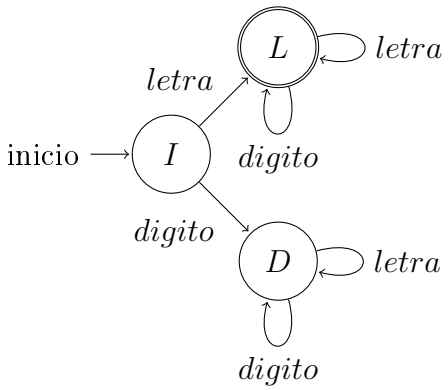
Capítulo 7

Automatas y expresiones regulares

7.1. Autómatas finitos

7.1.1. Diagrama de transiciones

El siguiente diagrama representa a un autómata que acepta cadenas que comienzan con una letra, y siguen con letras o números:



Diremos que una cadena es aceptada si sus símbolos corresponden a una secuencia de arcos (flechas) que conducen del circulo inicial a uno doble. Un diagrama de transiciones puede ser usado como herramienta de diseño para producir rutinas de análisis léxico.

7.1.2. Autómata de estado finito determinista

7.1.2.1. Definición

Un autómata de estado FINITO determinista es una quintupla $(\Sigma, S, f, Ac, \sigma)$ donde:

- Σ es un conjunto finito de *símbolos de entradas*.
- S es un conjunto FINITO de *estados*.
- $f : S \times \Sigma \to S$ es una *función de transición*.
- $Ac \subseteq S$ es un conjunto de *estados de aceptación*.
- $\sigma \in S$ es el estado inicial.

La función de transición del diagrama anterior estaría definida de la siguiente forma:

- $f(\sigma_1, digito) = \sigma_2$.
- $f(\sigma_1, letra) = \sigma_3$.
- $f(\sigma_2, digito) = \sigma_2$.
- $f(\sigma_2, letra) = \sigma_2$.
- $f(\sigma_3, digito) = \sigma_3$.
- $f(\sigma_3, letra) = \sigma_3$.

Observación Nótese que cada estado del diagrama de transiciones de un autómata de estado finito DETERMINISTA solo debe tener un arco que salga para cada símbolo del alfabeto; de lo contrario, un autómata que llega a ese estado se enfrentara a una elección de cual debe ser el arco a seguir. Ademas, dicho diagrama deberá estar completamente definido, es decir, debe existir por lo menos un arco para cada símbolo del alfabeto; de lo contrario, un autómata que llega a ese estado puede enfrentarse a una situación donde no puede aplicar ninguna transición. Estos requisitos están reflejados en la definición formal primero porque f es una función y, segundo porque su dominio es $S \times \Sigma$.

7.1.2.2. Palabra aceptada por un AEF

Sean $A = (\Sigma, S, f, Ac, \sigma)$ un AEF y $p = p(1)\,p(2)\ldots p(n)$ una palabra sobre Σ, diremos que dicha palabra es aceptada por el autómata A si existen $\sigma_0, \sigma_1, \ldots, \sigma_n \in S$ tales que:

1. $\sigma_0 = \sigma$.
2. $\sigma_i = f(\sigma_{i-1}, p(i))$ para $i = 1, \ldots, n$.
3. $\sigma_n \in Ac$.

Observación La palabra vaciá es aceptada si y solo si el estado inicial es de aceptación ($\sigma \in Ac$).

7.1.2.3. Función de transición extendida

Dado $A = (\Sigma, S, f, Ac, \sigma)$ un AEF, se define $F : S \times \Sigma^* \to S$ sobre una palabra $p = cl$ donde c es una cadena y l un caracter de forma recursiva:

- $F(s, \lambda) = s$.
- $F(s, p) = f[F(s, c), l]$.

7.1.2.4. Lenguaje aceptado por un AEF

Definimos al lenguaje aceptado por un autómata A como el conjunto: $\mathcal{AC}(A) = \{p \in \Sigma^* / p \text{ es aceptada por } A\}$.

7.1.2.5. Equivalencia de autómatas

Sean A_1, A_2 autómatas de estado finito, diremos que A_1 es equivalente a A_2 y lo notaremos $A_1 \equiv A_2$ si y solo si $\mathcal{AC}(A_1) = \mathcal{AC}(A_2)$, es decir, si y solo si aceptan el mismo lenguaje.

7.1.2.6. Regularidad de lenguajes aceptados por AEF

Enunciado Todo lenguaje aceptado por un AEF es regular, es decir:

$$\{L \in \mathcal{L} / L = \mathcal{AC}(A) \text{ para algun } A \in AEF\} \subseteq \mathcal{L}_3$$

Demostración Sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF, definimos $G = (N, T, P, \sigma)$ donde $N = S, T = \Sigma, \sigma = \sigma_0$ y reglas de producción:

- $U \to xV \iff f(U, x) = V$.
- $U \to \lambda \iff U \in Ac$

Debemos probar que $L(G) = \mathcal{AC}(A)$ es decir que:

$$\sigma \Rightarrow^* l_1 l_2 \dots l_n \iff F(\sigma_0, l_1 \dots l_n) \in Ac$$

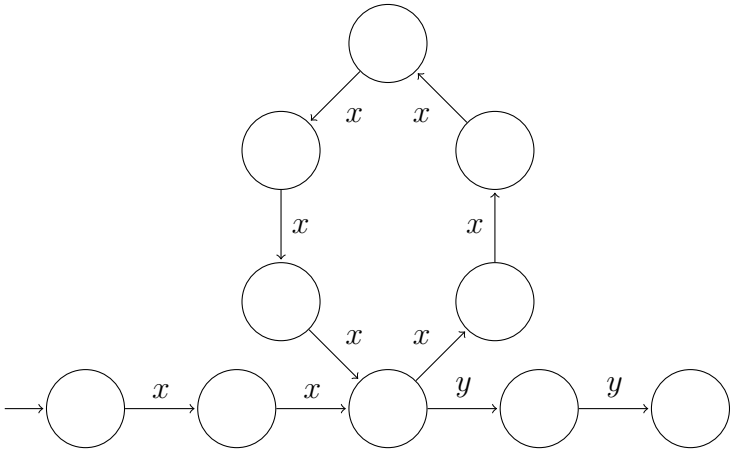
Queda como ejercicio al lector completar esta demostración.

Observación Este teorema nos indica que los AEF no sirven para reconocer lenguajes independientes de contexto.

7.1.2.7. Lema del bombeo para AEF

Enunciado Sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF, $L = \mathcal{AC}(A)$ y $x, y \in \Sigma$, si L contiene infinitas cadenas de la forma $x^n y^n$ entonces también contiene infinitas cadenas de la forma $x^m y^n$ con $m \neq n$.

Demostración El numero de estados de A ($|S|$) es finito. Sea $n > |S| : x^n y^n \in L$. Observemos que durante el proceso de aceptación de la palabra $x^n y^n$ (en particular, de la subcadena x^n) el autómata A deberá pasar sucesivamente por n estados conectados por el caracter x. Los estados visitados en dicha secuencia no tienen por que ser, en general, necesariamente distintos. Sin embargo, como el numero de transiciones es mayor que el de estados disponibles, al menos un estado debe aparecer repetido como muestra el siguiente diagrama:



De aquí concluimos que A acepta también la palabra $x^{n+m}y^n$ (donde m denota la longitud del bucle).

7.1.2.8. Existencia de lenguajes libres de contexto

Enunciado No existe un AEF tal que $\mathcal{AC}(A) = \{a^n b^n / n \in \mathbb{N}_0\}$.

Demostración Supongamos por el absurdo que $\exists A \in AEF / \mathcal{AC}(A) = \{a^n b^n / n \in \mathbb{N}_0\}$. Dado que $\mathcal{AC}(A)$ contiene infinitas cadenas de la forma $a^n b^n$, por el lema del bombeo concluimos que $\mathcal{AC}(A)$ también contiene cadenas de la forma $a^m b^n$ con $m \neq n$ lo cual contradice la definición del lenguaje.

Observación Una consecuencia de este teorema es que los AEF no sirven como analizadores léxicos de expresiones aritméticas que contienen paréntesis, pues el autómata debería aceptar cadenas de la forma $(^n)^n$ pero de ser así también aceptaría por ejemplo $((^n)^n$ que es incorrecta.

7.1.3. Autómata de estado finito no determinista

7.1.3.1. Definición

Un AEF no determinista ($AEFND$) es una quintupla $A = (\Sigma, S, R, Ac, \sigma)$ donde:

- Σ es un conjunto finito de *símbolos de entrada*.
- S es un conjunto finito de *estados*.
- $R \subseteq S \times \Sigma \times S$ es una *relación de transición*.

- $Ac \subseteq S$ es un conjunto de *estados de aceptación.*
- $\sigma \in S$ es el *estado inicial.*

Alternativamente podríamos pensar a R como una función $g : S \times \Sigma \to \mathcal{P}(S)$, es decir $g(\sigma, c) = \{\sigma' \in S / (\sigma, c, \sigma') \in R\}$.

Observación Como hemos cambiado una relación por una función, ahora al leer un nuevo símbolo se podrán seguir múltiples caminos. Ademas no es necesario que desde cada estado, se puedan leer todas las letras del alfabeto.

7.1.3.2. Lenguaje aceptado por un $AEFND$

Sea $A = (\Sigma, S, R, Ac, \sigma)$ un $AEFND$ y $p \in \Sigma^*$, luego:

- $p = \lambda \in \mathcal{AC}(A) \iff \sigma \in Ac$.
- $p = p(1)\, p(2) \ldots p(n) \in \mathcal{AC}(A) \iff \exists \sigma_0, \sigma_1, \ldots, \sigma_n$ tales que:
  1. $\sigma_0 = \sigma$.
  2. $(\sigma_{i-1}, p(i), \sigma_i) \in R \ \forall i = 1, \ldots, n$.
  3. $\sigma_n \in Ac$.

A toda secuencia $(\sigma_0, \sigma_1, \ldots, \sigma_n)$ que cumple con (1) y (2) pero no necesariamente con (3) se la llama secuencia que representa a p. En otras palabras, en un $AEFND$ una cadena p sera:

- Aceptada si existe un camino que la represente y termine en Ac.
- No aceptada si no existe camino que la represente o bien todo camino que la representa termina en $s \in S - Ac$.

7.1.3.3. Equivalencia entre AEF y $AEFND$

Dado que toda función es a su vez una relación, resulta que todo AEF es también un $AEFND$, por lo que:

$$\{L / \exists A \in AEF : \mathcal{AC}(A) = L\} \subseteq \{L / \exists A \in AEFND : \mathcal{AC}(A) = L\}$$

Mas precisamente sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF definimos $A' = (\Sigma, S, R, Ac, \sigma_0)$ con R dado por $(P, l, Q) \in R \iff f(P, l) = Q$, luego A' es $AEFND$ y $\mathcal{AC}(A') = \mathcal{AC}(A)$. Veremos a continuación que en efecto, ambos conjuntos son idénticos.

Teorema de Kleene-Rabin-Scott Para cada $AEFND$ existe un AEF que acepta el mismo lenguaje.

Demostración Sea $A = (\Sigma, S, R, Ac, \sigma_0)$ un $AEFND$ y $A' = (\Sigma, S', f, Ac', s'_0)$ donde:

- $S' = \mathcal{P}(S)$.
- $Ac' = \{s' \in \mathcal{P}(S) / s' \cap Ac \neq \emptyset\}$.
- $s'_0 = \{s_0\}$.
- $f : S' \times \Sigma \to S'$ tal que $f(s', l) = \{s \in S / \exists u \in s' : (u, l, s) \in R\}$.

Debemos mostrar ahora que $\mathcal{AC}(A) = \mathcal{AC}(A')$. Para mostrar que $p \in \mathcal{AC}(A) \iff p \in \mathcal{AC}(A')$ haremos inducción sobre la cantidad de letras de p, mas precisamente mostraremos que $\forall n \in \mathbb{N}_0$ vale el siguiente enunciado E_n: «*Para cada ruta en A que va de su estado inicial s_0 a un estado s_n, existe una ruta en A' que va de su estado inicial $s'_0 = \{s_0\}$ a un estado s'_n tal que $s_n \in s'_n$. Recíprocamente, para cada ruta en A' que va de s'_0 a un estado s'_n , y para cada $s_n \in s'_n$, existe una ruta en A que va de s_0 a s_n*».

- Caso base $n = 0$: Trivial. En A, $s_n = s_0$ corresponde al caso en que la entrada es λ y en A' $s'_n = s'_0$.
- Paso inductivo: Supongamos que vale E_n y veamos que vale E_{n+1}. Por H. I. existe una ruta en A' de la forma $s'_0, \ldots, s'_n$ tal que $s_n \in s'_n$ y dado que $(s_n, l_{n+1}, s_{n+1}) \in R$ existe un arco en A' rotulado l_{n+1} de s'_n a un estado que contiene a s_{n+1}. Llamemoslo s'_{n+1}. Por lo tanto existe una ruta en A', de s'_0 a s'_{n+1} tal que $s_{n+1} \in s'_{n+1}$. Recíprocamente consideremos una ruta en A' de la forma $s'_0, \ldots, s'_n, s'_{n+1}$ que recorre los arcos rotulados $l_1, \ldots, l_{n+1}$. Para todas las transiciones sobre esta ruta tenemos que $f(s'_i, l_{i+1}) = s'_{i+1}$; en particular, para la ultima de ellas $f(s'_n, l_{n+1}) = s'_{n+1}$.
  Por H. I., para cada $s_n \in s'_n$ debe existir una ruta en A que va de s_0 a s_n y dado que $f(s'_n, l_{n+1}) = s'_{n+1}$, por definición de f tenemos que s'_{n+1} es el conjunto de estados $s \in S$ a los que se puede llegar desde un estado de s'_n siguiendo un arco con etiqueta l_{n+1}. Por lo tanto, para cada $s \in s'_{n+1}$ existe una ruta en A que va desde s_0 a s.

7.1.3.4. Igualdad entre $\mathcal{AC}(AEF)$, $\mathcal{AC}(AEFND)$ y $\mathcal{L}_3$

Puesto que toda función es ademas relación sabíamos que $\mathcal{AC}(AEF) \subseteq \mathcal{AC}(AEFND)$ y por el teorema de Kleene-Rabin-Scott sabemos que $\mathcal{AC}(AEFND) \subseteq \mathcal{AC}(AEF)$ por lo que $\mathcal{AC}(AEF) = \mathcal{AC}(AEFND)$. Ademas también probamos que dado un AEF existe una gramática regular que genera el mismo lenguaje. En síntesis: $\mathcal{AC}(AEFND) = \mathcal{AC}(AEF) \subseteq \mathcal{L}_3$. A continuación veremos que estos tres conjuntos son iguales.

Enunciado Para todo lenguaje regular, existe un $AEFND$ que lo acepta.

Demostración Queda como ejercicio al lector completar esta demostración.

Conclusión Como $\mathcal{AC}(AEFND) = \mathcal{AC}(AEF) \subseteq \mathcal{L}_3$ y acabamos de probar $\mathcal{L}_3 \subseteq \mathcal{AC}(AEFND)$ resulta: $\mathcal{AC}(AEF) = \mathcal{AC}(AEFND) = \mathcal{L}_3$.

7.2. Expresiones regulares

7.2.1. Definición

Una expresión regular (ER) sobre un alfabeto Σ es una cadena sobre el alfabeto $\Sigma \cup \{(,), \circ, \cup, *, \emptyset\}$ definida inductivamente como:

- $\emptyset \in ER$.
- $x \in \Sigma \Rightarrow x \in ER$.
- $p, q \in ER \Rightarrow (p \cup q) \in ER$.
- $p, q \in ER \Rightarrow (p \circ q) \in ER$.
- $p \in ER \Rightarrow (p^*) \in ER$.

Para reducir el numero de paréntesis, convenimos el siguiente orden de precedencia: $*, \circ, \cup$.

Observación Nótese que una expresión regular es una cadena de símbolos, no un lenguaje.

7.2.2. Lenguaje asociado

El lenguaje asociado a una expresión esta dado por la función $L : ER \to \mathcal{L}$ que tiene las siguientes propiedades:

- $L(\emptyset) = \emptyset$.
- $x \in ER \wedge x \in \Sigma \Rightarrow L(x) = \{x\}$.
- $p, q \in ER \Rightarrow L(p \cup q) = L(p) \cup L(q)$.
- $p, q \in ER \Rightarrow L(p \circ q) = L(p) \circ L(q)$.
- $p \in ER \Rightarrow L(p^*) = L(p)^*$.

Definimos ademas $L_{ER} = \{L \in \mathcal{L} / \exists e \in ER : L(e) = L\}$, el conjunto de todos los lenguajes asociados a expresiones regulares.

7.2.3. Igualdad entre lenguajes asoc. a L_{ER} y $\mathcal{L}_3$

Enunciado El conjunto de todos los lenguajes asociados a expresiones regulares es igual al conjunto de todos los lenguajes regulares.

Demostración Deberemos probar una doble contención:

- $L_{ER} \subseteq \mathcal{L}_3$: Lo demostraremos por inducción sobre ER
  - $e = \emptyset \Rightarrow L(e) = \emptyset \in \mathcal{L}_3$.
  - $e = x \Rightarrow L(e) = \{x\} \in \mathcal{L}_3$.
  - $e = p \cup q \Rightarrow$: Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \cup q) = L(p) \cup L(q) \in \mathcal{L}_3$ y la unión es cerrada en $\mathcal{L}_3$. (Ejercicio).
  - $e = p \circ q \Rightarrow$: Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \circ q) = L(p) \circ L(q) \in \mathcal{L}_3$ y la concatenación es cerrada en $\mathcal{L}_3$. (Ejercicio).
  - $e = p^* \Rightarrow$: Debemos probar que $L(p) \in \mathcal{L}_3 \Rightarrow L(p^*) = L(p)^* \in \mathcal{L}_3$ y la clausura de Kleene es cerrada en $\mathcal{L}_3$. (Ejercicio).
- $\mathcal{L}_3 \subseteq L_{ER}$: Consultar «J. Glenn Brookshear. *Teoría de la Computación. Lenguajes formales, autómatas y complejidad.*», pagina 63.