No determinismo

Juan Gutiérrez

August 26, 2022

Resumen

AFN

Relación entre AFN y AFD

Actividad en clases

Otro tipo de AFNs

Actividad en clases

Operaciones regulares

AFN

Autómatas finitos no deterministas (AFN)

- Tiene la capacidad de estar en varios estados a la vez

Autómatas finitos no deterministas (AFN)

- Tiene la capacidad de estar en varios estados a la vez
- Puede "conjeturar" algo sobre su entrada

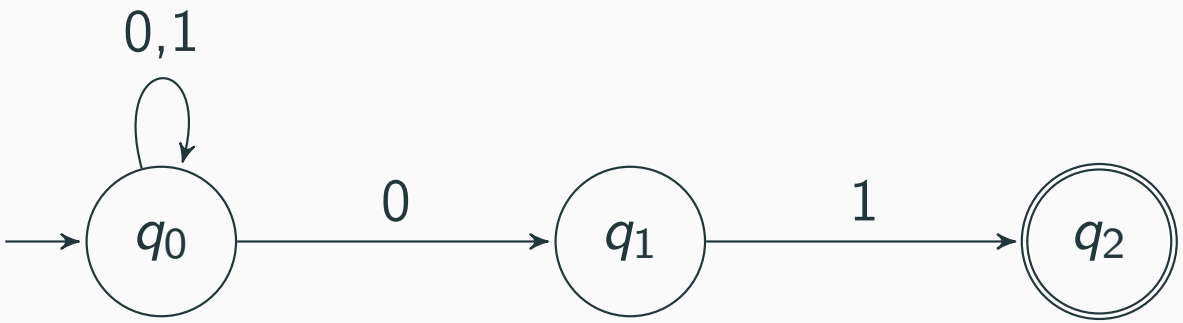
Autómatas finitos no deterministas (AFN)

- Tiene la capacidad de estar en varios estados a la vez
- Puede "conjeturar" algo sobre su entrada
- Tiene casi la misma definición que un AFD, excepto por la función de transición δ

Autómatas finitos no deterministas (AFN)

- Tiene la capacidad de estar en varios estados a la vez
- Puede "conjeturar" algo sobre su entrada
- Tiene casi la misma definición que un AFD, excepto por la función de transición δ
- Ahora δ devuelve un conjunto de estados, y no solo uno

Un AFN



Autómatas finitos no deterministas (AFN)

▸ Mario

Un AFN

Qué cadenas reconoce el autómata?

Definición formal de un AFN

Un AFN es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,

Definición formal de un AFN

Un AFN es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

Definición formal de un AFN

Un AFN es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

4. $q_0 \in Q$ es el *estado inicial*, y

Definición formal de un AFN

Un AFN es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

4. $q_0 \in Q$ es el *estado inicial*, y
5. $F \subseteq Q$ es el conjunto de *estados de aceptación*,

Definición formal de un AFN

Un AFN es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,
3. $\delta : Q \times \Sigma \rightarrow \{A : A \subseteq Q\}$ es la *función de transición*,
4. $q_0 \in Q$ es el *estado inicial*, y
5. $F \subseteq Q$ es el conjunto de *estados de aceptación*,

Definición formal de un AFN

Ahora δ devuelve un conjunto de estados !

Observaciones sobre la definición

- Note que, en la figura, $\delta(q_1, 0) = \emptyset$

Observaciones sobre la definición

- Note que, en la figura, $\delta(q_1, 0) = \emptyset$
- En ese caso, no existe la salida correspondiente

Observaciones sobre la definición

- Note que, en la figura, $\delta(q_1, 0) = \emptyset$
- En ese caso, no existe la salida correspondiente
- En un AFN puede ocurrir esto, pero nunca en un AFD

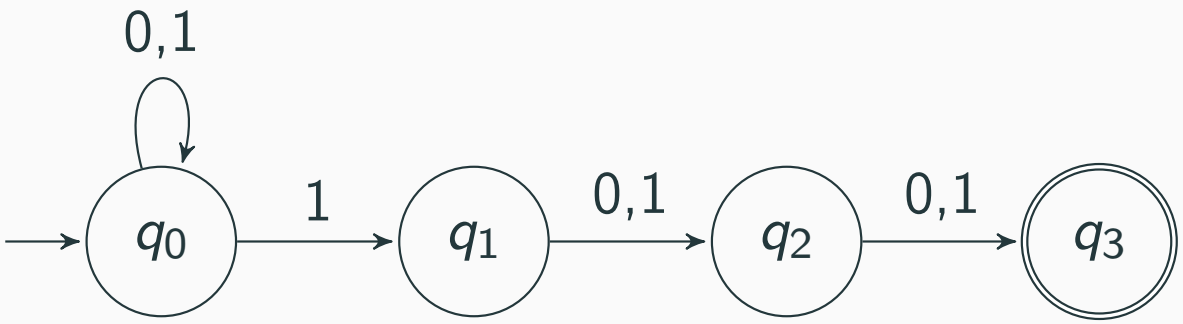
Observaciones sobre la definición

- Note que, en la figura, $\delta(q_1, 0) = \emptyset$
- En ese caso, no existe la salida correspondiente
- En un AFN puede ocurrir esto, pero nunca en un AFD
- Para aceptar una cadena, basta llegar a por lo menos un estado final (no es necesario llegar a todos)

Definición formal de un AFN

Cual es la definición formal del autómata de la figura?

Otro AFN



Un AFN

Qué cadenas reconoce el autómata?

Lenguaje aceptado por un AFN

Decimos que el AFN A reconoce el lenguaje L si
$L = \{w : A \text{ acepta } w\}$.

Es decir, al leer la cadena w, M llega a ***por lo menos*** *un estado final*

Ejercicios

Diseñe AFN's para los siguientes lenguajes

(i) Cadenas, con alfabeto $\{0, 1\}$, que tienen a 0101 como subcadena

(ii) Cadenas, con alfabeto $\{0, 1, 2, 3\}$, cuyo último dígito ya apareció antes

Relación entre AFN y AFD

Lenguaje aceptado por un AFN

Los AFN reconocen más lenguajes que los AFD?

Lenguaje aceptado por un AFN

Teorema
Todo AFN tiene un AFD equivalente, es decir, un AFD que reconoce el mismo lenguaje

Idea de la prueba

- Prueba por construcción

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- M debe simular N en cada paso

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- M debe simular N en cada paso
- Recuerde que N está en varios estados al mismo tiempo

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- M debe simular N en cada paso
- Recuerde que N está en varios estados al mismo tiempo
- M puede simular este comportamiento si en cada momento sabe todos los estados en los que está N

Idea de la prueba

- Prueba por construcción
- Sea N un AFN que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- M debe simular N en cada paso
- Recuerde que N está en varios estados al mismo tiempo
- M puede simular este comportamiento si en cada momento sabe todos los estados en los que está N
- Si N tiene k estados, entonces M tiene 2^k estados

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \bigcup_{p \in S} \delta(p, a)$

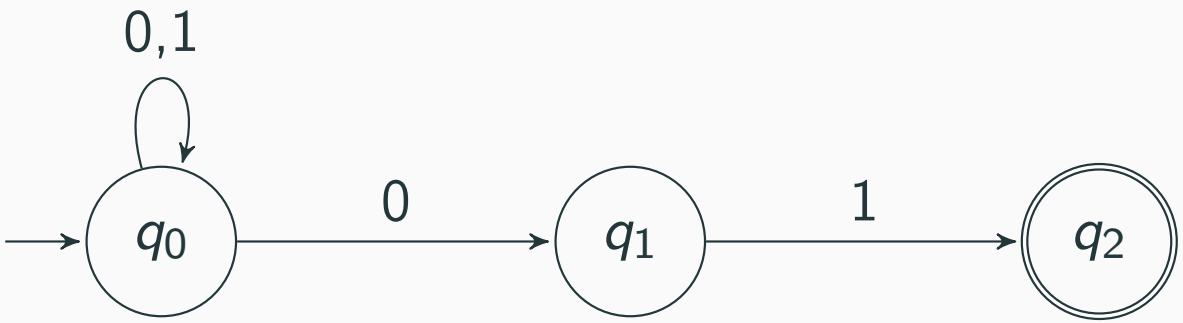
Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \bigcup_{p \in S} \delta(p, a)$
- $q_0' = \{q_0\}$

Idea de la prueba

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \bigcup_{p \in S} \delta(p, a)$
- $q_0' = \{q_0\}$
- $F' = \{S \in Q' : S$ tiene un estado de aceptación de $N\}$

Un ejemplo



Un ejemplo

Cual es el AFD resultante de aplicar el algoritmo?

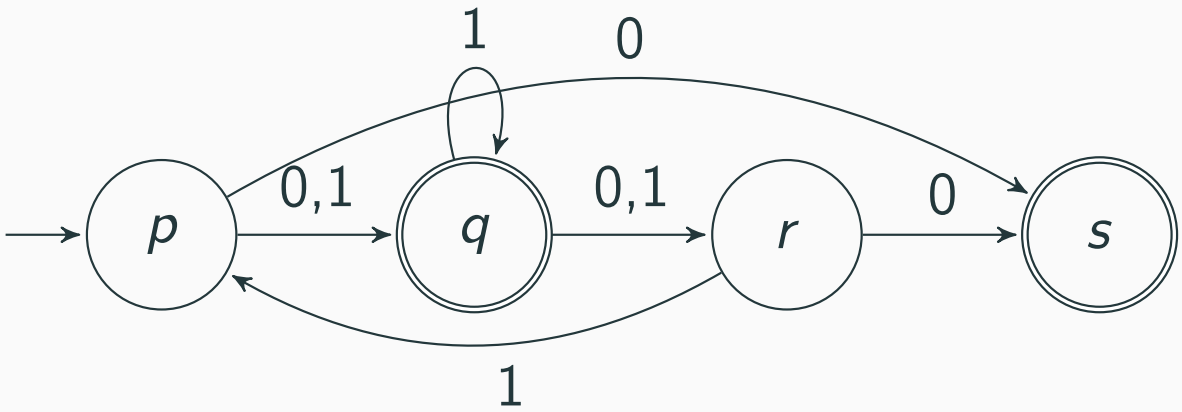
Un ejemplo

Podemos simplificar dicho AFD?

Actividad en clases

Actividad en clase

(a) Convertir el siguiente AFN en AFD:



(b) Diseñe un AFN para el siguiente lenguaje: cadenas, con alfabeto $\{0, 1\}$, que tienen a 0101 como subcadena. Luego conviértalo a un AFD con la técnica vista.

Otro tipo de AFNs

Recordemos la pregunta

La operación de concatenación de lenguajes regulares, es cerrada?
Es decir, si L_1 y L_2 son regulares, entonces $L_1 \cdot L_2$ es regular?

Recordemos la pregunta

Para responder dicha pregunta, necesitamos extender un poco la definición de AFN

AFN con transiciones ϵ (AFN-ϵ)

- El autómata hace transiciones sin recibir ningún símbolo de entrada

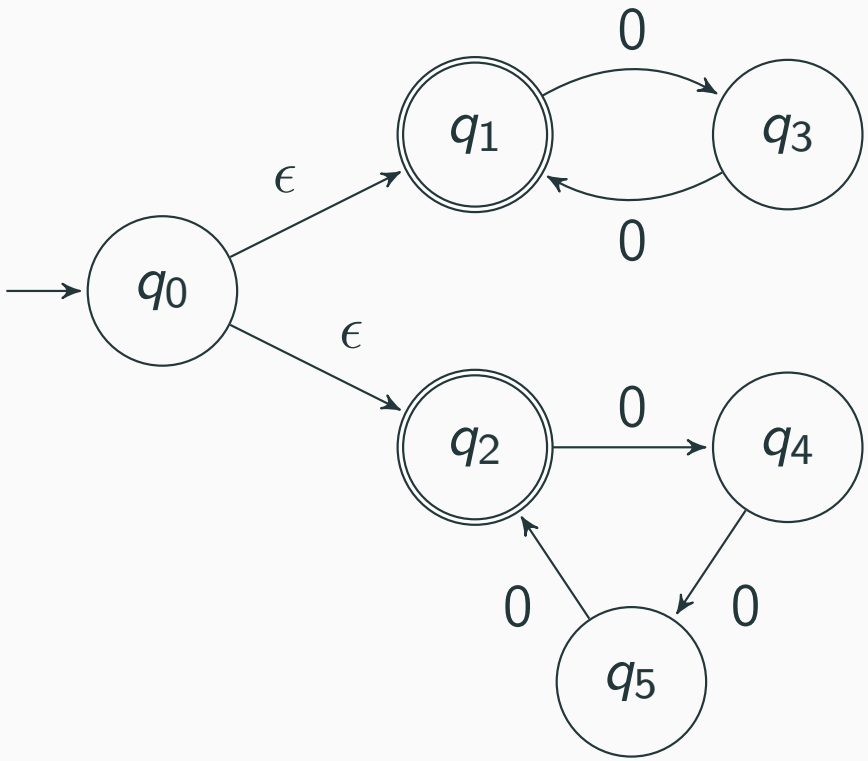
AFN con transiciones ϵ (AFN-ϵ)

- El autómata hace transiciones sin recibir ningún símbolo de entrada
- Transiciones ϵ: transiciones espontáneas

AFN con transiciones ϵ (AFN-ϵ)

- El autómata hace transiciones sin recibir ningún símbolo de entrada
- Transiciones ϵ: transiciones espontáneas
- No dan más poder computacional, pero proporcionan "facilidades de programación"

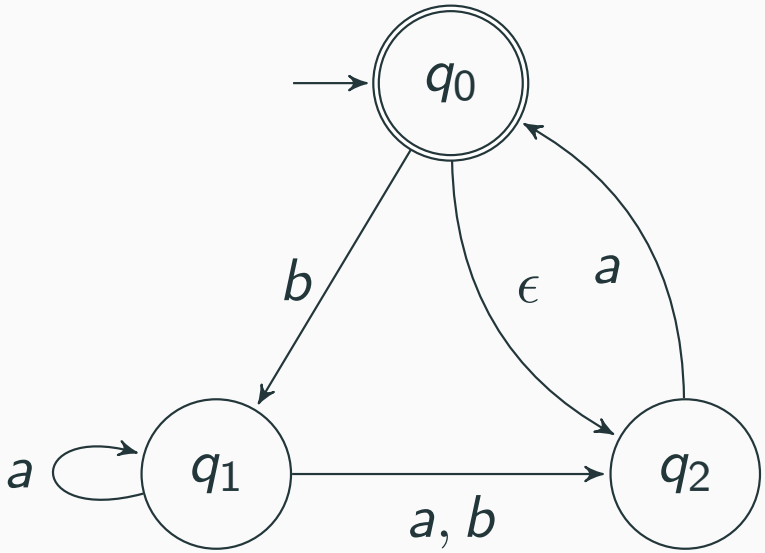
Un AFN-ϵ



Un AFN-ϵ

Qué cadenas reconoce el autómata?

Otro AFN-ϵ



Otro AFN-ϵ

Son aceptadas las cadenas aabbaa y babaaa?

Definición formal de un AFN-ϵ

Un AFN-ϵ es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,

Definición formal de un AFN-ϵ

Un AFN-ϵ es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

Definición formal de un AFN-ϵ

Un AFN-ϵ es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

4. $q_0 \in Q$ es el *estado inicial*, y

Definición formal de un AFN-ϵ

Un AFN-ϵ es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,

4. $q_0 \in Q$ es el *estado inicial*, y
5. $F \subseteq Q$ es el conjunto de *estados de aceptación*,

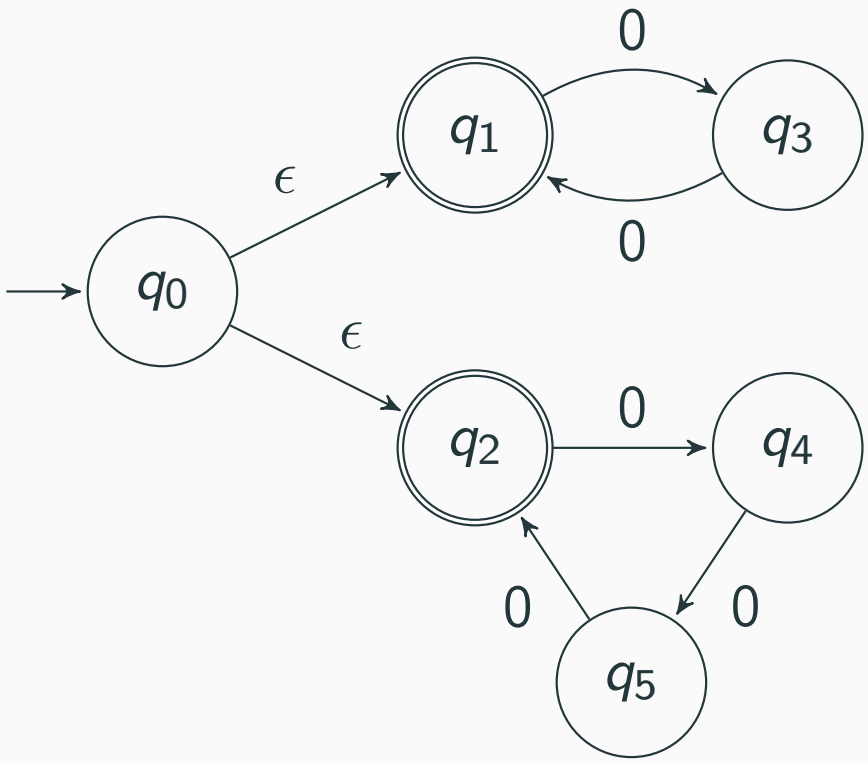
Definición formal de un AFN-ϵ

Un AFN-ϵ es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,
2. $\sum$ es un conjunto finito llamado *alfabeto*,
3. $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \{A : A \subseteq Q\}$ es la *función de transición*,
4. $q_0 \in Q$ es el *estado inicial*, y
5. $F \subseteq Q$ es el conjunto de *estados de aceptación*,

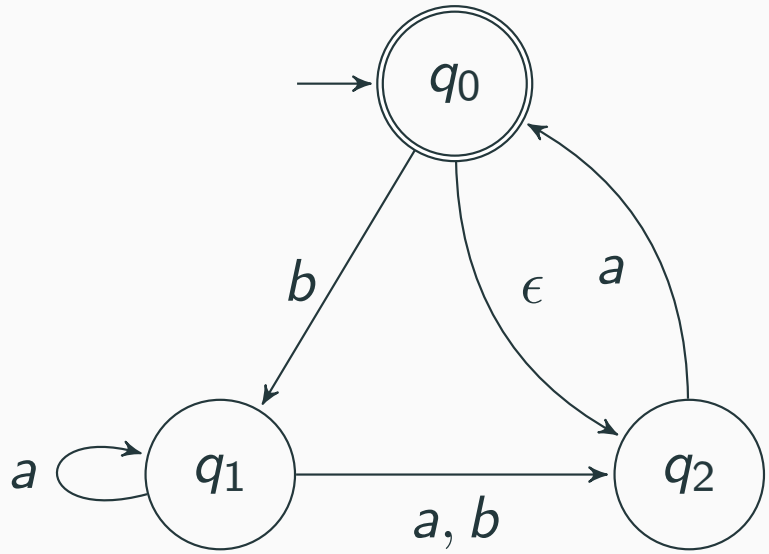
Definición formal de un AFN-ϵ

Cual es la definición formal del autómata presentado anteriormente?



Definición formal de un AFN-ϵ

Cual es la definición formal del autómata presentado anteriormente?



Definición formal de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ Sea w una cadena en el alfabeto $\sum$. Entonces N acepta w si podemos escribir w como $w = y_1 y_2 \ldots y_n$, donde cada y_i está en el alfabeto $\sum \cup \{\epsilon\}$ y existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,

Definición formal de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ Sea w una cadena en el alfabeto $\sum$. Entonces N acepta w si podemos escribir w como $w = y_1 y_2 \ldots y_n$, donde cada y_i está en el alfabeto $\sum \cup \{\epsilon\}$ y existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, para $i = 0, \ldots, n-1$, y

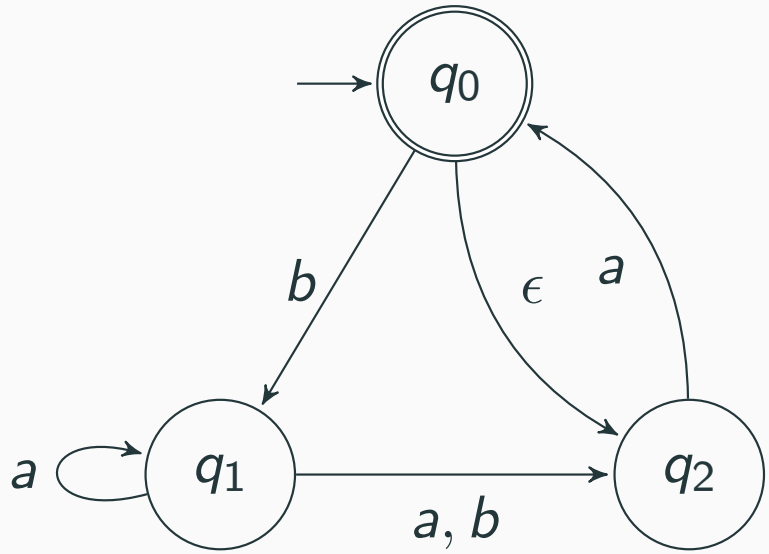
Definición formal de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ Sea w una cadena en el alfabeto $\sum$. Entonces N acepta w si podemos escribir w como $w = y_1 y_2 \ldots y_n$, donde cada y_i está en el alfabeto $\sum \cup \{\epsilon\}$ y existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, para $i = 0, \ldots, n-1$, y
3. $r_n \in F$.

Definición formal de computación en un AFN-ϵ

babaaa es aceptado por el siguente autómata?



Clausuras respecto de ϵ

- En el ejemplo anterior podemos descomponer babaaa como babaϵaϵa

Clausuras respecto de ϵ

- En el ejemplo anterior podemos descomponer babaaa como babaϵaϵa
- Pero no siempre es tan fácil ver cómo hacer esa descomposición al intentar computar una cadena

Clausuras respecto de ϵ

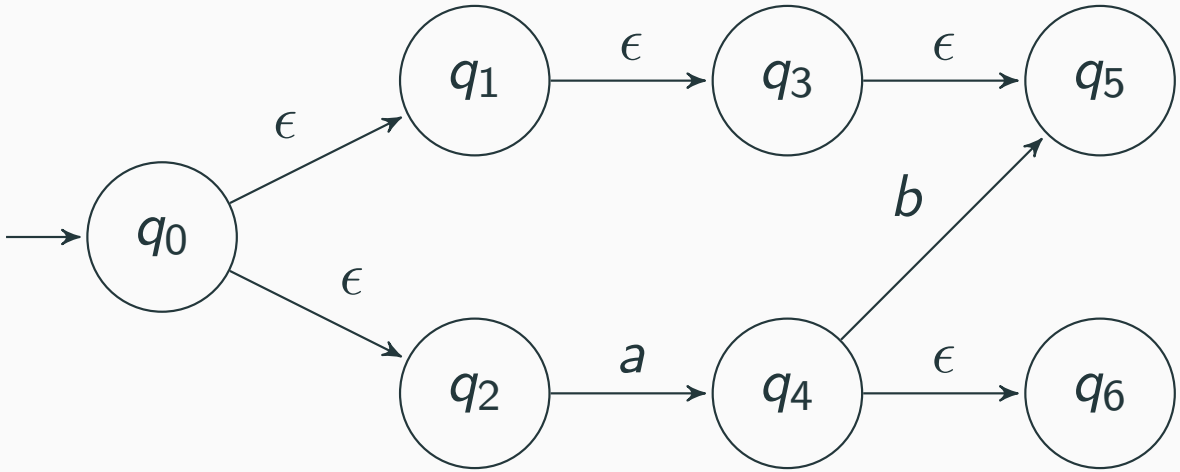
- En el ejemplo anterior podemos descomponer babaaa como babaϵaϵa
- Pero no siempre es tan fácil ver cómo hacer esa descomposición al intentar computar una cadena
- Podemos introducir un nuevo concepto: **Clausura de un estado** ($c\ell$)

Clausuras respecto de ϵ

- En el ejemplo anterior podemos descomponer babaaa como babaϵaϵa
- Pero no siempre es tan fácil ver cómo hacer esa descomposición al intentar computar una cadena
- Podemos introducir un nuevo concepto: **Clausura de un estado** ($c\ell$)
- $c\ell(p) = \{q :$
  q puede ser alcanzado por p usando 0 o más transiciones $\epsilon\}$

Clausuras respecto de ϵ

Indique cuales son las clausuras de cada uno de los estados del siguiente autómata



Clausuras respecto de ϵ

- Para facilidad, extenderemos un poco la definición de $c\ell$ para $\mathcal{C}\ell$

Clausuras respecto de ϵ

- Para facilidad, extenderemos un poco la definición de $c\ell$ para $\mathcal{C}\ell$
- Ahora $\mathcal{C}\ell$ puede ser aplicada a un *conjunto de estados*

Clausuras respecto de ϵ

- Para facilidad, extenderemos un poco la definición de $c\ell$ para $\mathcal{C}\ell$
- Ahora $\mathcal{C}\ell$ puede ser aplicada a un *conjunto de estados*
- En el ejemplo $\mathcal{C}\ell(\{q_3, q_4\}) = c\ell(q_3) \cup c\ell(q_4) = \{q_3, q_5\} \cup \{q_4, q_6\} = \{q_3, q_4, q_5, q_6\}$

Clausuras respecto de ϵ

- Para facilidad, extenderemos un poco la definición de $c\ell$ para $\mathcal{C}\ell$
- Ahora $\mathcal{C}\ell$ puede ser aplicada a un *conjunto de estados*
- En el ejemplo $\mathcal{C}\ell(\{q_3, q_4\}) = c\ell(q_3) \cup c\ell(q_4) = \{q_3, q_5\} \cup \{q_4, q_6\} = \{q_3, q_4, q_5, q_6\}$
- En general $\mathcal{C}\ell(S) = \bigcup_{q \in S} c\ell(q)$

Otra definición de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces N acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 \in c\ell(q_0)$,

Otra definición de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces N acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 \in c\ell(q_0)$,
2. $r_{i+1} \in \mathcal{C}\ell(\delta(r_i, w_{i+1}))$, para $i = 0, \ldots, n-1$, y

Otra definición de computación en un AFN-ϵ

Sea $N = (Q, \sum, \delta, q_0, F)$ un AFN-ϵ. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces N acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 \in c\ell(q_0)$,
2. $r_{i+1} \in \mathcal{C}\ell(\delta(r_i, w_{i+1}))$, para $i = 0, \ldots, n-1$, y
3. $r_n \in F$.

Lenguaje aceptado por un AFN-ϵ

Los AFN-ϵ tienen más poder computacional que los AFN? Es decir, los AFN-ϵ reconocen más lenguajes que los AFN?

Conversión AFN-ϵ a AFD

Teorema
Todo AFN-ϵ tiene un AFD equivalente, es decir, un AFD que reconoce el mismo lenguaje

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN
- Prueba por construcción

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN
- Prueba por construcción
- Sea N un AFN-ϵ que reconoce L

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN
- Prueba por construcción
- Sea N un AFN-ϵ que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN
- Prueba por construcción
- Sea N un AFN-ϵ que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- Si N tiene k estados, entonces M tiene 2^k estados

Conversión AFN-ϵ a AFD

- La prueba es escencialmente la misma que para el caso de AFN
- Prueba por construcción
- Sea N un AFN-ϵ que reconoce L
- Debemos construir un AFD M a partir de N, que también reconoce L
- Si N tiene k estados, entonces M tiene 2^k estados
- En este último paso vamos a simplificar el número de estados desde el comienzo

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \mathcal{C}\ell(\bigcup_{p \in S} \delta(p, a))$

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \mathcal{C}\ell(\bigcup_{p \in S} \delta(p, a))$
- $q_0' = c\ell(q_0)$

Conversión AFN-ϵ a AFD

- Sea $N = \{Q, \Sigma, \delta, q_0, F\}$
- Construímos $M = \{Q', \Sigma, \delta', q_0', F'\}$ como
- $Q' = \{S : S \subseteq Q\}$ (el conjunto potencia)
- $\delta'(S, a) = \mathcal{C}\ell(\bigcup_{p \in S} \delta(p, a))$
- $q_0' = c\ell(q_0)$
- $F' = \{S \in Q' : S$ tiene un estado de aceptación de $N\}$

Conversión AFN-ϵ a AFD

- Para evitar muchos estados, comenzamos desde q'_0 y seguimos por niveles

Conversión AFN-ϵ a AFD

- Para evitar muchos estados, comenzamos desde q_0' y seguimos por niveles
- **Paso 0**: $q_0' := c\ell(q_0)$, $P_1 := \{q_0'\}$, $i := 1$

Conversión AFN-ϵ a AFD

- Para evitar muchos estados, comenzamos desde q_0' y seguimos por niveles
- **Paso 0**: $q_0' := c\ell(q_0)$, $P_1 := \{q_0'\}$, $i := 1$
- **Paso i**: Para cada símbolo a y cada estado p en P_i, calculamos $\delta'(p, a)$

Conversión AFN-ϵ a AFD

- Para evitar muchos estados, comenzamos desde q_0' y seguimos por niveles
- **Paso 0**: $q_0' := c\ell(q_0)$, $P_1 := \{q_0'\}$, $i := 1$
- **Paso i**: Para cada símbolo a y cada estado p en P_i, calculamos $\delta'(p, a)$
- Sea P_{i+1} el conjunto de nuevos estados creados en el paso i

Conversión AFN-ϵ a AFD

- Para evitar muchos estados, comenzamos desde q_0' y seguimos por niveles
- **Paso 0**: $q_0' := c\ell(q_0)$, $P_1 := \{q_0'\}$, $i := 1$
- **Paso i**: Para cada símbolo a y cada estado p en P_i, calculamos $\delta'(p, a)$
- Sea P_{i+1} el conjunto de nuevos estados creados en el paso i
- Si $P_{i+1} = \emptyset$, entonces terminamos; caso contrario ejecutamos el Paso $i + 1$

Actividad en clase

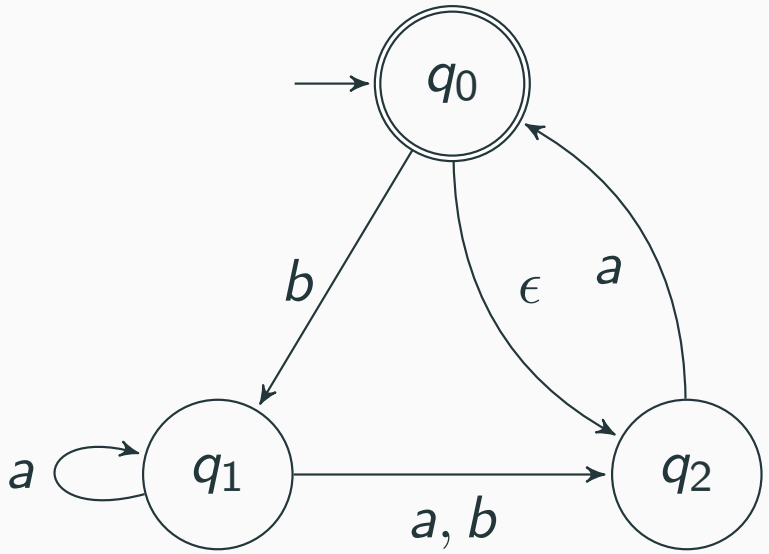
Ejercicio
Para cada uno de los siguientes autómatas:

- *Calcular la clausura de cada estado y cada conjunto de estados*
- *Computar la cadena babb*
- *Convertir el autómata en AFD*

Actividad en clases

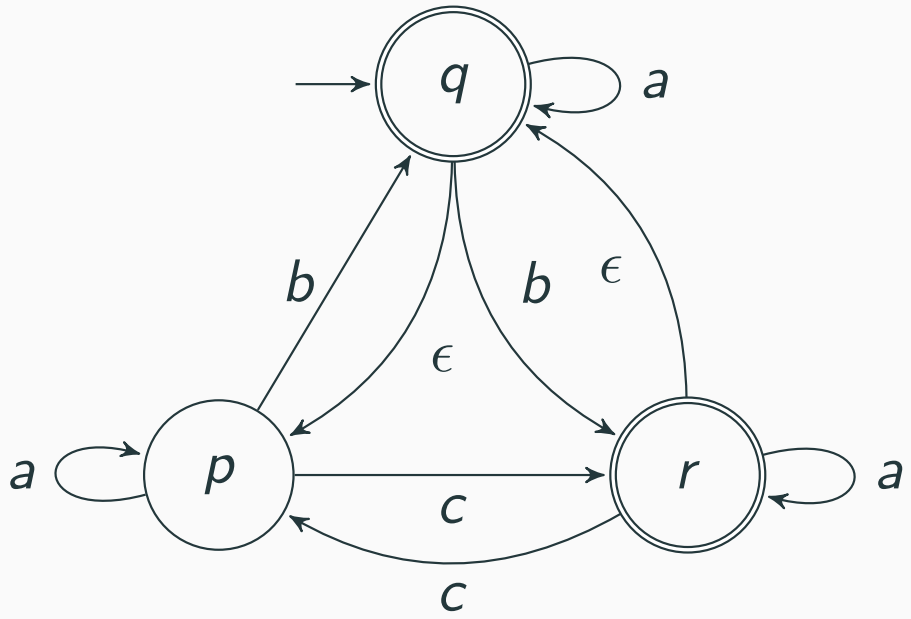
Actividad en clase

(a)



Actividad en clase

(b)



Operaciones regulares

Operaciones regulares

- Sean A y B dos lenguajes regulares

Operaciones regulares

- Sean A y B dos lenguajes regulares
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$

Operaciones regulares

- Sean A y B dos lenguajes regulares
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$
- **Concatenación**: $A \cdot B = \{xy : x \in A \text{ , } y \in B\}$

Operaciones regulares

- Sean A y B dos lenguajes regulares
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$
- **Concatenación**: $A \cdot B = \{xy : x \in A \ , \ y \in B\}$
- **Estrella**: $A^* = \{x_1 x_2 \dots x_k : x_i \in A\}$

Recordemos la pregunta

La operación de concatenación de lenguajes regulares, es cerrada?

Recordemos la pregunta

Ya podemos responder dicha pregunta, usando AFN-ϵ

Recordemos la pregunta

Primero, volvamos nuevamente a la operación de unión

Cerradura de la unión

Teorema

La clase de lenguajes regulares es cerrada bajo la operación de unión. Es decir, si L_1 y L_2 son lenguajes regulares, entonces $L_1 \cup L_2$ también es regular.

Cerradura de la unión

- Idea de la prueba.

Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1

Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2

Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cup L_2$

Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cup L_2$
- N tiene un nuevo estado inicial que salta hacia los estados iniciales de L_1 y L_2 usando transiciones ϵ

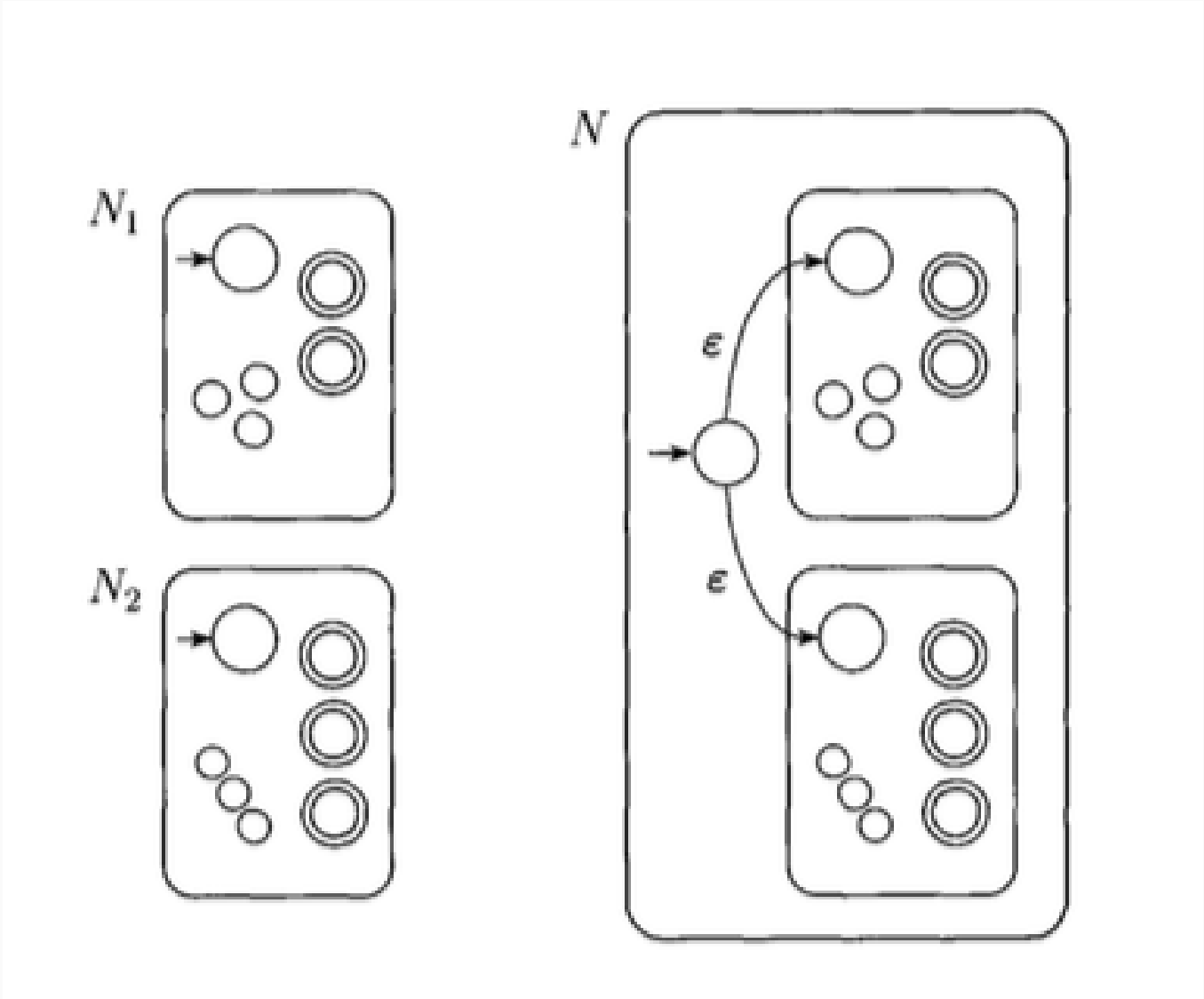
Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cup L_2$
- N tiene un nuevo estado inicial que salta hacia los estados iniciales de L_1 y L_2 usando transiciones ϵ
- N intenta adivinar de manera no deterministica cual de los dos AFN-ϵ acepta la cadena

Cerradura de la unión

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cup L_2$
- N tiene un nuevo estado inicial que salta hacia los estados iniciales de L_1 y L_2 usando transiciones ϵ
- N intenta adivinar de manera no deterministica cual de los dos AFN-ϵ acepta la cadena
- Si alguno de ellos acepta, N también acepta

Cerradura de la unión



Cerradura de la unión

- Prueba.

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1 \cup Q_2$

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1 \cup Q_2$
- $F = F_1 \cup F_2$

Cerradura de la unión

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1 \cup Q_2$
- $F = F_1 \cup F_2$
-

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{si } q \in Q_1, \\ \delta_2(q, a) & \text{si } q \in Q_2, \\ \{q_1, q_2\} & \text{si } q = q_0 \text{ y } a = \epsilon, \\ \emptyset & \text{si } q = q_0 \text{ y } a \neq \epsilon. \end{cases}$$

Cerradura de la concatenación

Teorema

La clase de lenguajes regulares es cerrada bajo la operación de concatenación. Es decir, si L_1 y L_2 son lenguajes regulares, entonces $L_1 \cdot L_2$ también es regular.

Cerradura de la concatenación

- Idea de la prueba.

Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1

Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2

Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cdot L_2$

Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cdot L_2$
- N simula N_1 y, al llegar a un estado final de N_1, intenta adivinar de manera no deterministica si la cadena restante es aceptada por N_2

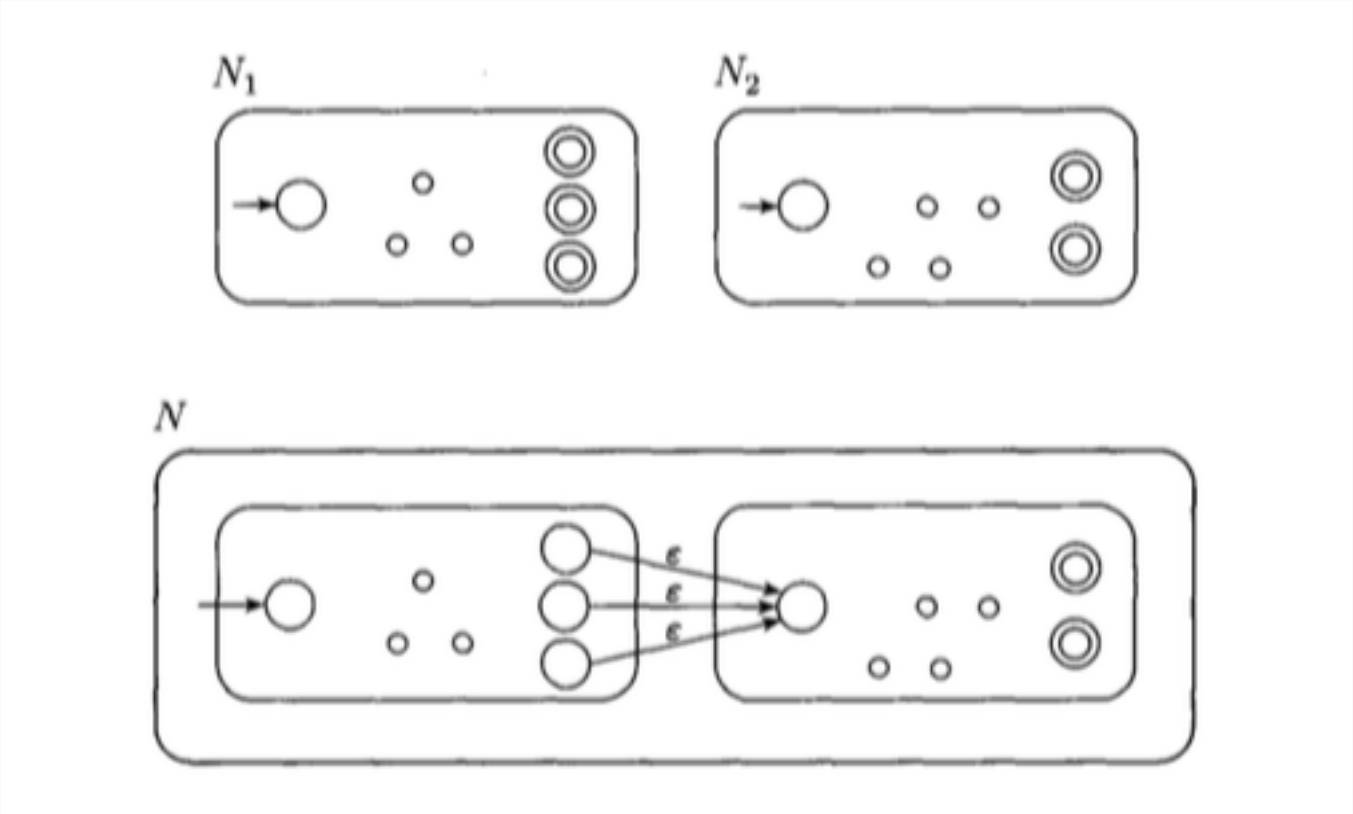
Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cdot L_2$
- N simula N_1 y, al llegar a un estado final de N_1, intenta adivinar de manera no deterministica si la cadena restante es aceptada por N_2
- Si N_2 acepta, entonces N también acepta

Cerradura de la concatenación

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Sea un AFN-ϵ N_2 que reconoce L_2
- Construimos un nuevo AFN-ϵ N que reconoce $L_1 \cdot L_2$
- N simula N_1 y, al llegar a un estado final de N_1, intenta adivinar de manera no deterministica si la cadena restante es aceptada por N_2
- Si N_2 acepta, entonces N también acepta
- Podemos pensar que N intenta adivinar de manera no determinística donde particionar la cadena

Cerradura de la concatenación



Cerradura de la concatenación

- Prueba.

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_1, F)$ según

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_1, F)$ según
- $Q = Q_1 \cup Q_2$

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_1, F)$ según
- $Q = Q_1 \cup Q_2$
- $F = F_2$

Cerradura de la concatenación

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Sea un AFN-ϵ $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ que reconoce L_2
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_1, F)$ según
- $Q = Q_1 \cup Q_2$
- $F = F_2$
-

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{si } q \in Q_1 \text{ y } q \notin F_1, \\ \delta_1(q,a) & \text{si } q \in F_1 \text{ y } a \neq \epsilon, \\ \delta_1(q,a) \cup \{q_2\} & \text{si } q \in F_1 \text{ y } a = \epsilon, \\ \delta_2(q,a) & \text{si } q \in Q_2. \end{cases}$$

Cerradura de la estrella

Teorema

La clase de lenguajes regulares es cerrada bajo la operación de estrella. Es decir, si L_1 es un lenguaje regular, entonces L_1^* también es regular.

Cerradura de la estrella

- Idea de la prueba.

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Construimos un nuevo AFN-ϵ N que reconoce L_1^*

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Construimos un nuevo AFN-ϵ N que reconoce L_1^*
- N aceptará la cadena si puede ser particionada en subcadenas, cada una de las cuales es aceptada por N_1

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Construimos un nuevo AFN-ϵ N que reconoce L_1^*
- N aceptará la cadena si puede ser particionada en subcadenas, cada una de las cuales es aceptada por N_1
- N simula N_1 y, al llegar a un estado final de N_1, regresa al inicio para intentar leer otra cadena aceptada por N_1

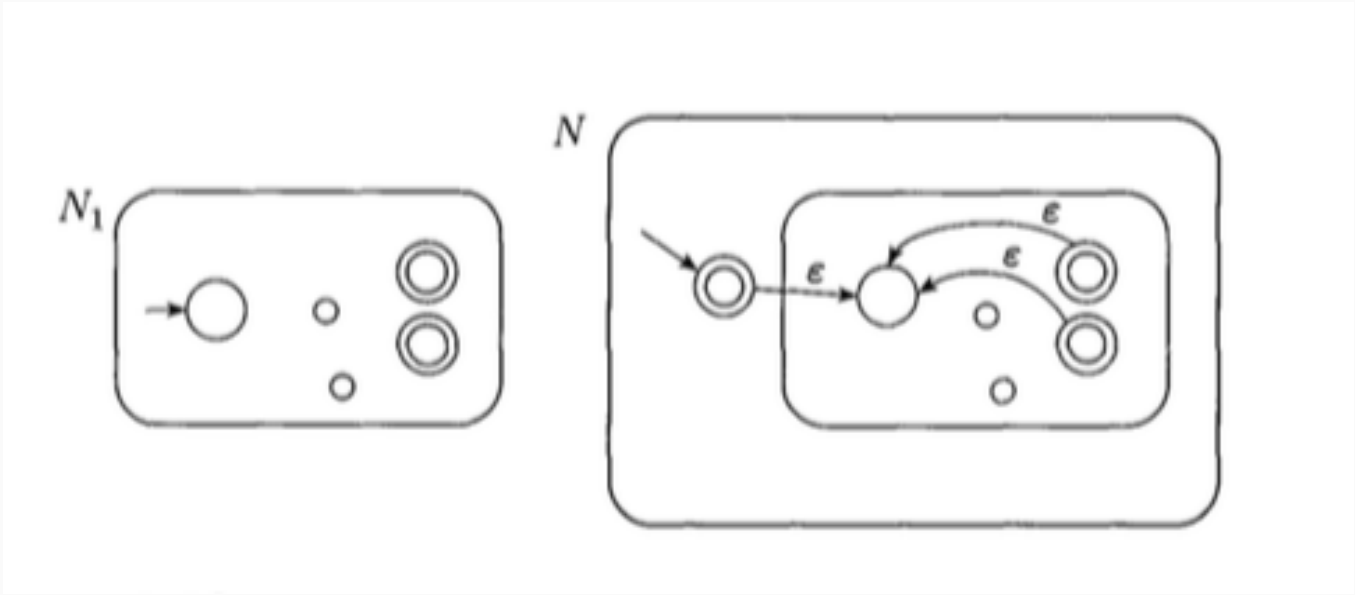
Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Construimos un nuevo AFN-ϵ N que reconoce L_1^*
- N aceptará la cadena si puede ser particionada en subcadenas, cada una de las cuales es aceptada por N_1
- N simula N_1 y, al llegar a un estado final de N_1, regresa al inicio para intentar leer otra cadena aceptada por N_1
- Si N_1 acepta, entonces N también acepta

Cerradura de la estrella

- Idea de la prueba.
- Sea un AFN-ϵ N_1 que reconoce L_1
- Construimos un nuevo AFN-ϵ N que reconoce L_1^*
- N aceptará la cadena si puede ser particionada en subcadenas, cada una de las cuales es aceptada por N_1
- N simula N_1 y, al llegar a un estado final de N_1, regresa al inicio para intentar leer otra cadena aceptada por N_1
- Si N_1 acepta, entonces N también acepta
- Como un caso aparte, N siempre debe aceptar la cadena vacía, para lo cual adicionamos un nuevo estado

Cerradura de la estrella



Cerradura de la estrella

- Prueba.

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1$

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1$
- $F = \{q_0\} \cup F_1$

Cerradura de la estrella

- Prueba.
- Sea un AFN-ϵ $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que reconoce L_1
- Construimos un nuevo AFN-ϵ $N = (Q, \Sigma, \delta, q_0, F)$ según
- $Q = \{q_0\} \cup Q_1$
- $F = \{q_0\} \cup F_1$
-

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{si } q \in Q_1 \text{ y } q \notin F_1, \\ \delta_1(q,a) & \text{si } q \in F_1 \text{ y } a \neq \epsilon, \\ \delta_1(q,a) \cup \{q_1\} & \text{si } q \in F_1 \text{ y } a = \epsilon, \\ \{q_1\} & \text{si } q = q_0 \text{ y } a = \epsilon, \\ \emptyset & \text{si } q = q_0 \text{ y } a \neq \epsilon \end{cases}$$

Ejercicios. Sean $L_1 = \{w : \text{la longitud de } w \text{ es máximo } 5\}$ y $L_2 = \{w : \text{termina en } 01\}$ sobre $\Sigma = \{0, 1\}$. Construir AFN-ϵ para los siguientes lenguajes

- $L_1 \cup L_2$

Ejercicios. Sean $L_1 = \{w : \text{la longitud de } w \text{ es máximo } 5\}$ y $L_2 = \{w : \text{termina en } 01\}$ sobre $\Sigma = \{0, 1\}$. Construir AFN-ϵ para los siguientes lenguajes

- $L_1 \cup L_2$
- $L_1 \cdot L_2$

Ejercicios. Sean $L_1 = \{w : \text{la longitud de } w \text{ es máximo } 5\}$ y $L_2 = \{w : \text{termina en } 01\}$ sobre $\Sigma = \{0, 1\}$. Construir AFN-ϵ para los siguientes lenguajes

- $L_1 \cup L_2$
- $L_1 \cdot L_2$
- L_1^* y L_2^*

Gracias