Autómatas en dos direcciones

Clase 10

IIC 2223

Prof. Cristian Riveros

¿cuánto se parece un autómata a un algoritmo?

¿cuáles son las diferencias?

1. Memoria.
2. “Movimiento” de la máquina.

En esta clase, veremos como extender autómatas con 2.

Outline

2DFA

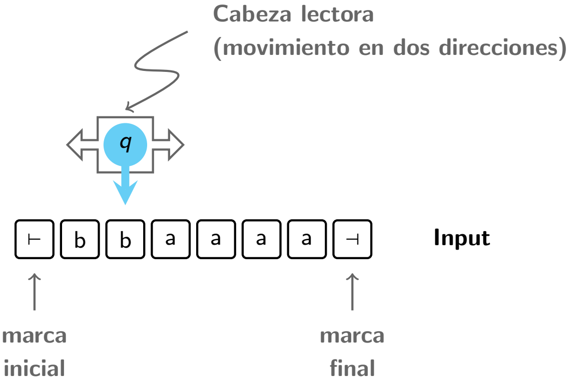
2DFA vs DFA

Outline

2DFA

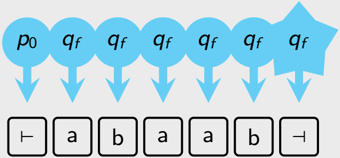
2DFA vs DFA

Autómatas finitos deterministas en **dos direcciones**



Autómatas finitos deterministas en dos direcciones

Ejemplo



Autómatas finitos deterministas en dos direcciones

Definición

Un autómata finito determinista en 2 direcciones (2DFA) es una estructura:

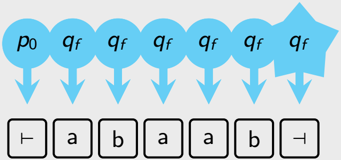
$$\mathcal{A} \quad = \quad (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_f)$$

- Q es un conjunto finito de estados.
- Σ es el alfabeto de input.
- $\vdash$ y $\dashv$ son las marcas (simbolos) iniciales y finales.
- $\delta : Q \times \left(\Sigma \cup \{\vdash, \dashv\}\right) \rightharpoonup Q \times \{\boldsymbol{\leftarrow}, \boldsymbol{\rightarrow}\}$ es la **función parcial de transición**.
- q_0 es el estado inicial.
- q_f es el estado final.

Autómatas finitos deterministas en dos direcciones

Ejemplo

$$L = \left\{ w \in \Sigma^* \;\middle|\; \big(\#a(w) = 0 \mod 3\big) \text{ y } \big(\#b(w) = 0 \mod 2\big) \right\}$$



	$\vdash$	a	b	$\dashv$
q_0	$(q_0, \rightarrow)$	$(q_1, \rightarrow)$	$(q_0, \rightarrow)$	$(p_0, \leftarrow)$
q_1	-	$(q_2, \rightarrow)$	$(q_1, \rightarrow)$	-
q_2	-	$(q_0, \rightarrow)$	$(q_2, \rightarrow)$	-
p_0	$(q_f, \rightarrow)$	$(p_0, \leftarrow)$	$(p_1, \leftarrow)$	-
p_1	-	$(p_1, \leftarrow)$	$(p_0, \leftarrow)$	-
q_f	-	$(q_f, \rightarrow)$	$(q_f, \rightarrow)$	-

Configuración de un 2DFA

Sea:

- Un 2DFA $\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_f)$.
- Una palabra $w = a_1 a_2 \ldots a_n \in \Sigma^*$.

Defina $a_0 = \vdash$ y $a_{n+1} = \dashv$ tal que el input se define como:

$$a_0\, a_1\, \ldots\, a_n\, a_{n+1} \;=\; \vdash \cdot\, w \cdot \dashv$$

Una **configuración** de $\mathcal{A}$ sobre w viene dado por un par:

$$(q, i) \;\in\; Q \times \{0, \ldots, n+1\}$$

- q es el **estado actual** del autómata.
- i es la **posición actual** de la cabeza lectora.

Configuración de un 2DFA

Sea:

- Un 2DFA $\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_f)$.
- Una palabra $w = a_1 a_2 \dots a_n \in \Sigma^*$.

Se define la relación de **siguiente configuración** $\xmapsto{\mathcal{A}}$ de $\mathcal{A}$ sobre w como:

$$(p, i) \xmapsto{\mathcal{A}} (q, j)$$

tal que:

- Si $\delta(p, a_i) = (q, \rightarrow)$, entonces $(p, i) \xmapsto{\mathcal{A}} (q, i + 1)$.
- Si $\delta(p, a_i) = (q, \leftarrow)$, entonces $(p, i) \xmapsto{\mathcal{A}} (q, i - 1)$.

¿cómo ejecuto mi 2DFA?

Sea:

- Un 2DFA $\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_f)$.
- El input $w = a_1 a_2 \dots a_n \in \Sigma^*$.

Una **ejecución** (o run) ρ de $\mathcal{A}$ sobre w es una secuencia de configuraciones:

$$\rho : (p_0, i_0) \to (p_1, i_1) \to \dots \to (p_m, i_m)$$

- $p_0 = q_0$ y $i_0 = 0$.
- $(p_j, i_j) \overset{\mathcal{A}}{\longmapsto} (p_{j+1}, i_{j+1}) \quad \forall\, j \in [0, m-1]$

Una ejecución ρ de $\mathcal{A}$ sobre w es de **aceptación** si:

$$p_m = q_f \quad \text{y} \quad i_m = n + 1$$

Lenguaje aceptado por un 2DFA

Sea un autómata $\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_f)$ y $w \in \Sigma^*$.

Definiciones

- $\mathcal{A}$ **acepta** w si hay una ejecución de $\mathcal{A}$ sobre w que es de **aceptación**.
- El **lenguaje aceptado por** $\mathcal{A}$ se define como:

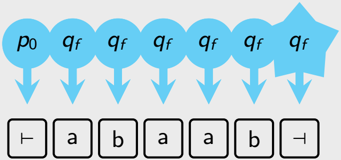
$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ acepta } w\}$$

Un 2DFA puede parar por error o NO parar nunca!

¿cuál es la ejecución de este autómata?

Ejemplo

$$L = \left\{ w \in \Sigma^* \;\middle|\; \big(\#a(w) = 0 \mod 3\big) \text{ y } \big(\#b(w) = 0 \mod 2\big) \right\}$$



	$\vdash$	a	b	$\dashv$
q_0	$(q_0, \rightarrow)$	$(q_1, \rightarrow)$	$(q_0, \rightarrow)$	$(p_0, \leftarrow)$
q_1	-	$(q_2, \rightarrow)$	$(q_1, \rightarrow)$	-
q_2	-	$(q_0, \rightarrow)$	$(q_2, \rightarrow)$	-
p_0	$(q_f, \rightarrow)$	$(p_0, \leftarrow)$	$(p_1, \leftarrow)$	-
p_1	-	$(p_1, \leftarrow)$	$(p_0, \leftarrow)$	-
q_f	-	$(q_f, \rightarrow)$	$(q_f, \rightarrow)$	-

Outline

2DFA

2DFA vs DFA

2DFA vs lenguajes regulares

Para todo lenguaje regular L existe un 2DFA $\mathcal{A}$:

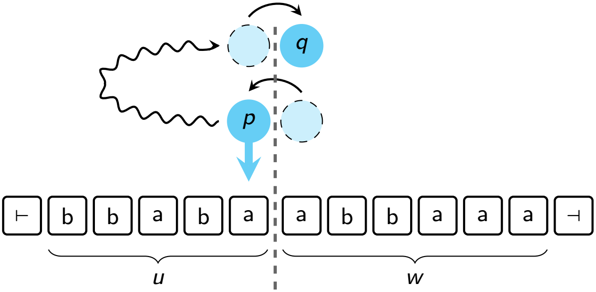
$$L \;=\; \mathcal{L}(\mathcal{A})$$

En otras palabras, DFA $\subseteq$ 2DFA.

¿són los 2DFA mas poderosos que los DFA?

Demostraremos que NO!

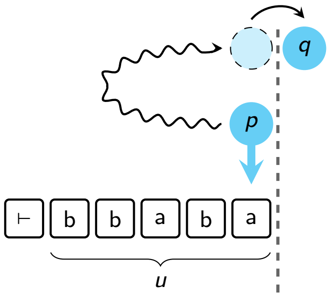
¿cuánta información puede almacenar un 2DFA?



"Cada vez que $\mathcal{A}$ cruce de w a u en el estado p*,*
$\mathcal{A}$ cruzará de regreso en el estado q*."*

¡este comportamiento solo depende de u y no de w!

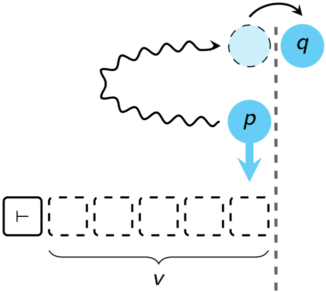
¿cuánta información puede almacenar un 2DFA?



Para cada $u \in \Sigma^*$ definimos la función $T_u : Q \cup \{\bullet\} \to Q \cup \{\perp\}$ tal que:

- $T_u(p) = q$ ssi desde $(p, |u|)$ cruza en la config. $(q, |u|+1)$.
- $T_u(p) = \perp$ ssi desde $(p, |u|)$ nunca cruza de u.
- $T_u(\bullet) = q$ ssi desde $(q_0, 0)$ cruza por 1era vez con $(q, |u|+1)$.
- $T_u(\bullet) = \perp$ ssi desde $(q_0, 0)$ nunca cruza de u.

¿cuánta información puede almacenar un 2DFA?



Suponga ahora que tenemos una palabra v tal que:

$$T_v = T_u$$

Entonces, v es **indistinguible** de u según $\mathcal{A}$

En otras palabras, $u \cdot w \in \mathcal{L}(\mathcal{A}) \Leftrightarrow v \cdot w \in \mathcal{L}(\mathcal{A})$ para todo $w \in \Sigma^*$.

Recordatorio: Teorema de Myhill-Nerode

Sea $L \subseteq \Sigma^*$ cualquier lenguaje.

Definición (Relación de Myhill-Nerode)

Una relación de equivalencia $\equiv$ en Σ^* es de **Myhill-Nerode** para L si:

1. $\equiv$ es una **congruencia por la derecha**:

$$u \equiv v \;\; \text{entonces} \;\; u \cdot w \equiv v \cdot w \qquad \forall w \in \Sigma^*$$

2. $\equiv$ **refina** L.

$$u \equiv v \;\; \text{entonces} \;\; \big(u \in L \;\Leftrightarrow\; v \in L\big)$$

3. El número de clases de equivalencia de $\equiv$ es **finita**.

Recordatorio: Teorema de Myhill-Nerode

Definición

Dado un lenguaje $L \subseteq \Sigma^*$, se define la relación de equivalencia $\equiv_L$ como:

$$u \equiv_L v \quad \text{si, y solo si,} \quad \big(u \cdot w \in L \Leftrightarrow v \cdot w \in L \big) \quad \forall w \in \Sigma^*$$

Teorema de Myhill-Nerode

Sea $L \subseteq \Sigma^*$. Las siguientes propiedades son equivalentes:

1. L es **regular**.
2. existe una **relación de Myhill-Nerode** para L.
3. la relación $\equiv_L$ tiene una cantidad **finita** de clases de equivalencia.

¿cuánta información puede almacenar un 2DFA?

1. La relación $\equiv_T$ entre palabras en Σ^* tal que:

$$u \equiv_T v \quad \text{si, y solo si,} \quad T_u = T_v$$

   es una **relación de equivalencia**. ¿por qué?

2. $\equiv_T$ es una **congruencia por la derecha**. $(u \equiv_T v \;\Rightarrow\; \forall w.\; u \cdot w \equiv_T v \cdot w)$
   ¿por qué?

3. $\equiv_T$ **refina** a $\mathcal{L}(\mathcal{A})$. $(u \equiv_T v \;\Rightarrow\; (u \in L \;\Leftrightarrow\; v \in L))$
   ¿por qué?

4. La relación $\equiv_T$ tiene una **cantidad finita** de clases de equivalencia.

$$T : Q \cup \{\bullet\} \to Q \cup \{\perp\}$$

   ¿cuántas funciones T existen?

$\equiv_{\mathcal{L}(\mathcal{A})}$ tiene una cantidad **finita** de **clases de equivalencia**.

2DFA aceptan solo lenguajes regulares

Teorema

Para todo 2DFA $\mathcal{A}$ existe un DFA $\mathcal{A}'$ tal que:

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$$

Por lo tanto, 2DFA $\equiv$ DFA.

¿cómo construimos el DFA?

- Usando el Teorema de Myhill-Nerode.
- Construyendolo a partir de las funciones T_u.

Ejercicio