Autómatas de pila

UNIVERSIDAD DE CHILE

Arturo Merino

Recuerdo / Plan

Recuerdo/Plan

- **Recuerdo:**

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares
  - Propiedades de cerradura

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares
  - Propiedades de cerradura

- **Plan:**

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares
  - Propiedades de cerradura

- **Plan:**
  - Autómatas de pila

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares
  - Propiedades de cerradura

- **Plan:**
  - Autómatas de pila
  - ¿Cómo computan?

Recuerdo/Plan

- **Recuerdo:**
  - Gramáticas libres de contexto
  - Lenguajes libres de contexto
  - Relación con los lenguajes regulares
  - Propiedades de cerradura

- **Plan:**
  - Autómatas de pila
  - ¿Cómo computan?
  - Relación con los lenguajes regulares

Autómata de pila

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`
- ¿Reconocer $0^n 1^n$?

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`
- ¿Reconocer 0^n1^n?
- Modelo más limpio: un string como pila, con acceso por la derecha.

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`
- ¿Reconocer $0^n 1^n$?
- Modelo más limpio: un string como pila, con acceso por la derecha.
  - `push(x)`: $\varepsilon \rightarrow x$.

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`
- ¿Reconocer $0^n 1^n$?
- Modelo más limpio: un string como pila, con acceso por la derecha.
  - `push(x)`: $\varepsilon \rightarrow x$.
  - `pop()==x`: $x \rightarrow \varepsilon$

Autómatas de pila

- AF $\leftarrow$ un computador sin memoria.
- **Extendamos** dichos modelos.
  - Acceso a alguna estructura de datos $\rightarrow$ pila
  - Demos a un AFND acceso a una pila
- Pila: estructura LIFO con las operaciones:
  - `push(x)`
  - `pop()`
  - `peek()? replace(a,b)? is_empty()?`
- ¿Reconocer 0^n1^n?
- Modelo más limpio: un string como pila, con acceso por la derecha.
  - `push(x)`: $\varepsilon \rightarrow x$.
  - `pop()==x`: $x \rightarrow \varepsilon$
  - `pop()==x, push(y)`: $x \rightarrow y$.

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,
- Γ es un conjunto finito llamado el *alfabeto de pila*,

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,
- Γ es un conjunto finito llamado el *alfabeto de pila*,
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to 2^{Q \times (\Gamma \cup \{\varepsilon\})}$ es la *f. de transición*,

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,
- Γ es un conjunto finito llamado el *alfabeto de pila*,
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to 2^{Q \times (\Gamma \cup \{\varepsilon\})}$ es la *f. de transición*,
- $q_0 \in Q$ es el *estado inicial*, y

Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,
- Γ es un conjunto finito llamado el *alfabeto de pila*,
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to 2^{Q \times (\Gamma \cup \{\varepsilon\})}$ es la *f. de transición*,
- $q_0 \in Q$ es el *estado inicial*, y
- $F \subseteq Q$ son los *estados de aceptación (o finales)*.

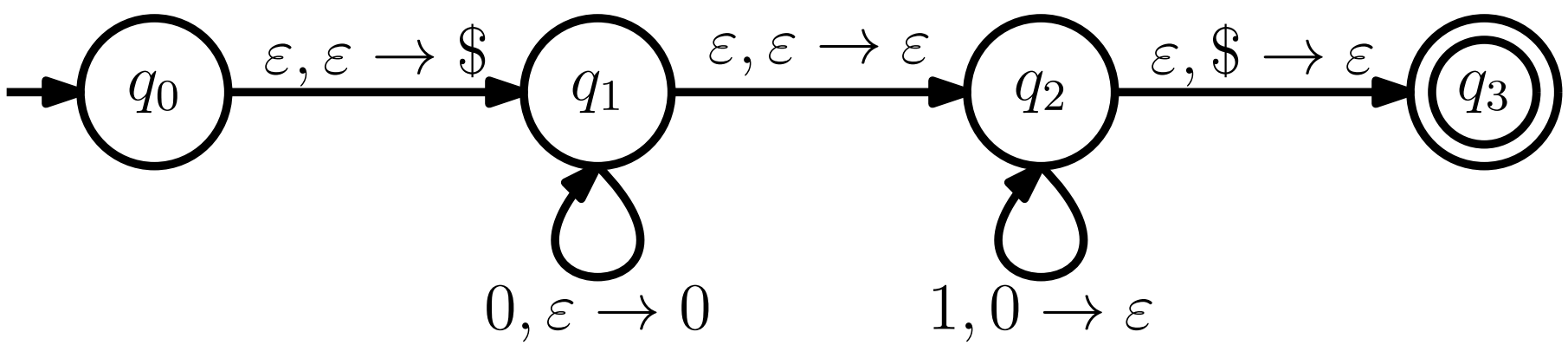
Autómata de pila

Def.

Un *autómata de pila (AP)* es una 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, donde

- Q es un conjunto finito de *estados*,
- Σ es un conjunto finito llamado *alfabeto (de entrada)*,
- Γ es un conjunto finito llamado el *alfabeto de pila*,
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to 2^{Q \times (\Gamma \cup \{\varepsilon\})}$ es la *f. de transición*,
- $q_0 \in Q$ es el *estado inicial*, y
- $F \subseteq Q$ son los *estados de aceptación (o finales)*.

- **Ej:**



Computación

Def.

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

Computación

Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

Computación

Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

- $r_0 = q_0$ y $s_0 = \varepsilon$ (partimos en la config. inicial)

Computación

Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

- $r_0 = q_0$ y $s_0 = \varepsilon$ (partimos en la config. inicial)
- para todo $i \in \{1, \dots, m\}$, tenemos que $(r_i, b) \in \delta(r_{i-1}, x_i, a)$ y $s_{i-1} = ta$ y $s_i = tb$ para algún $t \in \Gamma^*$ (transiciones válidas)

Computación

Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

- $r_0 = q_0$ y $s_0 = \varepsilon$ (partimos en la config. inicial)
- para todo $i \in \{1, \dots, m\}$, tenemos que $(r_i, b) \in \delta(r_{i-1}, x_i, a)$ y $s_{i-1} = ta$ y $s_i = tb$ para algún $t \in \Gamma^*$ (transiciones válidas)
- $r_m \in F$ (terminamos en aceptación)

Computación

Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

- $r_0 = q_0$ y $s_0 = \varepsilon$ (partimos en la config. inicial)
- para todo $i \in \{1, \dots, m\}$, tenemos que $(r_i, b) \in \delta(r_{i-1}, x_i, a)$ y $s_{i-1} = ta$ y $s_i = tb$ para algún $t \in \Gamma^*$ (transiciones válidas)
- $r_m \in F$ (terminamos en aceptación)

- **Obs:** distintas nociones: pila vacía/estado final y pila vacía.

Computación

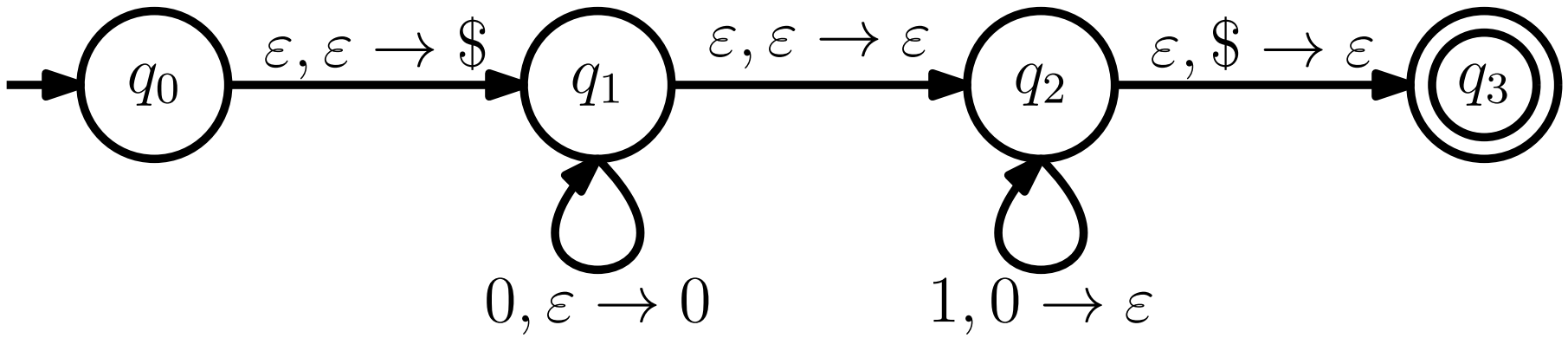
Def.

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

Diremos que *M acepta w* si podemos escribir $w = x_1 x_2 \dots x_m$ donde cada $x_i \in \Sigma \cup \{\varepsilon\}$, existen estados $r_0, \dots, r_m \in Q$ y existen strings $s_0, \dots, s_m \in \Gamma^*$ tales que

- $r_0 = q_0$ y $s_0 = \varepsilon$ (partimos en la config. inicial)
- para todo $i \in \{1, \dots, m\}$, tenemos que $(r_i, b) \in \delta(r_{i-1}, x_i, a)$ y $s_{i-1} = ta$ y $s_i = tb$ para algún $t \in \Gamma^*$ (transiciones válidas)
- $r_m \in F$ (terminamos en aceptación)

- **Obs:** distintas nociones: pila vacía/estado final y pila vacía.
- **Ej:** ¿01?



Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

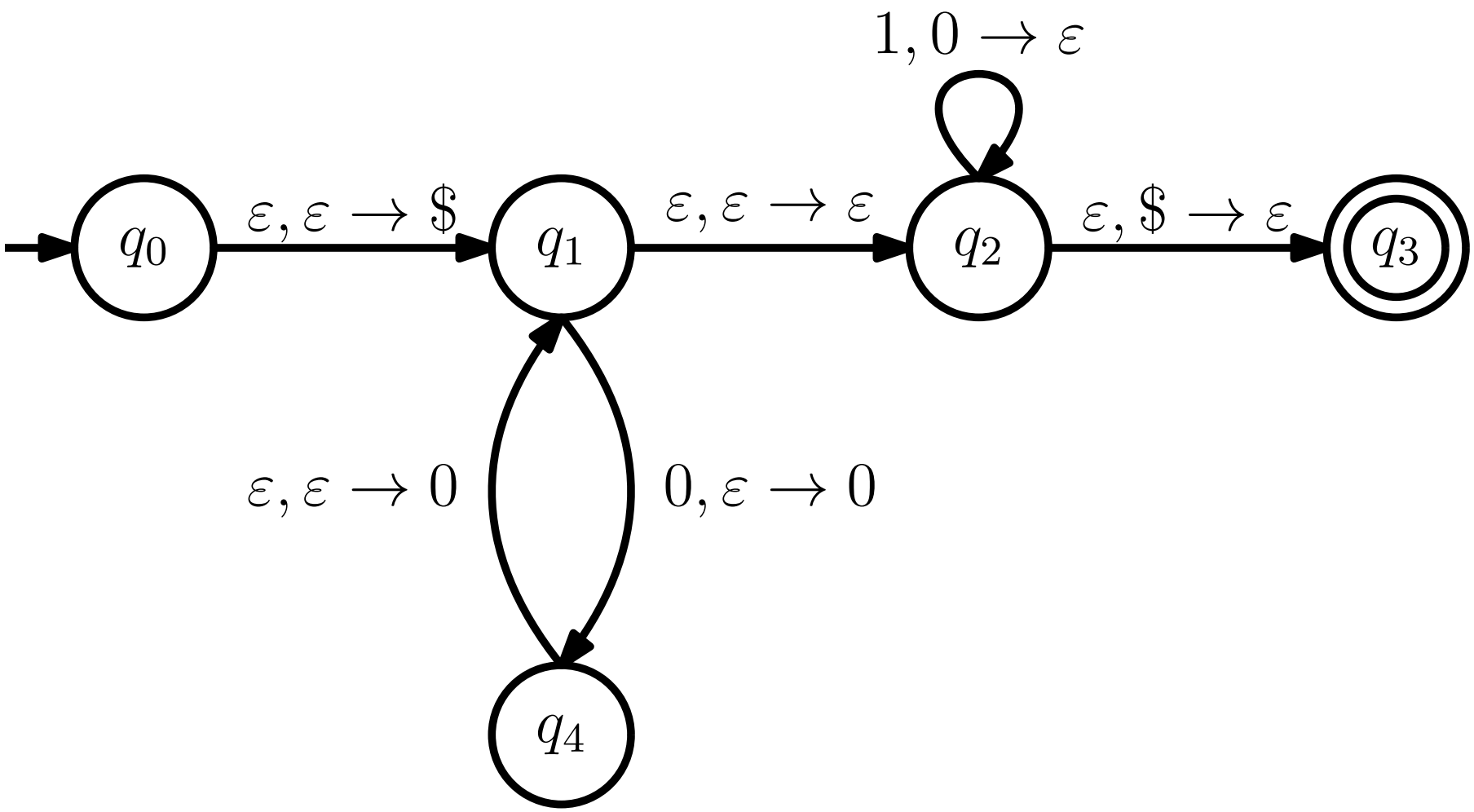
Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

- **Ej:** Lenguaje aceptado por



Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

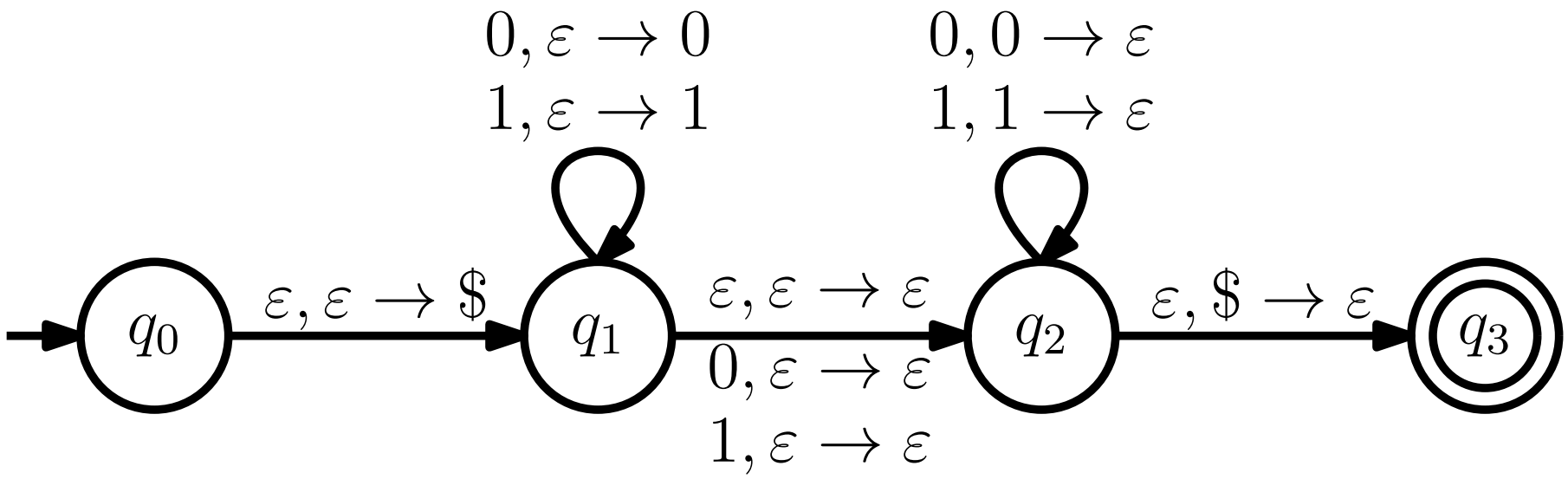
Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

- **Ej:** Lenguaje aceptado por



Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

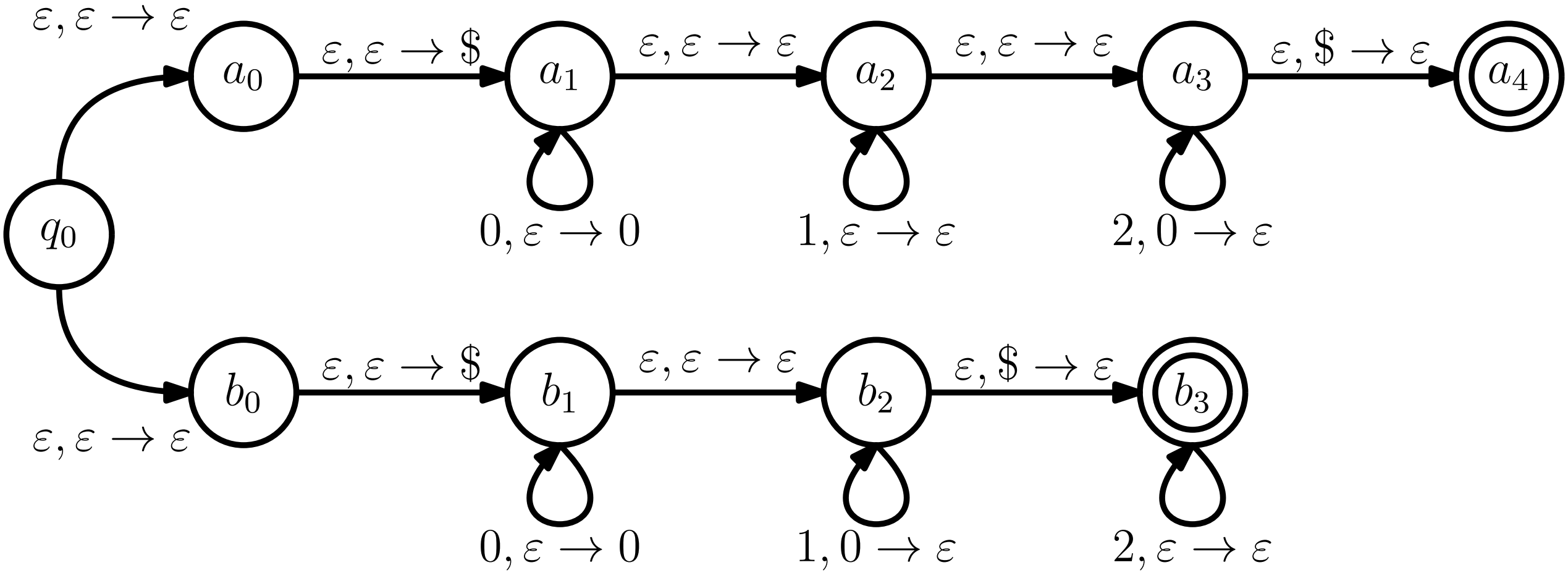
Autómatas vía sus lenguajes

Def.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AP. El *lenguaje aceptado* por M es

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

- **Ej:** Lenguaje aceptado por



Diseño de AP

- ¿Cómo diseño autómatas de pila?

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:
  - $L_1 = \{a^i b^j \mid i \geq j\}$

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:
  - $L_1 = \{a^i b^j \mid i \geq j\}$
  - $L_2 = \{a^i b^j c^j d^i \mid i, j \geq 0\}$

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:
  - $L_1 = \{a^i b^j \mid i \geq j\}$
  - $L_2 = \{a^i b^j c^j d^i \mid i, j \geq 0\}$
  - $L_3 = \{a^i \# a^j \# a^{i+j} \mid i, j \geq 0\}$

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:
  - $L_1 = \{a^i b^j \mid i \geq j\}$
  - $L_2 = \{a^i b^j c^j d^i \mid i, j \geq 0\}$
  - $L_3 = \{a^i \# a^j \# a^{i+j} \mid i, j \geq 0\}$
  - $L_4 = \{w \in \{(,)\}^* : w$ está bien balanceado$\}$

Diseño de AP

- ¿Cómo diseño autómatas de pila?
- Tips sobre el no determinismo: "adivinar" partes clave de la computación.
- Usar la pila para recordar información
- Diseñar autómatas para:
  - $L_1 = \{a^i b^j \mid i \geq j\}$
  - $L_2 = \{a^i b^j c^j d^i \mid i, j \geq 0\}$
  - $L_3 = \{a^i \# a^j \# a^{i+j} \mid i, j \geq 0\}$
  - $L_4 = \{w \in \{(,)\}^* : w$ está bien balanceado$\}$
  - $L_5 = \{w \in \{0,1\}^* : w$ tiene la misma cantidad de ceros que de unos$\}$

Lenguajes libres de contexto

- ¿Relación con las GLC?

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

- ¿AP **deterministas**?

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

- ¿AP **deterministas**?
  - Más débiles

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

- ¿AP **deterministas**?
  - Más débiles
  - GLC determinista / GLC con *endmarks*.

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

- ¿AP **deterministas**?
  - Más débiles
  - GLC determinista / GLC con *endmarks*.
  - ¿Realista?

Lenguajes libres de contexto

- ¿Relación con las GLC?

Teo.

Sea $L \subseteq \Sigma^*$.

L es libre de contexto si y sólo si existe un AP M tal que $\mathcal{L}(M) = L$.

- **Dem:** próxima clase.
- AP y GLC son *equivalentes*.

- ¿AP **deterministas**?
  - Más débiles
  - GLC determinista / GLC con *endmarks*.
  - ¿Realista?
  - Algoritmo de Valiant.

Dudas/Preguntas

Wrap-Up

- **Hoy:**

Wrap-Up

- **Hoy:**
  - Autómatas de pila

Wrap-Up

- **Hoy:**
  - Autómatas de pila
  - ¿Cómo computan?

Wrap-Up

- **Hoy:**
  - Autómatas de pila
  - ¿Cómo computan?
  - Relación con los lenguajes regulares

Wrap-Up

- **Hoy:**
  - Autómatas de pila
  - ¿Cómo computan?
  - Relación con los lenguajes regulares

- **Próx. clase:**

Wrap-Up

- **Hoy:**
  - Autómatas de pila
  - ¿Cómo computan?
  - Relación con los lenguajes regulares

- **Próx. clase:**
  - Equivalencia AP-GLC