \in \Sigma_2 \cup \{\lambda\}$$
$$d(f_1, \lambda) = \{q_2\}$$



$\boldsymbol{r_1^*}$**:** $M_0 = \langle Q_0, \Sigma_1, \delta_0, q_0, \{f_0\} \rangle$:

- $Q_0 = Q_1 \cup \{f_0, q_0\}$
- $\delta_1 : Q_0 \times \Sigma_1 \to \mathcal{P}(Q_0)$

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



Para el caso r_1^+ es el mismo autómata que para este caso sin la transición $q_0 \xrightarrow{\lambda} f_0$.

3.2. AFD a expresión regular

Dado un AFD $M = \langle \{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F\rangle$, que acepta el lenguaje $\mathcal{L}$, existe una expresión regular que denota el mismo lenguaje.

3.2.1. Demostración

Nombremos $R^k_{i,j}$ a la expresión regular cuyo lenguaje $\omega \subseteq \Sigma^*$ son las cadenas que llevan al autómata M desde el estado q_i al estado q_j pasando solo por estados q_l con $l \leq k$. En particular $R^n_{i,j}$ es la expresión regular que representa todas las cadenas que permiten ir del estado i al estado j.

Vamos a buscar como construir $R^k_{i,j}$ para cada $k \in \{0, \ldots, n\}$ de manera inductiva. Suponiendo que demostramos la existencia de esta expresión regular, podemos concluir que la unión $R^n_{1,f_1}|R^n_{1,f_2}|...|R^n_{1,f_m}$ (con $f_1 \ldots f_m \in F$) es la expresión regular que representa el lenguaje $\mathcal{L}$:

Caso base ($k = 0$): Como todos los estados están enumerados del 1 para arriba, $k = 0$ significa que no debe haber estados intermedios en el camino entre q_i y q_j, por lo que pueden ser de dos formas:

- Una arco del estado i al estado j.
- Un camino de longitud cero que solo contiene el estado i.

Si $i \neq j$, entonces solo es posible la primera opción. Debemos examinar el AFD y encontrar aquellos simbolos que nos permitan ir del estado i al estado j.

1. Si no existe tal símbolo, entonces $R^0_{i,j} = \varnothing$.
2. Si existe exactamente un símbolo a, entonces $R^0_{i,j} = a$.
3. Si existen más de un símbolo, entonces $R^0_{i,j} = a_1|a_2|...|a_n$.

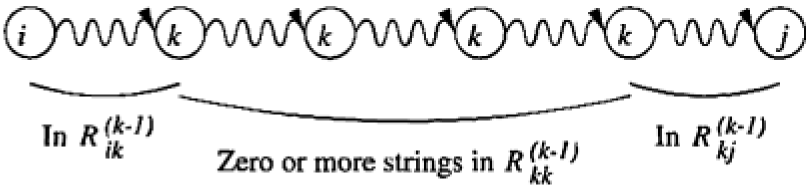
Ahora, si $i = j$ entonces los caminos de longitud cero también son posibles, por lo que habría que agregar a cada una de las expresiones recién mencionadas el simbolo λ:

1. $R^0_{i,j} = \lambda$.
2. $R^0_{i,j} = a|\lambda$.
3. $R^0_{i,j} = a_1|a_2|...|a_n|\lambda$.

Paso inductivo: Supongamos que hay un camino desde el estado i al estado j que no pasa por estados mas grandes k. Entonces podemos considerar las siguientes dos opciones:

1. El camino no pasa por el estado k, por lo que el lenguaje de $R^{k-1}_{i,j}$ contiene a ese camino.

2. El camino pasa por el estado k por lo menos una vez. Entonces podemos partir el camino en varias partes:



La primer parte, va desde el estado i al estado k sin pasar por k, la última parte es desde el estado k al estado j sin pasar por k, y todas las partes intermedia s van desde el estado k al estado k sin pasar por k. Cada una de estas partes ya tiene una expresión regular asociada: $R_{i,k}^{k-1}$, $R_{k,k}^{k-1}$, $R_{k,j}^{k-1}$, por lo que podemos unirlas para obtener la expresión regular que representa el camino completo de la siguiente forma:

$$R_{i,k}^{k-1}\left(R_{k,k}^{k-1}\right)^{*}R_{k,j}^{k-1}$$

Entonces $R_{i,j}^{k}$ es la unión de las expresiones de los dos tipos de caminos que acabamos de describir:

$$R_{i,j}^{k}=R_{i,j}^{k-1}\cup R_{i,k}^{k-1}\left(R_{k,k}^{k-1}\right)^{*}R_{k,j}^{k-1}$$

Finalmente, si construimos en orden todas estas expresiones regulares desde $R_{i,j}^{0}$, eventualmente llegaremos hasta $R_{i,j}^{n}$.

Y como dijimos, más arriba si calculamos $R_{1,j}^{0}$ para cada $q_j \in F$ y unimos todas las expresiones, obtendremos la expresión regular que representa el lenguaje $\mathcal{L}$.

3.3. Gramática regular a AFND

Dada una grámatica regular $G = \langle V_N, V_T, P, S\rangle$, podemos construir un AFND $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ que reconozca el lenguaje generado por G

3.3.1. Demostración

Vamos a constuir el autómata finito no determinista M y demostrar que reconoce el lenguaje generado por G.

Construcción de M: Construyamos M de la siguiente manera:

- $Q = V_N \cup \{q_f\}$
- $\Sigma = V_T$
- $q_0 = q_S$
- Si $A, B \in V_N$ y $a \in \Sigma$, entonces:
  - $q_B \in \delta(q_A, a) \iff A \to aB \in P$
  - $q_f \in \delta(q_A, a) \iff A \to a \in P$
  - $q_A \in F \iff A \to \lambda \in P$
  - $q_f \in F$

Equivalencia clausura transitiva de producciones y δ: Vamos a probar por inducción que

$$A \overset{*}{\Rightarrow} \alpha B \iff q_B \in \hat{\delta}(q_A, \alpha)$$

- **Caso base $\alpha = \lambda$:**
  - $A \overset{*}{\Rightarrow} \alpha B$, pero las gramáticas regulares no acentan producciones que vayan de un no terminal a otro sin pasar por un terminal, por lo que $B = A$. Osea $A \overset{*}{\Rightarrow} \alpha A$.
  - Además, como es un AFND, no tiene transiciones lambda, osea que $\delta(q_A, \lambda) = \{q_A\}$, por lo que $q_A \in \hat{\delta}(q_A, \alpha)$.
- **Caso inductivo $\alpha = \beta a$:**

$$A \overset{*}{\Rightarrow} \alpha B \iff A \overset{*}{\Rightarrow} \beta a B \underbrace{\iff}_{\text{def.}} \exists C \in V_N : A \overset{*}{\Rightarrow} \beta C \land C \to aB$$

$$\underbrace{\iff}_{\text{H.I. y def } M} \exists q_C \in Q, q_c \in \hat{\delta}(q_A, \alpha) \land q_B \in \delta(q_C, a) \iff q_B \in \delta(\hat{\delta}(q_A, \alpha), a)$$

$$\iff q_B \in \hat{\delta}(q_A, \beta a) \iff q_B \in \hat{\delta}(q_A, \alpha)$$

Demostración de la equivalencia: Vamos a demostrar que el lenguaje generado por G y M son iguales, osea que $\alpha a \in \mathcal{L}(M) \iff S \overset{*}{\Rightarrow} \alpha a$ Como G es una grámatica regular, hay solo dos formas de llegar desde S hasta αa:

1. $\exists A \in V_N : S \overset{*}{\Rightarrow} \alpha A \wedge A \to a \in P$
2. $\exists B \in V_N : S \overset{*}{\Rightarrow} \alpha a B \wedge B \to \lambda \in P$

Entonces:

$$
\begin{aligned}
& S \overset{*}{\Rightarrow} \alpha a \\
& \underbrace{\iff}_{\text{def. gramatica regular}} (\exists A \in V_N : S \overset{*}{\Rightarrow} \alpha A \wedge A \to a \in P) \vee (\exists B \in V_N : S \overset{*}{\Rightarrow} \alpha a B \wedge B \to \lambda \in P) \\
& \underbrace{\iff}_{\text{Equiv. anterior}} (\exists q_A \in Q, q_A \in \hat{\delta}(q_0, \alpha) \wedge q_f \in \delta(q_A, a)) \vee (\exists q_B \in Q, q_B \in \hat{\delta}(q_0, \alpha a) \wedge q_B \in F) \\
& \underbrace{\iff}_{\text{def. } \delta} q_f \in \delta(q_S, \alpha a) \vee (\exists q_B \in Q, q_B \in \hat{\delta}(q_0, \alpha a) \wedge q_B \in F) \\
& \iff \alpha a \in \mathcal{L}(M)
\end{aligned}
$$

Falta ver que pasa si $\lambda \in \mathcal{L}(G)$:

$$\lambda \in \mathcal{L}(G) \iff S \overset{*}{\Rightarrow} \lambda \iff S \to \lambda \in P \iff q_S \in F \iff \lambda \in \mathcal{L}(M)$$