Teoría de la Computación

Clase 3: Equivalencia entre NFAs y DFAs

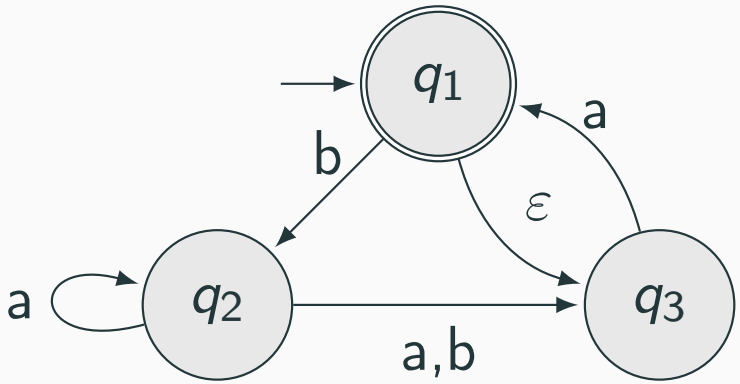
Mauro Artigiani

2 agosto 2021

Universidad del Rosario, Bogotá

Repaso

Un ejemplo de NFA



Definición

Somos listo para dar la definición formal de un `NFA`. Escribimos $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ y $\mathcal{P}(A)$ para el conjunto de potencia del conjunto A.

Definición

Un autómata finito no determinista es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado los estados;
2. Σ es un conjunto finito llamado alfabeto;
3. $\delta\colon Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ es la función de transición;
4. $q_0 \in Q$ es el estado inicial;
5. $F \subseteq Q$ es el conjunto de los estados aceptados;

Equivalencia entre NFAs y DFAs

Equivalencia entre NFAs y DFAs

Claramente, un autómata finito determinista es un caso especial de un autómata no determinista.

Equivalencia entre NFAs y DFAs

Claramente, un autómata finito determinista es un caso especial de un autómata no determinista. El hecho sorprendente y (muy) útil es que también se puede construir un autómata finito determinista que sea equivalente, es decir que reconozca el mismo lenguaje de un autómata no determinista.

Teorema

Cada autómata finito no determinista tiene un equivalente autómata finito determinista.

Demostración

Sea $N = (Q, \Sigma, \delta, q_0, F)$ un `NFA` que reconoce un lenguaje A.
Queremos construir un `DFA` $M = (Q', \Sigma, \delta', q_0', F')$ que reconozca el mismo lenguaje.

Demostración

Sea $N = (Q, \Sigma, \delta, q_0, F)$ un `NFA` que reconoce un lenguaje A. Queremos construir un `DFA` $M = (Q', \Sigma, \delta', q_0', F')$ que reconozca el mismo lenguaje.

Supongamos, por un momento, que no haya ninguna arista con etiqueta ε en N.

Demostración

Sea $N = (Q, \Sigma, \delta, q_0, F)$ un `NFA` que reconoce un lenguaje A. Queremos construir un `DFA` $M = (Q', \Sigma, \delta', q_0', F')$ que reconozca el mismo lenguaje.

Supongamos, por un momento, que no haya ninguna arista con etiqueta ε en N.

En cada momento, un autómata no determinista no está en un único estado, sino puede estar en varios estados en paralelo. Esto pero no es posible en un `DFA`. Para solucionar esto, definimos $Q' = \mathcal{P}(Q)$.

Demostración

Vamos a definir la función de transición δ'. Sean $R \in Q' = \mathcal{P}(Q)$ y $a \in \Sigma$, tenemos que definir $\delta'(R, a)$. Siendo R una colección de estados de N, la idea natural es seguir *todas* las aristas con etiqueta a que parten de los estados en R. Esto nos dará un subconjunto de estados de N, es decir un elemento de $\mathcal{P}(Q) = Q'$. En símbolos:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

Demostración

Vamos a definir la función de transición δ'. Sean $R \in Q' = \mathcal{P}(Q)$ y $a \in \Sigma$, tenemos que definir $\delta'(R, a)$. Siendo R una colección de estados de N, la idea natural es seguir *todas* las aristas con etiqueta a que parten de los estados en R. Esto nos dará un subconjunto de estados de N, es decir un elemento de $\mathcal{P}(Q) = Q'$. En símbolos:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

Definimos $q_0' = \{q_0\}$.

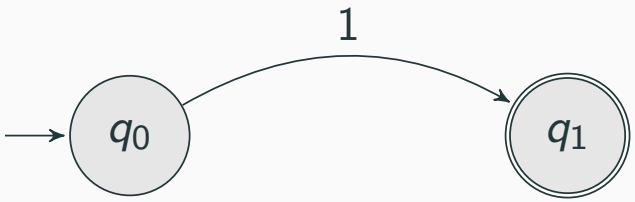
Demostración

Vamos a definir la función de transición δ'. Sean $R \in Q' = \mathcal{P}(Q)$ y $a \in \Sigma$, tenemos que definir $\delta'(R, a)$. Siendo R una colección de estados de N, la idea natural es seguir *todas* las aristas con etiqueta a que parten de los estados en R. Esto nos dará un subconjunto de estados de N, es decir un elemento de $\mathcal{P}(Q) = Q'$. En símbolos:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

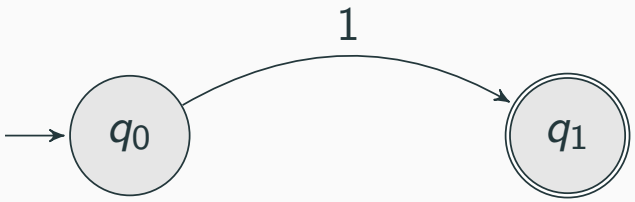
Definimos $q_0' = \{q_0\}$. Para terminar:
$F' = \{R \in Q', R \text{ contiene un estado final de } N\}$.

Interludio: ejemplo



	0	1
q_0	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\emptyset$

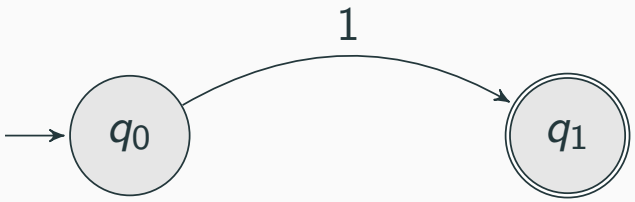
Interludio: ejemplo



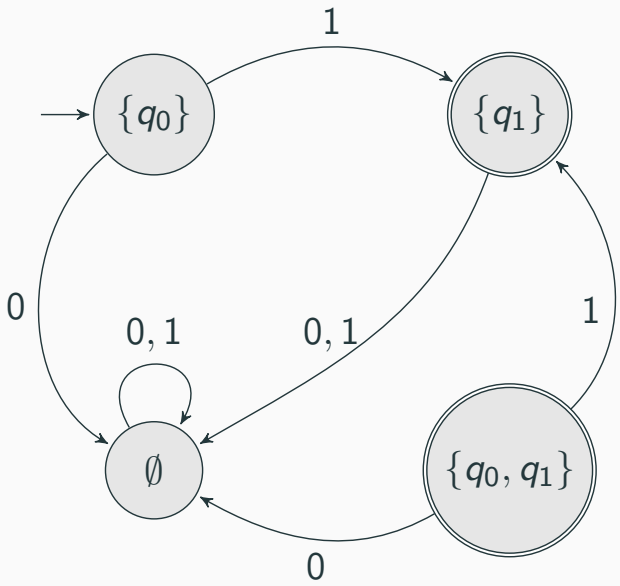
	0	1
q_0	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\emptyset$

	0	1
$\{q_0\}$	$\emptyset$	$\{q_1\}$
$\{q_1\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\emptyset$	$\{q_1\}$
$\emptyset$	$\emptyset$	$\emptyset$

Interludio: ejemplo

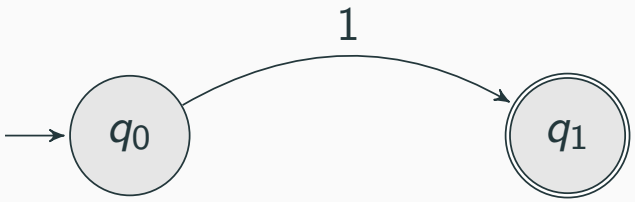


	0	1
q_0	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\emptyset$

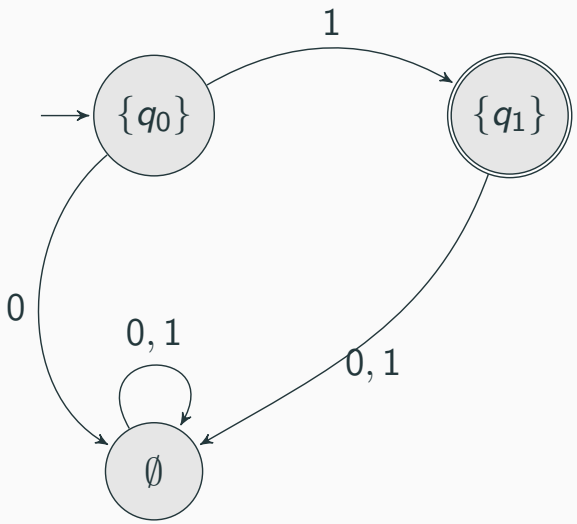


	0	1
$\{q_0\}$	$\emptyset$	$\{q_1\}$
$\{q_1\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\emptyset$	$\{q_1\}$
$\emptyset$	$\emptyset$	$\emptyset$

Interludio: ejemplo



	0	1
q_0	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\emptyset$



	0	1
$\{q_0\}$	$\emptyset$	$\{q_1\}$
$\{q_1\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\emptyset$	$\{q_1\}$
$\emptyset$	$\emptyset$	$\emptyset$

Demostración

Ajustamos ahora esta construcción para el caso donde sí haya etiquetas ε en N.

Demostración

Ajustamos ahora esta construcción para el caso donde sí haya etiquetas ε en N. El problema es que cuando un `NFA` llega a un estado con una arista con etiqueta ε, el autómata *inmediatamente* se divide y sigue en varias copias las aristas con ε.

Demostración

Ajustamos ahora esta construcción para el caso donde sí haya etiquetas ε en N. El problema es que cuando un `NFA` llega a un estado con una arista con etiqueta ε, el autómata *inmediatamente* se divide y sigue en varias copias las aristas con ε.

Para solucionar este problema, definimos, para $R \subset Q$,

$E(R) = \{q,$ se puede llegar a q desde R siguiendo 0 o más aristas $\varepsilon\}$.

a dónde me lleva dicho nodo

Demostración

Ajustamos ahora esta construcción para el caso donde sí haya etiquetas ε en N. El problema es que cuando un `NFA` llega a un estado con una arista con etiqueta ε, el autómata *inmediatamente* se divide y sigue en varias copias las aristas con ε.

Para solucionar este problema, definimos, para $R \subset Q$,

$E(R) = \{q,$ se puede llegar a q desde R siguiendo 0 o más aristas $\varepsilon\}$.

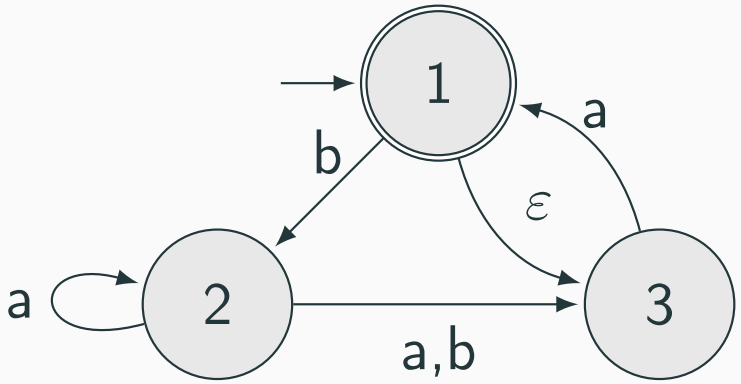
Finalmente

$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)),$$

y $E(\{q_0\})$ es el nuevo estado inicial. $\square$

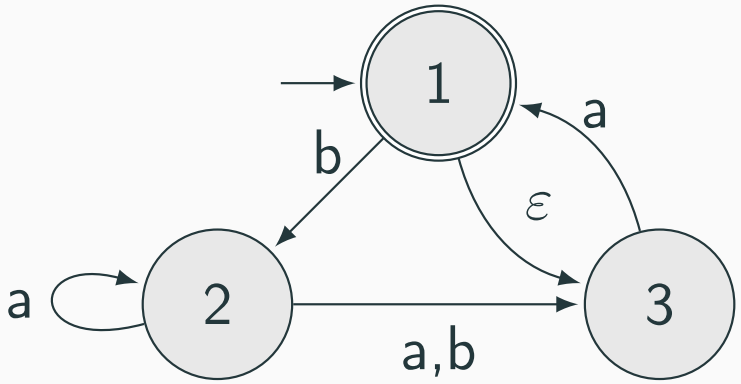
Un ejemplo

Vamos a aplicar esta construcción en un ejemplo. Sea N el `NFA` siguiente:



Un ejemplo

Vamos a aplicar esta construcción en un ejemplo. Sea N el `NFA` siguiente:

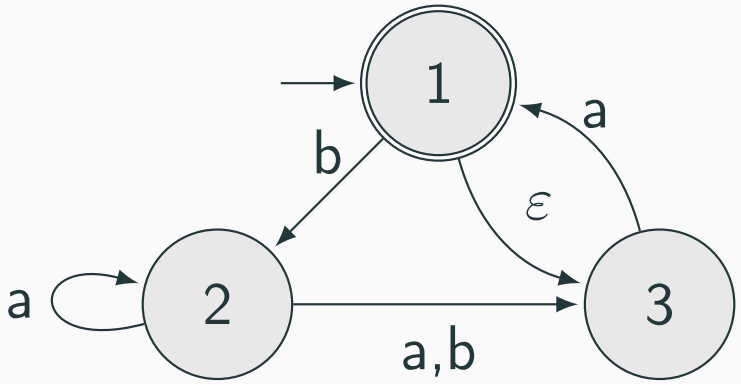


Empezamos con los estados:

$Q' = \mathcal{P}(Q) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.

Un ejemplo

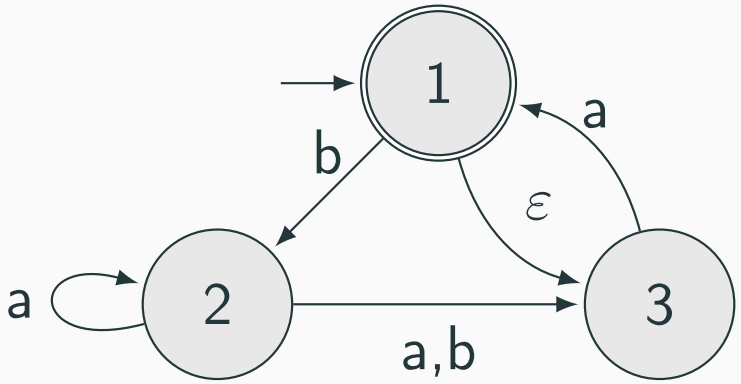
Vamos a aplicar esta construcción en un ejemplo. Sea N el `NFA` siguiente:



Empezamos con los estados:
$Q' = \mathcal{P}(Q) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.
Ahora, el estado inicial es $E(\{1\}) = \{1,3\}$.

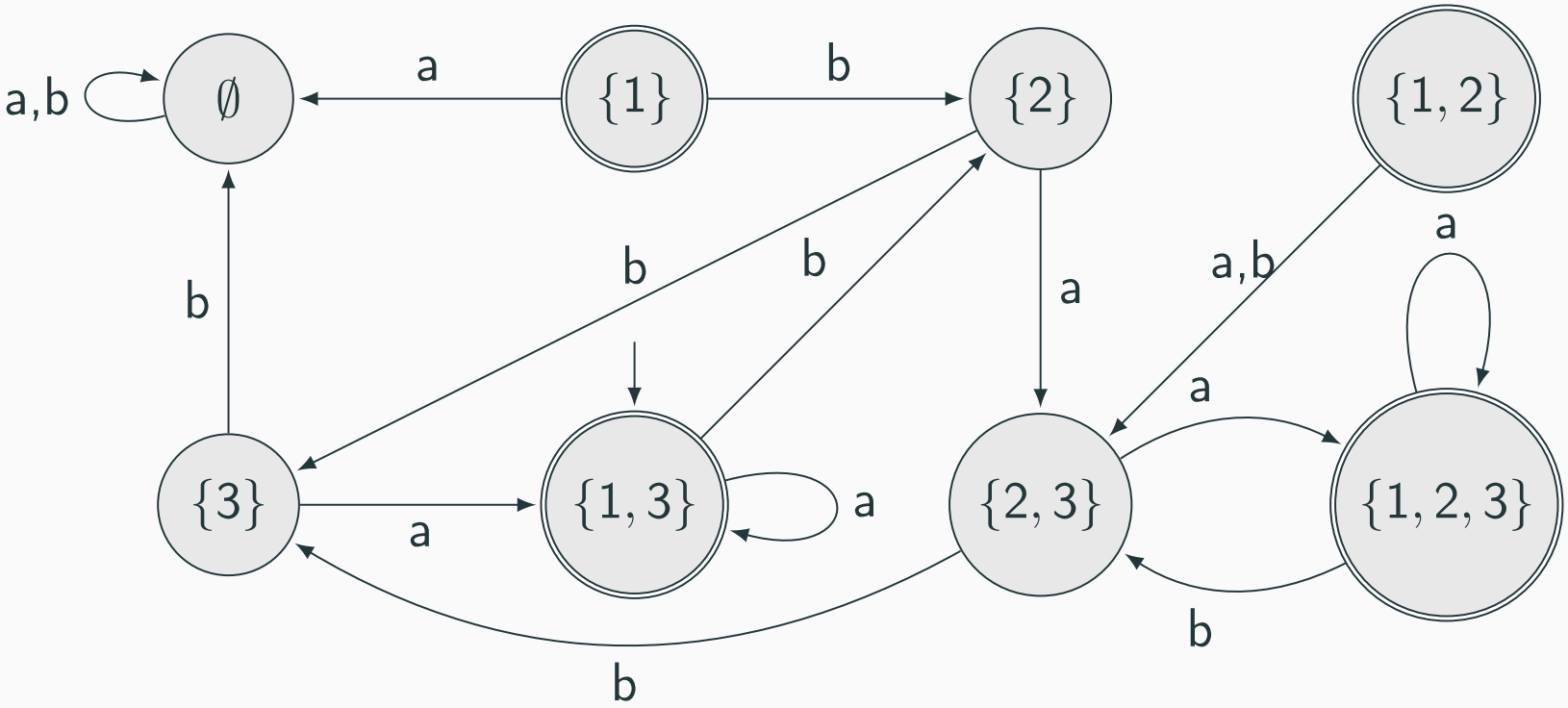
Un ejemplo

Vamos a aplicar esta construcción en un ejemplo. Sea N el `NFA` siguiente:

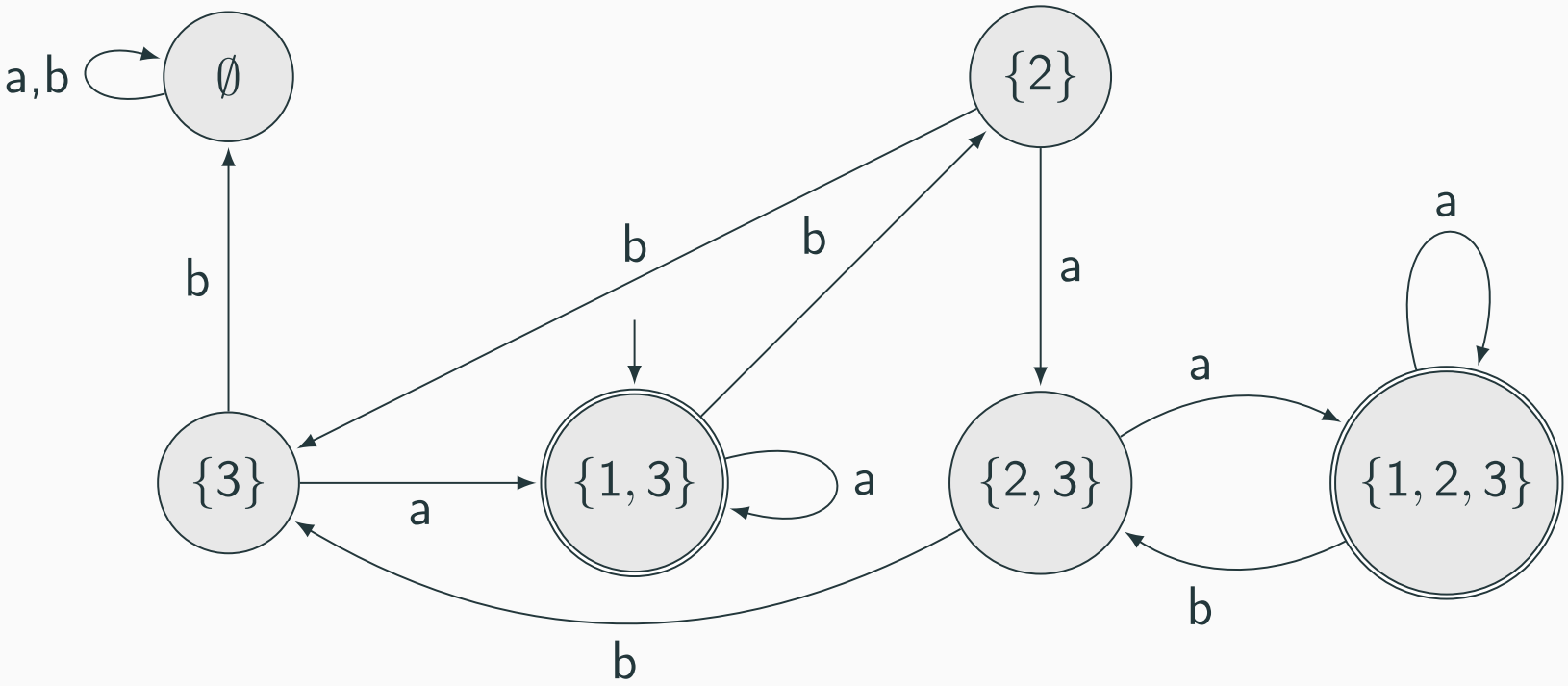


Empezamos con los estados:
$Q' = \mathcal{P}(Q) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.
Ahora, el estado inicial es $E(\{1\}) = \{1,3\}$. Los estados de aceptación son $F = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$.

Un ejemplo



Un ejemplo



Cierre bajo operaciones regulares

Concatenación

Ahora que sabemos que `NFA`s y `DFA`s son equivalentes podemos utilizar cualquiera de los dos para mostrar que un lenguaje es regular.

Concatenación

Ahora que sabemos que `NFAs` y `DFAs` son equivalentes podemos utilizar cualquiera de los dos para mostrar que un lenguaje es regular.

Teorema

La clase de los lenguajes regulares es cerrada bajo concatenación. Es decir, si A_1 y A_2 son dos lenguajes regulares, entonces también $A_1 \circ A_2$ es un lenguaje regular.

Concatenación

Ahora que sabemos que NFAs y DFAs son equivalentes podemos utilizar cualquiera de los dos para mostrar que un lenguaje es regular.

Teorema

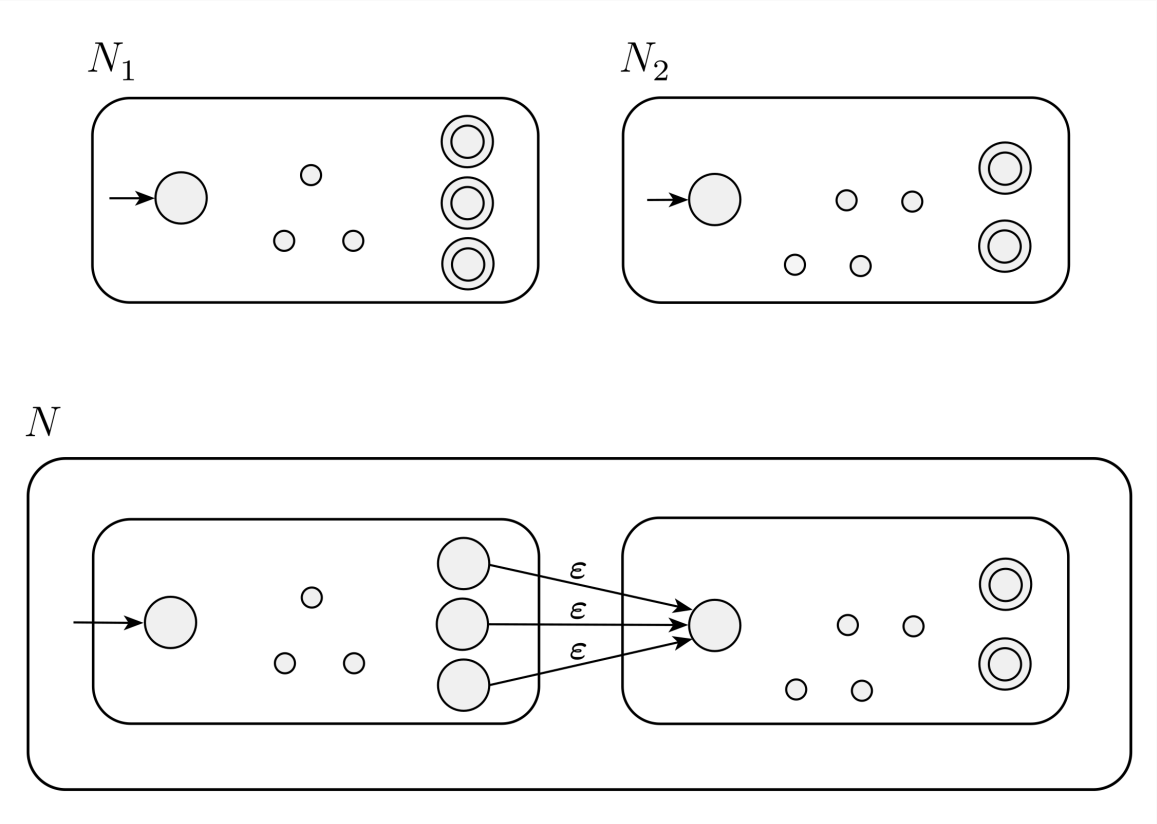
La clase de los lenguajes regulares es cerrada bajo concatenación. Es decir, si A_1 y A_2 son dos lenguajes regulares, entonces también $A_1 \circ A_2$ es un lenguaje regular.

La idea es construir un NFA que reconozca $A_1 \circ A_2$. Este autómata debería aceptar palabras construidas concatenando una palabra de A_1 con una de A_2. Supongamos que $w = xy \in A_1 \circ A_2$. El autómata no puede saber hasta donde llegue la palabra x y donde empiece la palabra y, por esto vamos a aprovechar del no determinismo.

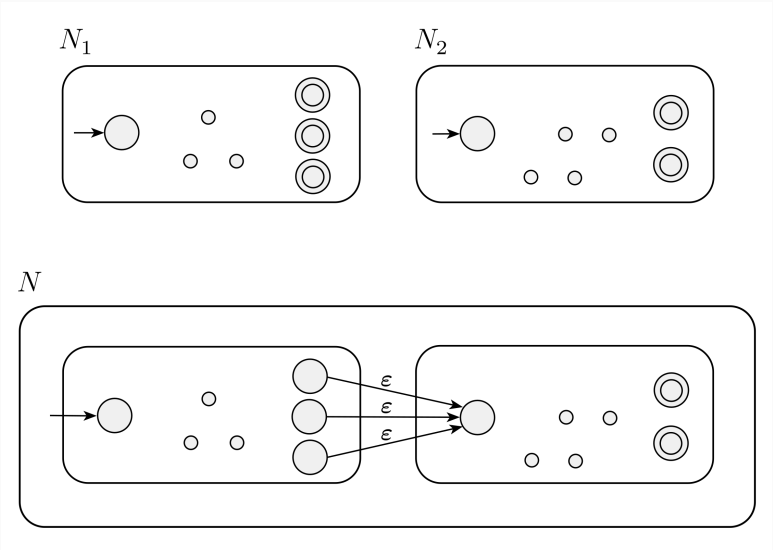
Demostración

Sea $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ un `NFA` que reconoce A_1 y $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ un `NFA` que reconoce A_2.

Vamos a construir un `NFA` N que reconozca $A_1 \circ A_2$ *concatenando* N_1 con N_2.



Demostración



Formalmente $N = (Q, \Sigma, \delta, q_1, F_2)$, donde: $Q = Q_1 \cup Q_2$; q_1 es el estado inicial de N_1; F_2 son los estados finales de N_2 y

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

□

Potencia

Teorema

La clase de los lenguajes regulares es cerrada bajo potencia. Es decir, si A es un lenguaje regular entonces también A^* lo es.

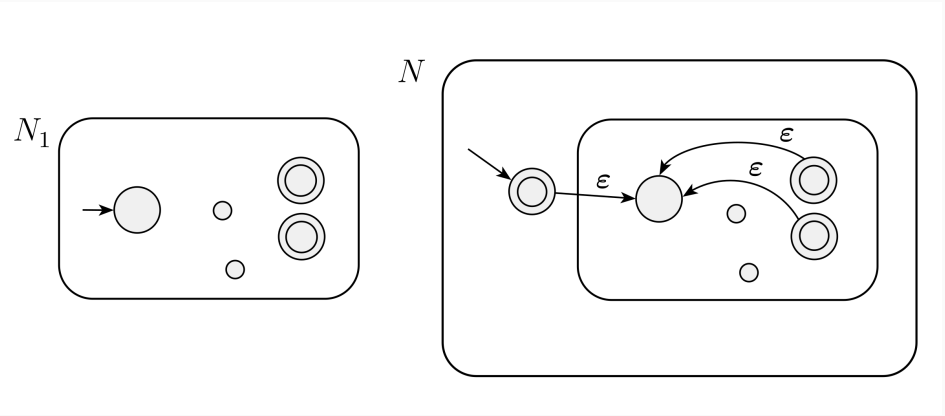
Potencia

Teorema

La clase de los lenguajes regulares es cerrada bajo potencia. Es decir, si A es un lenguaje regular entonces también A^* lo es.

Siendo A un lenguaje regular existe un `NFA` $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ que lo reconoce. Debemos construir un `NFA` N que reconozca la palabra vacía y cualquier concatenación de palabras en A. Es decir: debemos detectar que una palabra sea formada de subpalabras en A.

Demostración



Construimos $N = (Q, \Sigma, \delta, q_0, F)$, donde $Q = \{q_0\} \cup Q_1$; q_0 es un *nuevo* estadio inicial; $F = \{q_0\} \cup F_1$ y:

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1 \text{ y } q \notin F_1; \\ \delta_1(q,a), & q \in F_1 \text{ y } a \neq \varepsilon; \\ \delta_1(q,a) \cup \{q_1\}, & q \in F_1 \text{ y } a = \varepsilon; \\ \{q_1\}, & q = q_0 \text{ y } a = \varepsilon; \\ \emptyset, & q = q_0 \text{ y } a \neq \varepsilon. \end{cases} \qquad \square$$

Resumen

Resumen

Hoy aprendimos:

- A construir un `DFA` equivalente a un `NFA` dado;
- A construir un `NFA` que reconozca la concatenación y la potencia de lenguajes regulares.