No determinismo

UNIVERSIDAD DE CHILE

Arturo Merino

Recuerdo / Plan

Recuerdo/Plan

- **Recuerdo:**

Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos $\rightarrow A = (Q, \Sigma, \delta, q_0, F)$.

Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos $\rightarrow A = (Q, \Sigma, \delta, q_0, F)$.
  - Lenguajes regulares

Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos $\rightarrow A = (Q, \Sigma, \delta, q_0, F)$.
  - Lenguajes regulares
  - Cerrados bajo $\cap$, $\cup$ y complemento.

Recuerdo/Plan

- **Recuerdo:**
  - Autómatas finitos $\rightarrow A = (Q, \Sigma, \delta, q_0, F)$.
  - Lenguajes regulares
  - Cerrados bajo $\cap$, $\cup$ y complemento.

- **Plan:**

Recuerdo/Plan

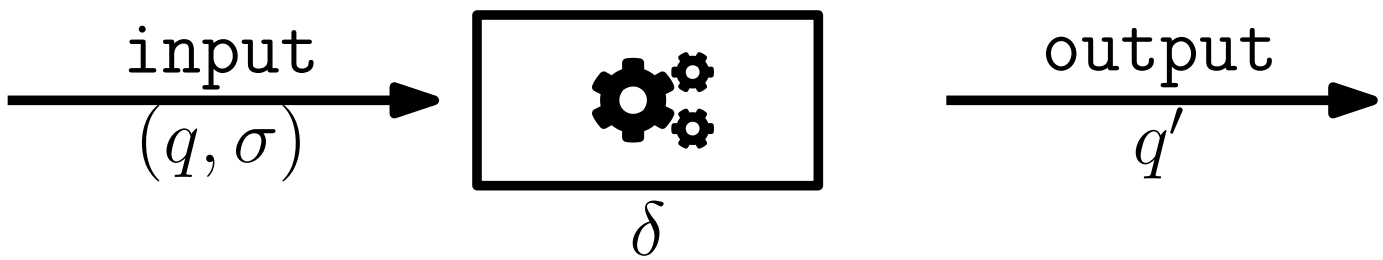
- **Recuerdo:**
  - Autómatas finitos $\rightarrow A = (Q, \Sigma, \delta, q_0, F)$.
  - Lenguajes regulares
  - Cerrados bajo $\cap$, $\cup$ y complemento.

- **Plan:**
  - Autómatas finitos **no deterministas**

Autómatas finitos no deterministas

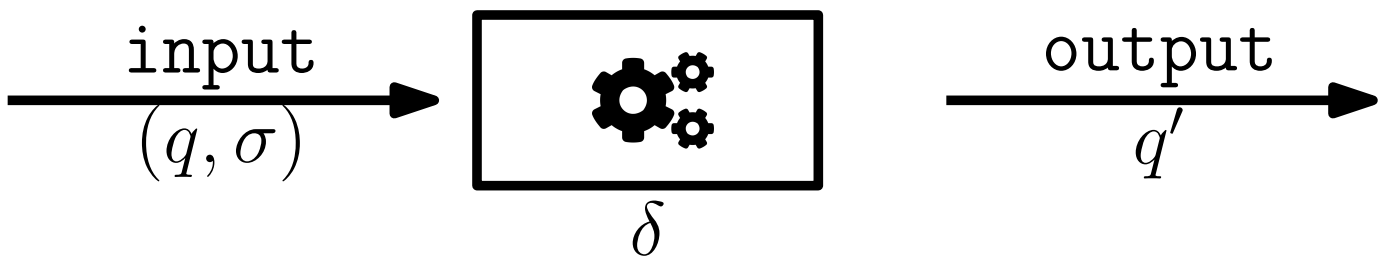
Determinismo y alternativas

- **Determinismo:** dado input conozco output inequívocamente.



Determinismo y alternativas

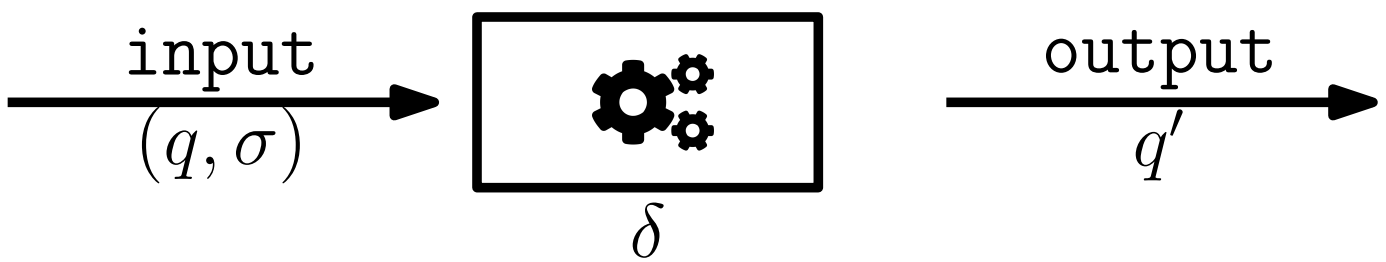
- **Determinismo:** dado input conozco output inequívocamente.



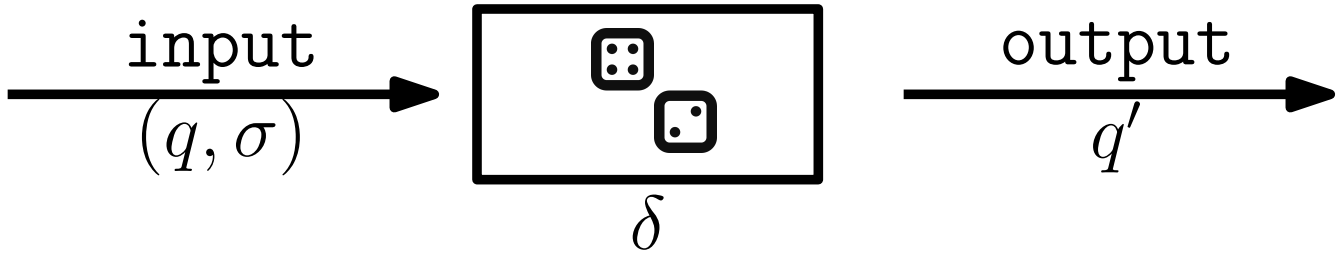
- **Alternativas:**

Determinismo y alternativas

- **Determinismo:** dado input conozco output inequívocamente.

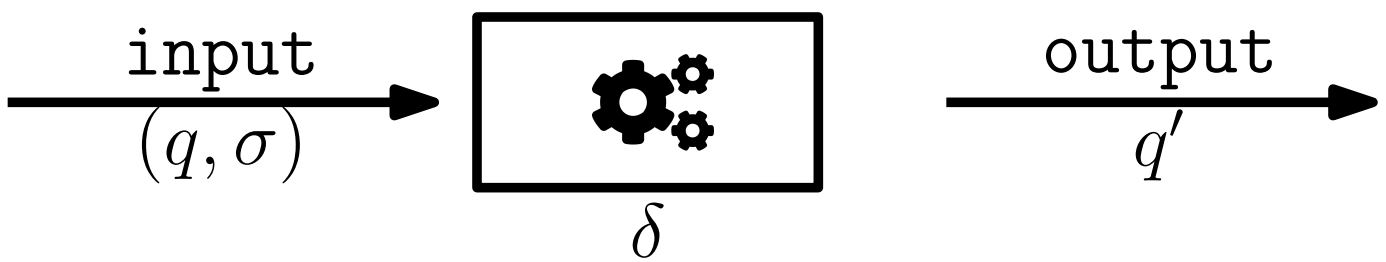


- **Alternativas:**
  - Aleatoriedad

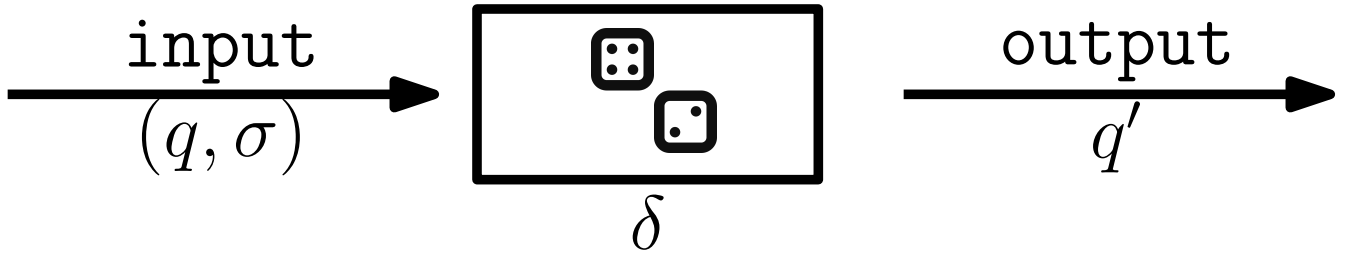


Determinismo y alternativas

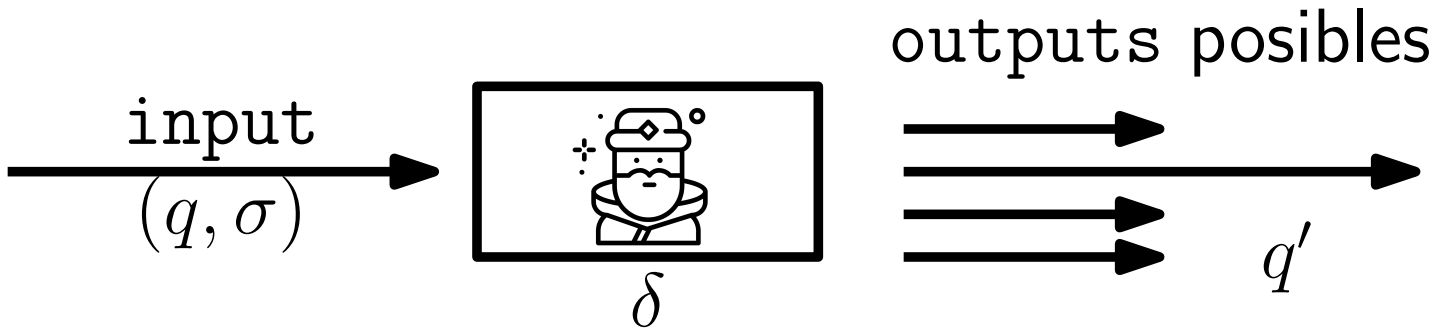
- **Determinismo:** dado input conozco output inequívocamente.



- **Alternativas:**
  - Aleatoriedad

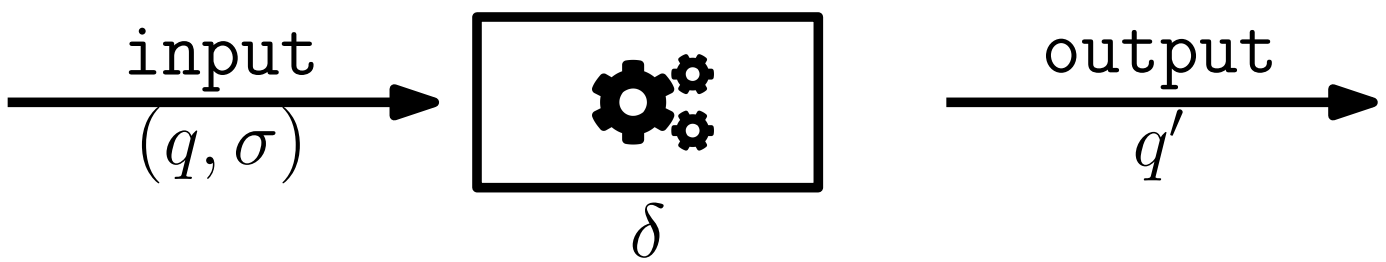


  - No determinismo

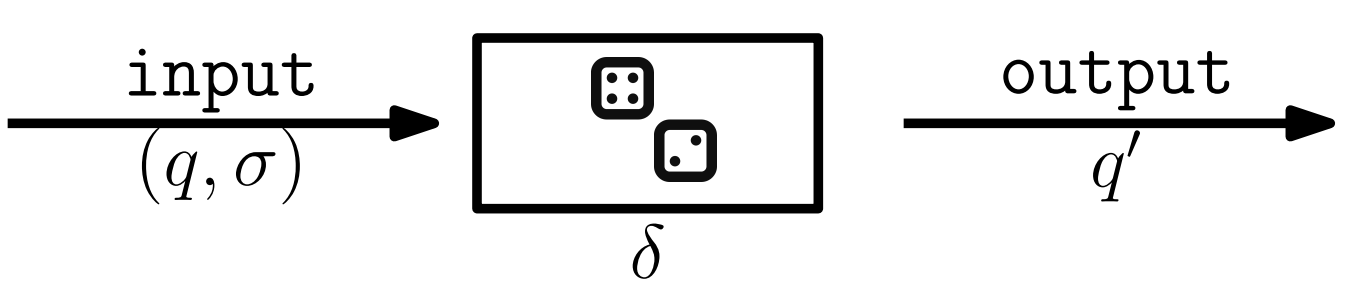


Determinismo y alternativas

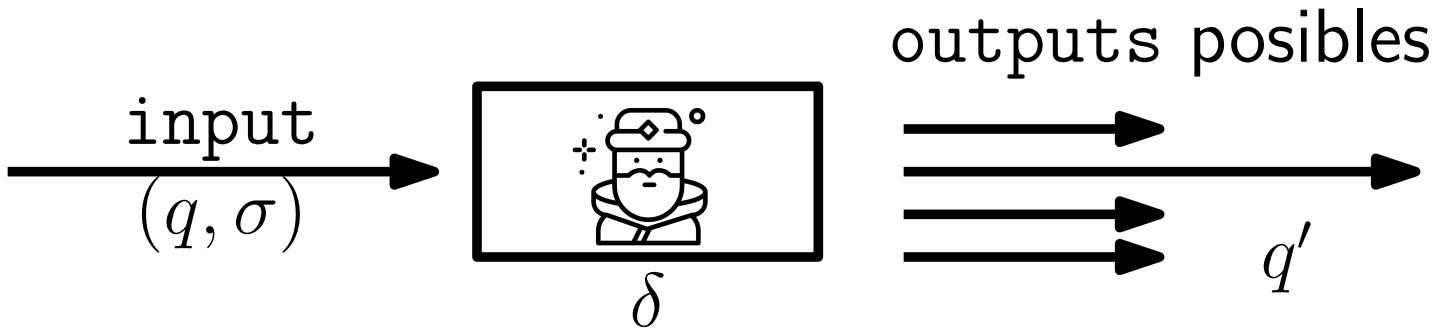
- **Determinismo:** dado input conozco output inequívocamente.



- **Alternativas:**
  - Aleatoriedad



  - No determinismo



  - **Clave:** chequear v/s computar, entender modelos, etc...

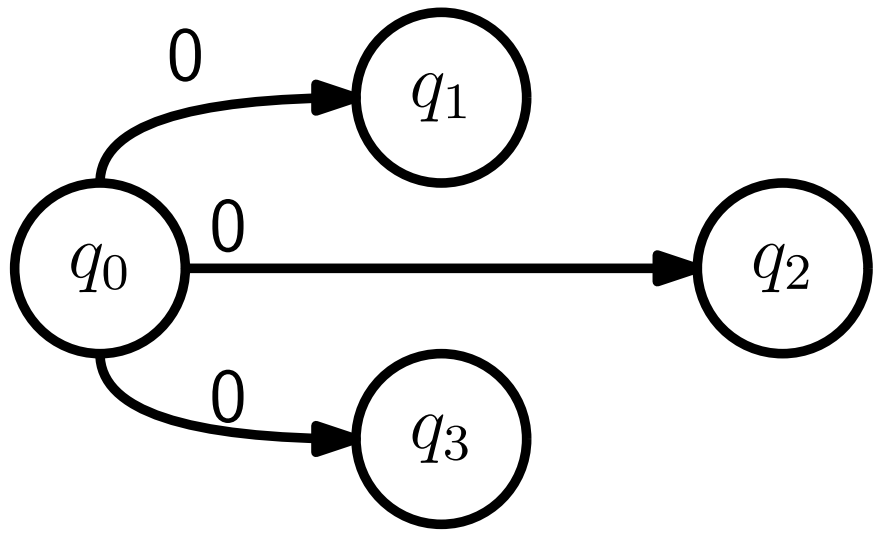
No determinismo

- ¿Varios outputs posibles?

No determinismo

- ¿Varios outputs posibles?

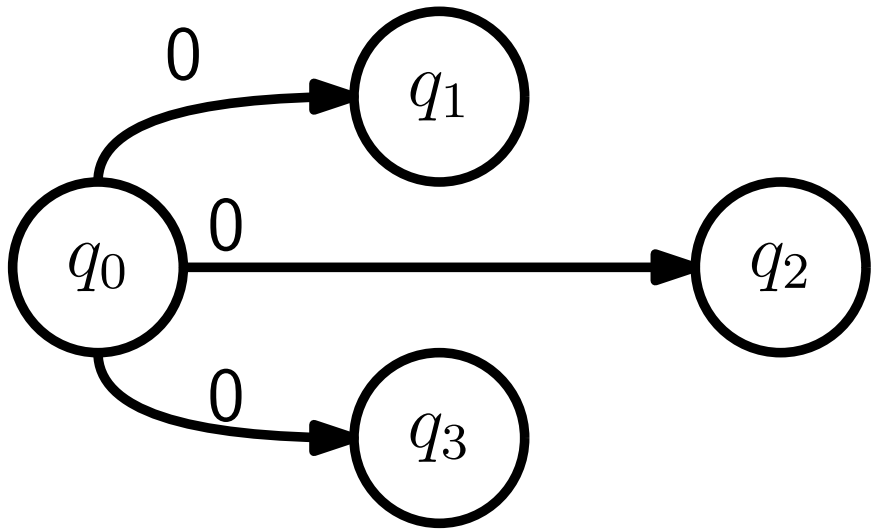
- Mismo carácter, varios estados



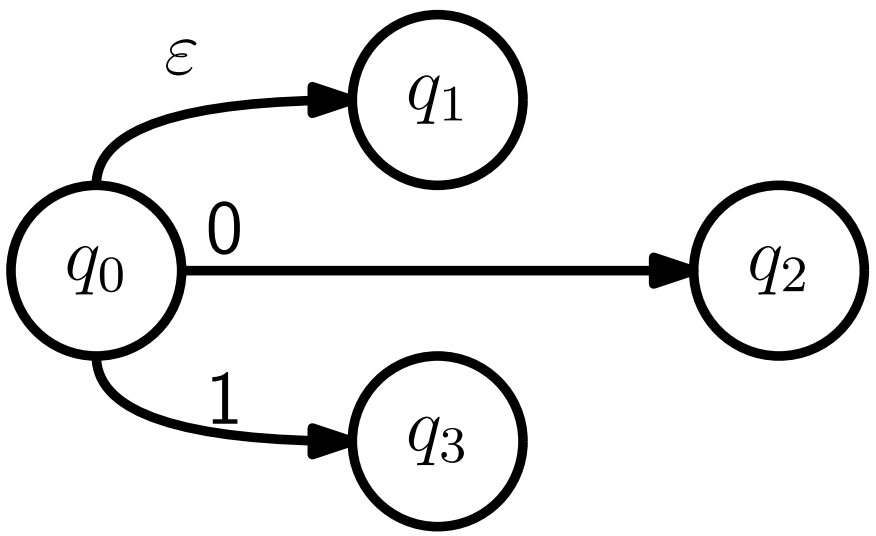
No determinismo

- ¿Varios outputs posibles?

- Mismo carácter, varios estados

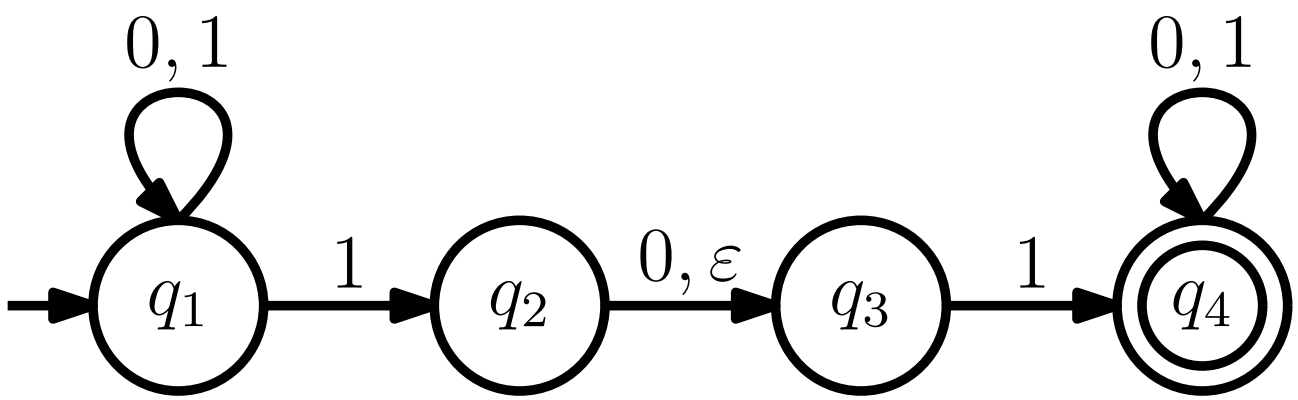


- Mover sin consumir string



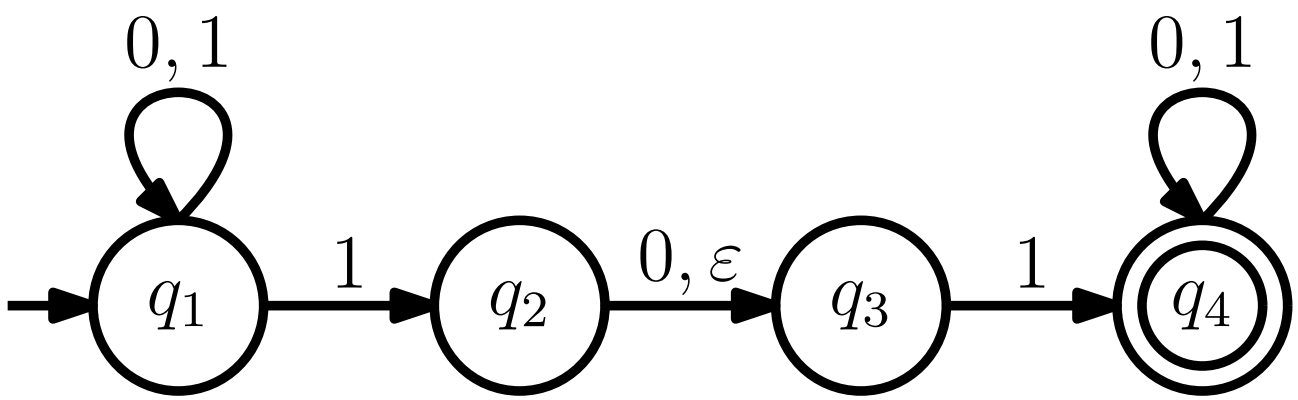
Computación

- ¿Cómo computa?



Computación

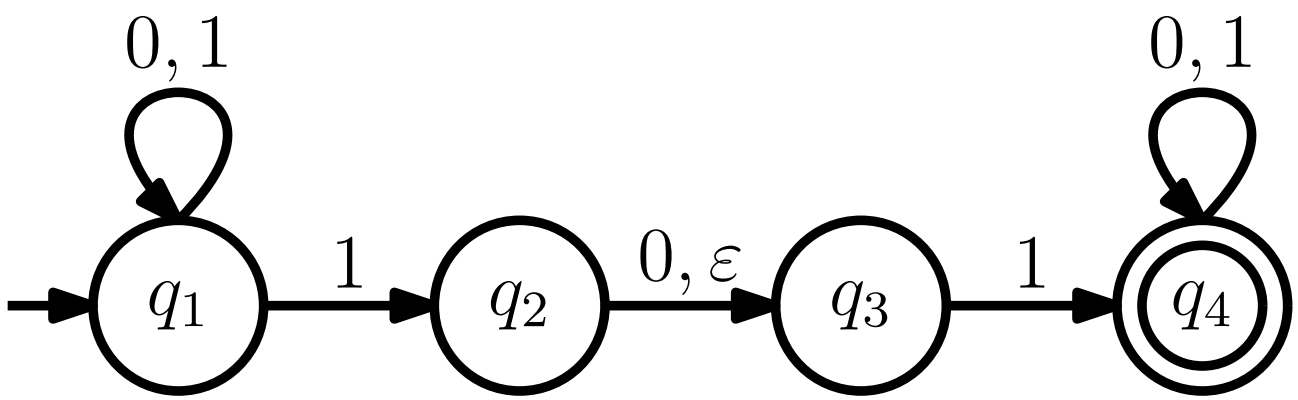
- ¿Cómo computa?



- **Nuevo:** Transiciones no definidas

Computación

- ¿Cómo computa?

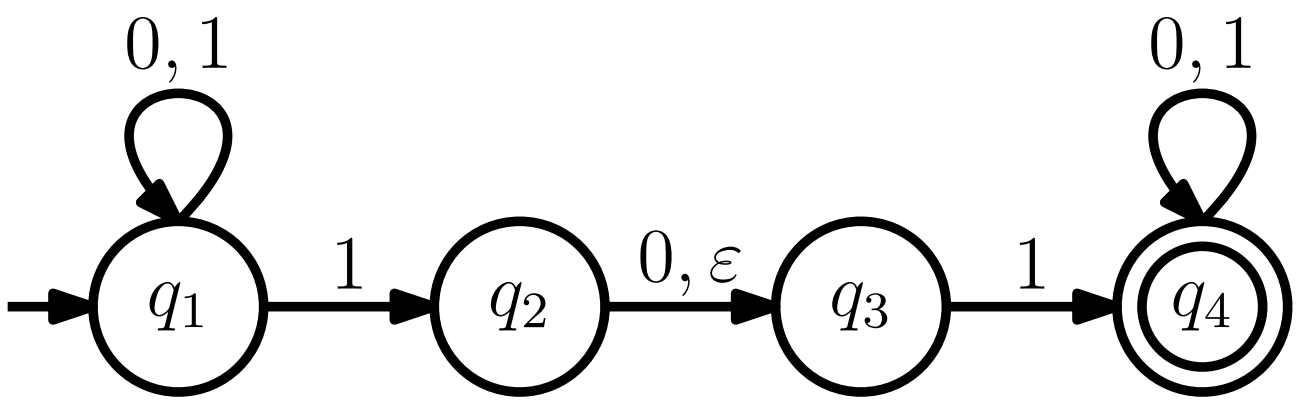


- **Nuevo:** Transiciones no definidas

- Varias interpretaciones

Computación

- ¿Cómo computa?

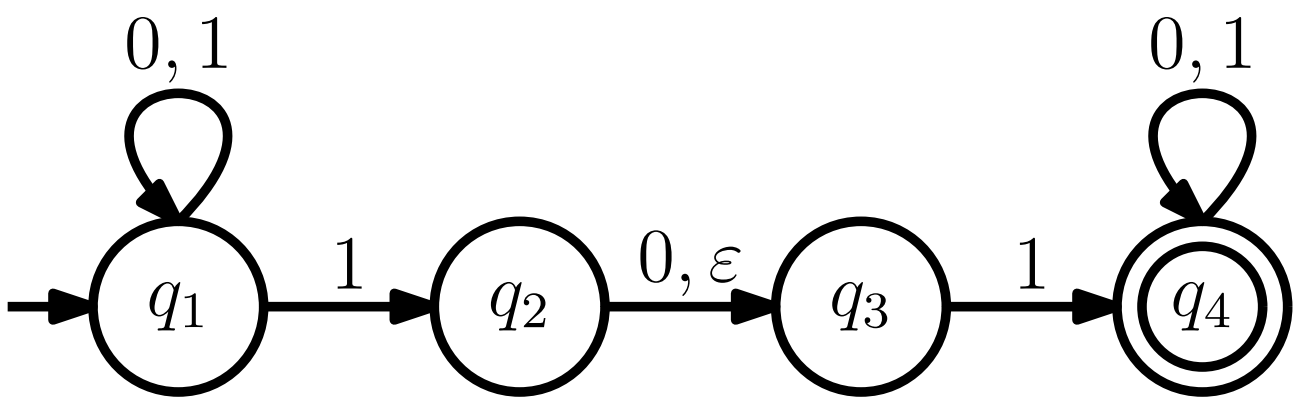


- **Nuevo:** Transiciones no definidas

- Varias interpretaciones
  - Paralelismo

Computación

- ¿Cómo computa?

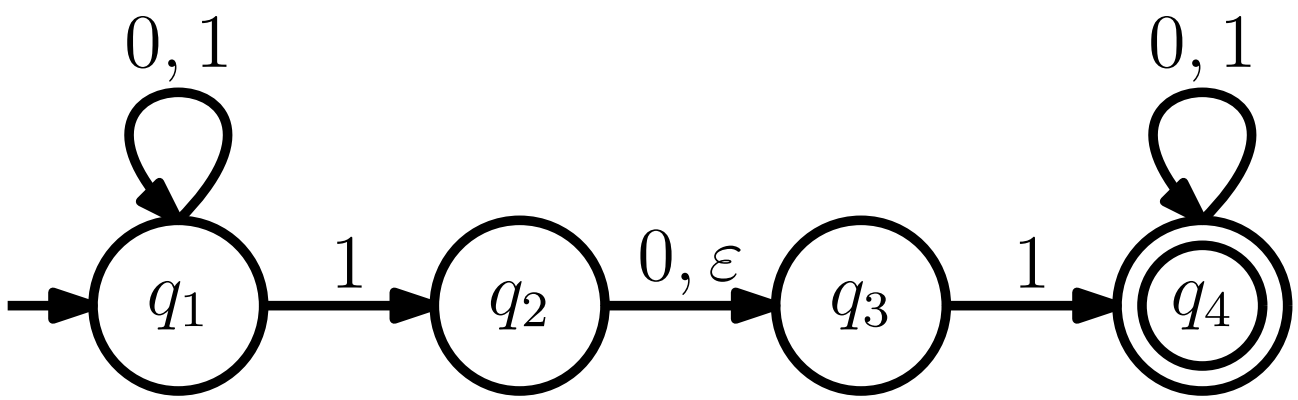


- **Nuevo:** Transiciones no definidas

- Varias interpretaciones
  - Paralelismo
  - Árbol de cómputo

Computación

- ¿Cómo computa?

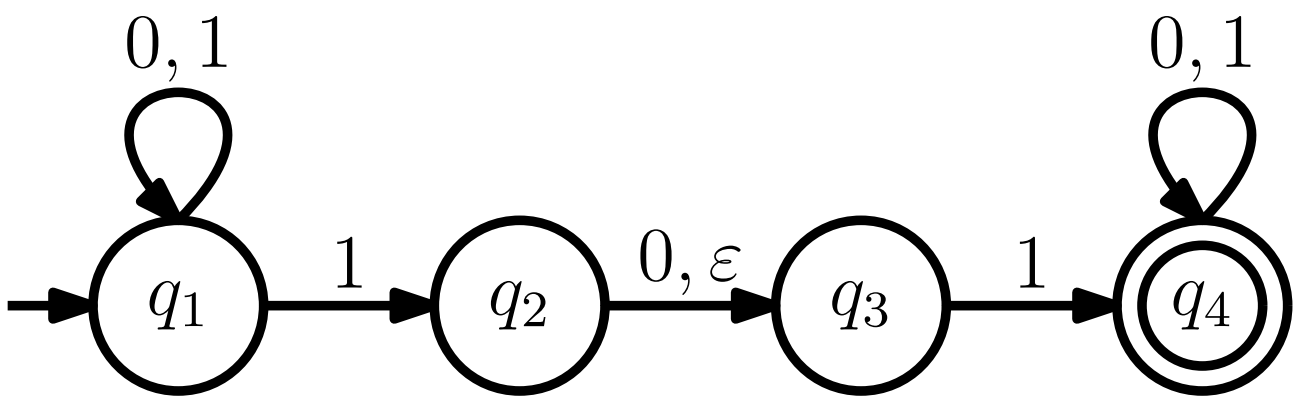


- **Nuevo:** Transiciones no definidas

- Varias interpretaciones
  - Paralelismo
  - Árbol de cómputo
  - Adivinar

Computación

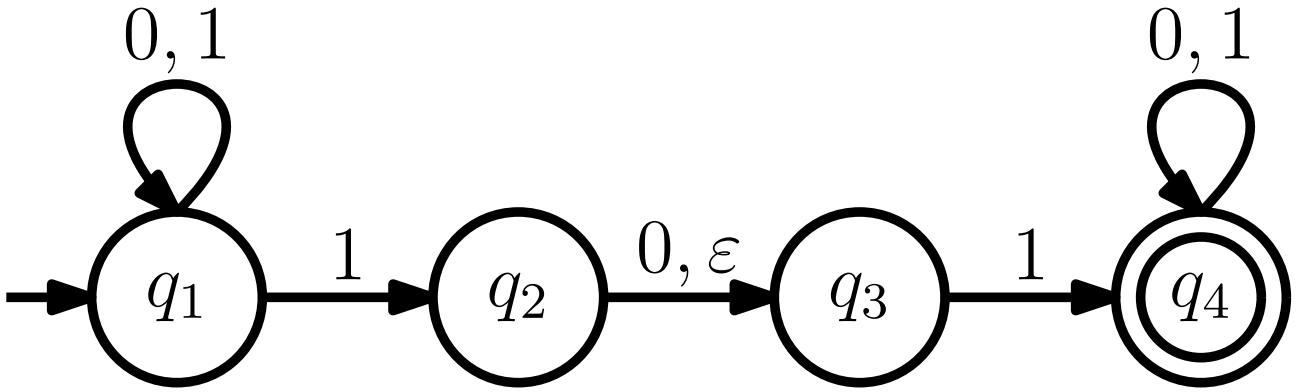
- ¿Cómo computa?

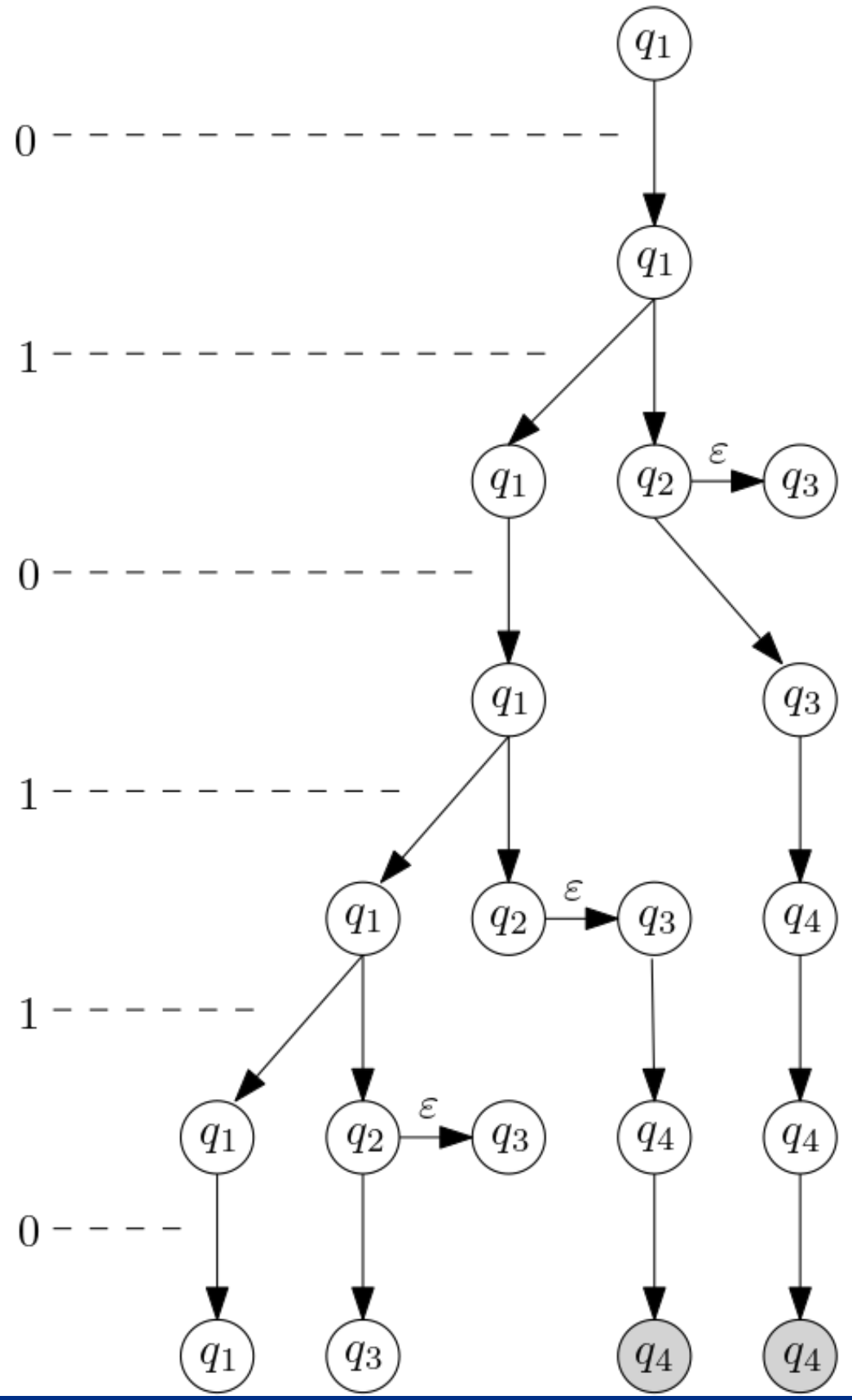


- **Nuevo:** Transiciones no definidas

- Varias interpretaciones
  - Paralelismo
  - Árbol de cómputo
  - Adivinar
  - Chequear

Árbol de cómputo





Definición formal

- ¿Qué cambió respecto al AFD?

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, F)$, donde

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* 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*

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* 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*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ es la *función de transición*.

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* 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*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ es la *función de transición*.
- $q_0 \in Q$ es el *estado inicial*, y

Definición formal

- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* 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*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ es la *función de transición*.
- $q_0 \in Q$ es el *estado inicial*, y
- $F \subseteq Q$ son los *estados de aceptación (o finales)*.

Definición formal

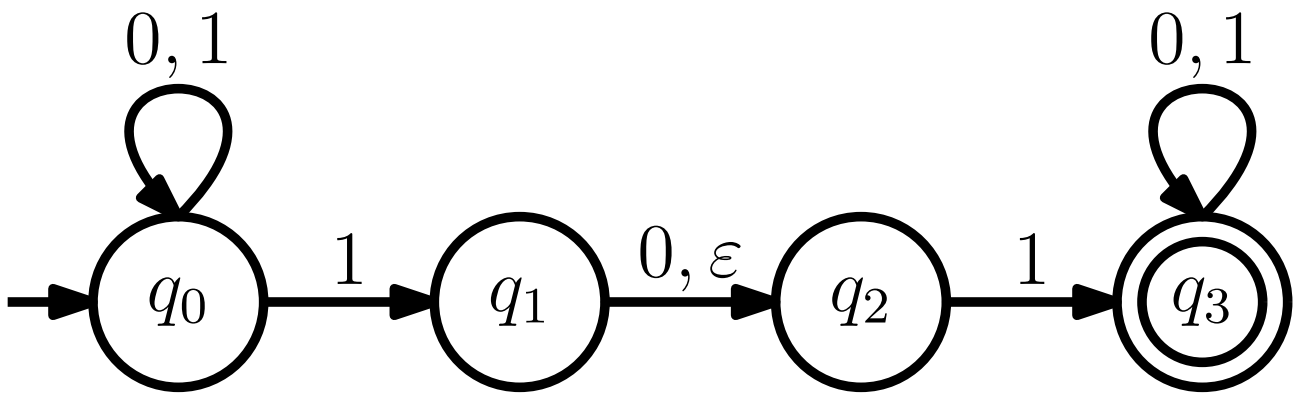
- ¿Qué cambió respecto al AFD?

Def.

Un *autómata finito no determinista (AFND)* 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*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ 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:**



Computación

- ¿Qué cambió en cómo computa?

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

- $r_0 = q_0$ (partimos en el estado inicial)

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

- $r_0 = q_0$ (partimos en el estado inicial)
- $r_i \in \delta(r_{i-1}, x_i)$ para todo $i \in \{1, \dots, m\}$ (transiciones válidas)

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

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

Computación

- ¿Qué cambió en cómo computa?

Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

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

- La secuencia **certifica**.

Computación

- ¿Qué cambió en cómo computa?

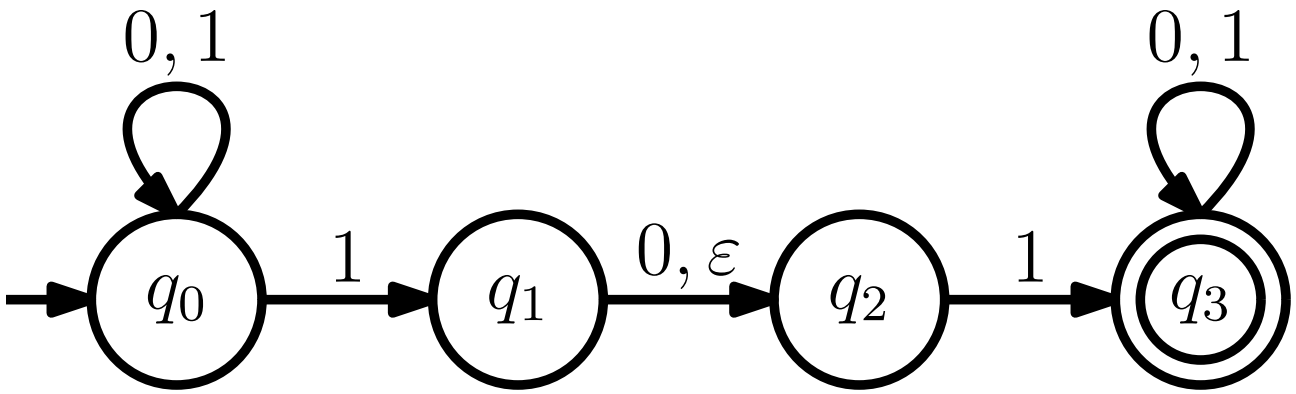
Def.

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

Diremos que M *acepta* w si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$ y existen estados $r_0, \dots, r_m \in Q$ tal que

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

- La secuencia **certifica**.
- **Ej:**



$w = 010110$

Entendiendo autómatas

Def.

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

Entendiendo autómatas

Def.

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

El *lenguaje aceptado por* M es

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

Entendiendo autómatas

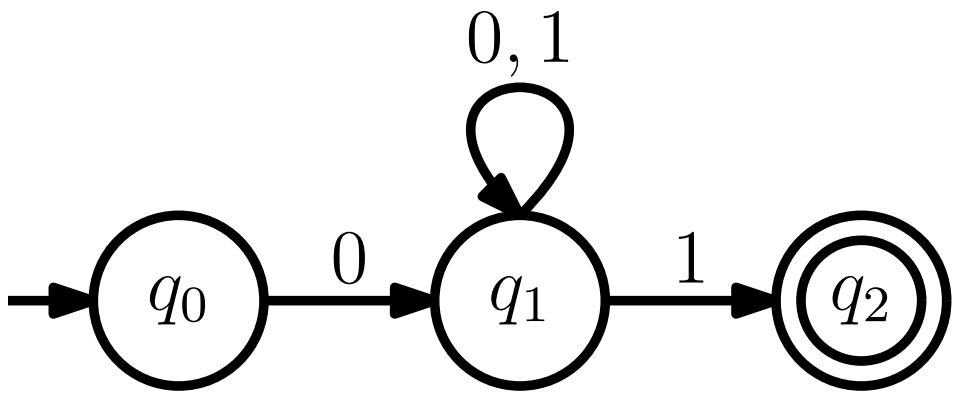
Def.

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

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 AFND.

Entendiendo autómatas

Def.

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

El *lenguaje aceptado por* M es

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

Entendiendo autómatas

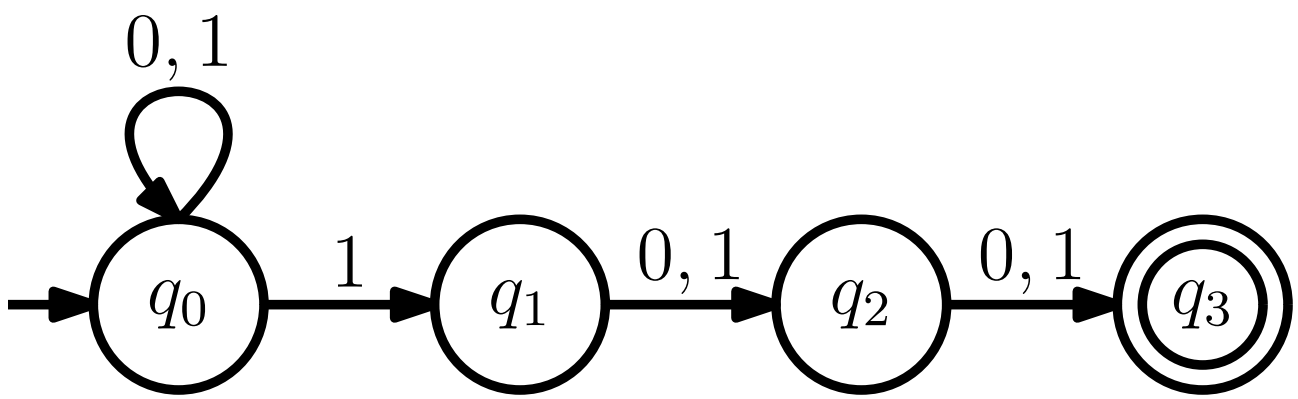
Def.

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

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 AFND.

Entendiendo autómatas

Def.

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

El *lenguaje aceptado por* M es

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

Entendiendo autómatas

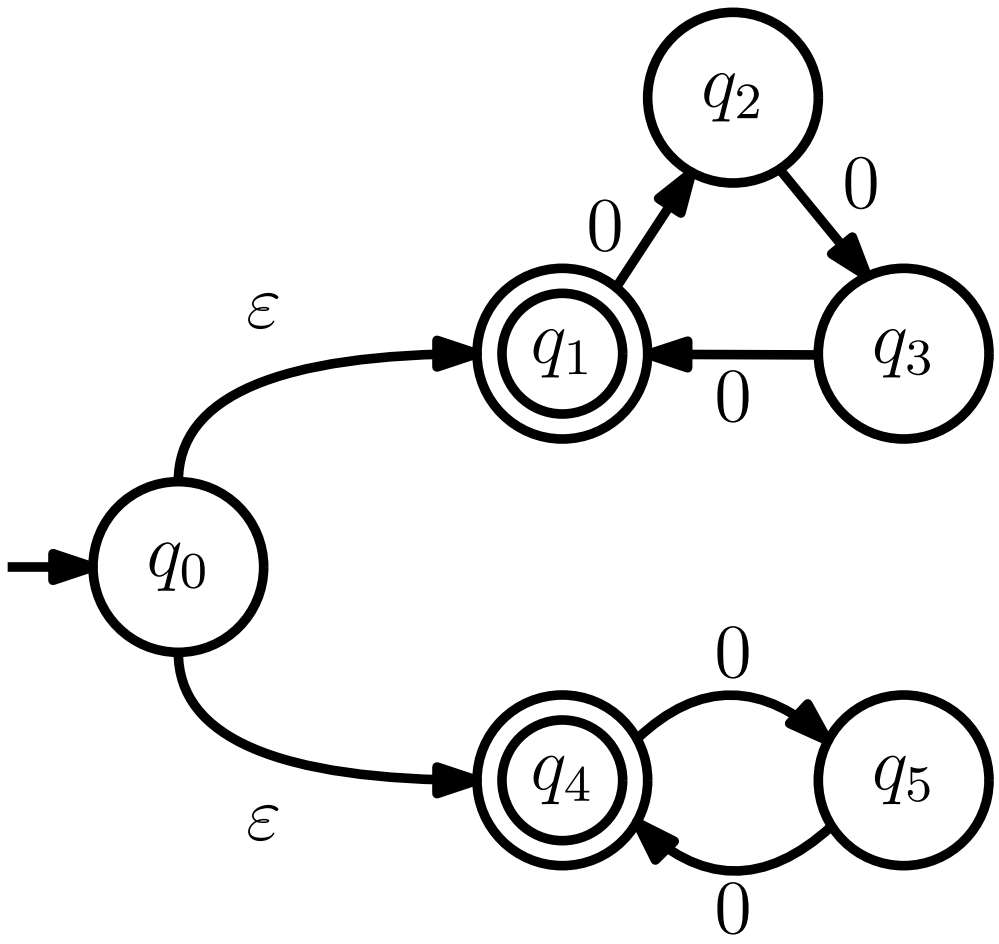
Def.

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

El *lenguaje aceptado por* M es

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

- **Ej:**



Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

- **Ej:** Diseñe AFND que acepten los siguientes lenguajes

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

- **Ej:** Diseñe AFND que acepten los siguientes lenguajes
  - $L_1 = \{w \in \{0,1\}^* \mid w$ termina con 00$\}$

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

- **Ej:** Diseñe AFND que acepten los siguientes lenguajes
  - $L_1 = \{w \in \{0,1\}^* \mid w \text{ termina con } 00\}$
  - $L_2 = \{w \in \{\texttt{b}, \texttt{e}, \texttt{r}\}^* \mid w \text{ contiene } \texttt{beer} \text{ como substring}\}$

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

- **Ej:** Diseñe AFND que acepten los siguientes lenguajes
  - $L_1 = \{w \in \{0,1\}^* \mid w$ termina con 00$\}$
  - $L_2 = \{w \in \{\texttt{b}, \texttt{e}, \texttt{r}\}^* \mid w$ contiene `beer` como substring$\}$
  - $L_3 = \{w \in \{0\}^* \mid w = 0^a 0^b$con $a = 1 \mod 3$ y $b = 0 \mod 2\}$

Diseñando autómatas no deterministas

- ¿Cómo programo "sin memoria" + no determinismo?
  - ¿Qué información es importante adivinar?
  - Diseñar estados
  - Ver evolución de la función de transición

- **Ej:** Diseñe AFND que acepten los siguientes lenguajes
  - $L_1 = \{w \in \{0,1\}^* \mid w$ termina con 00$\}$
  - $L_2 = \{w \in \{\texttt{b}, \texttt{e}, \texttt{r}\}^* \mid w$ contiene `beer` como substring$\}$
  - $L_3 = \{w \in \{0\}^* \mid w = 0^a 0^b$con $a = 1 \mod 3$ y $b = 0 \mod 2\}$
  - $L_4 = \{w \in \{0,1\}^* \mid w$ tiene par 0's, o exactamente dos 1's$\}$

Expresividad

- ¿Qué poder de cómputo tienen los AFND?

Expresividad

- ¿Qué poder de cómputo tienen los AFND?
  - **Mismo poder** que un AFD.

Expresividad

- ¿Qué poder de cómputo tienen los AFND?
  - **Mismo poder** que un AFD.

Def.

Diremos que dos máquinas A y B son equivalentes si $\mathcal{L}(A) = \mathcal{L}(B)$.

Expresividad

- ¿Qué poder de cómputo tienen los AFND?
  - **Mismo poder** que un AFD.

Def.

Diremos que dos máquinas A y B son equivalentes si $\mathcal{L}(A) = \mathcal{L}(B)$.

Teo.

Todo AFND tiene un AFD equivalente.

Expresividad

- ¿Qué poder de cómputo tienen los AFND?
  - **Mismo poder** que un AFD.

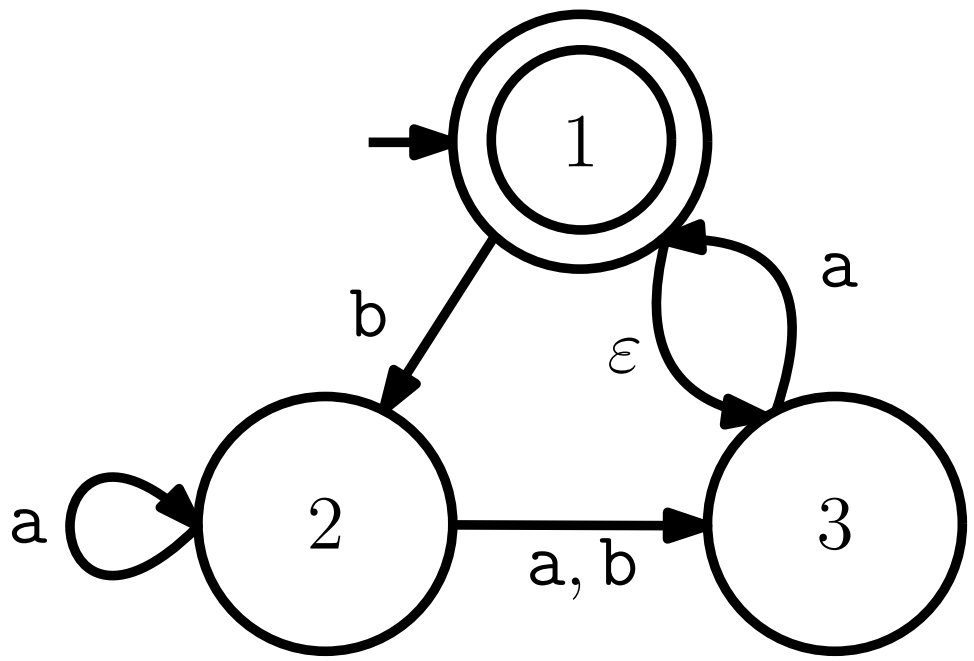
Def.

Diremos que dos máquinas A y B son equivalentes si $\mathcal{L}(A) = \mathcal{L}(B)$.

Teo.

Todo AFND tiene un AFD equivalente.

- **Ej:**



De AFND a AFD

Teo.

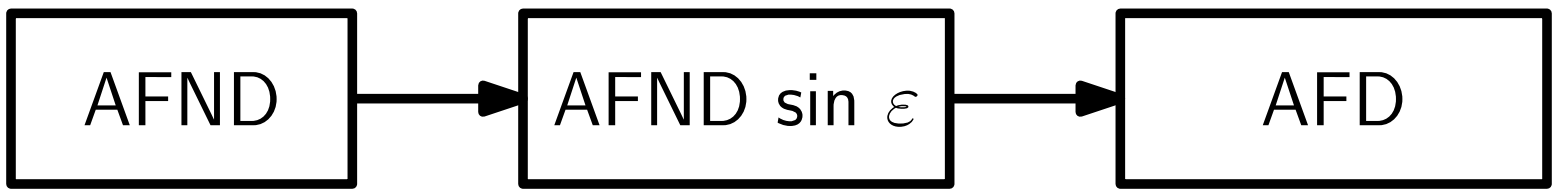
Todo AFND tiene un AFD equivalente.

De AFND a AFD

Teo.

Todo AFND tiene un AFD equivalente.

- **Dem:** (próxima clase.)



De AFND a AFD

Teo.

Todo AFND tiene un AFD equivalente.

- **Dem:** (próxima clase.)

Cor.

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

De AFND a AFD

Teo.

Todo AFND tiene un AFD equivalente.

- **Dem:** (próxima clase.)

Cor.

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

L es regular si y sólo si existe un AFND M tal que $\mathcal{L}(M) = L$.

De AFND a AFD

Teo.

Todo AFND tiene un AFD equivalente.

- **Dem:** (próxima clase.)

Cor.

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

L es regular si y sólo si existe un AFND M tal que $\mathcal{L}(M) = L$.

- Entender AFD vía AFND.

De AFND a AFD

Teo.

Todo AFND tiene un AFD equivalente.

- **Dem:** (próxima clase.)

Cor.

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

L es regular si y sólo si existe un AFND M tal que $\mathcal{L}(M) = L$.

- Entender AFD vía AFND.
- Probar que lenguaje es regular $\leftarrow$ diseñar AFND

Dudas/Preguntas

Problema del día

2. Dada $w \in \{0,1,?\}^*$, diremos que x es una *completación* de w si x se obtiene de w reemplazando cada símbolo ? por un 0 o un 1.

   a) Dado un lenguaje $L \subseteq \{0,1\}^*$, definimos

   $$\text{COMP}_\exists(L) = \{w \in \{0,1,?\}^* : \text{existe una completación de } w \text{ en } L\}.$$

   Sea

   $$L_0 = \{w \in \{0,1\}^* : w \text{ contiene 01 como substring}\}.$$

   Muestre que $\text{COMP}_\exists(L_0)$ es regular.

   b) Muestre que si L es regular, entonces $\text{COMP}_\exists(L)$ también es regular.

   c) Definimos ahora

   $$\text{COMP}_\forall(L) = \{w \in \{0,1,?\}^* : \text{todas las completaciones de } w \text{ están en } L\}.$$

   Muestre que si L es regular, entonces $\text{COMP}_\forall(L)$ también es regular.

   d) Definimos finalmente

   $$\text{COMP}_{\text{mix}}(L) = \{w \in \{0,1,?\}^* : \text{existe una completación de } w \text{ en } L \text{ y una en } L^c\}.$$

   Muestre que si L es regular, entonces $\text{COMP}_{\text{mix}}(L)$ también es regular.

Wrap-Up

- **Hoy:**

Wrap-Up

- **Hoy:**
  - Autómatas finitos **no** deterministas

Wrap-Up

- **Hoy:**
  - Autómatas finitos **no** deterministas
  - Computar

Wrap-Up

- **Hoy:**
  - Autómatas finitos **no** deterministas
  - Computar
  - Equivalencia AFD, AFND

Wrap-Up

- **Hoy:**
  - Autómatas finitos **no** deterministas
  - Computar
  - Equivalencia AFD, AFND

- **Próx. clase:**

Wrap-Up

- **Hoy:**
  - Autómatas finitos **no** deterministas
  - Computar
  - Equivalencia AFD, AFND

- **Próx. clase:**
  - Operaciones y expresiones regulares: $\cup, \cdot, *$.