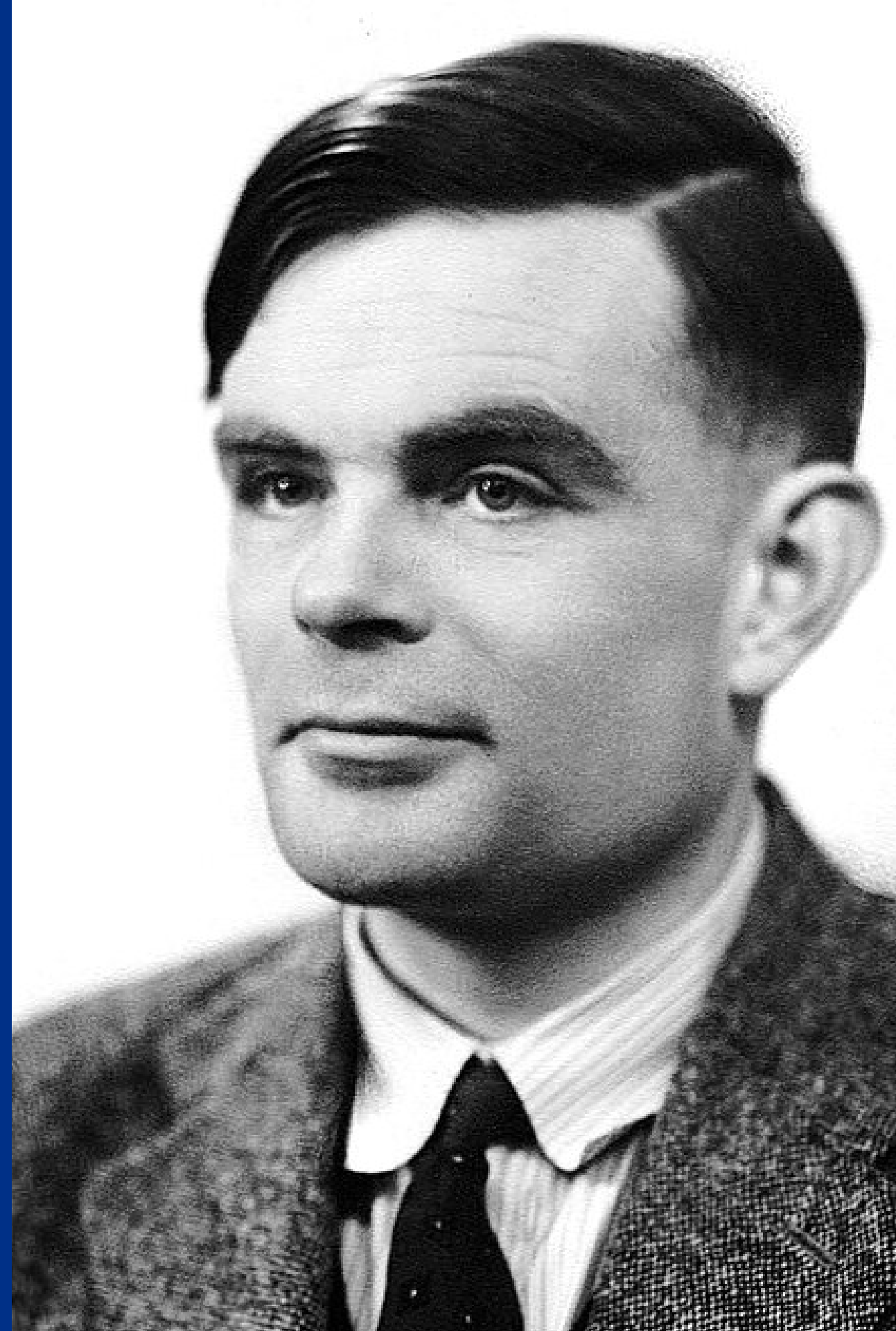
Máquinas de Turing

UNIVERSIDAD DE CHILE

Arturo Merino



Recuerdo / Plan

Recuerdo/plan

- **Recuerdo:**

Recuerdo/plan

- **Recuerdo:**
  - Lenguajes no libres de contexto.

Recuerdo/plan

- **Recuerdo:**
  - Lenguajes no libres de contexto.
  - Lema de Bombeo

Recuerdo/plan

- **Recuerdo:**
  - Lenguajes no libres de contexto.
  - Lema de Bombeo
- **Plan:**

Recuerdo/plan

- **Recuerdo:**
  - Lenguajes no libres de contexto.
  - Lema de Bombeo
- **Plan:**
  - Máquinas de Turing

Máquinas de Turing

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**
  - $\{0^n 1^n : n \in \mathbb{N}\}$ y $\{w\#w : w \in \{0,1\}^*\}$.

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**
  - $\{0^n 1^n : n \in \mathbb{N}\}$ y $\{w\#w : w \in \{0,1\}^*\}$.
  - ¿Qué falla?

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**
  - $\{0^n 1^n : n \in \mathbb{N}\}$ y $\{w\#w : w \in \{0,1\}^*\}$.
  - ¿Qué falla?
- AFD: tokenización/procesos. AP/GLC: sintaxis y estructura recursiva.

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**
  - $\{0^n 1^n : n \in \mathbb{N}\}$ y $\{w\#w : w \in \{0,1\}^*\}$.
  - ¿Qué falla?
- AFD: tokenización/procesos. AP/GLC: sintaxis y estructura recursiva.
  - Útiles, demasiado restringidos como modelo general.

Objetivo del curso

- **Objetivo del curso:** formalizar el modelo de computación.
- Entendimos el poder y los límites de AFD y AP/GLC.
- **Ej:**
  - $\{0^n 1^n : n \in \mathbb{N}\}$ y $\{w\#w : w \in \{0,1\}^*\}$.
  - ¿Qué falla?
- AFD: tokenización/procesos. AP/GLC: sintaxis y estructura recursiva.
  - Útiles, demasiado restringidos como modelo general.
- **Plan:** extender a un modelo de computación general.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.
- ¿Qué significa ejecutar un procedimiento mecánico?

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

- ¿Qué significa ejecutar un procedimiento mecánico?
- En los años 1930 se busca precisar la noción de *computar mecánicamente*.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

- ¿Qué significa ejecutar un procedimiento mecánico?
- En los años 1930 se busca precisar la noción de *computar mecánicamente*.
  - Turing: una persona que calcula paso a paso, con papel y símbolos.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

- ¿Qué significa ejecutar un procedimiento mecánico?
- En los años 1930 se busca precisar la noción de *computar mecánicamente*.
  - Turing: una persona que calcula paso a paso, con papel y símbolos.
  - Church: cálculo lambda.

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

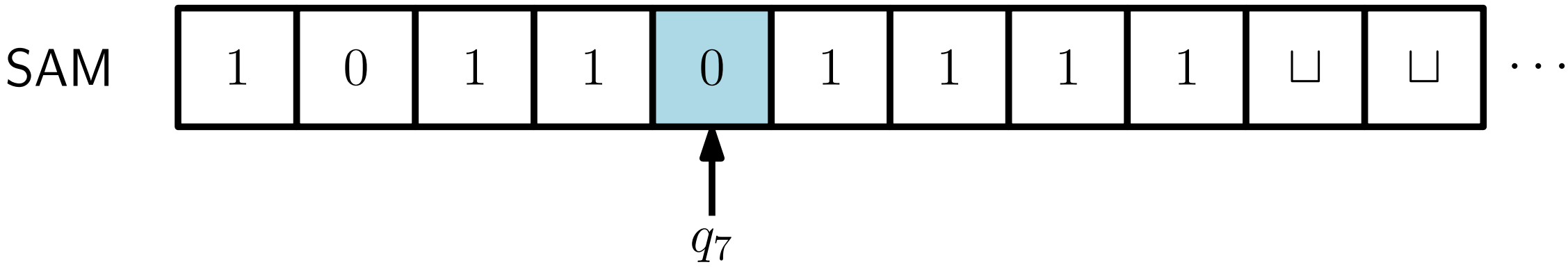
- ¿Qué significa ejecutar un procedimiento mecánico?
- En los años 1930 se busca precisar la noción de *computar mecánicamente*.
  - Turing: una persona que calcula paso a paso, con papel y símbolos.
  - Church: cálculo lambda.
  - Control finito + cinta = CPU + memoria

En busca de un modelo

- ¿Cómo extendemos nuestro modelo?
  - Agregar una segunda pila.
  - Usar una cola (u otra estructura de datos)
  - Dar acceso a memoria.
- Exploraremos la tercera idea
- La pregunta es anterior a los computadores.

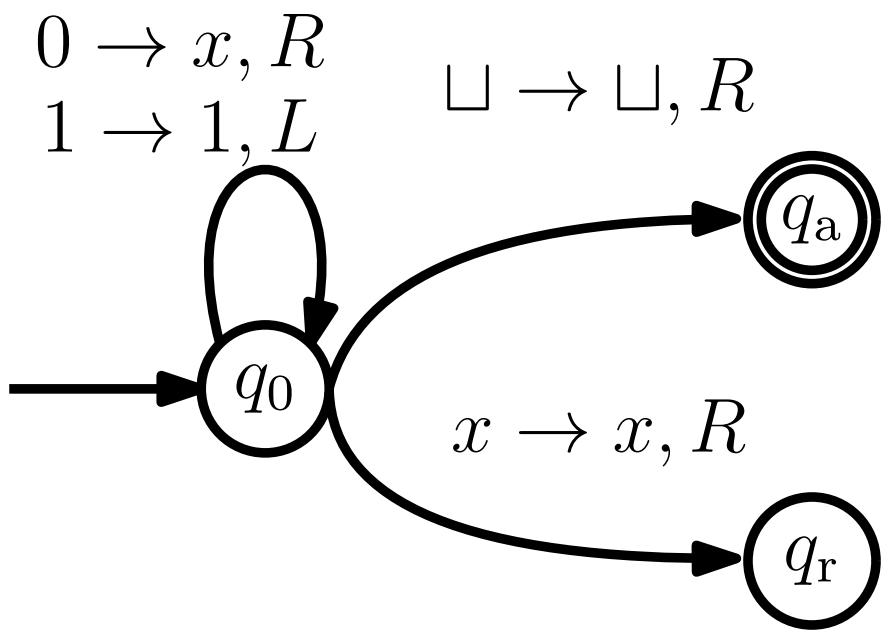
- ¿Qué significa ejecutar un procedimiento mecánico?
- En los años 1930 se busca precisar la noción de *computar mecánicamente*.
  - Turing: una persona que calcula paso a paso, con papel y símbolos.
  - Church: cálculo lambda.
  - Control finito + cinta = CPU + memoria
- Buscamos algo "con el mismo poder" que un lenguaje de programación moderno.

El acceso a memoria

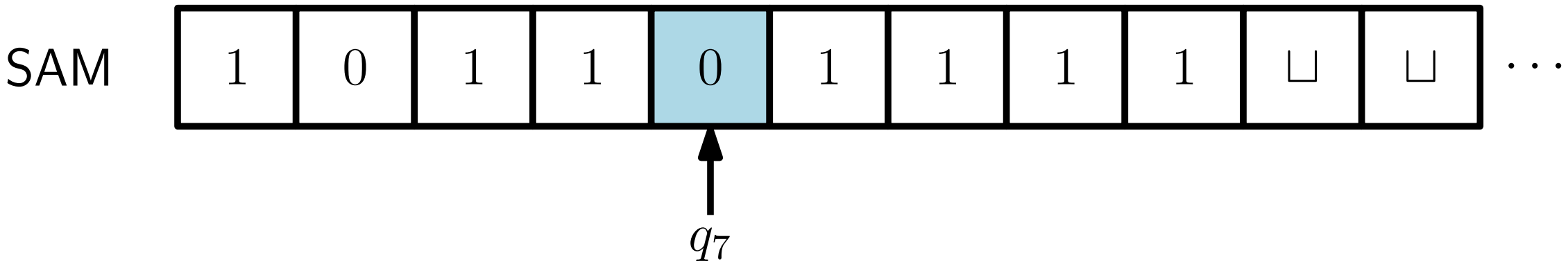


- **Ej:** M_0, con entrada en $\{0,1\}^*$.

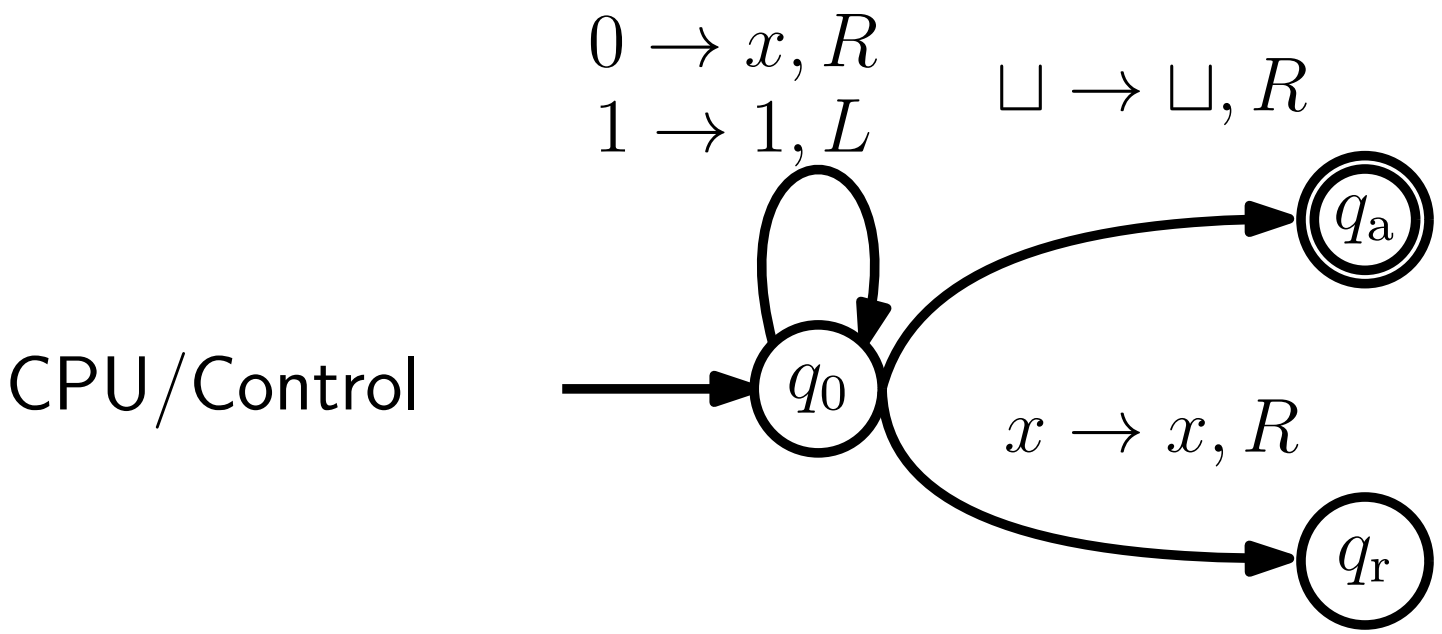
CPU/Control



El acceso a memoria

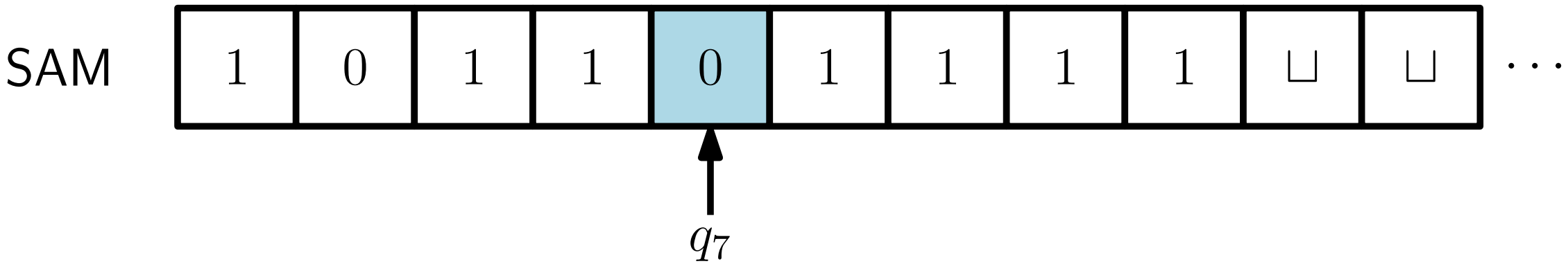


- **Ej:** M_0, con entrada en $\{0,1\}^*$.

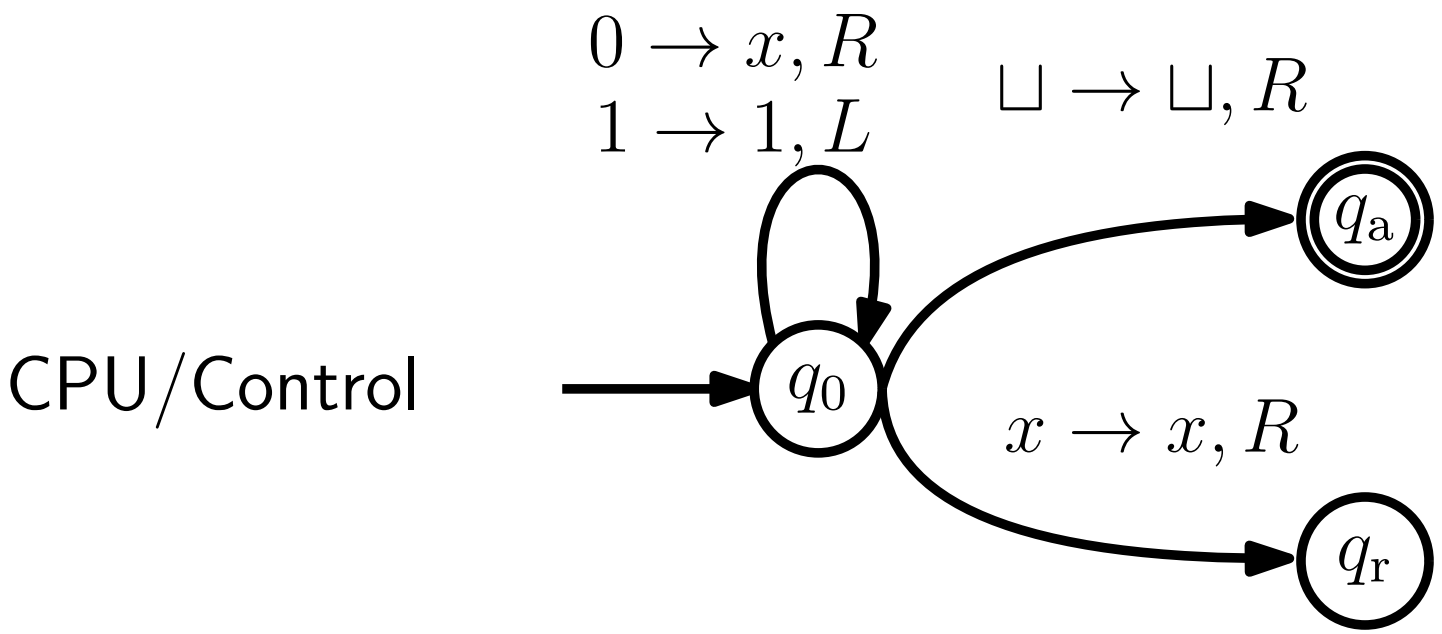


- $a \to b, D$: leer a, escribir b y mover el cabezal según D.

El acceso a memoria

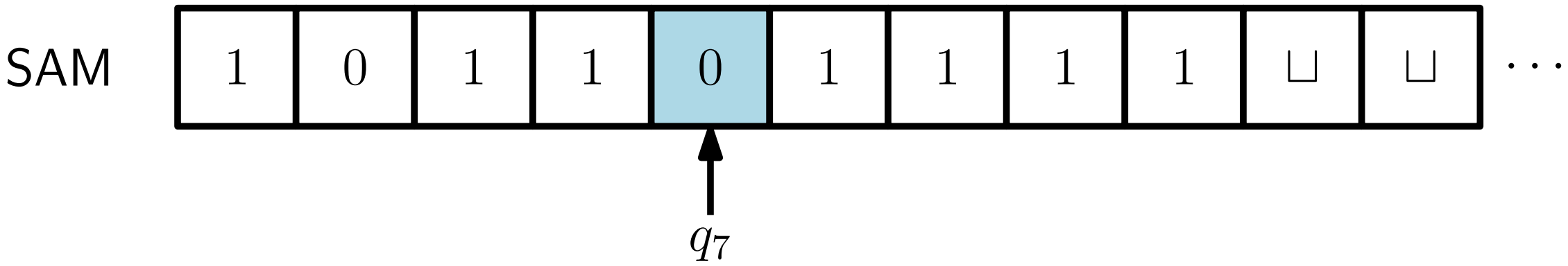


- **Ej:** M_0, con entrada en $\{0,1\}^*$.

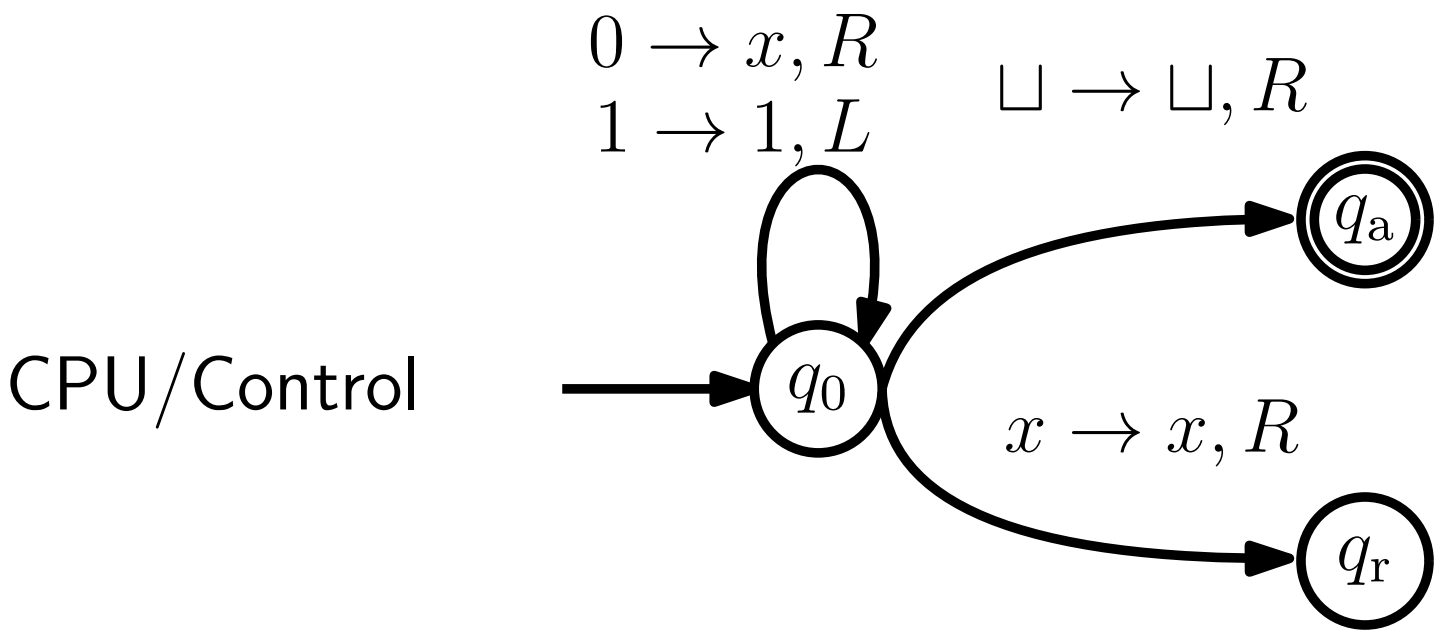


- $a \to b, D$: leer a, escribir b y mover el cabezal según D.
  - L: izquierda; R: derecha. $a \to D$ conserva el símbolo.

El acceso a memoria

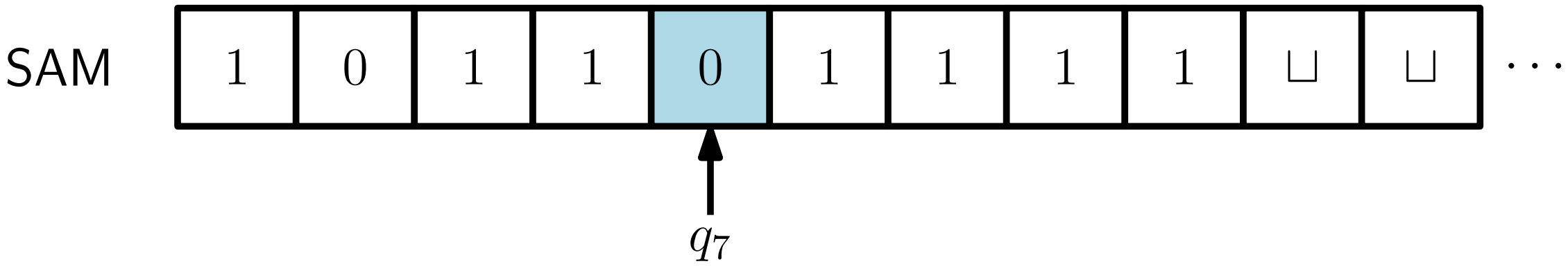


- **Ej:** M_0, con entrada en $\{0,1\}^*$.

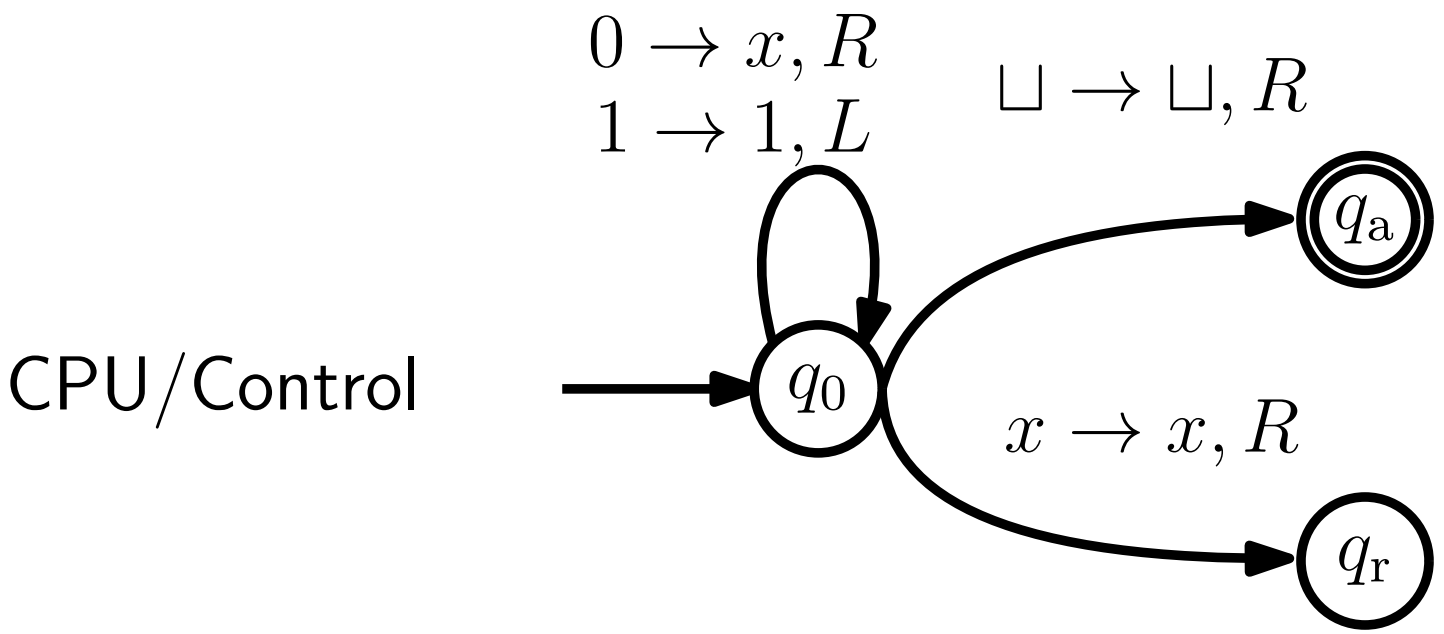


- $a \to b, D$: leer a, escribir b y mover el cabezal según D.
  - L: izquierda; R: derecha. $a \to D$ conserva el símbolo.
- En q_{a} devuelve `True`; en q_{r} `False`.

El acceso a memoria



- **Ej:** M_0, con entrada en $\{0,1\}^*$.



- $a \to b, D$: leer a, escribir b y mover el cabezal según D.
  - L: izquierda; R: derecha. $a \to D$ conserva el símbolo.
- En q_{a} devuelve `True`; en q_{r} `False`.
- Díficil de asimilar.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.
- **Ej:** M_0. Repetir según el símbolo que lee:

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.
- **Ej:** M_0. Repetir según el símbolo que lee:
  - Si es 0, escribir x y mover a la derecha.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.
- **Ej:** M_0. Repetir según el símbolo que lee:
  - Si es 0, escribir x y mover a la derecha.
  - Si es 1, conservarlo y mover a la izquierda.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.

- **Ej:** M_0. Repetir según el símbolo que lee:
  - Si es 0, escribir x y mover a la derecha.
  - Si es 1, conservarlo y mover a la izquierda.
  - Si es x, rechazar.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.

- **Ej:** M_0. Repetir según el símbolo que lee:
  - Si es 0, escribir x y mover a la derecha.
  - Si es 1, conservarlo y mover a la izquierda.
  - Si es x, rechazar.
  - Si es blanco, aceptar.

Describir una máquina

- Diagramas de estado $\leftarrow$ complejos.
  - Usualmente describiremos la máquina en español o pseudocódigo.
- **Ej:** M_0. Repetir según el símbolo que lee:
  - Si es 0, escribir x y mover a la derecha.
  - Si es 1, conservarlo y mover a la izquierda.
  - Si es x, rechazar.
  - Si es blanco, aceptar.
- La descripción debe poder traducirse a estados y transiciones.

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición,
5. $q_0 \in Q$ es el estado inicial,

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición,
5. $q_0 \in Q$ es el estado inicial,
6. $q_{\mathrm{a}} \in Q$ es el estado de aceptación, y

Máquinas de Turing

Def.

Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición,
5. $q_0 \in Q$ es el estado inicial,
6. $q_{\mathrm{a}} \in Q$ es el estado de aceptación, y
7. $q_{\mathrm{r}} \in Q$ es el estado de rechazo, donde $q_{\mathrm{r}} \neq q_{\mathrm{a}}$.

Máquinas de Turing

Def.

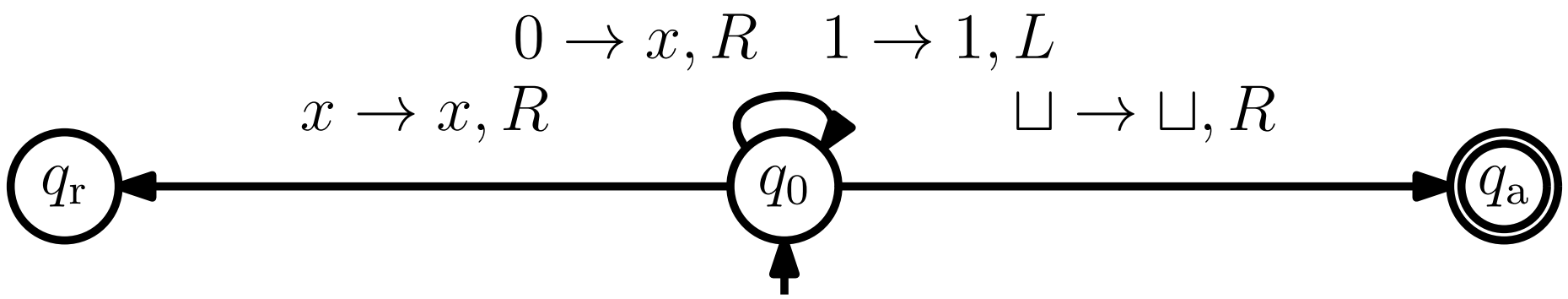
Una *máquina de Turing* es una 7-tupla,

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}}),$$

donde Q, Σ, Γ son conjuntos finitos y

1. Q es el conjunto de estados,
2. Σ es el alfabeto de entrada, que no contiene el símbolo blanco $\sqcup$,
3. Γ es el alfabeto de cinta, donde $\sqcup \in \Gamma$ y $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición,
5. $q_0 \in Q$ es el estado inicial,
6. $q_{\mathrm{a}} \in Q$ es el estado de aceptación, y
7. $q_{\mathrm{r}} \in Q$ es el estado de rechazo, donde $q_{\mathrm{r}} \neq q_{\mathrm{a}}$.

- **Ej:** M_0.



La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.

La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.

La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.
  - Las demás contienen $\sqcup$; el cabezal parte en la primera celda.

La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.
  - Las demás contienen $\sqcup$; el cabezal parte en la primera celda.
  - El estado inicial es q_0.

La cinta y el cabezal

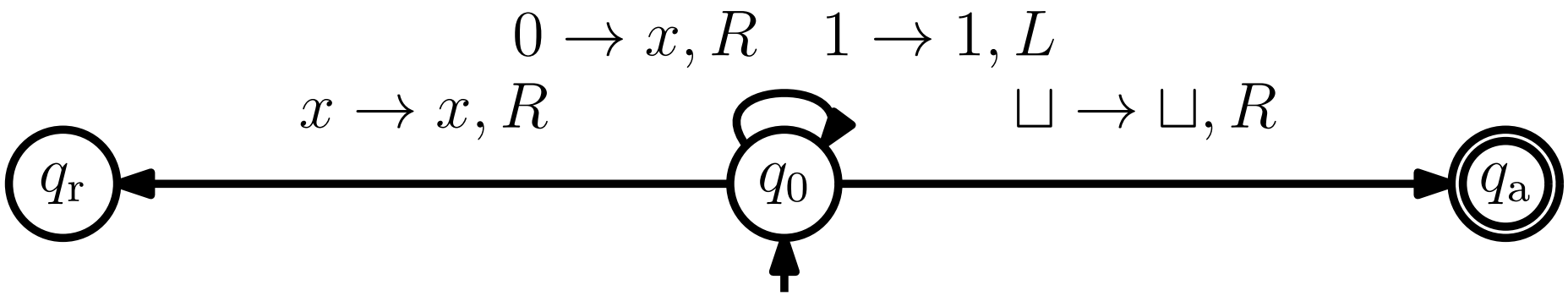
- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.
  - Las demás contienen $\sqcup$; el cabezal parte en la primera celda.
  - El estado inicial es q_0.
- Intentar ir "más a la izquierda" deja el cabezal en la primera celda.

La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.
  - Las demás contienen $\sqcup$; el cabezal parte en la primera celda.
  - El estado inicial es q_0.
- Intentar ir "más a la izquierda" deja el cabezal en la primera celda.
  - En q_{a} o q_{r} la máquina se detiene.

La cinta y el cabezal

- La cinta tiene una primera celda y es infinita hacia la derecha.
- Al inicio, el input $w = w_1 \cdots w_n \in \Sigma^*$ ocupa las primeras n celdas.
  - Las demás contienen $\sqcup$; el cabezal parte en la primera celda.
  - El estado inicial es q_0.
- Intentar ir "más a la izquierda" deja el cabezal en la primera celda.
  - En q_{a} o q_{r} la máquina se detiene.
- **Ej:** M_0 con entrada 001. ¿Con ε?



Configuraciones

- **Configuración:** describe el status de la computación.

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila
- **MT**: estado + cabezal + cinta.

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila
- **MT**: estado + cabezal + cinta.

Def.

Sea M una MT.

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila
- **MT**: estado + cabezal + cinta.

Def.

Sea M una MT.
Una *configuración* es un string $\alpha\ q\ \beta \in \Gamma^* Q \Gamma^*$ dónde α son los contenidos de la cinta a la izquierda del cabezal, q es el estado actual de la máquina, y β es el resto de la cinta.

Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila
- **MT**: estado + cabezal + cinta.

Def.

Sea M una MT.
Una *configuración* es un string $\alpha\ q\ \beta \in \Gamma^* Q \Gamma^*$ dónde α son los contenidos de la cinta a la izquierda del cabezal, q es el estado actual de la máquina, y β es el resto de la cinta.

- **Obs:** no anotamos los infinitos $\sqcup$ del final.

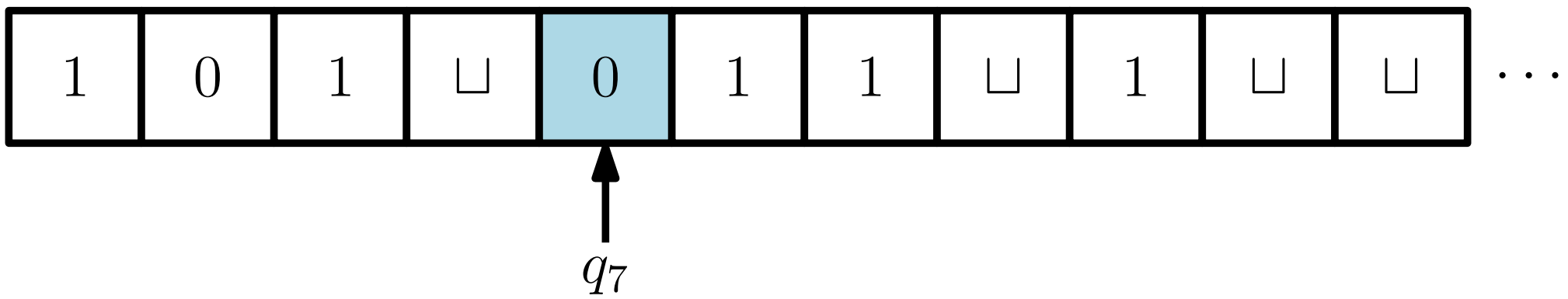
Configuraciones

- **Configuración:** describe el status de la computación.
- **AFD:** estado
- **AP:** estado + pila
- **MT**: estado + cabezal + cinta.

Def.

Sea M una MT.
Una *configuración* es un string $\alpha\ q\ \beta \in \Gamma^* Q \Gamma^*$ dónde α son los contenidos de la cinta a la izquierda del cabezal, q es el estado actual de la máquina, y β es el resto de la cinta.

- **Obs:** no anotamos los infinitos $\sqcup$ del final.
- **Ej:**



Un paso de computación

- ¿Cómo computa una MT?

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$ una MT.

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{a}}, q_{\text{r}})$ una MT.

Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_{\text{a}}, q_{\text{r}}\}$ y $r \in Q$.

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\text{a}, q_\text{r})$ una MT.

Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_\text{a}, q_\text{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $ua\ q\ bv$:

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $ua\ q\ bv$:

- La configuración $uac\ r\ v$ si $\delta(q, b) = (r, c, R)$.

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\text{a}, q_\text{r})$ una MT.

Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_\text{a}, q_\text{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $ua\ q\ bv$:

- La configuración $uac\ r\ v$ si $\delta(q, b) = (r, c, R)$.
- La configuración $u\ r\ acv$ si $\delta(q, b) = (r, c, L)$.

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $ua\ q\ bv$:

- La configuración $uac\ r\ v$ si $\delta(q, b) = (r, c, R)$.
- La configuración $u\ r\ acv$ si $\delta(q, b) = (r, c, L)$.

Si M computa desde la config. C la config. D en un paso, lo denotaremos por $C \vdash_M D$

Un paso de computación

- ¿Cómo computa una MT?

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{a}}, q_{\text{r}})$ una MT.

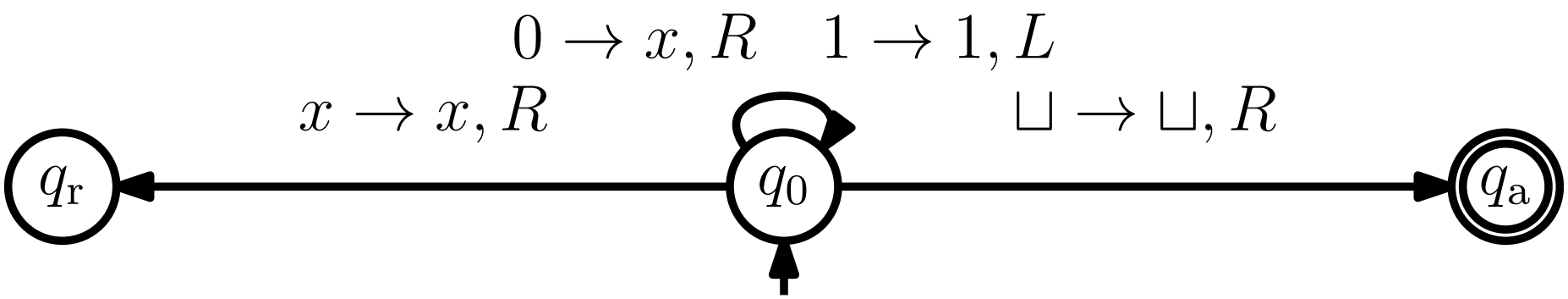
Sea $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q \in Q \setminus \{q_{\text{a}}, q_{\text{r}}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $ua\ q\ bv$:

- La configuración $uac\ r\ v$ si $\delta(q, b) = (r, c, R)$.
- La configuración $u\ r\ acv$ si $\delta(q, b) = (r, c, L)$.

Si M computa desde la config. C la config. D en un paso, lo denotaremos por $C \vdash_M D$

- **Ej:** en M_0, qué ocurre con $x\ q_0 01$ y $x\ q_0\ 1$.



Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$ una MT.

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $v \in \Gamma^*$, $b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $v \in \Gamma^*$, $b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $q\ bv$:

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $v \in \Gamma^*$, $b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $q\ bv$:

- La configuración $c\ r\ v$ si $\delta(q, b) = (r, c, R)$.

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $v \in \Gamma^*$, $b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $q\ bv$:

- La configuración $c\ r\ v$ si $\delta(q, b) = (r, c, R)$.
- La configuración $r\ cv$ si $\delta(q, b) = (r, c, L)$.

Extremo izquierdo de la cinta

- Caso borde: extremo izquierdo de la cinta.

Def.

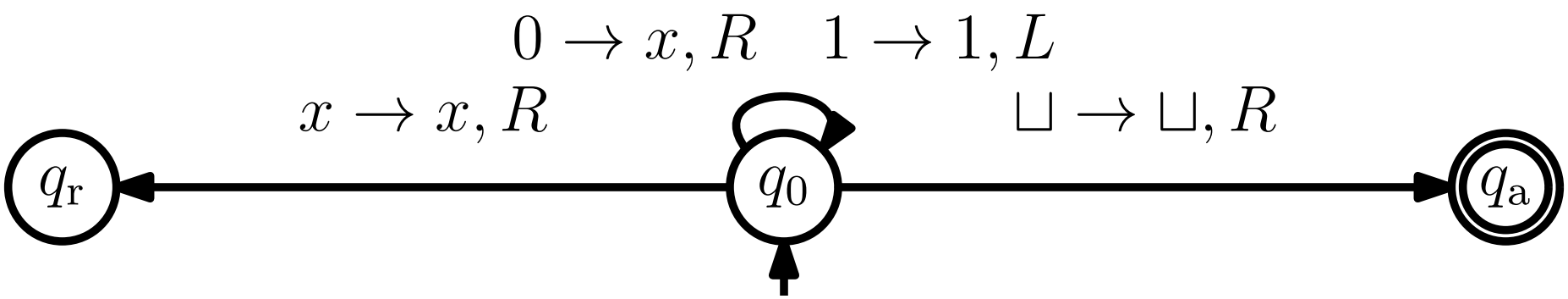
Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT.

Sea $v \in \Gamma^*$, $b \in \Gamma$, $q \in Q \setminus \{q_\mathrm{a}, q_\mathrm{r}\}$ y $r \in Q$.

Diremos que M *computa en un paso* desde $q\ bv$:

- La configuración $c\ r\ v$ si $\delta(q, b) = (r, c, R)$.
- La configuración $r\ cv$ si $\delta(q, b) = (r, c, L)$.

- **Ej:** en M_0, ¿$q_0\,1$? ¿q_0?



Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$ una MT y C, D dos configuraciones.

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Def.

Si una configuración está en estado q_a será *de aceptación*, si está en estado q_r será *de rechazo.*

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Def.

Si una configuración está en estado q_a será *de aceptación*, si está en estado q_r será *de rechazo.*

- Una MT M *acepta* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de aceptación.

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Def.

Si una configuración está en estado q_a será *de aceptación*, si está en estado q_r será *de rechazo.*

- Una MT M *acepta* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de aceptación.
- Una MT M *rechaza* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de rechazo.

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Def.

Si una configuración está en estado q_a será *de aceptación*, si está en estado q_r será *de rechazo.*

- Una MT M *acepta* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de aceptación.
- Una MT M *rechaza* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de rechazo.

Si M acepta o rechaza w diremos que se *detiene* en w.

Una computación

Def.

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, q_\mathrm{a}, q_\mathrm{r})$ una MT y C, D dos configuraciones.

Diremos que M *computa* D desde C si $C \vdash_M^* D$.

Es decir, existen $k \in \mathbb{N}$ configuraciones $C_1, \ldots, C_k$ tal que

$$C = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_k = D.$$

- ¿Config. inicial? ¿Aceptación? ¿Rechazo?

Def.

Si una configuración está en estado q_a será *de aceptación*, si está en estado q_r será *de rechazo.*

- Una MT M *acepta* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de aceptación.
- Una MT M *rechaza* $w \in \Sigma^*$ si $q_0 w \vdash_M^* C$ con C de rechazo.

Si M acepta o rechaza w diremos que se *detiene* en w.

- **Notación:** $M(w) = 1$ para aceptación, $M(w) = 0$ para rechazo.

Aceptar una entrada

Def.

El *lenguaje reconocido* por una MT M es

Aceptar una entrada

Def.

El *lenguaje reconocido* por una MT M es

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

Aceptar una entrada

Def.

El *lenguaje reconocido* por una MT M es

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

Un lenguaje es *Turing-reconocible* si alguna máquina de Turing lo reconoce.

Aceptar una entrada

Def.

El *lenguaje reconocido* por una MT M es

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

Un lenguaje es *Turing-reconocible* si alguna máquina de Turing lo reconoce.

- También se usa el nombre *recursivamente enumerable*.

Aceptar una entrada

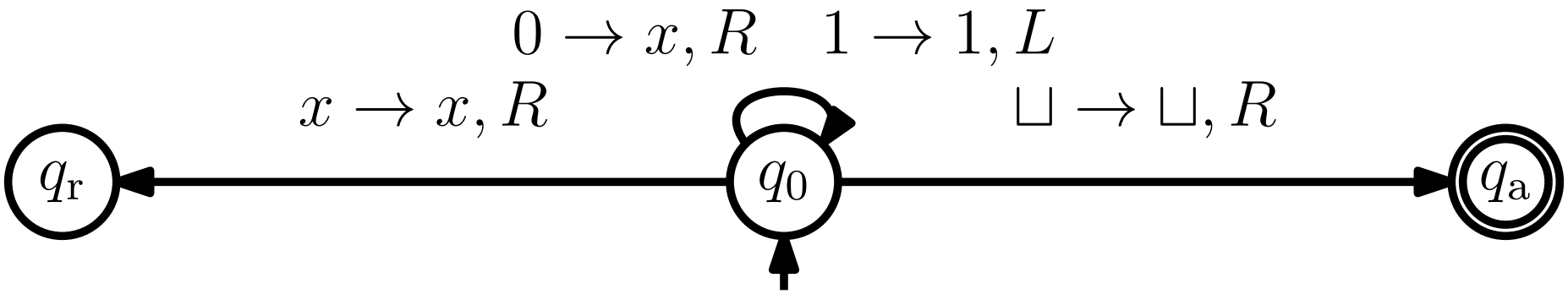
Def.

El *lenguaje reconocido* por una MT M es

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

Un lenguaje es *Turing-reconocible* si alguna máquina de Turing lo reconoce.

- También se usa el nombre *recursivamente enumerable*.
- **Ej:** M_0 con ε, 00, 01 y 1. ¿Qué entradas acepta?



Aceptar una entrada

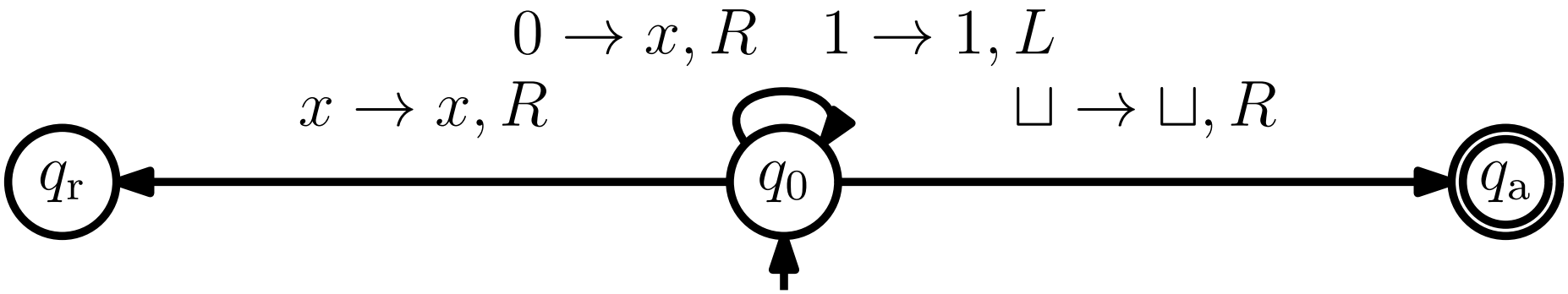
Def.

El *lenguaje reconocido* por una MT M es

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

Un lenguaje es *Turing-reconocible* si alguna máquina de Turing lo reconoce.

- También se usa el nombre *recursivamente enumerable*.
- **Ej:** M_0 con ε, 00, 01 y 1. ¿Qué entradas acepta?



- Strings aceptados/rechazados ¿Única opción?

Decidir un lenguaje

Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Decidir un lenguaje

Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Decidir un lenguaje

Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Un lenguaje es *Turing-decidible*, o *decidible*, si alguna MT lo decide.

Decidir un lenguaje

Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Un lenguaje es *Turing-decidible*, o *decidible*, si alguna MT lo decide.

- También se usa el nombre *recursivo*.

Decidir un lenguaje

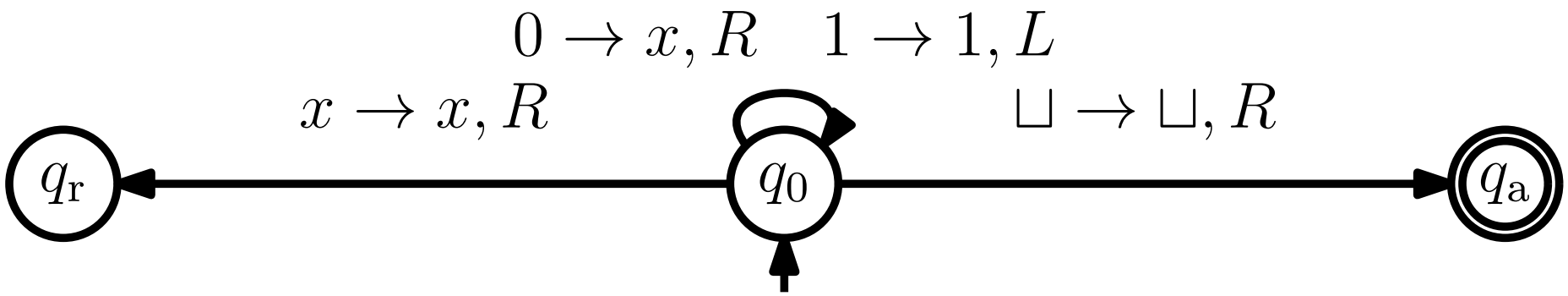
Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Un lenguaje es *Turing-decidible*, o *decidible*, si alguna MT lo decide.

- También se usa el nombre *recursivo*.
- **Ej:** M_0. ¿Es un decididor?



Decidir un lenguaje

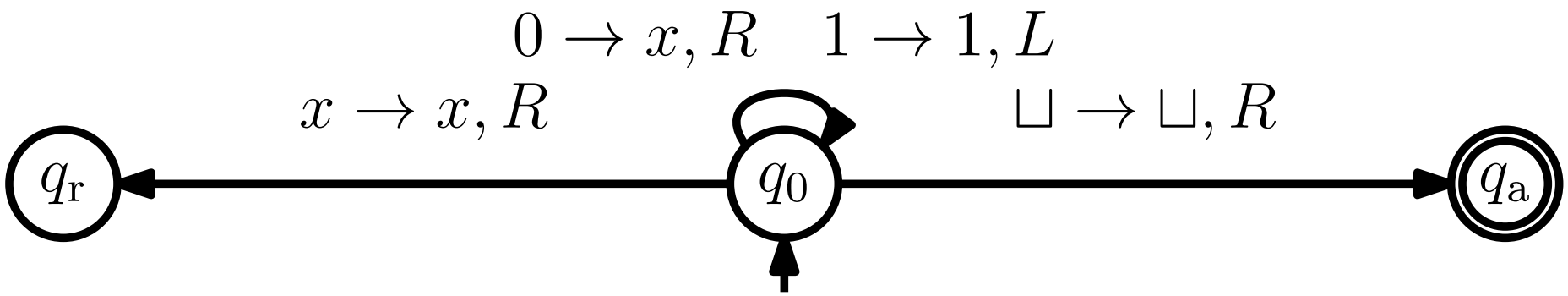
Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Un lenguaje es *Turing-decidible*, o *decidible*, si alguna MT lo decide.

- También se usa el nombre *recursivo*.
- **Ej:** M_0. ¿Es un decididor?



- Todo lenguaje decidible es Turing-reconocible.

Decidir un lenguaje

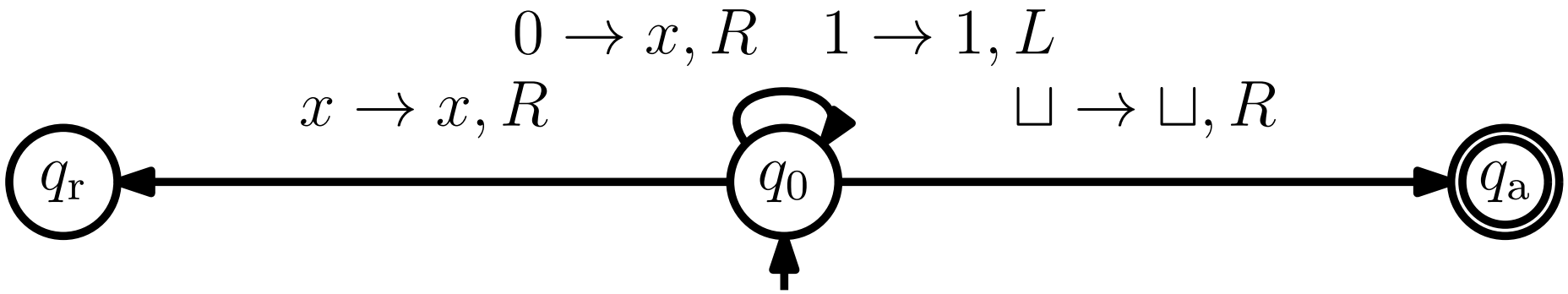
Def.

Un *decisor/decididor* es una MT que se detiene en toda entrada de Σ^*.

Si además reconoce L, decimos que *decide* L.

Un lenguaje es *Turing-decidible*, o *decidible*, si alguna MT lo decide.

- También se usa el nombre *recursivo*.
- **Ej:** M_0. ¿Es un decididor?



- Todo lenguaje decidible es Turing-reconocible.
  - Más adelante veremos que la recíproca no vale.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.
  - Regresar a la izquierda y continuar.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.
  - Regresar a la izquierda y continuar.
- Al terminar: si quedan símbolos a la derecha, rechazar; si no, aceptar.

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.
  - Regresar a la izquierda y continuar.
- Al terminar: si quedan símbolos a la derecha, rechazar; si no, aceptar.
- ¿Qué lenguaje reconoce M_1? ¿Siempre termina?

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.
  - Regresar a la izquierda y continuar.
- Al terminar: si quedan símbolos a la derecha, rechazar; si no, aceptar.
- ¿Qué lenguaje reconoce M_1? ¿Siempre termina?

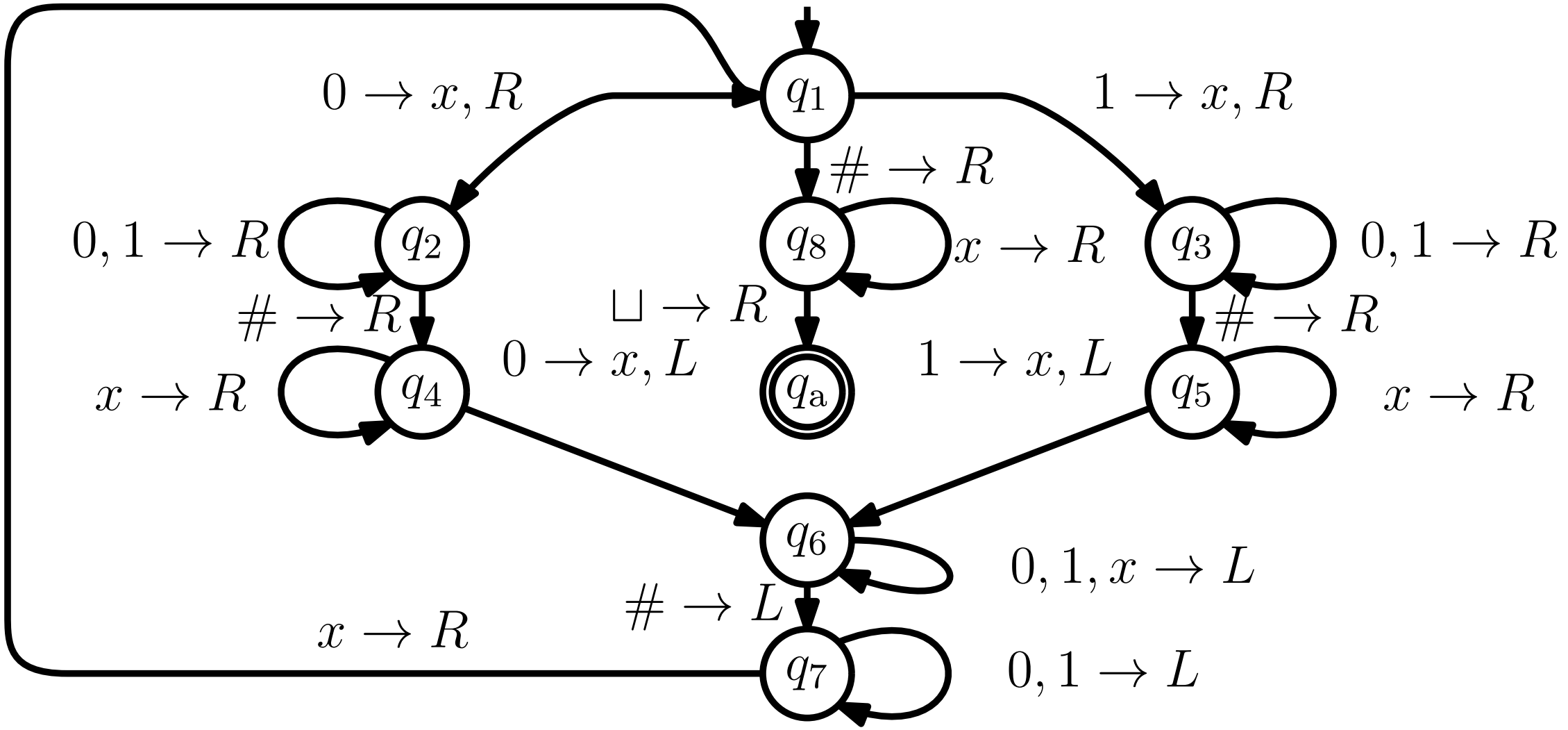
- **Ej:** ¿Cómo procesa estas entradas?

Una máquina en palabras

- **Ej:** M_1, con entrada $w \in \{0, 1, \#\}^*$.
- Si no aparece $\#$, rechazar.
- Mientras queden símbolos sin tachar antes del primer $\#$:
  - Comparar el primero con el siguiente sin tachar de la derecha.
  - Si difieren o falta su par, rechazar; si no, tachar ambos.
  - Regresar a la izquierda y continuar.
- Al terminar: si quedan símbolos a la derecha, rechazar; si no, aceptar.
- ¿Qué lenguaje reconoce M_1? ¿Siempre termina?

- **Ej:** ¿Cómo procesa estas entradas?
  - $011000\#011000, \quad 01\#01, \quad 01\#10, \quad \#, \quad \varepsilon.$

Diagrama de estados



Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

- Repetir el siguiente recorrido:

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.
- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.
- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.
- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.
- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.
  - En otro caso, volver al extremo izquierdo y repetir.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.
  - En otro caso, volver al extremo izquierdo y repetir.

- ¿Qué lenguaje reconoce? ¿Qué cambia en cada recorrido?

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.
  - En otro caso, volver al extremo izquierdo y repetir.

- ¿Qué lenguaje reconoce? ¿Qué cambia en cada recorrido?

- **Ej:** simular los recorridos sobre la entrada 0000.

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.
  - En otro caso, volver al extremo izquierdo y repetir.

- ¿Qué lenguaje reconoce? ¿Qué cambia en cada recorrido?

- **Ej:** simular los recorridos sobre la entrada 0000.
  - Comparar con 0, 000 y 00000000. ¿Siempre termina?

Otra máquina en palabras

- **Ej:** M_2. Primero, entradas no vacías en $\{0\}^*$.

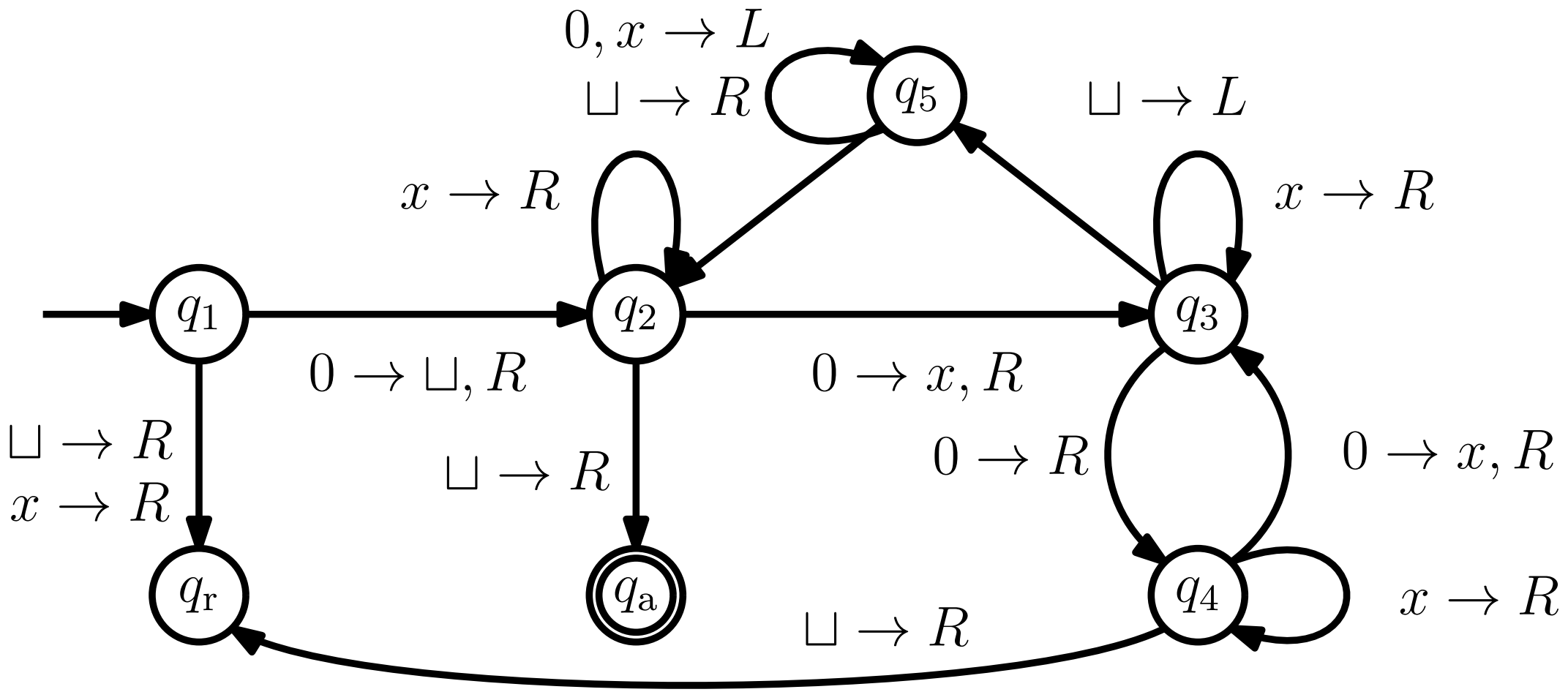
- Repetir el siguiente recorrido:
  - Ir a la derecha tachando alternadamente los 0 aún sin tachar.
  - Si al empezar había sólo uno, aceptar.
  - Si había más de uno y la cantidad era impar, rechazar.
  - En otro caso, volver al extremo izquierdo y repetir.

- ¿Qué lenguaje reconoce? ¿Qué cambia en cada recorrido?

- **Ej:** simular los recorridos sobre la entrada 0000.
  - Comparar con 0, 000 y 00000000. ¿Siempre termina?

- ¿Cómo completaríamos la descripción para la entrada ε?

Diagrama de estados



Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.

- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.

- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.
  - Si son iguales, rechazar; si no, avanzar la marca derecha al próximo $\#$.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.
  - Si son iguales, rechazar; si no, avanzar la marca derecha al próximo $\#$.
  - Si ya no hay otro $\#$, avanzar la izquierda al siguiente $\#$

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.
  - Si son iguales, rechazar; si no, avanzar la marca derecha al próximo $\#$.
  - Si ya no hay otro $\#$, avanzar la izquierda al siguiente $\#$
  - y poner la derecha en el posterior; si no existe, aceptar.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.
  - Si son iguales, rechazar; si no, avanzar la marca derecha al próximo $\#$.
  - Si ya no hay otro $\#$, avanzar la izquierda al siguiente $\#$
  - y poner la derecha en el posterior; si no existe, aceptar.
- Marcas son partes del alfabeto: $\#$ y $\dot{\#}$ son símbolos distintos.

Tercer lenguaje

- **Ej:** M_3, con entrada $w \in \{0, 1, \#\}^*$.
- Marcar el primer símbolo: si era blanco, aceptar; si no era $\#$, rechazar.
- Buscar y marcar el siguiente $\#$; si antes hay blanco, aceptar.
- Repetir:
  - Comparar los strings después de los dos $\#$ marcados.
  - Si son iguales, rechazar; si no, avanzar la marca derecha al próximo $\#$.
  - Si ya no hay otro $\#$, avanzar la izquierda al siguiente $\#$
  - y poner la derecha en el posterior; si no existe, aceptar.
- Marcas son partes del alfabeto: $\#$ y $\dot{\#}$ son símbolos distintos.
- **Standard:** Marcar caracteres.

Diseñar máquinas de Turing

- **Ej:** diseñar un decisor para cada lenguaje.

Diseñar máquinas de Turing

- **Ej:** diseñar un decisor para cada lenguaje.
  - $\{0^n 1^n : n \in \mathbb{Z}_{\geq 0}\}$.

Diseñar máquinas de Turing

- **Ej:** diseñar un decisor para cada lenguaje.
  - $\{0^n 1^n : n \in \mathbb{Z}_{\geq 0}\}$.
  - $\{a^n b^n c^n : n \in \mathbb{Z}_{\geq 0}\}$.

Diseñar máquinas de Turing

- **Ej:** diseñar un decisor para cada lenguaje.
  - $\{0^n 1^n : n \in \mathbb{Z}_{\geq 0}\}$.
  - $\{a^n b^n c^n : n \in \mathbb{Z}_{\geq 0}\}$.
  - $\{0^{n^2} : n \in \mathbb{N}\}$

Dudas

Recuerdo/plan

- **Hoy:**

Recuerdo/plan

- **Hoy:**
  - Máquinas de Turing

Recuerdo/plan

- **Hoy:**
  - Máquinas de Turing
- **Próxima clase:**

Recuerdo/plan

- **Hoy:**
  - Máquinas de Turing
- **Próxima clase:**
  - Variantes