Teoría de la Computación

Clase 2: Autómatas finitos deterministas (`DFAs`) y no deterministas (`NFAs`)

Mauro Artigiani

30 julio 2021

Universidad del Rosario, Bogotá

Definición

Definición

Definición

Un autómata finito es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado los estados;
2. Σ es un conjunto finito llamado alfabeto;
3. $\delta\colon Q \times \Sigma \to Q$ es la función de transición;
4. $q_0 \in Q$ es el estado inicial;
5. $F \subseteq Q$ es el conjunto de los estados aceptados;

Definición

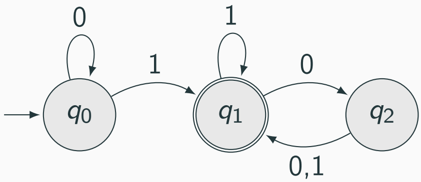
Definición

Un autómata finito es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado los estados;
2. Σ es un conjunto finito llamado alfabeto;
3. $\delta \colon Q \times \Sigma \to Q$ es la función de transición;
4. $q_0 \in Q$ es el estado inicial;
5. $F \subseteq Q$ es el conjunto de los estados aceptados;

Un autómata finito de este tipo se llama también autómata finito determinista o `DFA`.

Definición



Podemos describir formalmente este autómata en la siguiente manera: $Q = \{q_0, q_1, q_2\}$; $\Sigma = \{0, 1\}$; q_0 es el estado inicial; $F = \{q_1\}$; δ :

$\delta(\cdot, \cdot)$	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

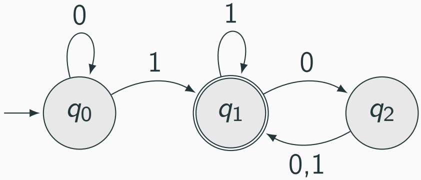
Lenguajes regulares

Lenguajes regulares

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un `DFA`. Consideramos una palabra w en el alfabeto Σ, $w = w_1 w_2 \dots w_n$. Decimos que M acepta w si existe una secuencia de estados r_0, r_1, ..., r_n en Q tal que:

1. $r_0 = q_0$;
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ para todos $i = 0$, ..., $n - 1$;
3. $r_n \in F$.

Lenguajes regulares



El autómata acepta la palabra $w = 1100$. En efecto, la secuencia de estados $r_0 = q_0$, $r_1 = q_1$, $r_2 = q_1$, $r_3 = q_2$, $r_4 = q_1$ satisface todos los requerimientos.

Lenguajes regulares

Decimos que M reconoce el lenguaje A si $A = \{w, M \text{ acepta } w\}$.

Lenguajes regulares

Decimos que M reconoce el lenguaje A si $A = \{w, M \text{ acepta } w\}$.

Definición

Un lenguaje A se llama un lenguaje regular si existe un autómata determinista finito M que reconoce A.

Lenguajes regulares

Decimos que M reconoce el lenguaje A si $A = \{w, M \text{ acepta } w\}$.

Definición

Un lenguaje A se llama un lenguaje regular si existe un autómata determinista finito M que reconoce A.

En la otra clase nos hemos visto que el lenguaje formado para las palabras w, en alfabeto $\Sigma = \{0, 1\}$, tales que w contiene por lo menos un 1 y hay un número pares de 0 después del último 1 es un lenguaje regular.

Operaciones entre lenguajes

Operaciones entre lenguajes

Como hay operaciones entre números, también hay operaciones entre lenguajes.

Definición

Sean A y B lenguajes. Definimos las siguientes operaciones entre ellos:

- Unión: $A \cup B = \{x, x \in A \text{ o } x \in B\}$;
- Intersección: $A \cap B = \{x, x \in A \text{ y } x \in B\}$;
- Concatenación: $A \circ B = \{xy, x \in A \text{ y } y \in B\}$;
- Potencia: $A^* = \{x_1 x_2 \ldots x_k, k \geq 0 \text{ y } x_i \in A\}$.

Ejemplo

Sea $\Sigma = \{0, 1\}$ un alfabeto. Sean $A = \{00, 11\}$ y $B = \{01, 00, 1\}$ dos lenguajes. Entonces

- $A \cup B = \{00, 11, 01, 1\}$;
- $A \cap B = \{00\}$;
- $A \circ B = \{0001, 0000, 001, 1101, 1100, 111\}$;
- $A^* = \{\varepsilon, 00, 11, 0000, 0011, 1100, 1111, \dots\}$.

Clausura de los lenguajes

Teorema

La clase de los lenguajes regulares es cerrada bajo la operación de unión. Es decir, si A_1 y A_2 son dos lenguajes regulares, entonces también $A_1 \cup A_2$ es un lenguajes regular.

Clausura de los lenguajes

Teorema

La clase de los lenguajes regulares es cerrada bajo la operación de unión. Es decir, si A_1 y A_2 son dos lenguajes regulares, entonces también $A_1 \cup A_2$ es un lenguajes regular.

Siendo A_1 y A_2 regulares, existen dos `DFA` M_1 y M_2 que los reconocen. La idea es construir un autómata M que *simule* tanto M_1 cuanto M_2 al mismo tiempo. Dado que M no sabe si la palabra que ha recibido sea en M_1 o M_2 (o en ninguno) necesitamos tener ambos casos en cuenta.

Demostración

Sea $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ un `DFA` que reconoce A_1 y $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ uno que reconoce A_2.

Demostración

Sea $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ un `DFA` que reconoce A_1 y $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ uno que reconoce A_2.

Vamos a construir formalmente el autómata $M = (Q, \Sigma, \delta, q_0, F)$ que reconozca $A_1 \cup A_2$.

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2), r_1 \in Q_1, r_2 \in Q_2\}$;
2. Σ es el mismo para M_1 y M_2. Si fueran distintos hubiéramos definido $\Sigma = \Sigma_1 \cup \Sigma_2$;
3. Sean $(r_1, r_2) \in Q$ un estado y $a \in \Sigma$ una letra. Definamos $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$;
4. $q_0 = (q_1, q_2)$;
5. $F = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(r_1, r_2), r_1 \in F_1 \text{ o } r_2 \in F_2\}$. $\square$

Demostración

Sea $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ un `DFA` que reconoce A_1 y $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ uno que reconoce A_2.

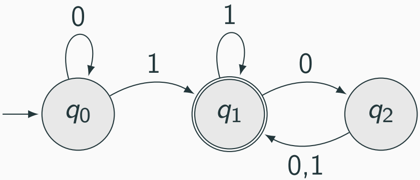
Vamos a construir formalmente el autómata $M = (Q, \Sigma, \delta, q_0, F)$ que reconozca $A_1 \cup A_2$.

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2), r_1 \in Q_1, r_2 \in Q_2\}$;
2. Σ es el mismo para M_1 y M_2. Si fueran distintos hubiéramos definido $\Sigma = \Sigma_1 \cup \Sigma_2$;
3. Sean $(r_1, r_2) \in Q$ un estado y $a \in \Sigma$ una letra. Definamos $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$;
4. $q_0 = (q_1, q_2)$;
5. $F = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(r_1, r_2), r_1 \in F_1 \text{ o } r_2 \in F_2\}$. $\square$

La misma construcción, con $F = F_1 \times F_2$ muestra que la intersección de lenguajes regulares es un lenguaje regular.

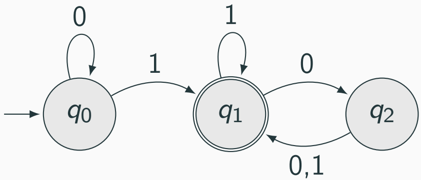
Un ejemplo

Sea A_1 el lenguaje regular reconocido por el `DFA` M_1:

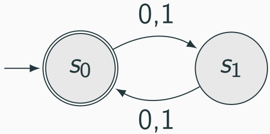


Un ejemplo

Sea A_1 el lenguaje regular reconocido por el `DFA` M_1:

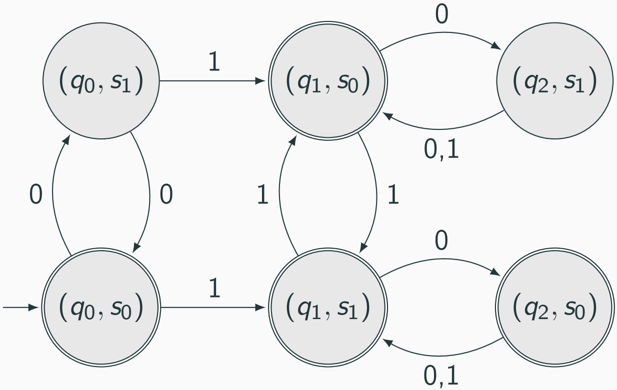


Sea A_2 el lenguaje regular reconocido por el `DFA` M_2:



Un ejemplo

El autómata M que reconoce $A_1 \cup A_2$ es el siguiente:

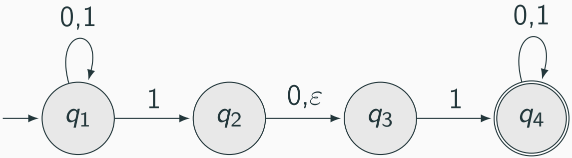


Clausura bajo las otras operaciones

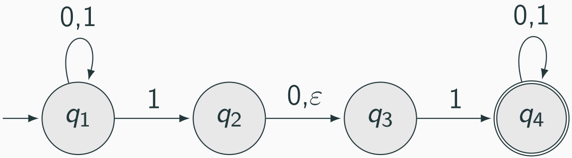
En las próximas clases mostraremos que los lenguajes regulares son cerrados también bajo las otras dos operaciones: concatenación y potencia. Demostrar esto es muy difícil con los `DFA`s. Por eso ahora introducimos los autómatas finitos no deterministas o `NFA`s.

Autómatas finitos no deterministas

Un ejemplo

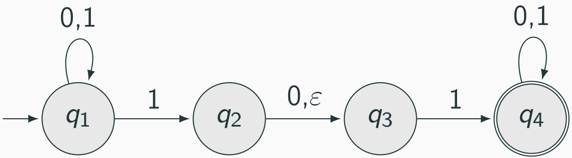


Un ejemplo



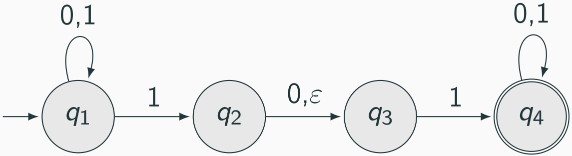
- No siempre hay 1 sola arista para cada símbolo de Σ que sale de cada estado.
- Hay aristas con ε como etiqueta.

Cómo calcula un NFA?



La palabra clave es *forking*.

Cómo calcula un NFA?



La palabra clave es *forking*.

El autómata no determinista empieza en el estado inicial. Cuando encuentra un símbolo repetido el autómata se divide en varias copias y en cada copia sigue una posible arista con ese símbolo. Desde este momento en adelante, el NFA sigue en paralelo todos estos caminos. Si en algún momento un proceso está en un estado que no tiene una arista con el símbolo que está leyendo el autómata, ese proceso *muere*.

Cómo calcula un `NFA`?

De manera parecida, si el autómata pasa por un estado que tiene una (o más) arista con etiqueta ε se divide en varias copias: una permanece en el mismo estado, cada otra copia sigue una arista con etiqueta ε.

Cómo calcula un `NFA`?

De manera parecida, si el autómata pasa por un estado que tiene una (o más) arista con etiqueta ε se divide en varias copias: una permanece en el mismo estado, cada otra copia sigue una arista con etiqueta ε.

Cuándo un autómata no determinista acepta una palabra?

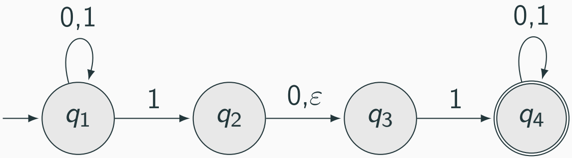
Cómo calcula un NFA?

De manera parecida, si el autómata pasa por un estado que tiene una (o más) arista con etiqueta ε se divide en varias copias: una permanece en el mismo estado, cada otra copia sigue una arista con etiqueta ε.

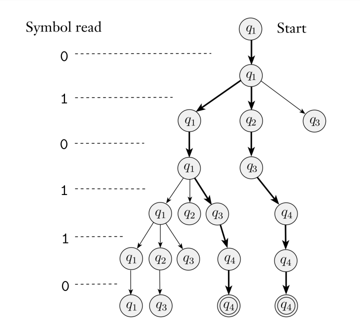
Cuándo un autómata no determinista acepta una palabra?

Cuando por lo menos un proceso termina en un estado de aceptación.

Cómo calcula un NFA?



Por ejemplo, si leemos la palabra 010110 esto pasa:

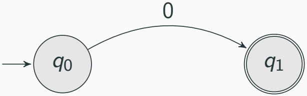


Ejemplos

Construimos un NFA M tal que $L(M) = \{0\}$

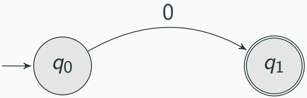
Ejemplos

Construimos un NFA M tal que $L(M) = \{0\}$



Ejemplos

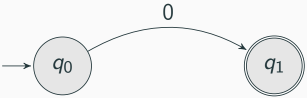
Construimos un `NFA` M tal que $L(M) = \{0\}$



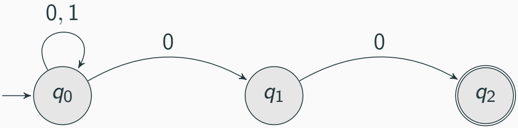
Construimos un `NFA` M tal que $L(M) = \{w \in \Sigma^* \colon w \text{ termina en } 00\}$

Ejemplos

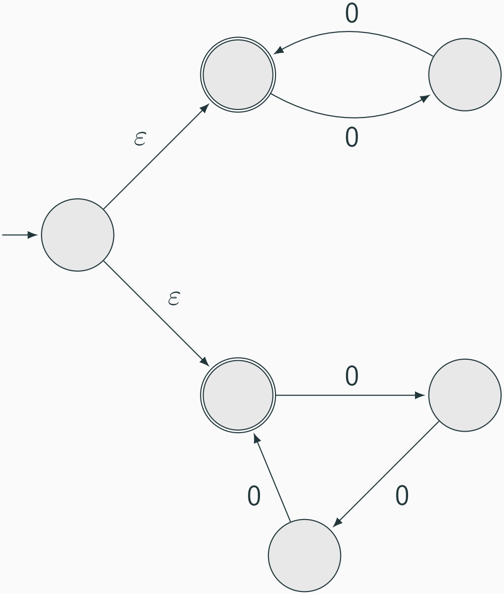
Construimos un `NFA` M tal que $L(M) = \{0\}$



Construimos un `NFA` M tal que $L(M) = \{w \in \Sigma^* \colon w \text{ termina en } 00\}$



Otro ejemplo



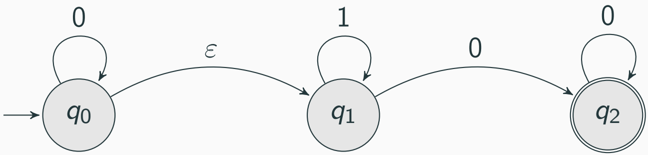
Diseño de NFAs

Ejemplo

Sea $L(M) = \{0^n 1^m 0^k \colon n, m \geq 0 \text{ y } k \geq 1\}$:

Ejemplo

Sea $L(M) = \{0^n 1^m 0^k \colon n, m \geq 0 \text{ y } k \geq 1\}$:



Definición de NFA

Definición

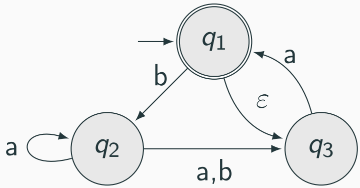
Somos listo para dar la definición formal de un `NFA`. Escribimos $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ y $\mathcal{P}(A)$ para el conjunto de potencia del conjunto A.

Definición

Un autómata finito no determinista es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado los estados;
2. Σ es un conjunto finito llamado alfabeto;
3. $\delta \colon Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ es la función de transición;
4. $q_0 \in Q$ es el estado inicial;
5. $F \subseteq Q$ es el conjunto de los estados aceptados;

Descripción formal de un NFA



Podemos describir formalmente este autómata en la siguiente manera: $Q = \{q_1, q_2, q_3\}$; $\Sigma = \{a, b\}$; q_1 es el estado inicial; $F = \{q_1\}$; δ :

$\delta(\cdot,\cdot)$	a	b	ε
q_1	$\emptyset$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_2, q_3\}$	$\{q_3\}$	$\emptyset$
q_3	$\{q_1\}$	$\emptyset$	$\emptyset$

Cuándo un NFA acepta una palabra

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un NFA. Consideramos una palabra w en el alfabeto Σ, tal que $w = y_1 y_2 \dots y_n$, con $y_i \in \Sigma_\varepsilon$. Decimos que M acepta w si existe una secuencia de estados r_0, r_1, ..., r_n en Q tal que:

1. $r_0 = q_0$;
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ para todos $i = 0$, ..., $n-1$;
3. $r_n \in F$.

Resumen

Resumen

Hoy aprendimos:

- Las definiciones formales de `DFA` y `NFA`;
- Las operaciones regulares;
- Como construir un `DFA` que reconozca la unión de dos lenguajes regulares;
- Como procesa un `NFA`.