

# Máquinas de Turing

## Matemáticas Computacionales (TC2020)

M.C. Xavier Sánchez Díaz  
sax@tec.mx



# Lenguajes

Lo que hemos visto hasta ahora

Un **lenguaje** es un conjunto de **palabras**.

¿Cómo **demostramos** que un lenguaje es **regular**?

¿Cómo **demostramos** que un lenguaje es **libre de contexto**?

# Lenguajes

Lo que hemos visto hasta ahora

Un **lenguaje** es un conjunto de **palabras**.

¿Cómo **demostramos** que un lenguaje es **regular**?

¿Cómo **demostramos** que un lenguaje es **libre de contexto**?

# Lenguajes

Lo que hemos visto hasta ahora

Un **lenguaje** es un conjunto de **palabras**.

¿Cómo **demostramos** que un lenguaje es **regular**?

¿Cómo **demostramos** que un lenguaje es **libre de contexto**?

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Los **Autómatas Finitos**, las **expresiones regulares** y las **gramáticas regulares** sirven para representar **lenguajes regulares**.

Los **Autómatas de Pila** y las **gramáticas libres de contexto** sirven para representar **lenguajes libres de contexto**.

Sin embargo, hay otros lenguajes que no podemos representar con ninguna de estas herramientas.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Los **Autómatas Finitos**, las **expresiones regulares** y las **gramáticas regulares** sirven para representar **lenguajes regulares**.

Los **Autómatas de Pila** y las **gramáticas libres de contexto** sirven para representar **lenguajes libres de contexto**.

Sin embargo, hay otros lenguajes que no podemos representar con ninguna de estas herramientas.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Los **Autómatas Finitos**, las **expresiones regulares** y las **gramáticas regulares** sirven para representar **lenguajes regulares**.

Los **Autómatas de Pila** y las **gramáticas libres de contexto** sirven para representar **lenguajes libres de contexto**.

Sin embargo, hay otros lenguajes que no podemos representar con ninguna de estas herramientas.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Por ejemplo, intentemos representar  $\{a^n b^n c^n | n \geq 0\}$  con un AP:

- Por cada  $a$  ponemos un contador en la pila
- Por cada  $b$  quitamos un contador de la pila
- No podemos contar las  $c$ s.

O si cambiamos el orden o usamos más contadores, simplemente no es posible hacerlo con un autómata de pila.



# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Por ejemplo, intentemos representar  $\{a^n b^n c^n | n \geq 0\}$  con un AP:

- Por cada  $a$  ponemos un contador en la pila
- Por cada  $b$  quitamos un contador de la pila
- No podemos contar las  $c$ s.

O si cambiamos el orden o usamos más contadores, simplemente no es posible hacerlo con un autómata de pila.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Por ejemplo, intentemos representar  $\{a^n b^n c^n | n \geq 0\}$  con un AP:

- Por cada  $a$  ponemos un contador en la pila
- Por cada  $b$  quitamos un contador de la pila
- No podemos contar las  $c$ s.

O si cambiamos el orden o usamos más contadores, simplemente no es posible hacerlo con un autómata de pila.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Por ejemplo, intentemos representar  $\{a^n b^n c^n | n \geq 0\}$  con un AP:

- Por cada  $a$  ponemos un contador en la pila
- Por cada  $b$  quitamos un contador de la pila
- No podemos contar las  $c$ s.

O si cambiamos el orden o usamos más contadores, simplemente no es posible hacerlo con un autómata de pila.

# Las máquinas y sus lenguajes

Lo que hemos visto hasta ahora

Por ejemplo, intentemos representar  $\{a^n b^n c^n | n \geq 0\}$  con un AP:

- Por cada  $a$  ponemos un contador en la pila
- Por cada  $b$  quitamos un contador de la pila
- No podemos contar las  $c$ s.

O si cambiamos el orden o usamos más contadores, simplemente no es posible hacerlo con un autómata de pila.

# Un mejor uso de la memoria

## Turing y su máquina

Claramente el problema que tenemos es de memoria: no tenemos cómo recordar las *cs*.

En un AP sólo leemos el tope de la pila. ¿Qué pasaría si leyéramos cualquier parte de la misma?

¿Cuál es el límite de la pila? ¿Cuántos símbolos puede guardar?

# Un mejor uso de la memoria

## Turing y su máquina

Claramente el problema que tenemos es de memoria: no tenemos cómo recordar las *cs*.

En un AP sólo leemos el tope de la pila. ¿Qué pasaría si leyéramos cualquier parte de la misma?

¿Cuál es el límite de la pila? ¿Cuántos símbolos puede guardar?

# Un mejor uso de la memoria

## Turing y su máquina

Claramente el problema que tenemos es de memoria: no tenemos cómo recordar las *cs*.

En un AP sólo leemos el tope de la pila. ¿Qué pasaría si leyéramos cualquier parte de la misma?

¿Cuál es el límite de la pila? ¿Cuántos símbolos puede guardar?

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.



# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Una nueva máquina

## Turing y su máquina

Una **máquina de Turing** (MT) soluciona estos problemas, *uniendo* el *input* y la memoria. Ahora nuestro Autómata Reloaded tiene los siguientes elementos:

- Un **conjunto de estados de control** que es **finito**
- Una **cinta infinita** que utiliza como su **memoria**
- Un **cabezal en la cinta** que puede **leer** y **escribir** en una celda a la vez.

En cada *paso* del cómputo, la máquina

- Escribe un símbolo en la celda donde está el cabezal,
- cambia de estado, y
- mueve el cabezal.

# Terminología y lenguaje

## Turing y su máquina

Sea  $M$  una máquina de Turing:

- $M$  **acepta** una palabra  $w$  si entra al estado de aceptación cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  **rechaza** una palabra  $w$  si entra al estado de rechazo cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  entra en **loop** con una palabra  $w$  si al leer  $w$  no entra ni al estado de aceptación ni al de rechazo. En este caso,  $M$  **no termina**.

¿Qué puede pasar si *no se acepta*? ¿Qué puede pasar si *no se rechaza*?



# Terminología y lenguaje

## Turing y su máquina

Sea  $M$  una máquina de Turing:

- $M$  **acepta** una palabra  $w$  si entra al estado de aceptación cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  **rechaza** una palabra  $w$  si entra al estado de rechazo cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  entra en **loop** con una palabra  $w$  si al leer  $w$  no entra ni al estado de aceptación ni al de rechazo. En este caso,  $M$  **no termina**.

¿Qué puede pasar si *no se acepta*? ¿Qué puede pasar si *no se rechaza*?

# Terminología y lenguaje

## Turing y su máquina

Sea  $M$  una máquina de Turing:

- $M$  **acepta** una palabra  $w$  si entra al estado de aceptación cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  **rechaza** una palabra  $w$  si entra al estado de rechazo cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  entra en **loop** con una palabra  $w$  si al leer  $w$  no entra ni al estado de aceptación ni al de rechazo. En este caso,  $M$  **no termina**.

¿Qué puede pasar si *no se acepta*? ¿Qué puede pasar si *no se rechaza*?

# Terminología y lenguaje

## Turing y su máquina

Sea  $M$  una máquina de Turing:

- $M$  **acepta** una palabra  $w$  si entra al estado de aceptación cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  **rechaza** una palabra  $w$  si entra al estado de rechazo cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  entra en **loop** con una palabra  $w$  si al leer  $w$  no entra ni al estado de aceptación ni al de rechazo. En este caso,  $M$  **no termina**.

¿Qué puede pasar si *no se acepta*? ¿Qué puede pasar si *no se rechaza*?

# Terminología y lenguaje

## Turing y su máquina

Sea  $M$  una máquina de Turing:

- $M$  **acepta** una palabra  $w$  si entra al estado de aceptación cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  **rechaza** una palabra  $w$  si entra al estado de rechazo cuando se lee  $w$ . En este caso,  $M$  **termina**.
- $M$  entra en **loop** con una palabra  $w$  si al leer  $w$  no entra ni al estado de aceptación ni al de rechazo. En este caso,  $M$  **no termina**.

¿Qué puede pasar si *no se acepta*? ¿Qué puede pasar si *no se rechaza*?

# Terminología y lenguaje

## Turing y su máquina

El lenguaje de una máquina de Turing  $M$ , denotado con  $L(M)$  o  $\mathcal{L}(M)$  es el conjunto de todas las palabras que  $M$  acepta:

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

Un lenguaje es **reconocible** si y solo si es el lenguaje de alguna máquina de Turing.

¿Existen más lenguajes?

# Terminología y lenguaje

## Turing y su máquina

El lenguaje de una máquina de Turing  $M$ , denotado con  $L(M)$  o  $\mathcal{L}(M)$  es el conjunto de todas las palabras que  $M$  acepta:

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

Un lenguaje es **reconocible** si y solo si es el lenguaje de alguna máquina de Turing.

¿Existen más lenguajes?

# Terminología y lenguaje

## Turing y su máquina

El lenguaje de una máquina de Turing  $M$ , denotado con  $L(M)$  o  $\mathcal{L}(M)$  es el conjunto de todas las palabras que  $M$  acepta:

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

Un lenguaje es **reconocible** si y solo si es el lenguaje de alguna máquina de Turing.

¿Existen más lenguajes?

# Turing y el Entscheidungsproblem

## Turing y su máquina

En 1928, David Hilbert y Wilhelm Ackermann—ambos matemáticos alemanes—propusieron un problema que llevó cerca de 8 años resolver:

### El Entscheidungsproblem

¿Existe algún algoritmo que tome como *input* una proposición de lógica de primer orden, y diga si es o no universalmente válido a partir de sus axiomas?



# Turing y el Entscheidungsproblem

## Turing y su máquina

En 1928, David Hilbert y Wilhelm Ackermann—ambos matemáticos alemanes—propusieron un problema que llevó cerca de 8 años resolver:

### El Entscheidungsproblem

¿Existe algún algoritmo que tome como *input* una proposición de lógica de primer orden, y diga si es o no universalmente válido a partir de sus axiomas?

# Turing y el Entscheidungsproblem

## Turing y su máquina

En 1936, Alonzo Church—matemático estadounidense—lanza una publicación definiendo el concepto de “calculabilidad efectiva” mediante el cálculo lambda. Esto da nombre a las funciones lambda en lenguajes de programación.

El mismo año, Alan Turing—matemático inglés—publica su trabajo *On Computable Numbers, with an application to the Entscheidungsproblem*<sup>1</sup>, donde propone la MT y delimita todo aquello que puede ser “computable”.

---

<sup>1</sup>Es *altamente* probable que tengan que leer el paper para el examen...

# Turing y el Entscheidungsproblem

## Turing y su máquina

En 1936, Alonzo Church—matemático estadounidense—lanza una publicación definiendo el concepto de “calculabilidad efectiva” mediante el cálculo lambda. Esto da nombre a las funciones lambda en lenguajes de programación.

El mismo año, Alan Turing—matemático inglés—publica su trabajo *On Computable Numbers, with an application to the Entscheidungsproblem*<sup>1</sup>, donde propone la MT y delimita todo aquello que puede ser “computable”.

---

<sup>1</sup>Es *altamente* probable que tengan que leer el paper para el examen...

# Turing y el Entscheidungsproblem

## Turing y su máquina

En 1936, Alonzo Church—matemático estadounidense—lanza una publicación definiendo el concepto de “calculabilidad efectiva” mediante el cálculo lambda. Esto da nombre a las funciones lambda en lenguajes de programación.

El mismo año, Alan Turing—matemático inglés—publica su trabajo *On Computable Numbers, with an application to the Entscheidungsproblem*<sup>1</sup>, donde propone la MT y delimita todo aquello que puede ser “computable”.

---

<sup>1</sup>Es *altamente* probable que tengan que leer el paper para el examen...

# ¿Qué puede hacer la máquina de Turing?

## Turing y su máquina

Aunque dos personas dieron la misma respuesta al *Entscheidungsproblem*—no existe algoritmo alguno para responder—usualmente nos quedamos con la versión de Turing, pues es más contundente al definir lo que es un algoritmo:

### Teorema 1

*Si lo puede hacer una máquina de Turing, entonces hay un algoritmo para ello.*

Lo que significa que la MT puede hacer operaciones aritméticas, trabajar con listas, aceptar palabras y hacer **funciones**.

# ¿Qué puede hacer la máquina de Turing?

## Turing y su máquina

Aunque dos personas dieron la misma respuesta al *Entscheidungsproblem*—no existe algoritmo alguno para responder—usualmente nos quedamos con la versión de Turing, pues es más contundente al definir lo que es un algoritmo:

### Teorema 1

*Si lo puede hacer una máquina de Turing, entonces hay un algoritmo para ello.*

Lo que significa que la MT puede hacer operaciones aritméticas, trabajar con listas, aceptar palabras y hacer **funciones**.

# ¿Qué puede hacer la máquina de Turing?

## Turing y su máquina

Aunque dos personas dieron la misma respuesta al *Entscheidungsproblem*—no existe algoritmo alguno para responder—usualmente nos quedamos con la versión de Turing, pues es más contundente al definir lo que es un algoritmo:

### Teorema 1

*Si lo puede hacer una máquina de Turing, entonces hay un algoritmo para ello.*

Lo que significa que la MT puede hacer operaciones aritméticas, trabajar con listas, aceptar palabras y hacer **funciones**.

# ¿Qué puede hacer la máquina de Turing?

## Turing y su máquina

Aunque dos personas dieron la misma respuesta al *Entscheidungsproblem*—no existe algoritmo alguno para responder—usualmente nos quedamos con la versión de Turing, pues es más contundente al definir lo que es un algoritmo:

### Teorema 1

*Si lo puede hacer una máquina de Turing, entonces hay un algoritmo para ello.*

Lo que significa que la MT puede hacer operaciones aritméticas, trabajar con listas, aceptar palabras y hacer **funciones**.



# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.

# Definición formal

## Formalidades y ejemplos

De nuevo, hay muchas maneras de abordar la definición formal. Por ello, nos enfocaremos en usar lo más parecido a lo que hayamos usado.

Primero, recordemos que una MT *para reconocer palabras de un lenguaje* tiene

- Un conjunto de estados.
- Un input.
- Una cinta que usa como memoria.
- Un estado inicial.
- Un estado de aceptación.
- Un estado de rechazo.
- Una función de transición que define cómo pasar de un estado a otro, dadas ciertas condiciones.



# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.

# Definición formal

## Formalidades y ejemplos

### Definición formal de una máquina de Turing

Una máquina de Turing  $M$  para reconocer palabras de un lenguaje es una tupla de la forma  $M = (Q, \Sigma, \Gamma, \delta, q, a, r)$  donde:

- $Q$  es un conjunto finito de **estados**,
- $\Sigma$  es el **alfabeto del input**, donde  $\square \notin \Sigma$ ,
- $\Gamma$  es el **alfabeto de la cinta**, donde  $\square \in \Gamma$  y  $\Sigma \subseteq \Gamma$ ,
- $q \in Q$  es el **estado inicial**,
- $a \in Q$  es el **estado de aceptación**,
- $r \in Q$  es el **estado de rechazo**, y
- $\delta$  es la función de transición.



# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un  $1$  en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un  $1$  en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_01 \rightarrow q_11R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha (*Right*)

# Definición formal

## Formalidades y ejemplos

La función de transición  $\delta$  es una función de la forma:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$$

### Ejemplo

Podemos escribir  $q_0 1 \rightarrow q_1 1R$  que significaría que:

- Estando en el estado  $q_0$ ,
- y al leer un 1 en la cinta

Entonces la MT

- cambia al estado  $q_1$ ,
- escribe un 1 en la celda actual, y
- mueve el cabezal hacia la derecha ( $R$ ight)

# Definición formal

## Formalidades y ejemplos

La verdad es que...

...es más fácil pensarlo como **flechas** que van **de un estado**  $q_0$  a **otro estado**  $q_1$  de la forma  $x \rightarrow y, D$ — si **lees**  $x$ , **escribes**  $y$  y te **mueves hacia**  $D$ :





# Definición formal

## Formalidades y ejemplos

La verdad es que...

...es más fácil pensarlo como **flechas** que van **de un estado  $q_0$  a otro estado  $q_1$**  de la forma  $x \rightarrow y, D$  — si **lees  $x$** , **escribes  $y$**  y te **mueves hacia  $D$** :

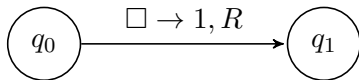


# Definición formal

## Formalidades y ejemplos

La verdad es que...

...es más fácil pensarlo como **flechas** que van **de un estado**  $q_0$  a **otro estado**  $q_1$  de la forma  $x \rightarrow y, D$  — si **lees**  $x$ , **escribes**  $y$  y te **mueves hacia**  $D$ :



# Configuración inicial

## Formalidades y ejemplos

Antes de comenzar el cómputo, la máquina de Turing debe estar en una configuración específica. En esta configuración:

- 1 La cinta está vacía, es decir que tiene sólo símbolos  $\square$  en ella.
- 2 La palabra de entrada **se copia** a algún lugar en la cinta.
- 3 El cabezal se mueve al **inicio** de la palabra de entrada.

# Configuración inicial

## Formalidades y ejemplos

Antes de comenzar el cómputo, la máquina de Turing debe estar en una configuración específica. En esta configuración:

- 1 La cinta está vacía, es decir que tiene sólo símbolos  $\square$  en ella.
- 2 La palabra de entrada **se copia** a algún lugar en la cinta.
- 3 El cabezal se mueve al **inicio** de la palabra de entrada.

# Configuración inicial

## Formalidades y ejemplos

Antes de comenzar el cómputo, la máquina de Turing debe estar en una configuración específica. En esta configuración:

- 1 La cinta está vacía, es decir que tiene sólo símbolos  $\square$  en ella.
- 2 La palabra de entrada **se copia** a algún lugar en la cinta.
- 3 El cabezal se mueve al **inicio** de la palabra de entrada.

# ¿Y los ejemplos?

Se irán agregando con el tiempo.

Mientras, hay que revisar otros enfoques:

- 1 Slides 26 de Ramón Brena/Santiago Conant.
- 2 Slides 18, 19, 20 y 21 de Keith Schwarz.
- 3 Capítulos 4 de Maheshwari y Smid.