Autómatas Finitos

Introducción a la Lógica y la Computación (3era Parte)

Docentes: Badano, Bustos, Costamagna, Tellechea, Zigaran

Año 2024

Autómatas Finitos (AF)

- Es un modelo computacional basado en la idea de **máquina automática secuencial** y cuyo poder computacional caracteriza a los lenguajes regulares.

Autómatas Finitos (AF)

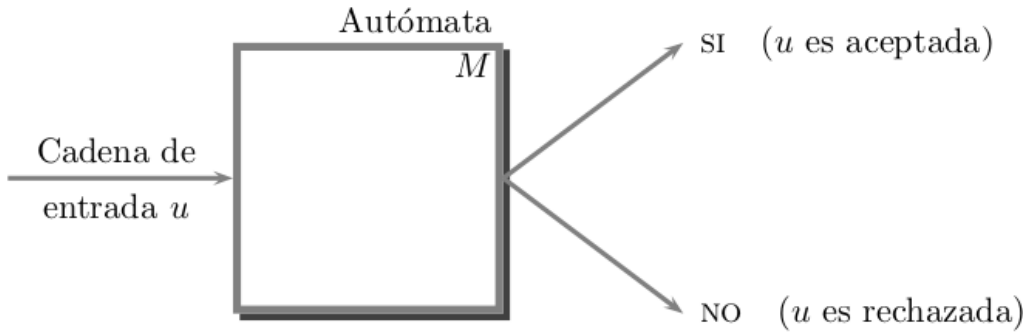
- Es un modelo computacional basado en la idea de **máquina automática secuencial** y cuyo poder computacional caracteriza a los lenguajes regulares.
- Primero, estudiaremos los autómatas finitos en si mismos, es decir, su funcionamiento, sus tipos y sus propiedades.

Autómatas Finitos (AF)

- Es un modelo computacional basado en la idea de **máquina automática secuencial** y cuyo poder computacional caracteriza a los lenguajes regulares.
- Primero, estudiaremos los autómatas finitos en si mismos, es decir, su funcionamiento, sus tipos y sus propiedades.
- Para luego, finalmente probar que es la máquina secuencial caracterizadora de los lenguajes regulares.

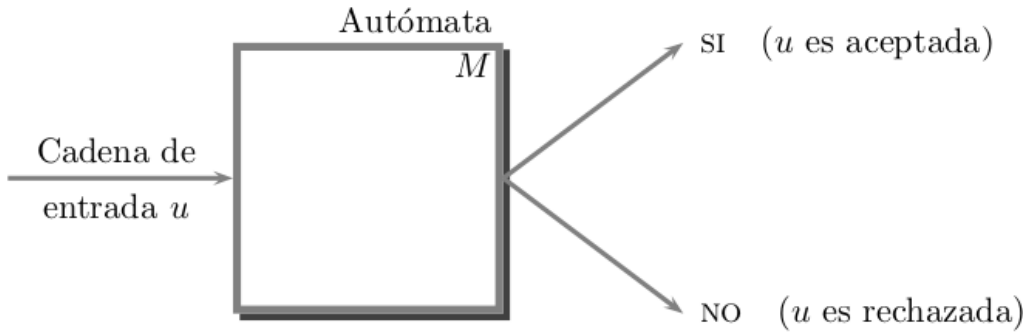
Qué computa un AF?

- Toma una cadena como input y luego de procesar todos sus símbolos devuelve si ésta es **“aceptada”** o **“rechazada”**.



Qué computa un AF?

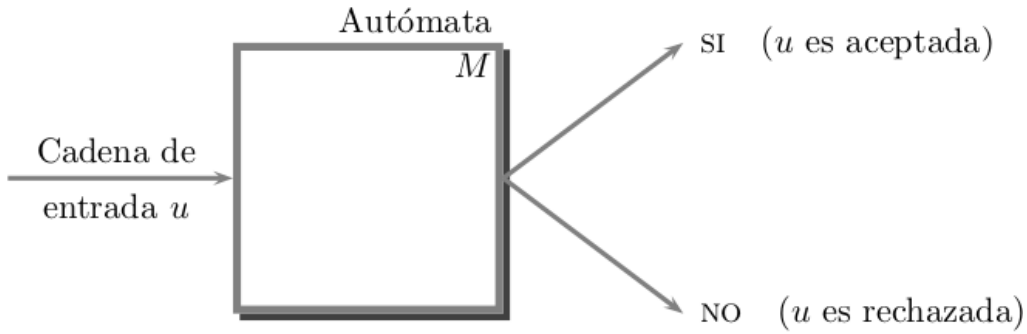
- Toma una cadena como input y luego de procesar todos sus símbolos devuelve si ésta es **"aceptada"** o **"rechazada"**.



- Estudiaremos 3 tipos de autómatas finitos:
  - Determinísticos (**AFD**)
  - No-Determinísticos (**AFN**)
  - No-Determinísticos con transiciones espontáneas (**AFN**ϵ)

Qué computa un AF?

- Toma una cadena como input y luego de procesar todos sus símbolos devuelve si ésta es **"aceptada"** o **"rechazada"**.



- Estudiaremos 3 tipos de autómatas finitos:
  - Determinísticos (**AFD**)
  - No-Determinísticos (**AFN**)
  - No-Determinísticos con transiciones espontáneas (**AFN**ϵ)
- Pero finalmente, todos **computacionalmente equivalentes** entre si.

Cómo funciona un AF?

- La máquina tiene un conjunto de **estados** posibles pero en todo momento se encuentra exactamente en uno de ellos.

Cómo funciona un AF?

- La máquina tiene un conjunto de **estados** posibles pero en todo momento se encuentra exactamente en uno de ellos.
- Tiene una cinta con celdas, llamada **cinta de entrada**, donde en cada celda se aloja un símbolo de la cadena a procesar, y un cabezal (de solo lectura) que apunta a la siguiente celda a procesar.

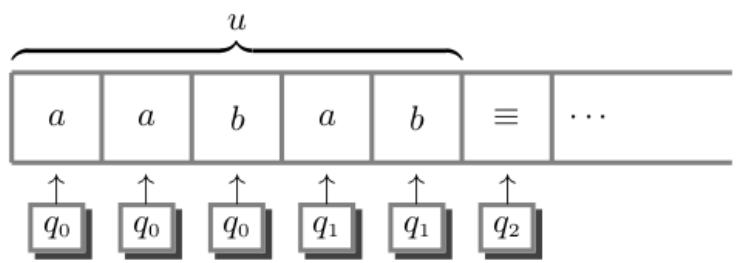
Cómo funciona un AF?

- La máquina tiene un conjunto de **estados** posibles pero en todo momento se encuentra exactamente en uno de ellos.
- Tiene una cinta con celdas, llamada **cinta de entrada**, donde en cada celda se aloja un símbolo de la cadena a procesar, y un cabezal (de solo lectura) que apunta a la siguiente celda a procesar.
- Cada vez que el cabezal lee un símbolo de la cinta, la máquina ejecuta un cambio de estado, llamado **transición**. Así sucesivamente, hasta que el cabezal lee el último símbolo de la cadena a procesar.

Cómo funciona un AF?

- La máquina tiene un conjunto de **estados** posibles pero en todo momento se encuentra exactamente en uno de ellos.
- Tiene una cinta con celdas, llamada **cinta de entrada**, donde en cada celda se aloja un símbolo de la cadena a procesar, y un cabezal (de solo lectura) que apunta a la siguiente celda a procesar.
- Cada vez que el cabezal lee un símbolo de la cinta, la máquina ejecuta un cambio de estado, llamado **transición**. Así sucesivamente, hasta que el cabezal lee el último símbolo de la cadena a procesar.
- El estado al que arriba la máquina, luego de procesar toda la cadena, definirá si la cadena es **aceptada o rechazada**.

$u = aabab.$



Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\delta : Q \times \Sigma \to Q$, *llamada función de transición, tal que,* $\delta(q, a) = q'$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata transiciona al estado* q'.

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\delta : Q \times \Sigma \to Q$, *llamada función de transición, tal que,* $\delta(q, a) = q'$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata transiciona al estado* q'.

Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M, *si el estado al que arriba* M, *luego de procesar todos los símbolos de la cadena* α, *partiendo desde el estado inicial* q_0, *es un estado de aceptación.*

Definición Autómata Finito Determinístico

Definición (AFD)

Un **Automata Finito Deterministico (AFD)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\delta : Q \times \Sigma \to Q$, *llamada función de transición, tal que,* $\delta(q, a) = q'$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata transiciona al estado* q'.

Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M, *si el estado al que arriba* M, *luego de procesar todos los símbolos de la cadena* α, *partiendo desde el estado inicial* q_0, *es un estado de aceptación.*

$$AFD^{\Sigma} = \{M : M \text{ es un AFD con alfabeto de entrada } \Sigma\}$$

Ejemplo de un AFD

Sea $M_1 = (\Sigma, Q, q_0, F, \delta)$ con $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2\}$, $F = \{q_0, q_2\}$ y δ definida de la siguiente manera:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

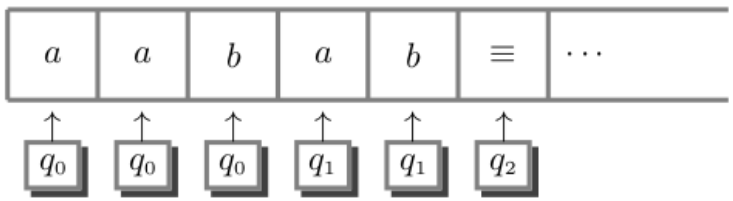
Ejemplo de un AFD

Sea $M_1 = (\Sigma, Q, q_0, F, \delta)$ con $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2\}$, $F = \{q_0, q_2\}$ y δ definida de la siguiente manera:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

La cadena $\alpha_1 =$ "*aabab*" es **aceptada** por M_1, porque luego de procesar todos los símblos de la cadena, el autómata arriba al estado $q_2 \in F$:



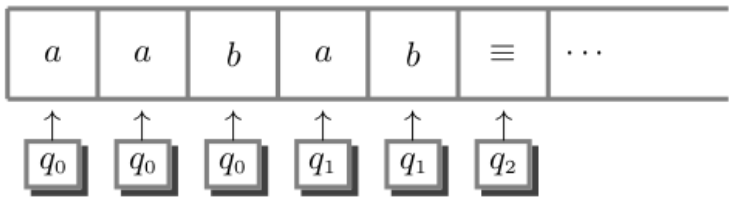
Ejemplo de un AFD

Sea $M_1 = (\Sigma, Q, q_0, F, \delta)$ con $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2\}$, $F = \{q_0, q_2\}$ y δ definida de la siguiente manera:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

La cadena $\alpha_1 =$ "*aabab*" es **aceptada** por M_1, porque luego de procesar todos los símblos de la cadena, el autómata arriba al estado $q_2 \in F$:



Mientras que la cadena $\alpha_2 =$ "*ababa*" es **rechazada** por M_1, pues el autómata arriba al estado $q_1 \notin F$:

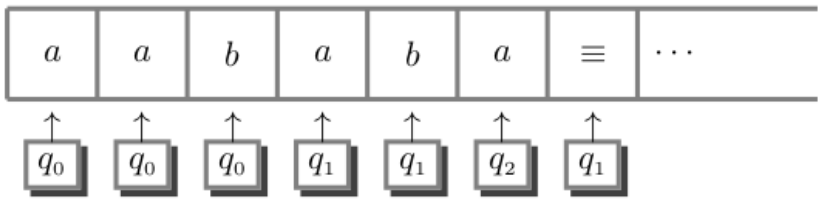


Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

- Los nodos son los estados del AF.

Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

- Los nodos son los estados del AF.
- El nodo que se corresponde con el estado inicial se marca con una flecha pequeña a su lado (por convención q_0).

Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

- Los nodos son los estados del AF.
- El nodo que se corresponde con el estado inicial se marca con una flecha pequeña a su lado (por convención q_0).
- Los nodos que se corresponden con los estados de aceptación se marcarán con un doble círculo.

Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

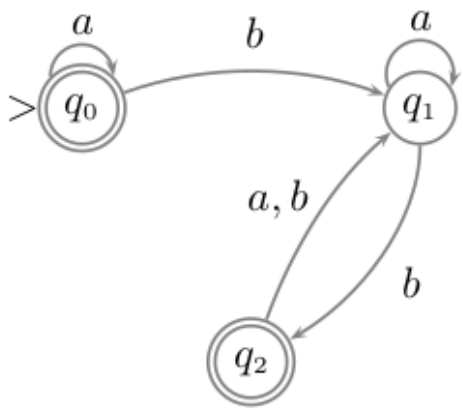
- Los nodos son los estados del AF.
- El nodo que se corresponde con el estado inicial se marca con una flecha pequeña a su lado (por convención q_0).
- Los nodos que se corresponden con los estados de aceptación se marcarán con un doble círculo.
- Existe una flecha desde el nodo "q" al nodo "q'", etiquetada con un símbolo "a", siempre que $\delta(q, a) = q'$.

Diagrama de Transición de un AF

Representaremos un AF mediante un sistema de transición etiquetado, o sea, un grafo donde:

- Los nodos son los estados del AF.
- El nodo que se corresponde con el estado inicial se marca con una flecha pequeña a su lado (por convención q_0).
- Los nodos que se corresponden con los estados de aceptación se marcarán con un doble círculo.
- Existe una flecha desde el nodo "q" al nodo "q'", etiquetada con un símbolo "a", siempre que $\delta(q, a) = q'$.

Por ejemplo, para el autómata anterior M_1 su diagrama de transición sería:



Lenguaje Aceptado de un AFD

Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD, entonces definimos $\overrightarrow{\delta}(q, \alpha)$ como la función que devuelve el estado al que arriba M, luego de procesar la cadena α, partiendo del estado q:

Lenguaje Aceptado de un AFD

Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD, entonces definimos $\overrightarrow{\delta}(q, \alpha)$ como la función que devuelve el estado al que arriba M, luego de procesar la cadena α, partiendo del estado q:

$\overrightarrow{\delta} : Q \times \Sigma^* \to Q$
$\overrightarrow{\delta}(q, \epsilon) = q$
$\overrightarrow{\delta}(q, a) = \delta(q, a)$
$\overrightarrow{\delta}(q, \alpha' a) = \delta(\overrightarrow{\delta}(q, \alpha'), a)$

Lenguaje Aceptado de un AFD

Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD, entonces definimos $\overrightarrow{\delta}(q, \alpha)$ como la función que devuelve el estado al que arriba M, luego de procesar la cadena α, partiendo del estado q:

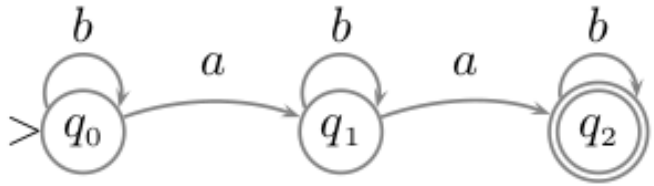
$\overrightarrow{\delta} : Q \times \Sigma^* \to Q$
$\overrightarrow{\delta}(q, \epsilon) = q$
$\overrightarrow{\delta}(q, a) = \delta(q, a)$
$\overrightarrow{\delta}(q, \alpha' a) = \delta(\overrightarrow{\delta}(q, \alpha'), a)$

Luego, definimos el **lenguaje aceptado** de M, denotado con $L(M)$, como el conjunto de todas cadenas aceptadas por M:

$$L(M) = \{\alpha \in \Sigma^* : \overrightarrow{\delta}(q_0, \alpha) \in F\}$$

Ejemplos de L(M)

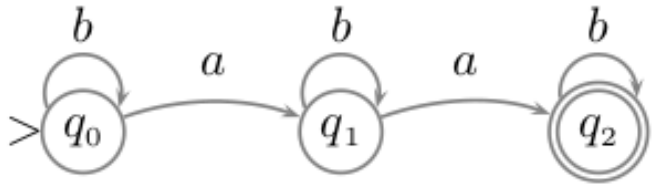
Por ejemplo, para el siguiente AFD M_1:



Tenemos que $L(M_1) = b^*ab^*ab^*$ (el lenguaje de todas las cadenas del alfabeto que tienen exactamente dos símbolos "a").

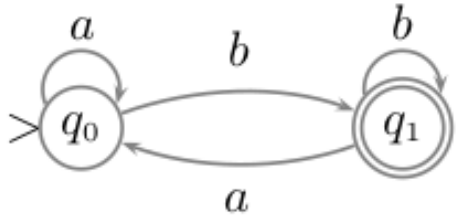
Ejemplos de L(M)

Por ejemplo, para el siguiente AFD M_1:



Tenemos que $L(M_1) = b^*ab^*ab^*$ (el lenguaje de todas las cadenas del alfabeto que tienen exactamente dos símbolos "a").

Para el AFD M_2:



Tenenemos que $L(M_2) = (a + b)^*b$ (el lenguaje de todas las cadenas del alfabeto que terminan con un símbolo "b").

Determinismo vs No-Determinismo

- **Determinismo**: dado un estado y un símbolo, existe un único estado al cual el autómata puede transicionar, por lo tanto, la traza de procesamiento de una cadena es única.

$$\delta(q, a) = q' \implies \overrightarrow{\delta}(q, \alpha) \text{ es un estado}$$

Determinismo vs No-Determinismo

- **Determinismo**: dado un estado y un símbolo, existe un único estado al cual el autómata puede transicionar, por lo tanto, la traza de procesamiento de una cadena es única.

$$\delta(q, a) = q' \implies \overrightarrow{\delta}(q, \alpha) \text{ es un estado}$$

´

- **No-Determinismo**: dado un estado y un símbolo, existen varios estados a los cuales el autómata puede transicionar, por lo tanto, existen varias trazas de procesamiento para una misma cadena.

$$\delta(q, a) = \{q1, \ldots, q_k\} \implies \overrightarrow{\delta}(q, \alpha) \text{ es un conjunto de estados.}$$

Determinismo vs No-Determinismo

- **Determinismo**: dado un estado y un símbolo, existe un único estado al cual el autómata puede transicionar, por lo tanto, la traza de procesamiento de una cadena es única.

$$\delta(q, a) = q' \implies \overrightarrow{\delta}(q, \alpha) \text{ es un estado}$$

´

- **No-Determinismo**: dado un estado y un símbolo, existen varios estados a los cuales el autómata puede transicionar, por lo tanto, existen varias trazas de procesamiento para una misma cadena.

$$\delta(q, a) = \{q1, \ldots, q_k\} \implies \overrightarrow{\delta}(q, \alpha) \text{ es un conjunto de estados.}$$

(Para evitar confusiones, denotaremos con Δ las funciones de transición no-deterministas, mientras que con δ las deterministas)

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, *llamada función de transición, tal que,* $\Delta(q, a) = \{q_1, \ldots, q_k\}$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata puede transicionar a cualquier estado* q_i.

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, *llamada función de transición, tal que,* $\Delta(q, a) = \{q_1, \ldots, q_k\}$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata puede transicionar a cualquier estado* q_i.

Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M, *si existe* **al menos una forma** *de procesar la cadena* α, *partiendo desde el estado inicial* q_0, *donde el estado al que arriba* M *es un estado de aceptación.*

Autómata Finito No-Determinístico (AFN)

Definición (AFN)

Un **Automata Finito No-Deterministico (AFN)** *es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

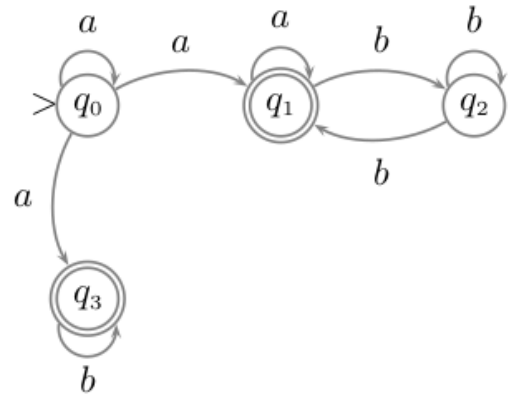
- Σ *es un alfabeto, llamado alfabeto de entrada.*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, *llamada función de transición, tal que,* $\Delta(q, a) = \{q_1, \ldots, q_k\}$ *significa que si el autómata se encuentra en el estado "q" y el cabezal lee un símbolo "a" en la cinta de entrada, entonces el autómata puede transicionar a cualquier estado* q_i.

Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M, *si existe* **al menos una forma** *de procesar la cadena* α, *partiendo desde el estado inicial* q_0, *donde el estado al que arriba* M *es un estado de aceptación.*

$$AFN^{\Sigma} = \{M : M \text{ es un AFN con alfabeto de entrada } \Sigma\}$$

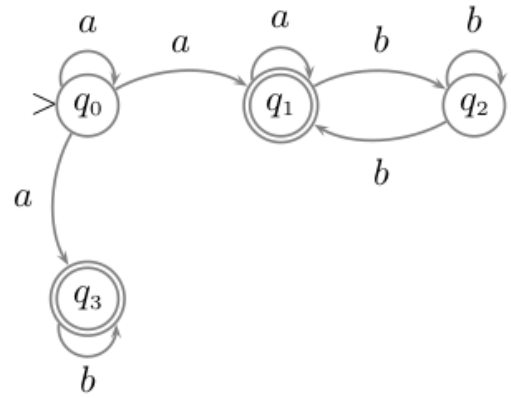
Ejemplo de AFN

Sea M_1 el siguiente AFN:

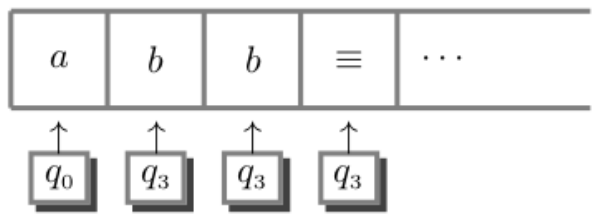


Ejemplo de AFN

Sea M_1 el siguiente AFN:

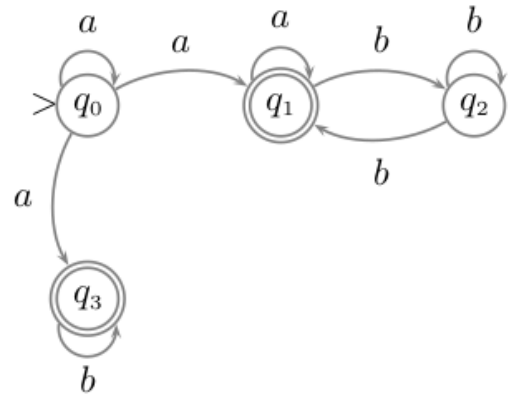


La cadena "abb" tiene una traza de procesamiento de aceptación:

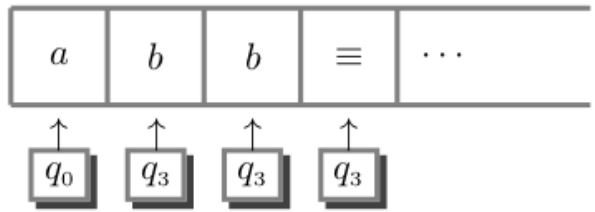


Ejemplo de AFN

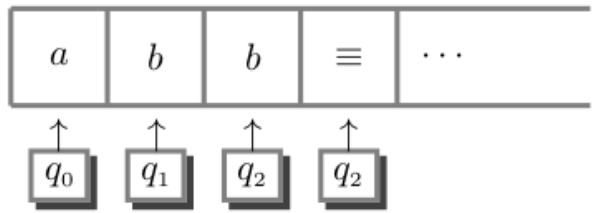
Sea M_1 el siguiente AFN:



La cadena "abb" tiene una traza de procesamiento de aceptación:



Y también, tiene una traza de procesamiento de rechazo:



Lenguaje Aceptado de un AFN

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN, primero extendemos la definición de Δ a un conjunto de estados $S \subseteq Q$:

$$\hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$$

Lenguaje Aceptado de un AFN

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN, primero extendemos la definición de Δ a un conjunto de estados $S \subseteq Q$:

$$\hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$$

Luego, definimos $\overrightarrow{\Delta}(q, a)$ como la función que devuelve el **conjunto de todos los estados posibles** al que puede arribar M luego de procesar la cadena α partiendo del estado q:

Lenguaje Aceptado de un AFN

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN, primero extendemos la definición de Δ a un conjunto de estados $S \subseteq Q$:

$$\hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$$

Luego, definimos $\overrightarrow{\Delta}(q, a)$ como la función que devuelve el **conjunto de todos los estados posibles** al que puede arribar M luego de procesar la cadena α partiendo del estado q:

$\overrightarrow{\Delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$
$\overrightarrow{\Delta}(q, \epsilon) = \{q\}$
$\overrightarrow{\Delta}(q, a) = \Delta(q, a)$
$\overrightarrow{\Delta}(q, \alpha' a) = \hat{\Delta}(\overrightarrow{\Delta}(q, \alpha'), a)$

Lenguaje Aceptado de un AFN

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN, primero extendemos la definición de Δ a un conjunto de estados $S \subseteq Q$:

$$\hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$$

Luego, definimos $\overrightarrow{\Delta}(q, a)$ como la función que devuelve el **conjunto de todos los estados posibles** al que puede arribar M luego de procesar la cadena α partiendo del estado q:

$$\overrightarrow{\Delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$$
$$\overrightarrow{\Delta}(q, \epsilon) = \{q\}$$
$$\overrightarrow{\Delta}(q, a) = \Delta(q, a)$$
$$\overrightarrow{\Delta}(q, \alpha' a) = \hat{\Delta}(\overrightarrow{\Delta}(q, \alpha'), a)$$

Para finalmente, definir el lenguaje aceptado de M como:

$$L(M) = \{\alpha \in \Sigma^* : \overrightarrow{\Delta}(q_0, \alpha) \cap F \neq \emptyset\}$$

Equivalencia entre AFD y AFN

Los AFD y los AFN son computacionalmente equivalentes.

Equivalencia entre AFD y AFN

Los AFD y los AFN son computacionalmente equivalentes.

Como toda equivalencia debemos probar una ida y una vuelta.

Equivalencia entre AFD y AFN

Los AFD y los AFN son computacionalmente equivalentes.

Como toda equivalencia debemos probar una ida y una vuelta.

Teorema ($AFD^{\Sigma} \equiv AFN^{\Sigma}$)

1. *(Ida) Si* $M \in AFD^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFD^{\Sigma}$ *tq* $L(M') = L(M)$.

Equivalencia entre AFD y AFN

Los AFD y los AFN son computacionalmente equivalentes.

Como toda equivalencia debemos probar una ida y una vuelta.

Teorema ($AFD^{\Sigma} \equiv AFN^{\Sigma}$)

1. *(Ida) Si* $M \in AFD^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFD^{\Sigma}$ *tq* $L(M') = L(M)$.

Demo 1. Trivial, pues un AFD puede ser visto como un AFN sin usar su no-determinismo (i.e. basta con definir $\Delta'(q, a) = \{\delta(q, a)\}$).

Equivalencia entre AFD y AFN

Los AFD y los AFN son computacionalmente equivalentes.

Como toda equivalencia debemos probar una ida y una vuelta.

Teorema ($AFD^{\Sigma} \equiv AFN^{\Sigma}$)

1. *(Ida) Si* $M \in AFD^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFD^{\Sigma}$ *tq* $L(M') = L(M)$.

Demo 1. Trivial, pues un AFD puede ser visto como un AFN sin usar su no-determinismo (i.e. basta con definir $\Delta'(q, a) = \{\delta(q, a)\}$).

Demo 2. Debemos probar que **siempre se puede determinizar un AFN preservando su lenguaje**. Y el truco consiste en que los estados del AFD son conjuntos de estados del AFN, de tal manera que si el AFD se encuentra en el estado $\{q1, \ldots, q_k\}$ luego de procesar un prefijo de la cadena, entonces el AFN se encuentra en algún q_i.

Sea $M = (\Sigma, Q, q_0, F, \Delta)$, entonces tomamos $M' = (\Sigma, Q', q_0', F', \delta')$ con:

Sea $M = (\Sigma, Q, q_0, F, \Delta)$, entonces tomamos $M' = (\Sigma, Q', q_0', F', \delta')$ con:

$Q' = \mathcal{P}(Q)$
$q_0' = \{q_0\}$
$F' = \{S \subseteq Q : S \cap F \neq \emptyset\}$
$\delta'(S, a) = \hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$

Sea $M = (\Sigma, Q, q_0, F, \Delta)$, entonces tomamos $M' = (\Sigma, Q', q_0', F', \delta')$ con:

$Q' = \mathcal{P}(Q)$
$q_0' = \{q_0\}$
$F' = \{S \subseteq Q : S \cap F \neq \emptyset\}$
$\delta'(S, a) = \hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$

Claramente, vale que $\overrightarrow{\delta'}(\{q_0\}, \alpha) = \overrightarrow{\Delta}(q_0, \alpha)$ para toda α, por definición de M'.

Sea $M = (\Sigma, Q, q_0, F, \Delta)$, entonces tomamos $M' = (\Sigma, Q', q_0', F', \delta')$ con:

$Q' = \mathcal{P}(Q)$
$q_0' = \{q_0\}$
$F' = \{S \subseteq Q : S \cap F \neq \emptyset\}$
$\delta'(S, a) = \hat{\Delta}(S, a) = \bigcup_{q \in S} \Delta(q, a)$

Claramente, vale que $\overrightarrow{\delta'}(\{q_0\}, \alpha) = \overrightarrow{\Delta}(q_0, \alpha)$ para toda α, por definición de M'.

Luego, $\alpha \in L(M)$
$\iff \exists q \in F \in$ tal que $q \in \overrightarrow{\Delta}(q_0, \alpha)$
$\iff \exists q \in F \in$ tal que $q \in \overrightarrow{\delta'}(\{q_0\}, \alpha)$
$\iff \overrightarrow{\delta'}(\{q_0\}, \alpha) \in F'$
$\iff \alpha \in L(M')$

$\therefore L(M) = L(M')$

Optimizando un AF por eliminación de estados

Hay estados que se pueden eliminar sin afectar el lenguaje aceptado de un AF:

Optimizando un AF por eliminación de estados

Hay estados que se pueden eliminar sin afectar el lenguaje aceptado de un AF:

- Estados que NO tienen un camino desde el estado inicial hasta ellos.
- Estados que NO tienen un camino desde ellos hasta un estado de aceptación.

Optimizando un AF por eliminación de estados

Hay estados que se pueden eliminar sin afectar el lenguaje aceptado de un AF:

- Estados que NO tienen un camino desde el estado inicial hasta ellos.
- Estados que NO tienen un camino desde ellos hasta un estado de aceptación.

Definición (Estados Alcanzables y Solubles)

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ *un AFN, entonces:*

- *Un estado* $q \in Q$ *es* **alcanzable** *sii* $\exists \alpha \in \Sigma^*$ *tq* $q \in \overrightarrow{\Delta}(q_0, \alpha)$.
- *Un estado* $q \in Q$ *es* **soluble** *sii* $\exists \alpha \in \Sigma^*$ *tq* $\overrightarrow{\Delta}(q, \alpha) \cap F \neq \emptyset$.

Optimizando un AF por eliminación de estados

Hay estados que se pueden eliminar sin afectar el lenguaje aceptado de un AF:

- Estados que NO tienen un camino desde el estado inicial hasta ellos.
- Estados que NO tienen un camino desde ellos hasta un estado de aceptación.

Definición (Estados Alcanzables y Solubles)

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ *un AFN, entonces:*

- *Un estado* $q \in Q$ *es* **alcanzable** *sii* $\exists \alpha \in \Sigma^*$ *tq* $q \in \overrightarrow{\Delta}(q_0, \alpha)$.
- *Un estado* $q \in Q$ *es* **soluble** *sii* $\exists \alpha \in \Sigma^*$ *tq* $\overrightarrow{\Delta}(q, \alpha) \cap F \neq \emptyset$.

Por lo tanto, podemos eliminar los estados **no alcanzables** y **no solubles** preservando completamente el lenguaje aceptado del autómata.

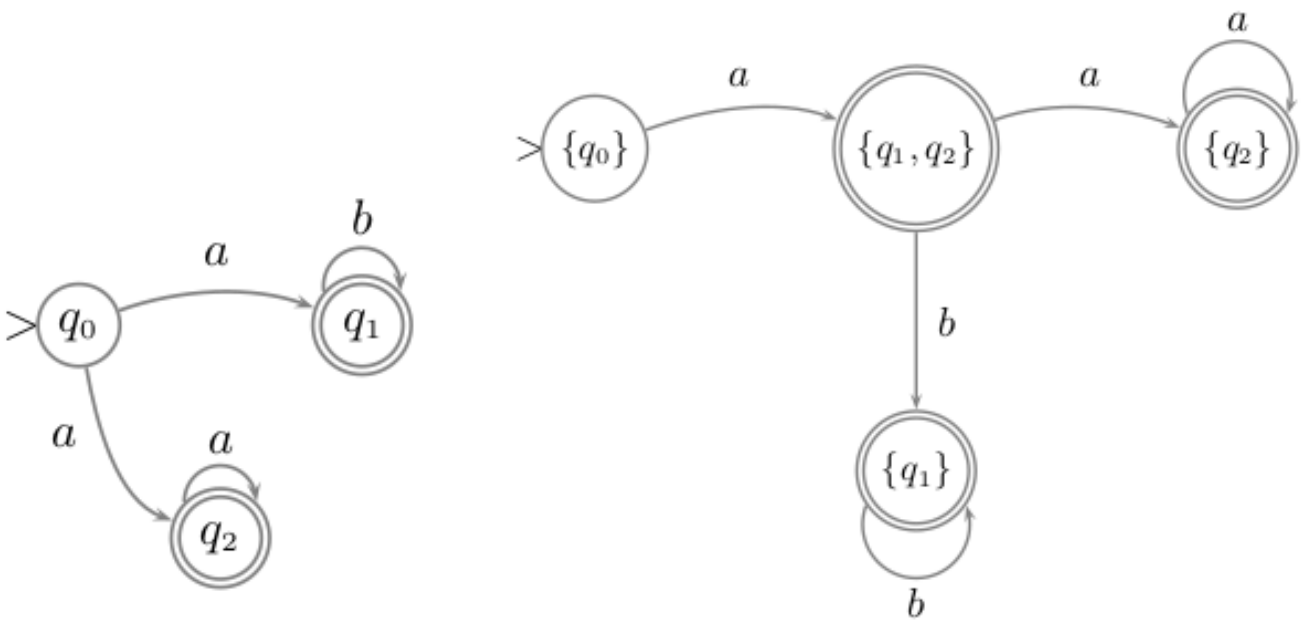
Determinizando un AFN

La última demo (2) nos brinda un algoritmo de determinización, pero, por lo general, con muchos estados no solubles y no alcanzables que se pueden eliminar sin afectar el lenguaje aceptado.

Determinizando un AFN

La última demo (2) nos brinda un algoritmo de determinización, pero, por lo general, con muchos estados no solubles y no alcanzables que se pueden eliminar sin afectar el lenguaje aceptado.

Por ejemplo, para el AFN que se muestra más a la izquierda, con lenguaje $ab^* + a^+$, aplicamos el algoritmo y luego de eliminar los estados no alcanzables y no solubles obtenemos a la derecha su AFD equivalente que acepta el mismo lenguaje:



Transiciones Espontáneas

- Permiten al autómata transicionar de estado, sin leer y desplazar el cabezal de la cinta de entrada.

Transiciones Espontáneas

- Permiten al autómata transicionar de estado, sin leer y desplazar el cabezal de la cinta de entrada.
- Una transición espontánea, si esta definida, puede tener lugar como no (por eso su nombre), en consecuencia, es en si misma una transición no-determinista.

Transiciones Espontáneas

- Permiten al autómata transicionar de estado, sin leer y desplazar el cabezal de la cinta de entrada.
- Una transición espontánea, si esta definida, puede tener lugar como no (por eso su nombre), en consecuencia, es en si misma una transición no-determinista.
- Se pueden pensar como una transición “gratis”, en el sentido de que el autómata cambia de estado sin procesar o consumir un símbolo de la cinta de entrada.

Transiciones Espontáneas

- Permiten al autómata transicionar de estado, sin leer y desplazar el cabezal de la cinta de entrada.
- Una transición espontánea, si esta definida, puede tener lugar como no (por eso su nombre), en consecuencia, es en si misma una transición no-determinista.
- Se pueden pensar como una transición "gratis", en el sentido de que el autómata cambia de estado sin procesar o consumir un símbolo de la cinta de entrada.
- Las denotaremos con el símbolo ϵ, así:

$$\Delta(q, \epsilon) = \{q1, \ldots, q_k\}$$

significa que el autómata puede libremente transicionar de q a cualquier q_i sin leer símbolo alguno de la cinta de entrada (y sin desplazar su cabezal).

Autómata Finito No-Determinístico con Transiciones Espontáneas

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

Autómata Finito No-Determinístico con Transiciones Espontáneas

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*

Autómata Finito No-Determinístico con Transiciones Espontáneas

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*

Autómata Finito No-Determinístico con Transiciones Espontáneas

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$*, llamado estado inicial*

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$*, llamado estado inicial*
- $F \subseteq Q$*, llamado conjunto de estados de aceptación o finales*

Autómata Finito No-Determinístico con Transiciones Espontáneas

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$, *llamada función de transición.*

Definición (AFNϵ)

Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$*, llamado estado inicial*
- $F \subseteq Q$*, llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$*, llamada función de transición.*

Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M*, si existe* **al menos una forma** *de procesar la cadena* α*, partiendo desde el estado inicial* q_0*, donde el estado al que arriba* M *es un estado de aceptación.*

Definición (AFNϵ)

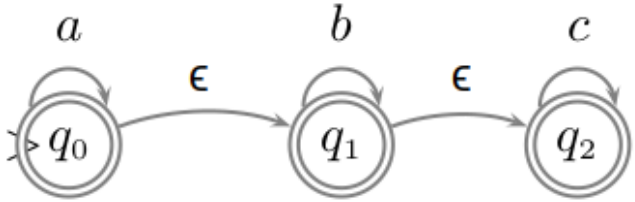
Un **Automata Finito No-Deterministico con transiciones espontáneas** *(AFNϵ) es un 5-upla* $M = (\Sigma, Q, q_0, F, \Delta)$ *donde:*

- Σ *es un alfabeto, llamado alfabeto de entrada*
- Q *es un conjunto (no vacio y finito) de estados*
- $q_0 \in Q$, *llamado estado inicial*
- $F \subseteq Q$, *llamado conjunto de estados de aceptación o finales*
- $\Delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$, *llamada función de transición.*

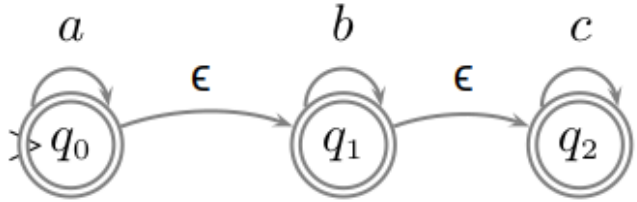
Diremos que una cadena $\alpha \in \Sigma^*$ *es* **aceptada** *por el autómata* M, *si existe* **al menos una forma** *de procesar la cadena* α, *partiendo desde el estado inicial* q_0, *donde el estado al que arriba* M *es un estado de aceptación.*

$$AFN\epsilon^{\Sigma} = \{M : M \text{ es un } AFN\epsilon \text{ con alfabeto de entrada } \Sigma\}$$

El AFNϵ que acepta el lenguaje $a^*b^*c^*$:



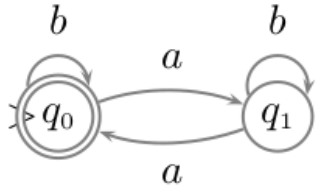
El AFNϵ que acepta el lenguaje $a^*b^*c^*$:



Las transiciones espontáneas son muy útiles ya que permiten muy fácilmente realizar empalmes entre autómatas!!!

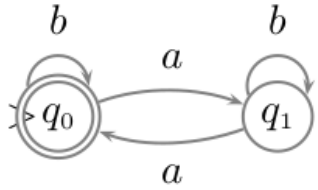
Combinando Autómatas

AFD que acepta el lenguaje de las cadenas con un número par de *a*'s:

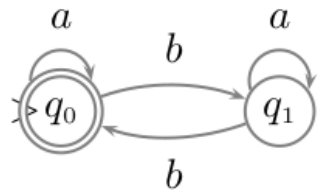


Combinando Autómatas

AFD que acepta el lenguaje de las cadenas con un número par de *a*'s:

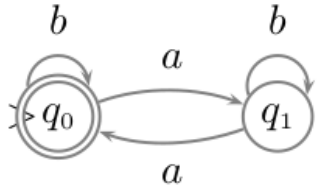


AFD que acepta el lenguaje de las cadenas con un número par de *b*'s:

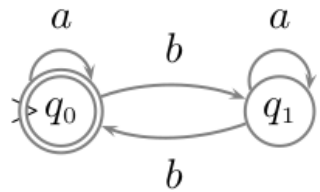


Combinando Autómatas

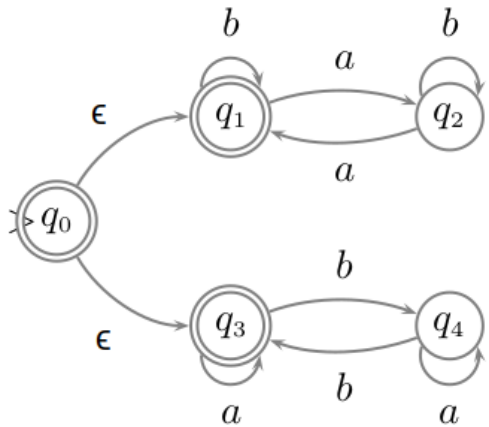
AFD que acepta el lenguaje de las cadenas con un número par de a's:



AFD que acepta el lenguaje de las cadenas con un número par de b's:



AFNϵ que acepta el lenguaje de las cadenas con un número par de a's ó b's:



Equivalencia entre AFN y AFNϵ

Los AFN y AFNϵ son computacionalmente equivalentes.

Equivalencia entre AFN y AFNϵ

Los AFN y AFNϵ son computacionalmente equivalentes.

Teorema ($AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma}$)

1. *(Ida) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFN\epsilon^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN\epsilon^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.

Equivalencia entre AFN y AFNϵ

Los AFN y AFNϵ son computacionalmente equivalentes.

Teorema ($AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma}$)

1. *(Ida) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFN\epsilon^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN\epsilon^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.

Demo 1. Trivial, pues un AFN puede ser visto como un AFNϵ sin utilizar transiciones espontáneas.

Equivalencia entre AFN y AFNϵ

Los AFN y AFNϵ son computacionalmente equivalentes.

Teorema ($AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma}$)

1. *(Ida) Si* $M \in AFN^{\Sigma}$, *entonces* $\exists M' \in AFN\epsilon^{\Sigma}$ *tq* $L(M') = L(M)$.
2. *(Vuelta) Si* $M \in AFN\epsilon^{\Sigma}$, *entonces* $\exists M' \in AFN^{\Sigma}$ *tq* $L(M') = L(M)$.

Demo 1. Trivial, pues un AFN puede ser visto como un AFNϵ sin utilizar transiciones espontáneas.
Demo 2. Debemos probar que **siempre se puede eliminar las transiciones espontáneas sin afectar el lenguaje**. El truco consiste en introducir transiciones que simulen los mismos efectos de las espontáneas. Para esto, debemos introducir primero la noción de **clausura de un estado**.

Clausura de un Estado

La **clausura** de un estado q, denotado con $[q]$, es el conjunto de estados al que puedo arribar desde q, utilizando solamente transiciones espontáneas:

Clausura de un Estado

La **clausura** de un estado q, denotado con $[q]$, es el conjunto de estados al que puedo arribar desde q, utilizando solamente transiciones espontáneas:

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_k \Longrightarrow q_i \in [q]$$

Clausura de un Estado

La **clausura** de un estado q, denotado con $[q]$, es el conjunto de estados al que puedo arribar desde q, utilizando solamente transiciones espontáneas:

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_k \Longrightarrow q_i \in [q]$$

Definimos formalmente la clausura de un estado q:

$$[q] = \{q' \in Q : q' \in \overrightarrow{\Delta}(q, \epsilon)\}$$

Clausura de un Estado

La **clausura** de un estado q, denotado con $[q]$, es el conjunto de estados al que puedo arribar desde q, utilizando solamente transiciones espontáneas:

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_k \Longrightarrow q_i \in [q]$$

Definimos formalmente la clausura de un estado q:

$$[q] = \{q' \in Q : q' \in \overrightarrow{\Delta}(q, \epsilon)\}$$

Extendemos la definición de clausura para conjunto de estados $S \subseteq Q$:

$$[S] = \bigcup_{q \in S} [q]$$

Clausura de un Estado

La **clausura** de un estado q, denotado con $[q]$, es el conjunto de estados al que puedo arribar desde q, utilizando solamente transiciones espontáneas:

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_k \Longrightarrow q_i \in [q]$$

Definimos formalmente la clausura de un estado q:

$$[q] = \{q' \in Q : q' \in \overrightarrow{\Delta}(q, \epsilon)\}$$

Extendemos la definición de clausura para conjunto de estados $S \subseteq Q$:

$$[S] = \bigcup_{q \in S} [q]$$

Para todo q, se cumple que:

1. $q \in [q]$
2. $[[q]] = [q]$

Ahora si probemos la vuelta (2).

Ahora si probemos la vuelta (2).

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ el AFNϵ, entonces podemos tomar el AFN $M' = (\Sigma, Q', q_0', F', \Delta')$ donde:

Ahora si probemos la vuelta (2).

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ el AFNϵ, entonces podemos tomar el AFN $M' = (\Sigma, Q', q_0', F', \Delta')$ donde:

Se preservan todos los estados y el estado inicial del AFNϵ:

$Q' = Q$
$q_0' = q_0$

Ahora si probemos la vuelta (2).

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ el AFNϵ, entonces podemos tomar el AFN $M' = (\Sigma, Q', q_0', F', \Delta')$ donde:

Se preservan todos los estados y el estado inicial del AFNϵ:

$Q' = Q$
$q_0' = q_0$

Los estados finales son aquellos tal que su clausura contiene al menos un estado de aceptación del AFNϵ:

$F' = \{q \in Q : [q] \cap F \neq \emptyset\}$

Ahora si probemos la vuelta (2).

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ el AFNϵ, entonces podemos tomar el AFN $M' = (\Sigma, Q', q_0', F', \Delta')$ donde:

Se preservan todos los estados y el estado inicial del AFNϵ:

$Q' = Q$
$q_0' = q_0$

Los estados finales son aquellos tal que su clausura contiene al menos un estado de aceptación del AFNϵ:

$F' = \{q \in Q : [q] \cap F \neq \emptyset\}$

Y por último, intuitivamente si en el AFNϵ tenemos un camino $q \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q'$, entonces en el AFN definimos la transición $q \xrightarrow{a} q'$:

$\Delta'(q, a) = [\hat{\Delta}([q], a)]$

Ahora si probemos la vuelta (2).

Sea $M = (\Sigma, Q, q_0, F, \Delta)$ el AFNϵ, entonces podemos tomar el AFN $M' = (\Sigma, Q', q_0', F', \Delta')$ donde:

Se preservan todos los estados y el estado inicial del AFNϵ:

$Q' = Q$
$q_0' = q_0$

Los estados finales son aquellos tal que su clausura contiene al menos un estado de aceptación del AFNϵ:

$F' = \{q \in Q : [q] \cap F \neq \emptyset\}$

Y por último, intuitivamente si en el AFNϵ tenemos un camino $q \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q'$, entonces en el AFN definimos la transición $q \xrightarrow{a} q'$:

$\Delta'(q, a) = [\hat{\Delta}([q], a)]$

Luego, $L(M) = L(M')$.

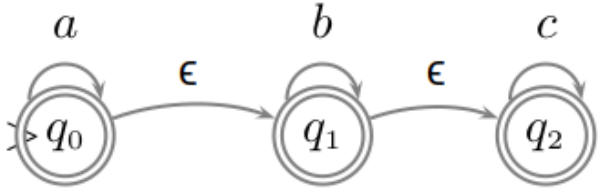
Algoritmo de Eliminacion de Transiciones Espontaneas

La prueba del iniciso 2 nos brinda un algoritmo para eliminar transiciones espontáneas.

Algoritmo de Eliminacion de Transiciones Espontaneas

La prueba del iniciso 2 nos brinda un algoritmo para eliminar transiciones espontáneas.

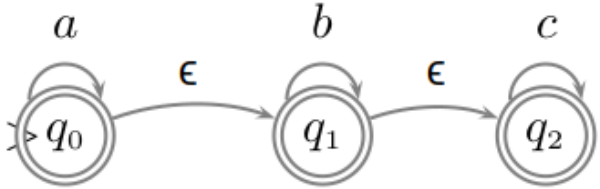
Por ejemplo, sea M el siguiente AFNϵ con L(M) $= a^*b^*c^*$.



Algoritmo de Eliminacion de Transiciones Espontaneas

La prueba del iniciso 2 nos brinda un algoritmo para eliminar transiciones espontáneas.

Por ejemplo, sea M el siguiente AFNϵ con L(M) $= a^*b^*c^*$.

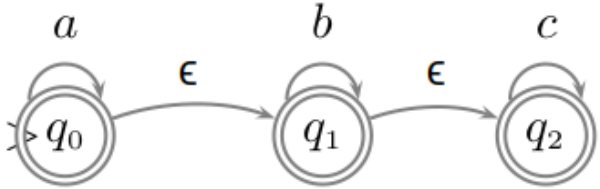


Obtengamos el AFN equivalente eliminando sus transiciones espontáneas.

Algoritmo de Eliminacion de Transiciones Espontaneas

La prueba del iniciso 2 nos brinda un algoritmo para eliminar transiciones espontáneas.

Por ejemplo, sea M el siguiente AFNϵ con L(M) = $a^*b^*c^*$.



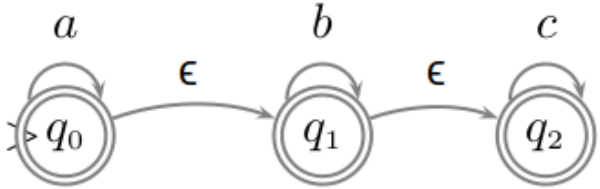
Obtengamos el AFN equivalente eliminando sus transiciones espontáneas.

Calculamos las clausuras de los estados:
$[q_0] = \{q_0, q_1, q_2\}$
$[q_1] = \{q_1, q_2\}$
$[q_2] = \{q_2\}$

Algoritmo de Eliminacion de Transiciones Espontaneas

La prueba del iniciso 2 nos brinda un algoritmo para eliminar transiciones espontáneas.

Por ejemplo, sea M el siguiente AFNϵ con L(M) = $a^*b^*c^*$.



Obtengamos el AFN equivalente eliminando sus transiciones espontáneas.

Calculamos las clausuras de los estados:
$[q_0] = \{q_0, q_1, q_2\}$
$[q_1] = \{q_1, q_2\}$
$[q_2] = \{q_2\}$

Calculamos los estados de aceptación:
$F' = \{q_0, q_1, q_2\}$

Y por último, calculamos la función de transición:

Y por último, calculamos la función de transición:

$$\Delta'(q_0, a) = [\hat{\Delta}([q_0], a)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, a)] = [\{q_0\}] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_0, b) = [\hat{\Delta}([q_0], b)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_0, c) = [\hat{\Delta}([q_0], c)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

Y por último, calculamos la función de transición:

$$\Delta'(q_0, a) = [\hat{\Delta}([q_0], a)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, a)] = [\{q_0\}] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_0, b) = [\hat{\Delta}([q_0], b)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_0, c) = [\hat{\Delta}([q_0], c)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

$$\Delta'(q_1, a) = [\hat{\Delta}([q_1], a)] = [\hat{\Delta}(\{q_1, q_2\}, a)] = [\emptyset] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_1, b) = [\hat{\Delta}([q_1], b)] = [\hat{\Delta}(\{q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_1, c) = [\hat{\Delta}([q_1], c)] = [\hat{\Delta}(\{q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

Y por último, calculamos la función de transición:

$$\Delta'(q_0, a) = [\hat{\Delta}([q_0], a)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, a)] = [\{q_0\}] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_0, b) = [\hat{\Delta}([q_0], b)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_0, c) = [\hat{\Delta}([q_0], c)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

$$\Delta'(q_1, a) = [\hat{\Delta}([q_1], a)] = [\hat{\Delta}(\{q_1, q_2\}, a)] = [\emptyset] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_1, b) = [\hat{\Delta}([q_1], b)] = [\hat{\Delta}(\{q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_1, c) = [\hat{\Delta}([q_1], c)] = [\hat{\Delta}(\{q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

$$\Delta'(q_2, a) = [\hat{\Delta}([q_2], a)] = [\hat{\Delta}(\{q_2\}, a)] = [\emptyset] = \emptyset$$
$$\Delta'(q_2, b) = [\hat{\Delta}([q_2], b)] = [\hat{\Delta}(\{q_2\}, b)] = [\emptyset] = \emptyset$$
$$\Delta'(q_2, c) = [\hat{\Delta}([q_2], c)] = [\hat{\Delta}(\{q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

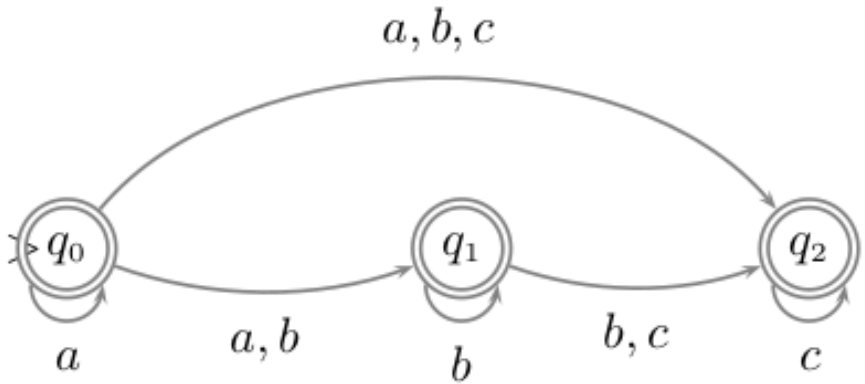
Y por último, calculamos la función de transición:

$$\Delta'(q_0, a) = [\hat{\Delta}([q_0], a)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, a)] = [\{q_0\}] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_0, b) = [\hat{\Delta}([q_0], b)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_0, c) = [\hat{\Delta}([q_0], c)] = [\hat{\Delta}(\{q_0, q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

$$\Delta'(q_1, a) = [\hat{\Delta}([q_1], a)] = [\hat{\Delta}(\{q_1, q_2\}, a)] = [\emptyset] = \{q_0, q_1, q_2\}$$
$$\Delta'(q_1, b) = [\hat{\Delta}([q_1], b)] = [\hat{\Delta}(\{q_1, q_2\}, b)] = [\{q_1\}] = \{q_1, q_2\}$$
$$\Delta'(q_1, c) = [\hat{\Delta}([q_1], c)] = [\hat{\Delta}(\{q_1, q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

$$\Delta'(q_2, a) = [\hat{\Delta}([q_2], a)] = [\hat{\Delta}(\{q_2\}, a)] = [\emptyset] = \emptyset$$
$$\Delta'(q_2, b) = [\hat{\Delta}([q_2], b)] = [\hat{\Delta}(\{q_2\}, b)] = [\emptyset] = \emptyset$$
$$\Delta'(q_2, c) = [\hat{\Delta}([q_2], c)] = [\hat{\Delta}(\{q_2\}, c)] = [\{q_2\}] = \{q_2\}$$

Obteniendo finalmente el siguiente AFN M' equivalente pues $L(M') = a^* b^* c^*$.



Resumen

- Presentamos tres tipos de autómatas finitos (AFD, AFN, AFNϵ) y probamos que son computacionalmente equivalentes :

$$AFD^{\Sigma} \equiv AFN^{\Sigma} \text{ y } AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma} \implies AFD^{\Sigma} \equiv AFN\epsilon^{\Sigma}$$

Resumen

- Presentamos tres tipos de autómatas finitos (AFD, AFN, AFNϵ) y probamos que son computacionalmente equivalentes :

  $$AFD^{\Sigma} \equiv AFN^{\Sigma} \text{ y } AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma} \implies AFD^{\Sigma} \equiv AFN\epsilon^{\Sigma}$$

- En la praćtica, convertimos un AFNϵ a su AFD equivalente, aplicando primero el algoritmo de eliminacion de transiciones espontáneas, y luego, el algoritmo de determinización.

Resumen

- Presentamos tres tipos de autómatas finitos (AFD, AFN, AFNϵ) y probamos que son computacionalmente equivalentes :

$$AFD^{\Sigma} \equiv AFN^{\Sigma} \text{ y } AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma} \implies AFD^{\Sigma} \equiv AFN\epsilon^{\Sigma}$$

- En la praćtica, convertimos un AFNϵ a su AFD equivalente, aplicando primero el algoritmo de eliminacion de transiciones espontáneas, y luego, el algoritmo de determinización.
- Por ser equivalentes, hablaremos de autómata finito (AF) sin especificar su tipo:

$$AF^{\Sigma} = AFD^{\Sigma} \cup AFN^{\Sigma} \cup AFN\epsilon^{\Sigma}$$

Resumen

- Presentamos tres tipos de autómatas finitos (AFD, AFN, AFNϵ) y probamos que son computacionalmente equivalentes :

$$AFD^{\Sigma} \equiv AFN^{\Sigma} \text{ y } AFN^{\Sigma} \equiv AFN\epsilon^{\Sigma} \implies AFD^{\Sigma} \equiv AFN\epsilon^{\Sigma}$$

- En la praćtica, convertimos un AFNϵ a su AFD equivalente, aplicando primero el algoritmo de eliminacion de transiciones espontáneas, y luego, el algoritmo de determinización.
- Por ser equivalentes, hablaremos de autómata finito (AF) sin especificar su tipo:

$$AF^{\Sigma} = AFD^{\Sigma} \cup AFN^{\Sigma} \cup AFN\epsilon^{\Sigma}$$

- Ahora estamos en condiciones de probar en la próxima clase que los AF caracterizan a los lenguajes regulares:

$$LR^{\Sigma} \equiv AF^{\Sigma}.$$

Bibliografía

Rodrigo De Castro Korgi.

"Teoria de la Computación". Lenguajes, Autómatas, Gramáticas.

Capítulo 2: Autómatas Finitos. Sección 2.1 hasta 2.7.