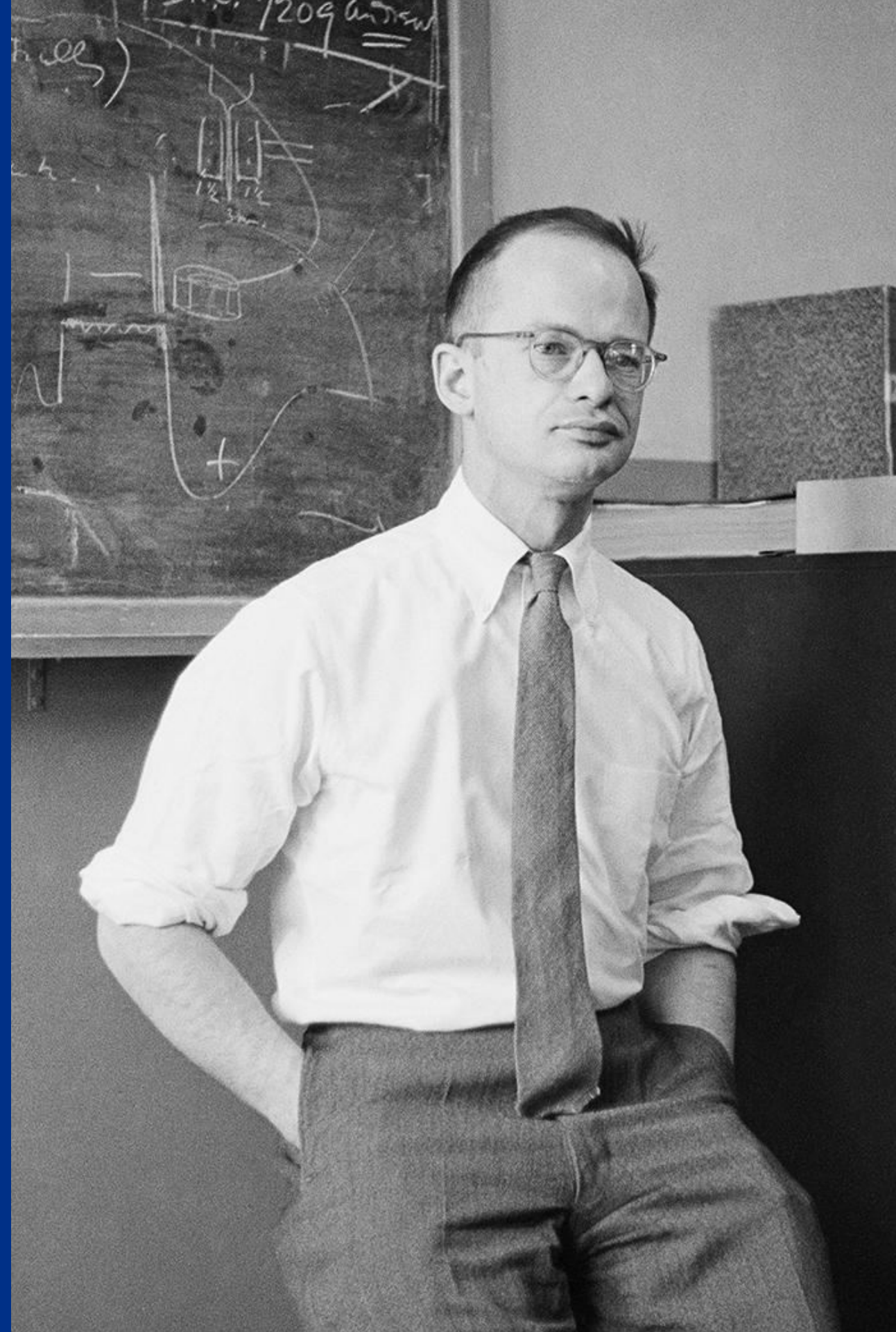


Autómatas finitos



UNIVERSIDAD
DE CHILE

Arturo Merino



Recuerdo / Plan

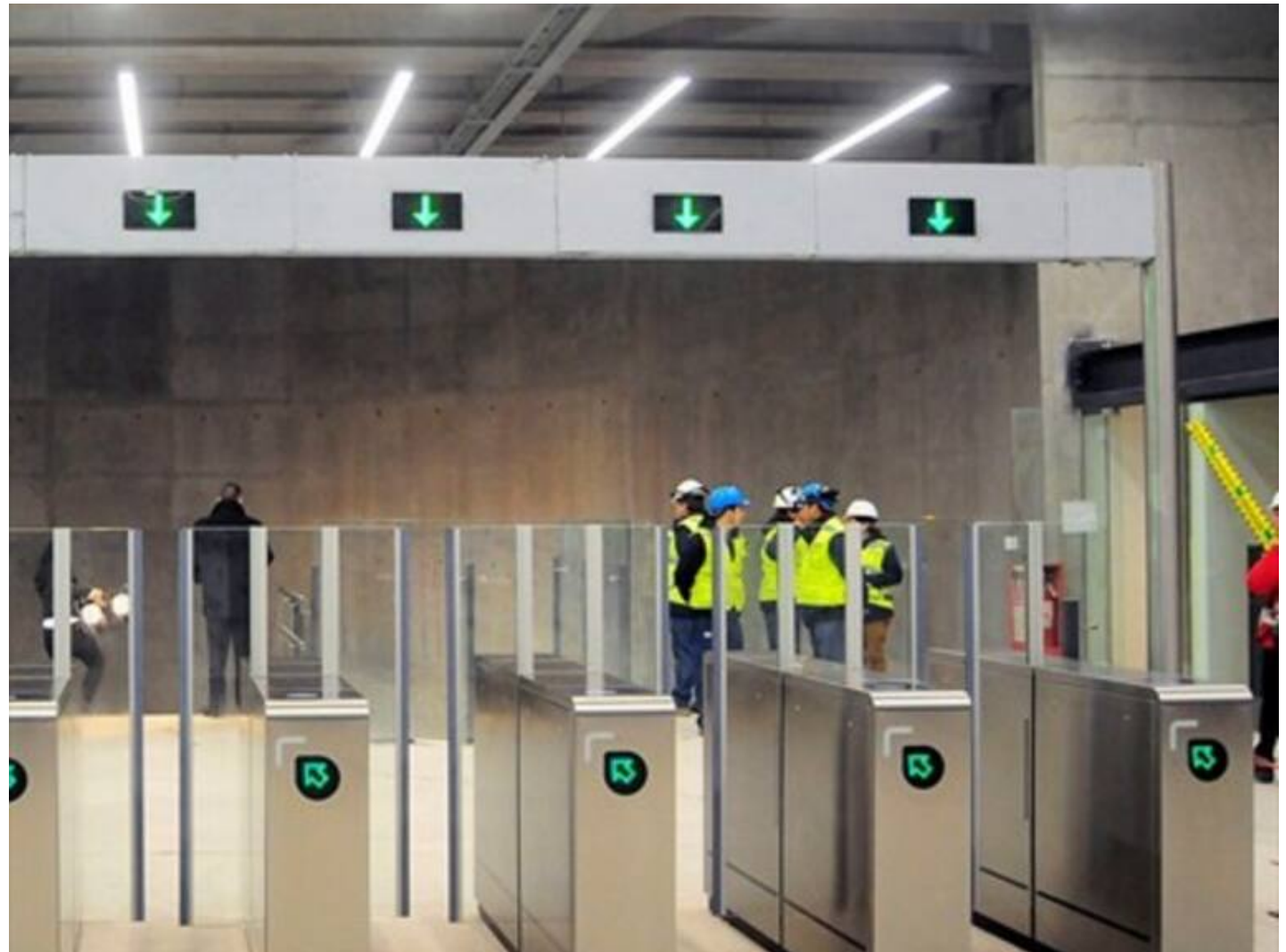
Recuerdo/Plan

- **Recuerdo:**
 - Lenguajes formales
- **Plan:**
 - Autómatas finitos

Autómatas finitos

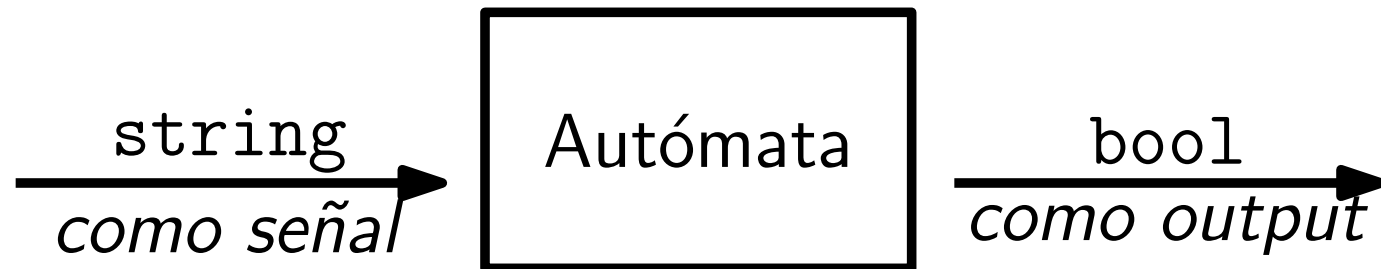
Autómatas

- **Entender** el poder de máquinas “sin” memoria.
- ¿Qué podemos hacer sin memoria?
- Varias cosas
 - Sensor BIP
 - Sensor cruce
 - ¿Estados?
 - ¿Dinámica?



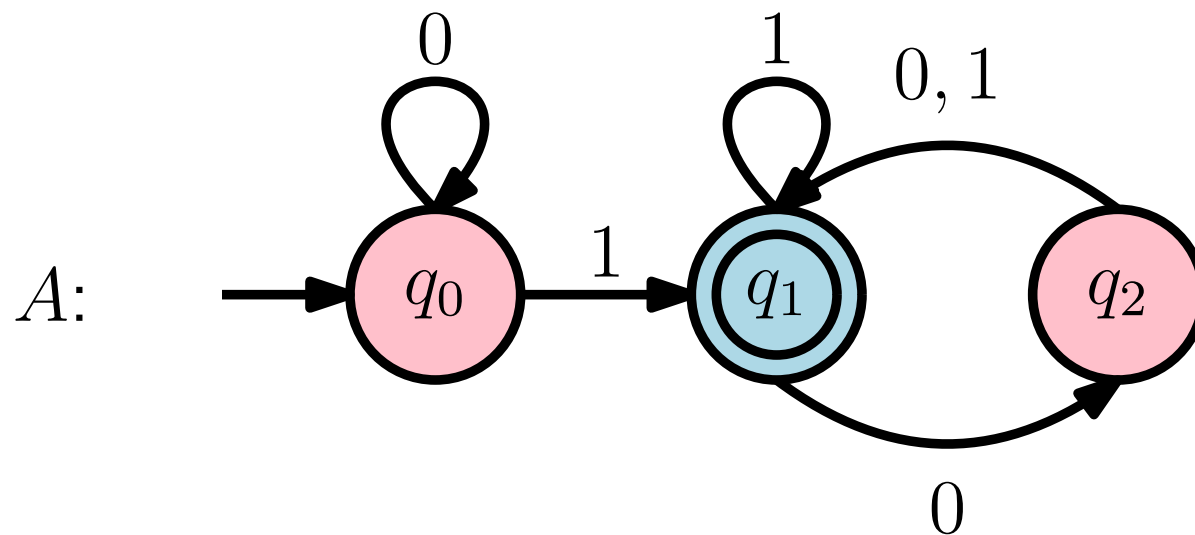
Problemas de decisión

- Nuestro interés: $\text{string} \rightarrow \text{bool}$.



- No recuerda el string, lo *consume*.

- Ej:



- ¿ $A(010011)$?
- ¿ $A(01111111111111111111000)$?
- ¿ $A(\varepsilon)$?

Definición formal

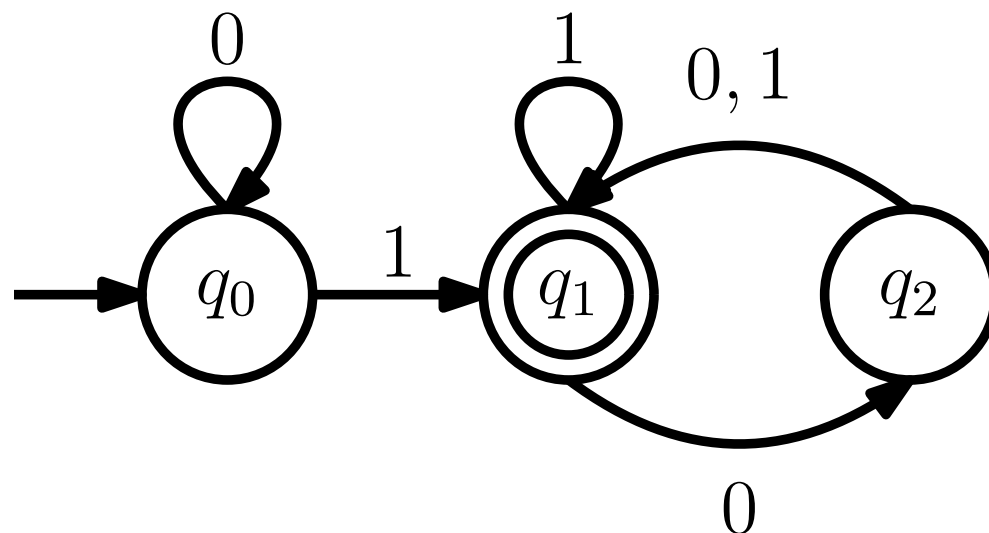
Def.

Un *autómata finito determinista (AFD)* es una 5-tupla

$M = (Q, \Sigma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*
- Σ es un conjunto finito llamado *alfabeto*
- $\delta: Q \times \Sigma \rightarrow Q$ es la *función de transición*.
- $q_0 \in Q$ es el *estado inicial*, y
- $F \subseteq Q$ son los *estados de aceptación (o finales)*.

• Ej:



- Usualmente descrito por el *diagrama de estados*.

Computación

- Formalicemos nuestra idea informal de computación

Def.

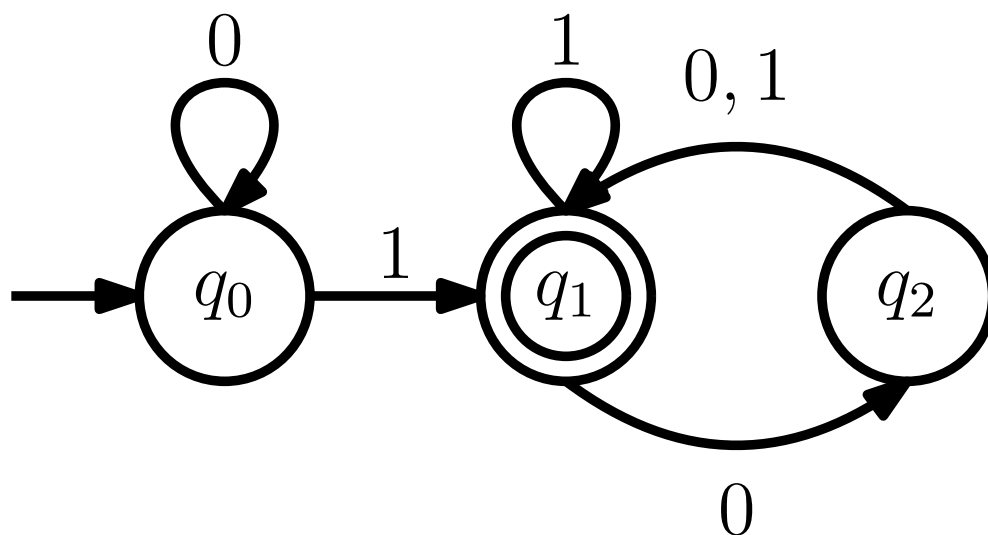
Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFD y $w = w_1 \dots w_n \in \Sigma^*$ un string.

Diremos que M *acepta* w si existen estados $r_0, r_1, \dots, r_n$ tal que

- $r_0 = q_0$ (partimos en el estado inicial)
- $r_i = \delta(r_{i-1}, w_i)$ para todo $i \in \{1, \dots, n\}$ (transiciones válidas)
- $r_n \in F$ (terminamos en aceptación)

- Not:** Usaremos $M(w) = 1$ y $M(w) = 0$ para decir si/no acepta.

- Ej:**



$w = 00101$

Entendiendo autómatas

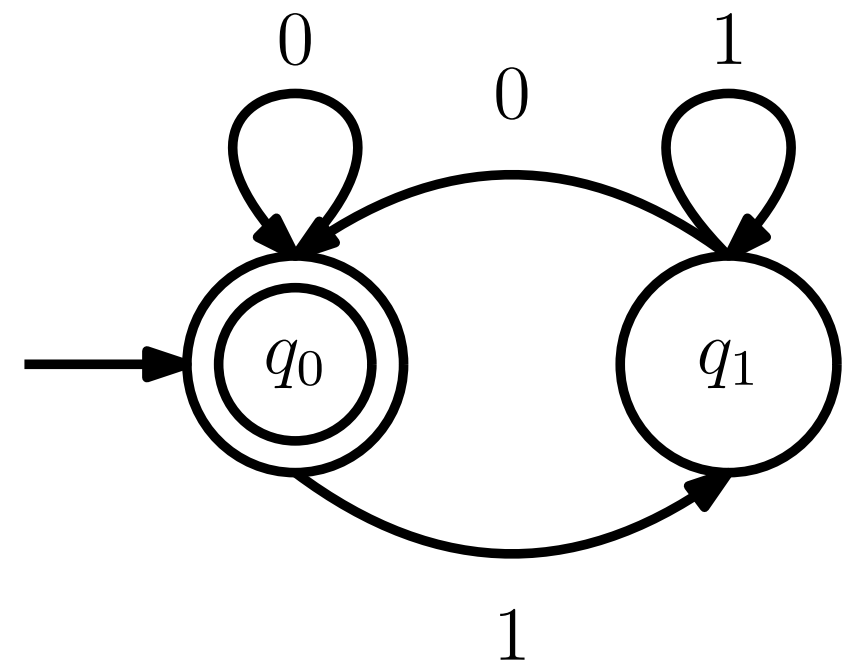
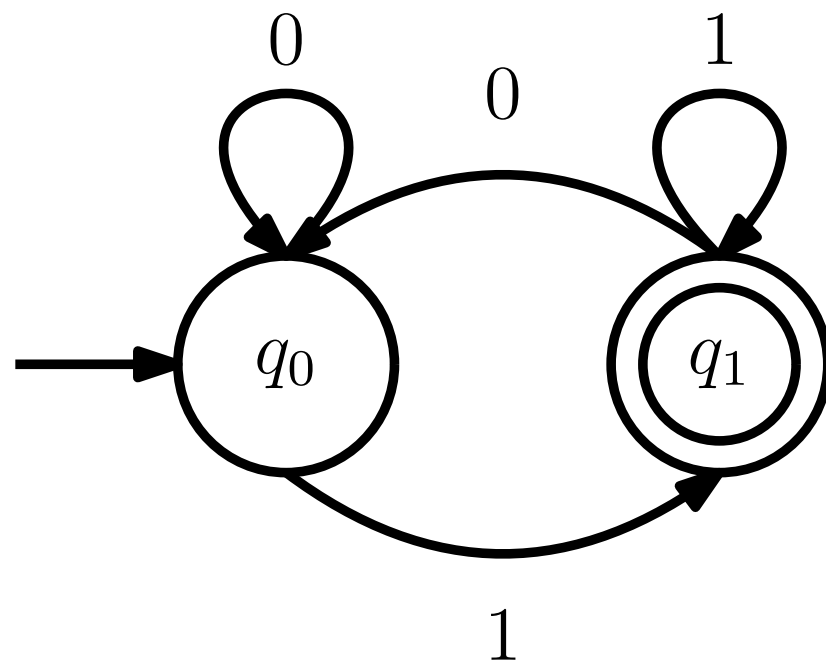
Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un autómata.

El *lenguaje aceptado por M* es

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M(w) = 1\}.$$

• Ej:



Entendiendo autómatas

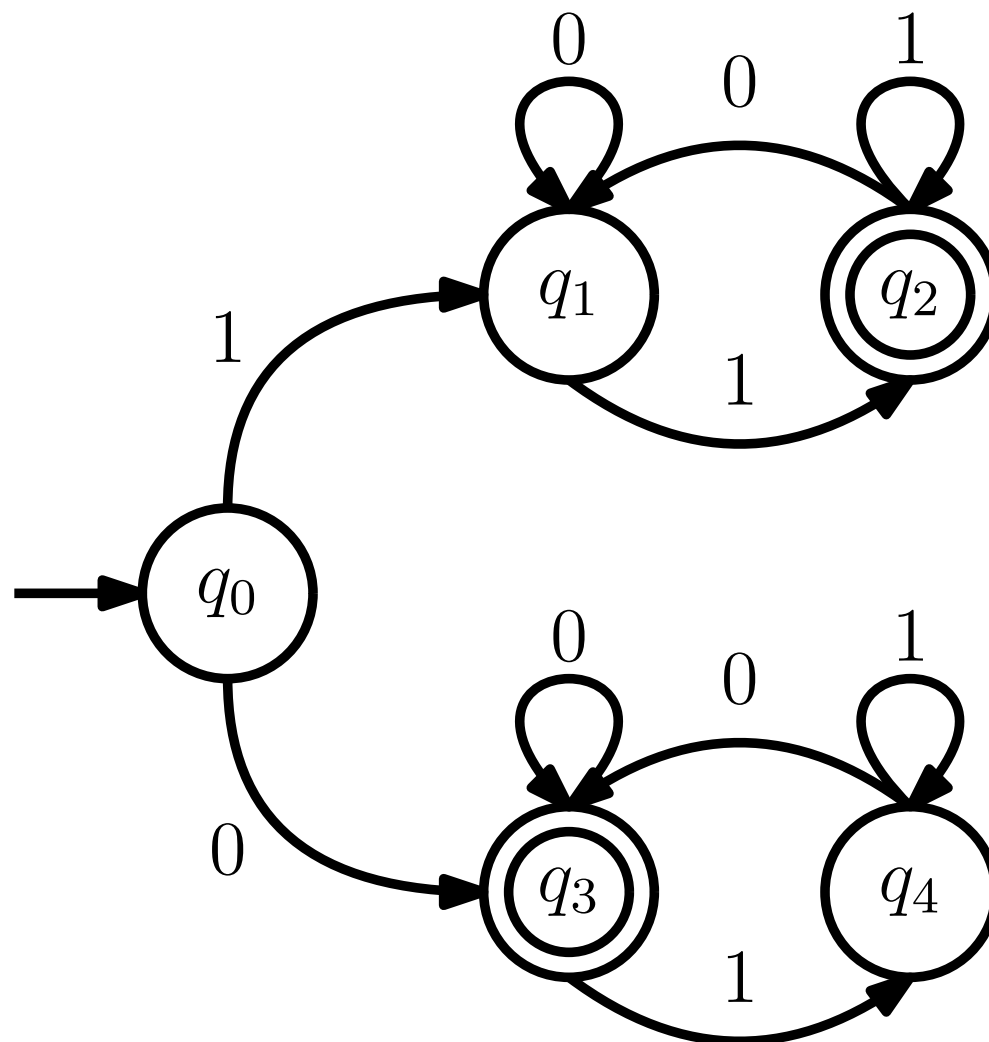
Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un autómata.

El *lenguaje aceptado por M* es

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M(w) = 1\}.$$

• Ej:



Entendiendo autómatas

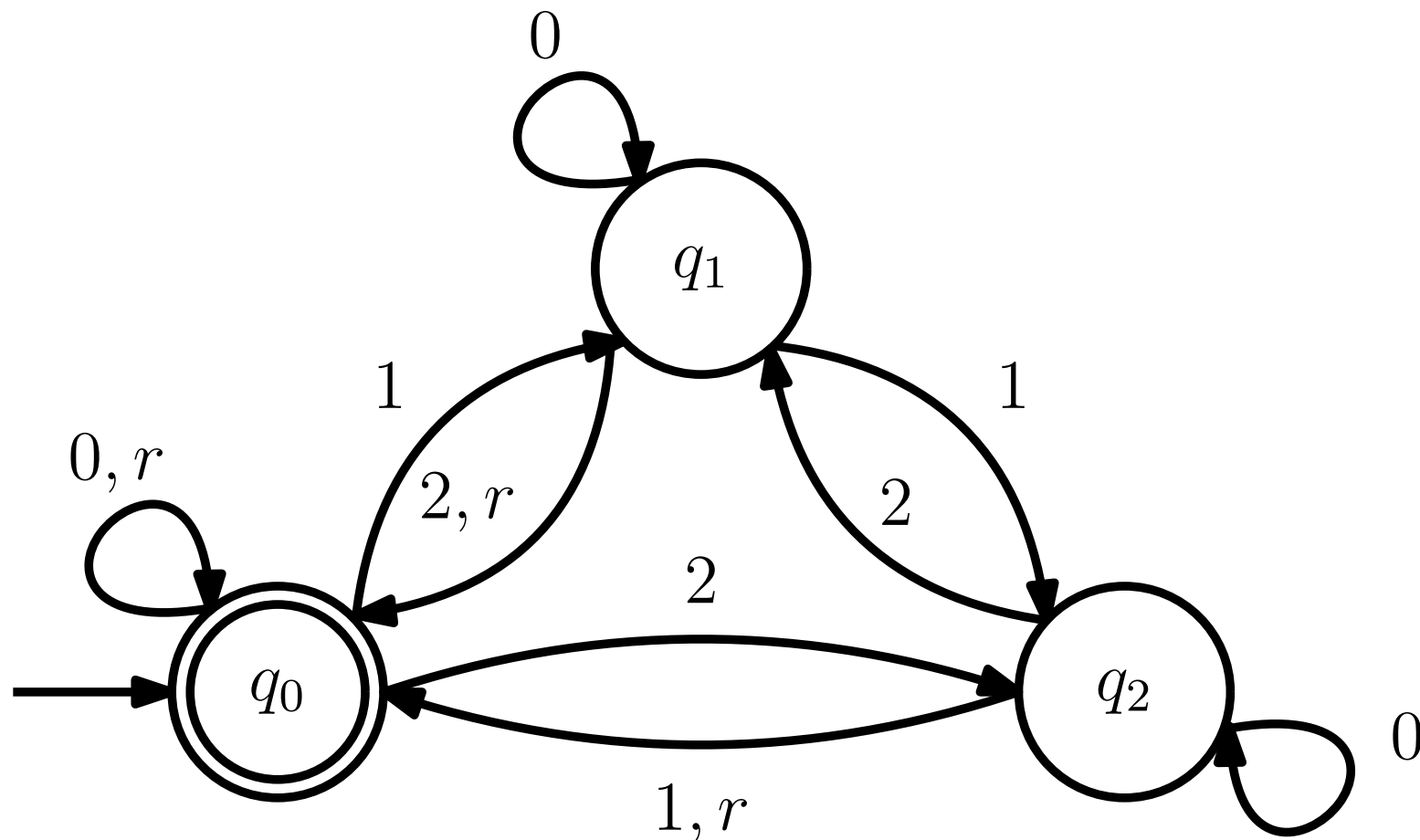
Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un autómata.

El *lenguaje aceptado por M* es

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M(w) = 1\}.$$

• Ej:



Lenguajes regulares

- Obj curso $\rightarrow$ clasificar problemas.

Input: $w \in \Sigma^*$

Output: 1 si $w \in L$, 0 si $w \notin L$.

Def.

Sea Σ un alfabeto.

Un lenguaje $L \subseteq \Sigma^*$ es *regular* si existe un autómata M tal que $\mathcal{L}(M) = L$.

- Probar que L es regular $\rightarrow$ diseñar un autómata que “resuelve” L .

Diseñando autómatas

- ¿Cómo programo “sin memoria”?
 - Puedo “guardar información” en los estados
 - Diseñar estados primero
 - Ver evolución de la función de transición
- **Ej:** Demostremos que los siguientes son regulares.
 - $L_1 = \{w \in \{0, 1\}^* \mid w \text{ tiene una cantidad par de 1's}\}$
 - $L_2 = \{w \in \{0, 1\}^* \mid w \text{ termina con } 00\}$
 - $L_3 = \{w \in \{\text{b, e, r}\}^* \mid w \text{ tiene como antepenúltimo carácter a e}\}$
 - $L_4 = \{w \in \{\text{b, e, r}\}^* \mid w \text{ contiene beer como substring}\}$

Operaciones de lenguajes

- ¿Qué ocurre al operar lenguajes regulares?

Teo.

Sea Σ un alfabeto y $L_1, L_2 \subseteq \Sigma^*$ dos lenguajes regulares.

Los siguientes son lenguajes regulares

- L_1^c
- $L_1 \cup L_2$
- $L_1 \cap L_2$

- Lenguajes regulares son cerrados bajo complemento, unión, intersección.
- **Dem:**
- **Ej:** Los siguientes son regulares.
 - $L_1 = \{w \in \{0, 1\}^* \mid w \text{ tiene una cantidad impar de 1's}\}$
 - $L_2 = \{w \in \{0, 1\}^* \mid w \text{ termina con } 00 \text{ y tiene par 1's}\}$
 - $L_3 = \{w \in \{0, 1\}^* \mid w \text{ no termina con } 00 \text{ o tiene impar 1's}\}$

Dudas/Preguntas

Problema del día

2. Para cada $n \geq 1$, considere el lenguaje sobre $\Sigma = \{0, 1\}$ definido como

$$L_n = \{w : |w| \geq n \text{ y el } n\text{-ésimo símbolo de } w, \text{ desde la derecha, es } 1\}.$$

El objetivo de esta pregunta es determinar la cantidad óptima de estados que necesita un AFD para decidir L_n .

1. Construya un AFD con 8 estados que acepte L_3 . ¿Qué información guarda su automata en los estados?
2. Construya un AFD con 2^n estados que acepte L_n . ¿Qué información guarda su automata en los estados?
3. Sea M cualquier AFD y $x, y \in \Sigma^*$. Demuestre que, si M queda en el mismo estado después de leer x y después de leer y , entonces $M(xz) = M(yz)$ para todo $z \in \Sigma^*$.
4. Suponga ahora que M reconoce L_n . Demuestre que dos strings distintos $x, y \in \{0, 1\}^n$ deben llevar a M a estados distintos. Concluya que cualquier AFD que reconozca L_n debe tener al menos 2^n estados.
5. Discuta que consecuencias tiene el inciso anterior sobre su automata del inciso b). ¿Qué dice esto sobre la cantidad de bits de memoria que se necesitan para aceptar L_n ?

Wrap-Up

- **Hoy:**

- Autómatas finitos deterministas
- Computar
- Lenguajes regulares
- Cerradura

- **Próx. clase:**

- Autómatas finitos **no** deterministas