

# Autómatas Finitos Deterministas

## Implementación de Métodos Computacionales (TC2037)

M.C. Xavier Sánchez Díaz  
mail@tec.mx



# Tabla de contenidos

1 DFA: Las bases

2 Diseñando un DFA

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.

# Definición

## DFA

### Definición 1

Un **autómata finito determinista** (DFA por sus siglas en inglés) es una **quíntupla** de la forma

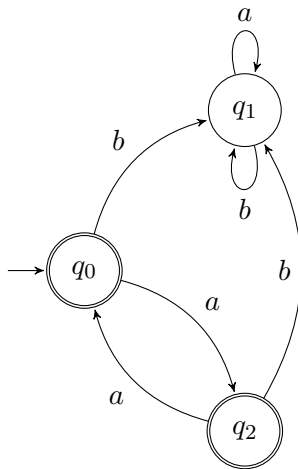
$$M = (Q, \Sigma, \delta, q, F)$$

- $Q$  es un **conjunto de estados** que es finito,
- $\Sigma$  es el **alfabeto** aceptado,
- $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
- $q \in Q$  es el **estado inicial**,
- $F \subseteq Q$  es un **conjunto de estados finales**.



# Ejemplo

DFA



# Lenguaje aceptado

## DFA

Un DFA **acepta** una palabra si al consumir todos sus caracteres llega a uno de sus estados finales (representados con un doble círculo en el diagrama).

El **lenguaje aceptado** por una máquina  $M$  es el conjunto de palabras aceptadas por dicha máquina.

# Lenguaje aceptado

## DFA

Un DFA **acepta** una palabra si al consumir todos sus caracteres llega a uno de sus estados finales (representados con un doble círculo en el diagrama).

El **lenguaje aceptado** por una máquina  $M$  es el conjunto de palabras aceptadas por dicha máquina.

# Corrección y Completez

## Diseñando un DFA

En ocasiones, el problema radica en construir un DFA a partir del lenguaje que debe aceptar.

El autómata diseñado debe ser **correcto** y **completo**.

# Corrección y Completez

## Diseñando un DFA

En ocasiones, el problema radica en construir un DFA a partir del lenguaje que debe aceptar.

El autómata diseñado debe ser **correcto** y **completo**.

# Corrección y Completez

## Diseñando un DFA

**Corrección:** se refiere a la cualidad de un sistema de ser **correcto**. En el caso de un autómata, se refiere a que acepta **sólo** las palabras que pertenecen al lenguaje.

**Completez:** se refiere a la cualidad de un sistema de ser **completo**. En el caso de un autómata, se refiere a que acepta **todas** las palabras que pertenecen al lenguaje.

# Corrección y Completez

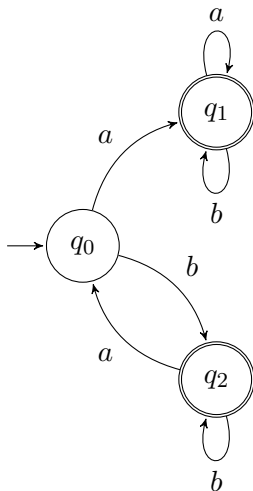
## Diseñando un DFA

**Corrección:** se refiere a la cualidad de un sistema de ser **correcto**. En el caso de un autómata, se refiere a que acepta **sólo** las palabras que pertenecen al lenguaje.

**Completez:** se refiere a la cualidad de un sistema de ser **completo**. En el caso de un autómata, se refiere a que acepta **todas** las palabras que pertenecen al lenguaje.

## Diseñando un DFA

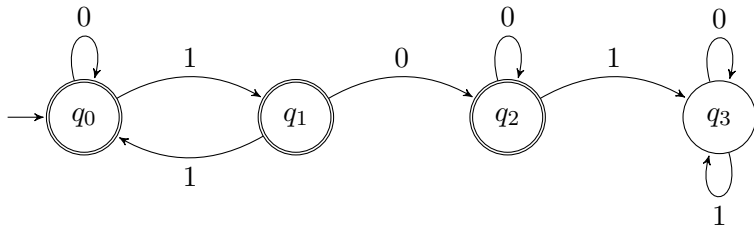
DFA para lenguaje en  $\{a, b\}$  de palabras que no tienen varias  $a$ s seguidas.





## Diseñando un DFA

DFA para lenguaje en  $\{0, 1\}$  de palabras que no contienen 101.



# Principios de diseño

## Diseñando un DFA

¿Qué alternativas tenemos?

- **Prueba y error.** Inconveniente dado a que hay demasiados casos a considerar.
- **Sistemático.** Por condiciones de estados.
- **Conjuntos de estados.** Agrupando estados similares.

# Principios de diseño

## Diseñando un DFA

¿Qué alternativas tenemos?

- **Prueba y error.** Inconveniente dado a que hay demasiados casos a considerar.
- **Sistemático.** Por condiciones de estados.
- **Conjuntos de estados.** Agrupando estados similares.

# Principios de diseño

## Diseñando un DFA

¿Qué alternativas tenemos?

- **Prueba y error.** Inconveniente dado a que hay demasiados casos a considerar.
- **Sistemático.** Por condiciones de estados.
- **Conjuntos de estados.** Agrupando estados similares.

# Principios de diseño

## Diseñando un DFA

¿Qué alternativas tenemos?

- **Prueba y error.** Inconveniente dado a que hay demasiados casos a considerar.
- **Sistemático.** Por condiciones de estados.
- **Conjuntos de estados.** Agrupando estados similares.

# Diseño sistemático

## Diseñando un DFA

- 1 Determinar de manera explícita qué condición necesita cada uno de los estados.
  - ▶ Los estados deben ser mutuamente excluyentes
  - ▶ Los estados deben ser exhaustivos (*comprehensive*)
- 2 Proponer las transiciones que permiten pasar de un estado a otro.

# Diseño sistemático

## Diseñando un DFA

- 1 Determinar de manera explícita qué condición necesita cada uno de los estados.
  - ▶ Los estados deben ser mutuamente excluyentes
  - ▶ Los estados deben ser exhaustivos (*comprehensive*)
- 2 Proponer las transiciones que permiten pasar de un estado a otro.

# Diseño sistemático

## Diseñando un DFA

- 1 Determinar de manera explícita qué condición necesita cada uno de los estados.
  - ▶ Los estados deben ser mutuamente excluyentes
  - ▶ Los estados deben ser exhaustivos (*comprehensive*)
- 2 Proponer las transiciones que permiten pasar de un estado a otro.



# Diseño sistemático

## Diseñando un DFA

- 1 Determinar de manera explícita qué condición necesita cada uno de los estados.
  - ▶ Los estados deben ser mutuamente excluyentes
  - ▶ Los estados deben ser exhaustivos (*comprehensive*)
- 2 Proponer las transiciones que permiten pasar de un estado a otro.

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .



# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.

Condiciones a recordar:

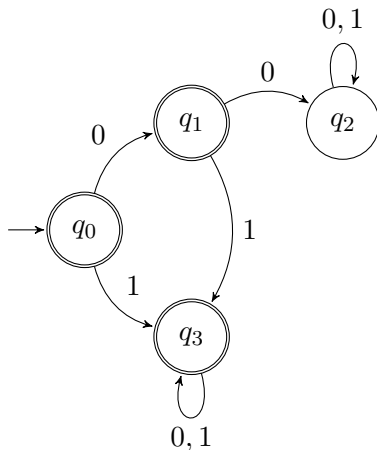
- $q_0$ : no se han recibido caracteres.
- $q_1$ : empieza con un solo 0.
- $q_2$ : empieza con dos o más 0s.
- $q_3$ : no empieza con dos 0s.

Estados finales:  $q_0, q_1, q_3$ .

# Ejemplo de diseño sistemático

## Diseñando un DFA

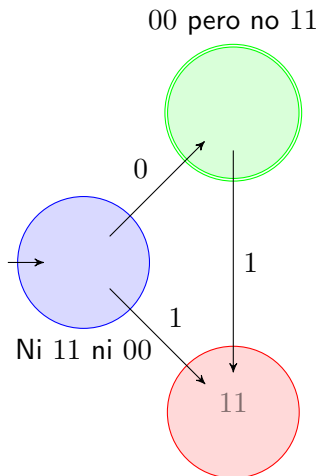
Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **no comienzan con 00**.



# Diseño por conjuntos de estados

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **tengan** 00 **pero no** 11.



# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).



# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Subdivisión: diseño por conjuntos de estados

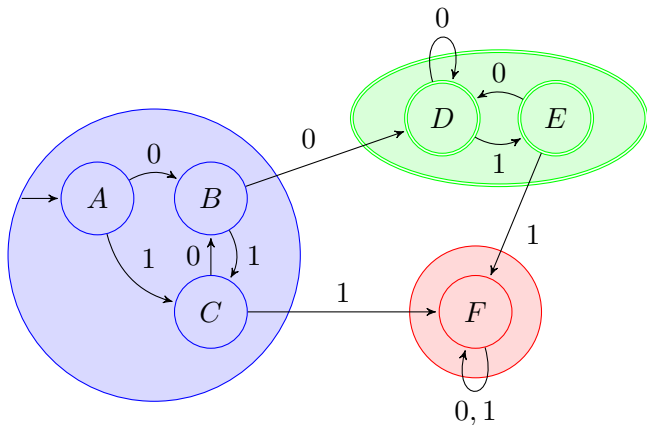
## Diseñando un DFA

- Los caracteres alimentados hasta el momento no contienen ni 00 ni 11:
  - ▶  $A$  — no se han recibido caracteres (estado inicial).
  - ▶  $B$  — se recibió el primer 0 de 00.
  - ▶  $C$  — se recibió el primer 1 de 11.
- Contienen 00 pero no 11:
  - ▶  $D$  — se recibió otro 0.
  - ▶  $E$  — se recibió el primer 1 de 11.
- Contienen 11:
  - ▶ No importa nada más (estado *infierno*).

# Diseño por conjuntos de estados

## Diseñando un DFA

Un DFA que acepte exactamente el lenguaje en el alfabeto  $\{0, 1\}$  de palabras que **tengan 00 pero no 11**.



# Diseño por complemento

## Diseñando un DFA

Si  $M$  es un autómata determinista que acepta un lenguaje regular  $L$ , para construir un autómata  $M^c$  que acepte el lenguaje  $L^c$ , basta con **intercambiar** los estados finales por estados no finales, y viceversa.

Cuando una palabra se rechaza en  $M$ , entonces se aceptará en  $M^c$ , y viceversa.

# Diseño por complemento

## Diseñando un DFA

Si  $M$  es un autómata determinista que acepta un lenguaje regular  $L$ , para construir un autómata  $M^c$  que acepte el lenguaje  $L^c$ , basta con **intercambiar** los estados finales por estados no finales, y viceversa.

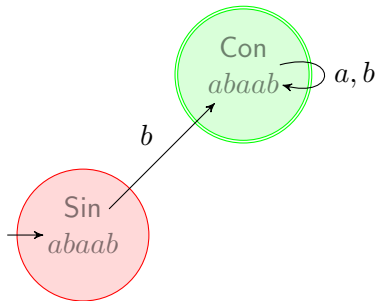
Cuando una palabra se rechaza en  $M$ , entonces se aceptará en  $M^c$ , y viceversa.



# Diseño por complemento

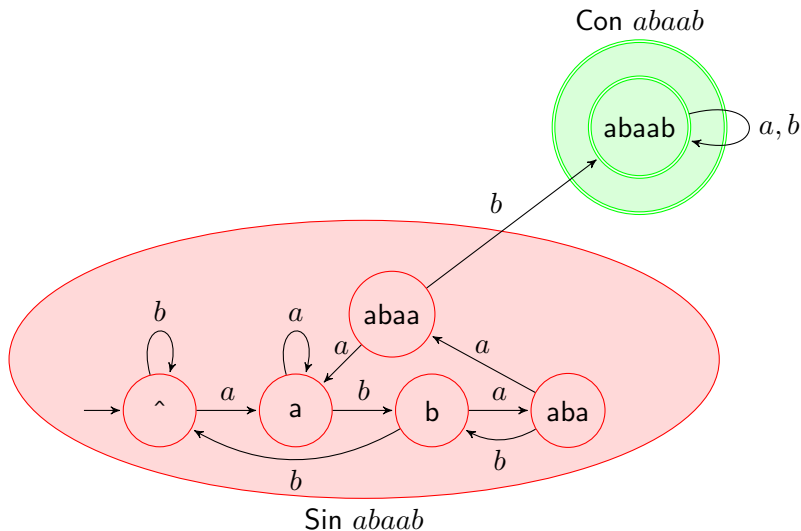
## Diseñando un DFA

Obtener un DFA para el lenguaje en  $\{a, b\}^*$  de las palabras que **no contienen la sub-cadena abaab**.



# Diseño por complemento

Diseñando un DFA



# Diseño por complemento

Diseñando un DFA

