Grado en Ingeniería Informática

Computabilidad y Algoritmia

Tema 2: Autómatas finitos y lenguajes regulares

F. de Sande

Curso 2024-2025

Indice

Indice

Introducción

Hasta ahora...

- La mayoría de los lenguajes considerados han sido bastante sencillos
- Los procesos vistos para determinar qué cadenas pertenecen a un lenguaje sobre un alfabeto Σ resultan laboriosos, excepto para Σ^* y algún otro lenguaje sencillo

Introducción

Hasta ahora...

- La mayoría de los lenguajes considerados han sido bastante sencillos
- Los procesos vistos para determinar qué cadenas pertenecen a un lenguaje sobre un alfabeto Σ resultan laboriosos, excepto para Σ^* y algún otro lenguaje sencillo

A partir de ahora...

- Nuestro objetivo será la definición de lenguajes:
  definir exactamente qué cadenas componen un lenguaje

Introducción

Hasta ahora...

- La mayoría de los lenguajes considerados han sido bastante sencillos
- Los procesos vistos para determinar qué cadenas pertenecen a un lenguaje sobre un alfabeto Σ resultan laboriosos, excepto para Σ^* y algún otro lenguaje sencillo

A partir de ahora...

- Nuestro objetivo será la definición de lenguajes: **definir exactamente qué cadenas componen un lenguaje**

Antes de empezar a definir lenguajes, estudiemos Σ^*

- Todos los lenguajes sobre Σ son sublenguajes de Σ^*
- ¿Cuántas cadenas tiene Σ^*?
- ¿Cuántos sublenguajes tiene Σ^* para un alfabeto Σ en particular?

¿Cuántas cadenas tiene Σ^*?

Orden lexicográfico para las cadenas

¿Cuántas cadenas tiene Σ^*?

Orden lexicográfico para las cadenas

- Establecer (arbitrariamente/alfabéticamente) un orden sobre los símbolos de Σ.

¿Cuántas cadenas tiene Σ^*?

Orden lexicográfico para las cadenas

- Establecer (arbitrariamente/alfabéticamente) un orden sobre los símbolos de Σ.
- Ordenar las cadenas en orden creciente de longitud.

¿Cuántas cadenas tiene Σ^*?

Orden lexicográfico para las cadenas

- Establecer (arbitrariamente/alfabéticamente) un orden sobre los símbolos de Σ.
- Ordenar las cadenas en orden creciente de longitud.
- Para cadenas de igual longitud considerar el orden de sus símbolos.

Ejemplo: sea $\Sigma = \{a, b\}$

ε	0
a	1
b	2
aa	3
ab	4
ba	5
bb	6
aaa	7
aab	8
...	

¿Cuántas cadenas tiene Σ^*?

De forma general, sea $\Sigma = \{a_1, a_2, ..., a_n\}$

Podremos numerar las cadenas de Σ^* de la misma forma:

$$\begin{array}{cc}
\varepsilon & 0 \\
a_1 & 1 \\
a_2 & 2 \\
... & ... \\
a_n & n \\
a_1a_1 & n+1 \\
a_1a_2 & n+2 \\
... & ...
\end{array}$$

¿Cuántas cadenas tiene Σ^*?

De forma general, sea $\Sigma = \{a_1, a_2, ..., a_n\}$

Podremos numerar las cadenas de Σ^* de la misma forma:

$$\begin{array}{cc} \varepsilon & 0 \\ a_1 & 1 \\ a_2 & 2 \\ ... & ... \\ a_n & n \\ a_1a_1 & n+1 \\ a_1a_2 & n+2 \\ ... & ... \end{array}$$

Forma de relacionar las cadenas de Σ^* con los números naturales

- Cada cadena está representada por un único número natural.
- Cada número natural representa a una única cadena.
- **Para todo alfabeto Σ, Σ^* es infinito numerable**.
- Por lo tanto, todo $L \subseteq \Sigma^*$ será finito o infinito numerable.

¿Cuántos sublenguajes de Σ^* existen?

- Equivale a determinar cuántos lenguajes hay sobre el alfabeto Σ
- ¿Existe un método para enumerarlos?

El conjunto de todos los lenguajes sobre Σ no es numerable

- No existe ningún método de especificación de lenguajes que sea capaz de definir *todos* los lenguajes sobre un alfabeto.
- Dado un método de representación de lenguajes, hay lenguajes que no son representables.
- Hay métodos que tienen mayor expresividad que otros (definen más lenguajes).
- Estudiaremos distintos mecanismos de especificación de lenguajes.

Lenguajes regulares

Definición

Sea Σ un alfabeto. El conjunto de los **lenguajes regulares** sobre Σ se define recursivamente:

a) $\emptyset$ es un lenguaje regular
b) $\{\varepsilon\}$ es un lenguaje regular
c) Para todo $a \in \Sigma$, $\{a\}$ es un lenguaje regular
d) Si A y B son lenguajes regulares, entonces $A \cup B$, $A \cdot B$ y A^* son lenguajes regulares
e) Ningún otro lenguaje sobre Σ es regular

El conjunto de los lenguajes regulares sobre Σ está formado por: $\emptyset$, $\{\varepsilon\}$, los lenguajes unitarios, y los obtenidos por unión, concatenación y cierre de Kleene de otros que sean regulares

Lenguajes regulares

Ejemplo

Sea $\Sigma = \{0, 1\}$, algunos ejemplos de lenguajes regulares sobre Σ son:

- $L_1 = \emptyset$
- $L_2 = \{\varepsilon\}$
- $L_3 = \{0\}$
- $L_4 = \{1\}$
- $L_5 = \{0, 1\}$
- $L_6 = \{\varepsilon, 0, 00, 000, 0000, ...\}$
- $L_7 = \{01, 10, 001\}$
    - $L_3 \cdot L_4 = \{0\} \cdot \{1\} = \{01\} = L_{A_1}$
    - $L_4 \cdot L_3 = \{1\} \cdot \{0\} = \{10\} = L_{A_2}$
    - $L_3 \cdot L_3 = \{0\} \cdot \{0\} = \{00\} = L_{A_3}$
    - $L_{A_3} \cdot L_4 = \{00\} \cdot \{1\} = \{001\} = L_{A_4}$
    - $L_{A_1} \cup L_{A_2} \cup L_{A_4} = L_7$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$
  - $L_8 = (\{0\} \cup \{1\})^*$
- $L_9 = \{1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, ...\}$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$
  - $L_8 = (\{0\} \cup \{1\})^*$
- $L_9 = \{1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, ...\}$
  - $L_9 = L_8 \cdot \{1\}$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$
  - $L_8 = (\{0\} \cup \{1\})^*$
- $L_9 = \{1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, ...\}$
  - $L_9 = L_8 \cdot \{1\}$
- $L_{10} = \{0^i \mid i \geq 0\} = \{\varepsilon, 0, 00, 000, ...\}$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$
  - $L_8 = (\{0\} \cup \{1\})^*$
- $L_9 = \{1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, ...\}$
  - $L_9 = L_8 \cdot \{1\}$
- $L_{10} = \{0^i \mid i \geq 0\} = \{\varepsilon, 0, 00, 000, ...\}$
- $L_{11} = \{0^i 1^j \mid i, j \geq 0\} = \{\varepsilon, 0, 1, 01, 001, 0011, ...\}$

Lenguajes regulares

Ejemplo

- $L_8 = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, ...\} = \Sigma^*$
  - $L_8 = \{0, 1\}^*$
  - $L_8 = (\{0\} \cup \{1\})^*$
- $L_9 = \{1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, ...\}$
  - $L_9 = L_8 \cdot \{1\}$
- $L_{10} = \{0^i \mid i \geq 0\} = \{\varepsilon, 0, 00, 000, ...\}$
- $L_{11} = \{0^i 1^j \mid i, j \geq 0\} = \{\varepsilon, 0, 1, 01, 001, 0011, ...\}$
- $L_{12} = \{(01)^i \mid i \geq 0\} = \{\varepsilon, 01, 0101, 010101, ...\}$

Expresiones regulares

Definición

Sea Σ un alfabeto, las expresiones regulares sobre Σ se definen recursivamente utilizando la notación siguiente:

- $\emptyset$ y ε son expresiones regulares
- Para todo $a \in \Sigma$, a es una expresión regular
- Si r y s son expresiones regulares, entonces:
  - $r \mid s$ es expresión regular
  - $r \cdot s$ es expresión regular
  - $r*$ es expresión regular
- No hay otra forma que las anteriores para construir una expresión regular

Expresiones regulares

Definición

Sea Σ un alfabeto, las expresiones regulares sobre Σ se definen recursivamente utilizando la notación siguiente:

- $\emptyset$ y ε son expresiones regulares
- Para todo $a \in \Sigma$, a es una expresión regular
- Si r y s son expresiones regulares, entonces:
  - $r \mid s$ es expresión regular
  - $r \cdot s$ es expresión regular
  - $r*$ es expresión regular
- No hay otra forma que las anteriores para construir una expresión regular

Las expresiones regulares son un mecanismo formal que permite describir (especificar) lenguajes regulares

Notación

Toda expresión regular **representa** a un lenguaje regular

- Sea r una expresión regular, el lenguaje regular representado por r se denota $L(r)$
- Si r y s son expresiones regulares sobre el mismo alfabeto, entonces si $L(r) = L(s)$ se dice que r y s son equivalentes $(r = s)$

Notación

Toda expresión regular **representa** a un lenguaje regular

- Sea r una expresión regular, el lenguaje regular representado por r se denota $L(r)$
- Si r y s son expresiones regulares sobre el mismo alfabeto, entonces si $L(r) = L(s)$ se dice que r y s son equivalentes $(r = s)$

Expresión regular r	**Lenguaje representado** $L(r)$
$\emptyset$	$\emptyset$
ε	$\{\varepsilon\}$
a	$\{a\}$
$r_1 \cdot r_2$	$L(r_1) \cdot L(r_2)$
$r_1 \mid r_2$	$L(r_1) \cup L(r_2)$
r_1*	$L(r_1)^*$

Orden de precedencia de los operadores

De mayor a menor precedencia

- $*$
- $\cdot$
- $|$

Orden de precedencia de los operadores

De mayor a menor precedencia

- $*$
- $\cdot$
- $|$

Ejemplos

- $((0(1*))|0) = 01*|0$
- $ab*c|e = (a(b*)c)|e$
- Para algunas expresiones será necesario usar paréntesis:
  - $(a|b)(a|b)$

Equivalencia de expresiones

Existen muchas expresiones regulares que representan el mismo lenguaje.

Equivalencia de expresiones

Existen muchas expresiones regulares que representan el mismo lenguaje.

Ejemplo

- $r = (a{*}b){*}$
- $s = \varepsilon \mid (a|b){*}b$

Equivalencia de expresiones

Existen muchas expresiones regulares que representan el mismo lenguaje.

Ejemplo

- $r = (a{*}b){*}$
- $s = \varepsilon \mid (a|b){*}b$
- $L(r) = L(s) = \{\varepsilon, b, ab, bb, aab, abab, aaab, ...\}$

Equivalencia de expresiones

Existen muchas expresiones regulares que representan el mismo lenguaje.

Ejemplo

- $r = (a{*}b){*}$
- $s = \varepsilon \mid (a|b){*}b$
- $L(r) = L(s) = \{\varepsilon, b, ab, bb, aab, abab, aaab, ...\}$
- Lenguaje sobre $\Sigma = \{a, b\}$ con cero o más aes y bes seguidas de b

Obsérvese la ambigüedad cuando un lenguaje se describe usando lenguaje natural.
¿Pertenece ϵ al lenguaje anterior?

Equivalencia de expresiones

Equivalencia entre expresiones regulares que se demuestran teniendo en cuenta propiedades de la *unión* de lenguajes

- $r|(s|t) = (r|s)|t$
- $r|s = s|r$
- $\emptyset|r = r|\emptyset = r$
- $r|r = r$

Equivalencia de expresiones

Equivalencia entre expresiones regulares que se demuestran teniendo en cuenta propiedades de la *unión* de lenguajes

- $r|(s|t) = (r|s)|t$
- $r|s = s|r$
- $\emptyset|r = r|\emptyset = r$
- $r|r = r$

Equivalencia entre expresiones regulares que se demuestran teniendo en cuenta propiedades de la *concatenación* de lenguajes

- $r(st) = (rs)t$
- $rs \neq sr$
- $\varepsilon \cdot r = r \cdot \varepsilon = r$
- $\emptyset \cdot r = r \cdot \emptyset = \emptyset$

Equivalencia de expresiones

Equivalencia entre expresiones regulares que se demuestran teniendo en cuenta propiedades de la *clausura* de lenguajes

- $\varepsilon * = \varepsilon$
- $\emptyset * = \varepsilon$
- $r* = r** = r* r* = (\varepsilon|r)* = r*(r|\varepsilon) = (r|\varepsilon)r* = \varepsilon|rr*$
- $rr* = r*r$
- $(r|s)* = (r*|s*)* = (r* s*)* = (r*s)* r* = r*(sr*)*$
- $r(sr)* = (rs)*r$
- $(r*s)* = \varepsilon|(r|s)*s$
- $(rs*)* = \varepsilon|r(r|s)*$
- $s(r|\varepsilon)*(r|\varepsilon)|s = sr*$

Equivalencia de expresiones

Demostración por reasociación

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1 r_1)...(s_n r_n)$ para algún $n \geq 0$

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs) * r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1r_1)...(s_nr_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1r_1)...(s_nr_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.
- Con lo cual $w = (r_0s_1)(r_1s_2)...(r_{n-1}s_n)r_n$

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)*r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1r_1)...(s_nr_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.
- Con lo cual $w = (r_0s_1)(r_1s_2)...(r_{n-1}s_n)r_n$
- Por lo tanto, $w \in L((rs)*r)$

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1 r_1)...(s_n r_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.
- Con lo cual $w = (r_0 s_1)(r_1 s_2)...(r_{n-1} s_n) r_n$
- Por lo tanto, $w \in L((rs)* r)$
- Así se prueba que L$(r(sr)*) \subseteq L((rs)* r)$

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1r_1)...(s_nr_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.
- Con lo cual $w = (r_0s_1)(r_1s_2)...(r_{n-1}s_n)r_n$
- Por lo tanto, $w \in L((rs)* r)$
- Así se prueba que L$(r(sr)*) \subseteq L((rs)* r)$
- $L((rs)* r) \subseteq L(r(sr)*)$ se podría demostrar de manera similar

Equivalencia de expresiones

Demostración por reasociación

Por ejemplo, $r(sr)* = (rs)* r$

- Si $w \in L(r(sr)*)$, entonces $w = r_0(s_1r_1)...(s_nr_n)$ para algún $n \geq 0$
- Puesto que la concatenación es asociativa, se puede reasociar la expresión.
- Con lo cual $w = (r_0s_1)(r_1s_2)...(r_{n-1}s_n)r_n$
- Por lo tanto, $w \in L((rs)* r)$
- Así se prueba que L$(r(sr)*) \subseteq L((rs)* r)$
- $L((rs)* r) \subseteq L(r(sr)*)$ se podría demostrar de manera similar

Demostración por uso de igualdades ya conocidas

También se pueden demostrar haciendo uso de igualdades ya conocidas

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*
- $| \equiv$ disyunción $\equiv$ *o*

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*
- $| \equiv$ disyunción $\equiv$ *o*
- $* \equiv$ asterisco $\equiv$ *cero o más repeticiones*

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*
- $| \equiv$ disyunción $\equiv$ *o*
- $* \equiv$ asterisco $\equiv$ *cero o más repeticiones*
- $+ \equiv$ más $\equiv$ *una o más repeticiones*

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot$ $\equiv$ concatenación $\equiv$ *seguido de*
- $|$ $\equiv$ disyunción $\equiv$ *o*
- $*$ $\equiv$ asterisco $\equiv$ *cero o más repeticiones*
- $+$ $\equiv$ más $\equiv$ *una o más repeticiones*

Ejemplos

- $(a|b)(a|b)(a|b)$: a *ó* b *seguido de* a *ó* b *seguido de* a *ó* b

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*
- $| \equiv$ disyunción $\equiv$ *o*
- $* \equiv$ asterisco $\equiv$ *cero o más repeticiones*
- $+ \equiv$ más $\equiv$ *una o más repeticiones*

Ejemplos

- $(a|b)(a|b)(a|b)$: a *ó* b *seguido de* a *ó* b *seguido de* a *ó* b
- $(a|b)*$: *cero o más repeticiones de* a *ó* b

Lectura intuitiva de expresiones regulares

Notación y lectura intuitiva

- $\cdot \equiv$ concatenación $\equiv$ *seguido de*
- $| \equiv$ disyunción $\equiv$ *o*
- $* \equiv$ asterisco $\equiv$ *cero o más repeticiones*
- $+ \equiv$ más $\equiv$ *una o más repeticiones*

Ejemplos

- $(a|b)(a|b)(a|b)$: a *ó* b *seguido de* a *ó* b *seguido de* a *ó* b
- $(a|b)*$: *cero o más repeticiones de* a *ó* b
- $aab(aa)*$: *La cadena* aab *seguida de cero o más repeticiones de* aa

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$: Cadenas binarias que no tienen dos 0 consecutivos y que comienzan por 1.

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$: Cadenas binarias que no tienen dos 0 consecutivos y que comienzan por 1.
- $1(1|0) * 1$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$: Cadenas binarias que no tienen dos 0 consecutivos y que comienzan por 1.
- $1(1|0) * 1$: Cadenas binarias que comienzan y acaban por 1 (con longitud ≥ 2).

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$: Cadenas binarias que no tienen dos 0 consecutivos y que comienzan por 1.
- $1(1|0) * 1$: Cadenas binarias que comienzan y acaban por 1 (con longitud $\geq$ 2).
- $1 * 01 * 01*$:

Ejemplos de expresiones regulares

Describir los conjuntos que representan las siguientes expresiones regulares

- $0*$: Cadenas de cero o más símbolos 0 (longitud arbitraria).
- $(10)*$: Secuencias de cadenas 10 (longitud arbitraria).
- $(0|1)*$: Cadenas binarias de longitud arbitraria.
- $(0|1) * 1(0|1)*$: Cadenas binarias que contienen al menos un 1.
- $(0|1) * 00(0|1)*$: Cadenas binarias con al menos dos 0 consecutivos.
- $(1|10)*$: Cadenas binarias que no tienen dos 0 consecutivos y que comienzan por 1.
- $1(1|0) * 1$: Cadenas binarias que comienzan y acaban por 1 (con longitud ≥ 2).
- $1 * 01 * 01*$: Cadenas binarias con sólo dos 0.

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1) * 0$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1) * 0$
- Cadenas binarias con sólo un 0:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1) * 0$
- Cadenas binarias con sólo un 0: $1 * 01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1: $1(1|0)*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1: $1(1|0)*$
- Cadenas binarias de longitud par:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1: $1(1|0)*$
- Cadenas binarias de longitud par: $((0|1)(0|1))*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1: $1(1|0)*$
- Cadenas binarias de longitud par: $((0|1)(0|1))*$
- Cadenas con un número par de 1:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas binarias que acaban en 0: $(0|1)*0$
- Cadenas binarias con sólo un 0: $1*01*$
- Cadenas que si contienen al menos un 1, todo 1 va seguido y precedido de 0: $((01)*0)*$
- Cadenas que comienzan por 1: $1(1|0)*$
- Cadenas binarias de longitud par: $((0|1)(0|1))*$
- Cadenas con un número par de 1: $0*(10*10*)*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0,1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1) * 0|1(0|1) * 1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1 * (011*) * (0|1*)$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0,1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1*(011*)*(0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1) * 0|1(0|1) * 1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1 * (011*) * (0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez: $1 * (011*) * 00(11 * 0) * 1*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1*(011*)*(0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez: $1*(011*)*00(11*0)*1*$
- Cadenas que contienen la subcadena 101:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1*(011*)*(0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez: $1*(011*)*00(11*0)*1*$
- Cadenas que contienen la subcadena 101: $(0|1)*101(0|1)*$

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1*(011*)*(0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez: $1*(011*)*00(11*0)*1*$
- Cadenas que contienen la subcadena 101: $(0|1)*101(0|1)*$
- Cadenas que no contienen la subcadena 01:

Ejemplos de expresiones regulares

Escribir expresiones regulares sobre $\{0, 1\}$ que representen a los siguientes conjuntos:

- Cadenas que comienzan y terminan con el mismo símbolo: $0(0|1)*0|1(0|1)*1|0|1$
- Cadenas en las que no aparece la subcadena 00: $1*(011*)*(0|1*)$
- Cadenas en las que la subcadena 00 aparece una sola vez: $1*(011*)*00(11*0)*1*$
- Cadenas que contienen la subcadena 101: $(0|1)*101(0|1)*$
- Cadenas que no contienen la subcadena 01: $1*0*$

Extensiones de operadores

Algunas extensiones de operadores

- $r+ = rr*$
- $r? = r|\varepsilon$
- ˆ $\equiv$ Comienzo de línea
- $\$$ $\equiv$ Final de línea
- . $\equiv$ Cualquier caracter distinto de retorno de carro
- \\. $\equiv$ El caracter punto
- * $\equiv$ El caracter asterisco

Extensiones de operadores

Algunas extensiones de operadores

- Sea Σ un conjunto ordenado:
  - $[a - z] \equiv a|b|c|...|z$ (rangos)
  - $[bdfg] \equiv b|d|f|g$ (un elemento de entre un conjunto de posibilidades)

Extensiones de operadores

Algunas extensiones de operadores

- Sea Σ un conjunto ordenado:
  - $[a-z] \equiv a|b|c|...|z$ (rangos)
  - $[bdfg] \equiv b|d|f|g$ (un elemento de entre un conjunto de posibilidades)
- $[\hat{}\, a-z] \equiv$ Ninguna letra minúscula

Extensiones de operadores

Algunas extensiones de operadores

- Sea Σ un conjunto ordenado:
  - $[a - z] \equiv a|b|c|...|z$ (rangos)
  - $[bdfg] \equiv b|d|f|g$ (un elemento de entre un conjunto de posibilidades)
- $[\hat{}\, a - z] \equiv$ Ninguna letra minúscula
- $[\hat{}\, bdfg] \equiv$ Cualquier símbolo excepto b, d, f, g

Extensiones de operadores

Algunas extensiones de operadores

- Sea Σ un conjunto ordenado:
  - $[a - z] \equiv a|b|c|...|z$ (rangos)
  - $[bdfg] \equiv b|d|f|g$ (un elemento de entre un conjunto de posibilidades)
- $[\hat{}\, a - z] \equiv$ Ninguna letra minúscula
- $[\hat{}\, bdfg] \equiv$ Cualquier símbolo excepto b, d, f, g

Extensiones de operadores

Algunas extensiones de operadores

- Sea Σ un conjunto ordenado:
  - $[a-z] \equiv a|b|c|...|z$ (rangos)
  - $[bdfg] \equiv b|d|f|g$ (un elemento de entre un conjunto de posibilidades)
- $[\hat{\ } a-z] \equiv$ Ninguna letra minúscula
- $[\hat{\ } bdfg] \equiv$ Cualquier símbolo excepto b, d, f, g

- Los metacaracteres pierden su significado y se convierten en literales cuando se encuentran dentro de los corchetes. Por ejemplo, dentro de los corchetes, el punto representa un caracter literal y no un metacaracter, por lo que no es necesario prefijarlo con la barra inversa
- El único carácter que es necesario prefijar con la barra inversa dentro de los corchetes es la propia barra inversa
- Incluso el guión, para que se tenga en cuenta como caracter literal basta con ponerlo al inicio o al final, de modo que tampoco se prefija

Extensiones de operadores

Ejemplos

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)

Extensiones de operadores

Ejemplos

- $.ol \ \rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol \ \rightarrow col$ o sol pero no rol

Extensiones de operadores

Ejemplos

- $.ol \ \rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol \ \rightarrow$ col o sol pero no rol
- $[\hat{\ } s]ol \ \rightarrow$ cadenas de tres letras que acaben en ol excepto sol

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{\ } s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{\ } sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{}\,s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{}\,sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{}\,[cs]ol$ $\rightarrow$ col o sol cuando están al comienzo de la cadena/línea

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{\ } s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{\ } sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{\ }[cs]ol$ $\rightarrow$ col o sol cuando están al comienzo de la cadena/línea
- $\hat{\ }[cs]ol\$$ $\rightarrow$ col o sol cuando están al final de la cadena/línea

Extensiones de operadores

Ejemplos

- $.ol \;\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol \;\rightarrow$ col o sol pero no rol
- $[\hat{}\, s]ol \;\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{}\, sr]ol \;\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{}\,[cs]ol \;\rightarrow$ col o sol cuando están al comienzo de la cadena/línea
- $\hat{}\,[cs]ol\$ \;\rightarrow$ col o sol cuando están al final de la cadena/línea
- $\backslash\,[.\,\backslash\,]$ $\rightarrow$ cualquier caracter (sólo uno) que esté precedido y continuado de "[" y "]" respectivamente

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{}\,s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{}\,sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{}\,[cs]ol$ $\rightarrow$ col o sol cuando están al comienzo de la cadena/línea
- $\hat{}\,[cs]ol\$$ $\rightarrow$ col o sol cuando están al final de la cadena/línea
- $\backslash\,[.\ \backslash\,]$ $\rightarrow$ cualquier caracter (sólo uno) que esté precedido y continuado de "[" y "]" respectivamente
- $[sc]?ol \rightarrow sol$, col y ol

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{}\,s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{}\,sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{}\,[cs]ol$ $\rightarrow$ col o sol cuando están al comienzo de la cadena/línea
- $\hat{}\,[cs]ol\$$ $\rightarrow$ col o sol cuando están al final de la cadena/línea
- $\backslash\,[.\,\backslash\,]$ $\rightarrow$ cualquier caracter (sólo uno) que esté precedido y continuado de "[" y "]" respectivamente
- $[sc]?ol \rightarrow sol$, col y ol
- $[sc]+ol \rightarrow sol, col, sssol, csol, \ldots$

Extensiones de operadores

Ejemplos

- $.ol$ $\rightarrow$ cadenas de tres letras que acaben en ol (ejemplo: sol, col, ...)
- $[cs]ol$ $\rightarrow$ col o sol pero no rol
- $[\hat{}\, s]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol
- $[\hat{}\, sr]ol$ $\rightarrow$ cadenas de tres letras que acaben en ol excepto sol y rol
- $\hat{}\,[cs]ol$ $\rightarrow$ col o sol cuando están al comienzo de la cadena/línea
- $\hat{}\,[cs]ol\$$ $\rightarrow$ col o sol cuando están al final de la cadena/línea
- $\backslash\, [. \backslash\,]$ $\rightarrow$ cualquier caracter (sólo uno) que esté precedido y continuado de "[" y "]" respectivamente
- $[sc]?ol \rightarrow sol$, col y ol
- $[sc] + ol \rightarrow sol, col, sssol, csol, \ldots$
- $[sc] * ol \rightarrow sol, col, sssol, csol, \ldots$, pero también ol

Usos más habituales de las expresiones regulares

Ejemplos

- Analizadores léxicos para compiladores

Usos más habituales de las expresiones regulares

Ejemplos

- Analizadores léxicos para compiladores
- Búsquedas de patrones en editores de texto

Usos más habituales de las expresiones regulares

Ejemplos

- Analizadores léxicos para compiladores
- Búsquedas de patrones en editores de texto

Recordatorio

Para un alfabeto Σ, los lenguajes regulares sobre Σ son el menor conjunto de lenguajes que contienen a $\{\varepsilon\}$, $\emptyset$, y los lenguajes unitarios $(\{a\}, a \in \Sigma)$ y que además son cerrados respecto a:

- Concatenación
- Unión
- Cierre de Kleene

Conjunto cerrado con respecto a una operación

Definición

Dados un conjunto A y una operación $\odot$ se dice que el conjunto es **cerrado** bajo esa operación, cuando al operar dos elementos del conjunto $x \odot y$ el resultado es **siempre** un elemento del conjunto A.
A es cerrado bajo $\odot$ si $\forall x, y \in A, x \odot y \in A$

Así, por ejemplo

- $\mathbb{N}$ es cerrado para la operación $+$ pero no lo es para la operación $-$
- $\mathbb{R}$ es cerrado para las operaciones de suma, resta, producto y división (entre otras)

Indice

¿Cómo saber si una cadena pertenece a un determinado lenguaje regular?

Ejemplo

Dado el lenguaje regular L representado por la expresión regular $c*(a|bc*)*$, ¿Pertenecen a L las siguientes cadenas w_1 y w_2?:

- $w_1 = abc^5c^3ab$
- $w_2 = cabac^3bc$

¿Cómo saber si una cadena pertenece a un determinado lenguaje regular?

Ejemplo

Dado el lenguaje regular L representado por la expresión regular $c*(a|bc*)*$, ¿Pertenecen a L las siguientes cadenas w_1 y w_2?:

- $w_1 = abc^5c^3ab$
- $w_2 = cabac^3bc$

Para saber si las cadenas pertenecen o no al lenguaje regular debemos analizar no sólo los símbolos que aparecen sino también su posición relativa

Ayuda para determinar los elementos del lenguaje

Diagrama de transición

- Tiene la forma de grafo dirigido pero incluye información adicional
- Los nodos se llaman *estados* y sirven para señalar hasta qué posición se ha analizado la cadena
- Las aristas del grafo se llaman *transiciones* y se etiquetan con símbolos del alfabeto
- Se comienza por el *estado inicial* y se procesan los símbolos de la cadena leyéndolos de izquierda a derecha
- Si el siguiente símbolo a procesar concuerda con la etiqueta de alguna transición que parte del estado actual se cambia de estado
- Si al finalizar los símbolos de la cadena, el estado actual es de aceptación se dice que la cadena ha sido aceptada (rechazada en caso contrario)

Diagrama de transición

Ejemplo

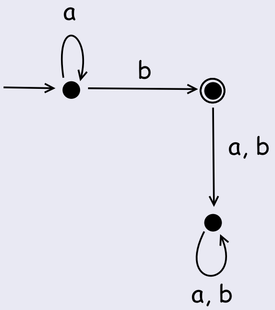
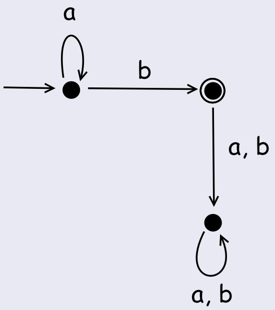


Diagrama de transición

Ejemplo



- El diagrama anterior acepta todas las cadenas que están formadas por 0 o más "aes" seguidas por una única b
- Lenguaje $L = \{a^k b \mid k \geq 0\}$ representado por la expresión regular $a{*}b$

Diagrama de transición

Ejemplo

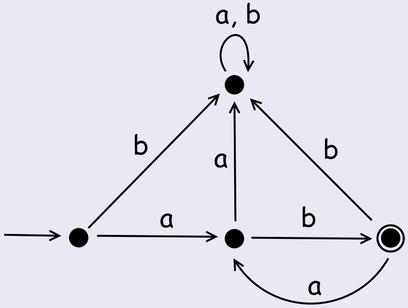
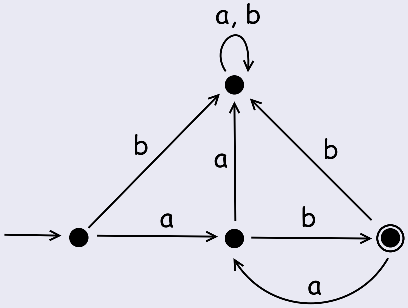


Diagrama de transición

Ejemplo



- Lenguaje $L = \{(ab)^i \mid i \geq 1\}$ representado por la expresión regular $ab(ab)* = (ab)+$

Diagrama de transición

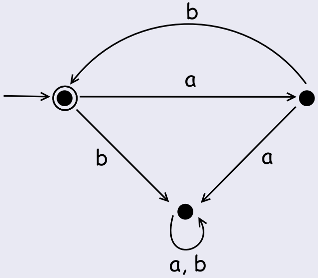
Ejemplo

- Lenguaje $L = \{(ab)^i \mid i \geq 0\}$ representado por la expresión regular $(ab)*$

Diagrama de transición

Ejemplo

- Lenguaje $L = \{(ab)^i \mid i \geq 0\}$ representado por la expresión regular $(ab)*$



- El estado inicial también es de aceptación: se acepta la cadena vacía

Diagrama de transición

Representación mediante una tabla

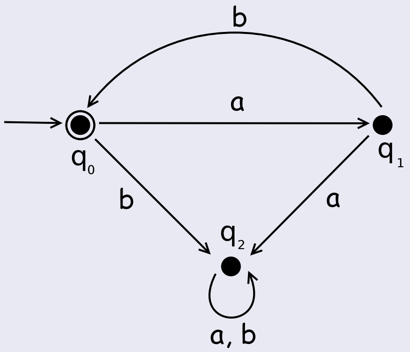
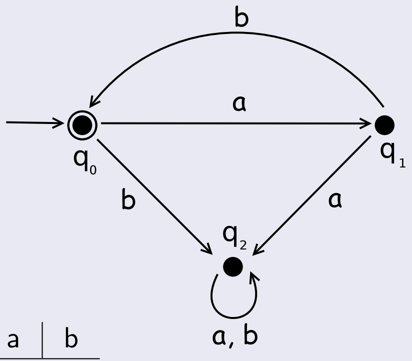


Diagrama de transición

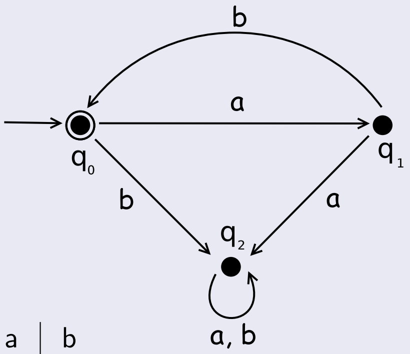
Representación mediante una tabla



Estado/Entrada	a	b

Diagrama de transición

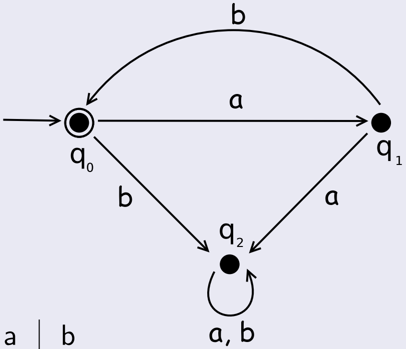
Representación mediante una tabla



Estado/Entrada	a	b
q_0	q_1	q_2

Diagrama de transición

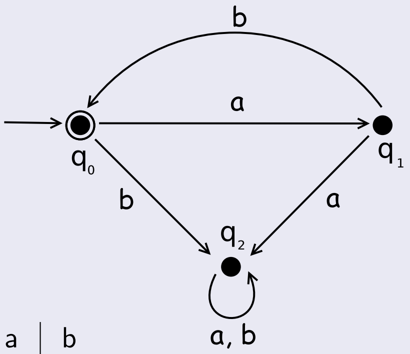
Representación mediante una tabla



Estado/Entrada	a	b
q_0	q_1	q_2
q_1	q_2	q_0

Diagrama de transición

Representación mediante una tabla



Estado/Entrada	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2

Autómata finito

Definición

- Para cada estado actual y símbolo de entrada se puede determinar cuál será el estado siguiente
- El diagrama representa la acción de una máquina que cambia de estado dependiendo de la entrada y del estado en el que se encuentre
- *Autómata finito*:
    - Determinista
    - No Determinista

Autómata finito determinista (AFD - DFA)

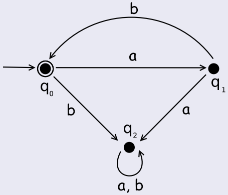
Definición

Formalmente, un *autómata finito determinista* M es una colección de cinco elementos:

- Un alfabeto de entrada Σ
- Una colección finita de estados Q
- Un estado inicial s
- Una colección F de estados finales o de aceptación
- Una función $\delta : Q \times \Sigma \rightarrow Q$ que determina el único estado siguiente para el par (q_i, σ) correspondiente al estado actual y la entrada

Autómata finito determinista (AFD - DFA)

Notación



El DFA del ejemplo anterior se representa mediante $M = (Q, \Sigma, s, F, \delta)$, donde:

- $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$

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

Características de un DFA

δ es una *función*

- δ se debe definir para todos los pares (q_i, σ) de $Q \times \Sigma$
- Sea cual sea el estado actual y el símbolo de la entrada *siempre* hay un estado siguiente y éste es *único*
- El estado siguiente está totalmente determinado por la información que proporciona el par (q_i, σ)

Crear un diagrama de transición a partir de la definición de un DFA

Pasos

Crear un diagrama de transición a partir de la definición de un DFA

Pasos

- Se dibuja un nodo para cada estado $q_i \in Q$ del autómata

Crear un diagrama de transición a partir de la definición de un DFA

Pasos

- Se dibuja un nodo para cada estado $q_i \in Q$ del autómata
- Si $\delta(q_i, \sigma) = q_j$ se dibuja un arco dirigido:
  - Origen: q_i
  - Destino: q_j
  - Etiqueta: σ

Crear un diagrama de transición a partir de la definición de un DFA

Pasos

- Se dibuja un nodo para cada estado $q_i \in Q$ del autómata
- Si $\delta(q_i, \sigma) = q_j$ se dibuja un arco dirigido:
  - Origen: q_i
  - Destino: q_j
  - Etiqueta: σ
- Se marca con una flecha de entrada el estado de arranque q_0

Crear un diagrama de transición a partir de la definición de un DFA

Pasos

- Se dibuja un nodo para cada estado $q_i \in Q$ del autómata
- Si $\delta(q_i, \sigma) = q_j$ se dibuja un arco dirigido:
  - Origen: q_i
  - Destino: q_j
  - Etiqueta: σ
- Se marca con una flecha de entrada el estado de arranque q_0
- Se marcan con doble trazo los estados pertenecientes a F

Crear un diagrama de transición a partir de la definición de un DFA

Ejemplo

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $s = q_0$
- $F = \{q_0\}$

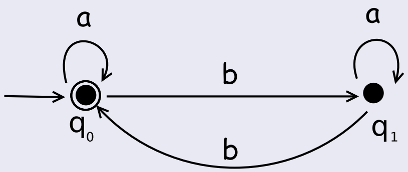
δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

Crear un diagrama de transición a partir de la definición de un DFA

Ejemplo

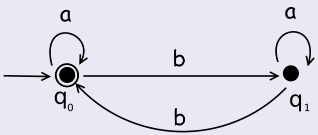
- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $s = q_0$
- $F = \{q_0\}$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0



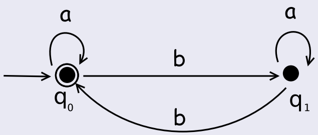
Crear un diagrama de transición a partir de la definición de un DFA

Ejemplo: si la cadena de entrada fuera $w = abaaba$



Crear un diagrama de transición a partir de la definición de un DFA

Ejemplo: si la cadena de entrada fuera $w = abaaba$



- El DFA transitaría por los estados $q_0, q_0, q_1, q_1, q_1, q_0, q_0$
- En este caso, se dice que el DFA "acepta" la cadena w porque, una vez leídos todos sus símbolos, el DFA acaba en un estado de aceptación
- ¿Qué cadenas acepta este DFA?
  - Cadenas de "aes" y "bes" con un número par de "bes"

Estados de muerte o absorción

Definición

Un estado q es de muerte o absorción si:

- $\delta(q, x) = q \ \forall x \in \Sigma$
- $q \notin F$

Estados de muerte o absorción

Definición

Un estado q es de muerte o absorción si:

- $\delta(q, x) = q \ \forall x \in \Sigma$
- $q \notin F$

Ejemplo: construir un DFA que acepte las cadenas del lenguaje representado por $(a|b)aab*$

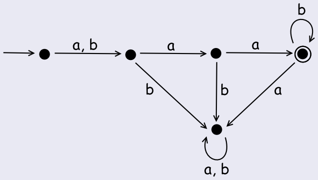
Estados de muerte o absorción

Definición

Un estado q es de muerte o absorción si:

- $\delta(q, x) = q \ \forall x \in \Sigma$
- $q \notin F$

Ejemplo: construir un DFA que acepte las cadenas del lenguaje representado por $(a|b)aab*$



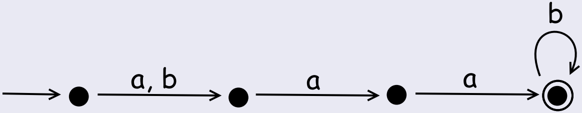
Estados de muerte o absorción

Ejemplo: construir un DFA que acepte las cadenas del lenguaje representado por $(a|b)aab*$

Estados de muerte o absorción

Ejemplo: construir un DFA que acepte las cadenas del lenguaje representado por $(a|b)aab*$

- Se aconseja dibujar todos los estados del autómata, aunque
- En algunas ocasiones no se dibujan los estados de muerte
- Los estados permiten al DFA "recordar" lo que ha ocurrido en la cadena de entrada



Función de transición extendida

- Se ha definido:
  $\delta : Q \times \Sigma \rightarrow Q$
  $\delta(q, a) = p$

Función de transición extendida

- Se ha definido:
  $\delta : Q \times \Sigma \to Q$
  $\delta(q, a) = p$
- Se extenderá la función de transición para operar sobre cadenas:
  $\hat{\delta} : Q \times \Sigma^* \to Q$
  $\hat{\delta}(q, w) = r$, donde

$$\hat{\delta}(q, w) = \begin{cases} q & \text{si } w = \varepsilon \\ \delta(\hat{\delta}(q, y), a) & \text{si } w = ya, \text{ con } a \in \Sigma, y \in \Sigma^* \end{cases}$$

Función de transición extendida

- Se ha definido:
  $\delta : Q \times \Sigma \to Q$
  $\delta(q, a) = p$
- Se extenderá la función de transición para operar sobre cadenas:
  $\hat{\delta} : Q \times \Sigma^* \to Q$
  $\hat{\delta}(q, w) = r$, donde

$$\hat{\delta}(q, w) = \begin{cases} q & \text{si } w = \varepsilon \\ \delta(\hat{\delta}(q, y), a) & \text{si } w = ya, \text{ con } a \in \Sigma, y \in \Sigma^* \end{cases}$$

- Se verifica que: $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \varepsilon), a) = \delta(q, a)$

Función de transición extendida

- Se ha definido:
  $\delta : Q \times \Sigma \rightarrow Q$
  $\delta(q, a) = p$
- Se extenderá la función de transición para operar sobre cadenas:
  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$
  $\hat{\delta}(q, w) = r$, donde

$$\hat{\delta}(q, w) = \begin{cases} q & \text{si } w = \varepsilon \\ \delta(\hat{\delta}(q, y), a) & \text{si } w = ya, \text{ con } a \in \Sigma, y \in \Sigma^* \end{cases}$$

- Se verifica que: $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \varepsilon), a) = \delta(q, a)$

$\hat{\delta}$ es la aplicación sucesiva de δ a los símbolos de la cadena

Si p. ej. $w = abaa$, entonces $\hat{\delta}(q_0, w) = \delta(\delta(\delta(\delta(q_0, a), b), a), a)$

Lenguaje aceptado por un DFA

Sea M un DFA definido por $M \equiv (Q, \Sigma, \delta, q_0, F)$

Definición

El *lenguaje aceptado* por el DFA M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Lenguaje aceptado por un DFA

Sea M un DFA definido por $M \equiv (Q, \Sigma, \delta, q_0, F)$

Definición

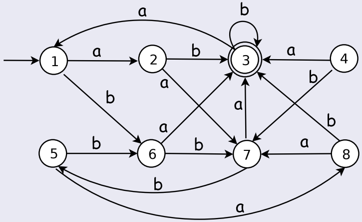
El *lenguaje aceptado* por el DFA M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

- $L(M)$ es el conjunto de **todas** las cadenas $w \in \Sigma^*$ que hacen transitar al DFA M desde su estado inicial q_0 hasta algún estado de aceptación
- Los DFAs tienen la capacidad de reconocer los lenguajes definidos por expresiones regulares (lenguajes regulares)
- Dos DFAs M_1 y M_2 son *equivalentes* si $L(M_1) = L(M_2)$

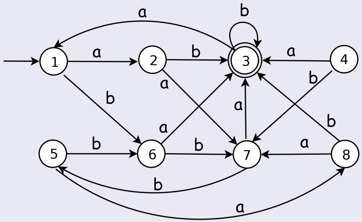
DFAs no mínimos

Ejemplo



DFAs no mínimos

Ejemplo



- Algunos estados se comportan del mismo modo para toda cadena de entrada
- Estando en el estado q_2 o q_8, al leer cualquier cadena de entrada no vacía ($w \in \Sigma^*, \Sigma = \{a, b\}, w \neq \varepsilon$) se llega siempre al mismo estado final
- La presencia de los estados q_2 y q_8 es redundante
- Para obtener un DFA con un número mínimo de estados sería conveniente evitar estados redundantes

Estados equivalentes y estados distinguibles

Consideremos un DFA $M \equiv (Q, \Sigma, \delta, s, F)$

Definición

Dos estados $p, q \in Q$ son **equivalentes** si:

$$\forall w \in \Sigma^*, \;\; \hat{\delta}(p, w) \in F \Longleftrightarrow \hat{\delta}(q, w) \in F$$

Estados equivalentes y estados distinguibles

Consideremos un DFA $M \equiv (Q, \Sigma, \delta, s, F)$

Definición

Dos estados $p, q \in Q$ son **equivalentes** si:

$$\forall w \in \Sigma^*, \ \hat{\delta}(p, w) \in F \Longleftrightarrow \hat{\delta}(q, w) \in F$$

Definición

Dos estados $p, q \in Q$ son **distinguibles** si:

$$\exists w \in \Sigma^* \mid \hat{\delta}(p, w) \in F \wedge \hat{\delta}(q, w) \notin F \text{ o viceversa}$$

DFA mínimo

Definición

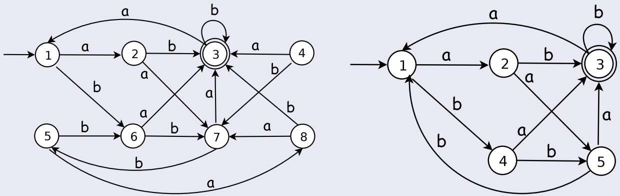
Si todos los pares de estados del DFA son distinguibles, el autómata no tiene estados redundantes y por lo tanto, se dice que es un DFA mínimo.

DFA mínimo

Definición

Si todos los pares de estados del DFA son distinguibles, el autómata no tiene estados redundantes y por lo tanto, se dice que es un DFA mínimo.

Ejemplo



(a) DFA con estados redundantes

(b) DFA mínimo

- Si el autómata contiene uno o más conjuntos de estados no distinguibles, se puede eliminar la redundancia reemplazando cada conjunto de estados por un único estado.

Algoritmo de minimización de estados

- Se expondrá un algoritmo para minimizar el numero de estados de un DFA
- El algoritmo funciona hallando todos los conjuntos de estados que pueden ser diferenciados por una cadena de entrada
- Cada grupo de estados no distinguibles se fusiona en un único estado
- El algoritmo mantiene y refina sucesivamente una partición del conjunto de estados
- Cada grupo de estados dentro de la partición está formado por estados que aún no han sido distinguidos, y por todos los pares de estados elegidos de entre grupos diferentes que han sido considerados distinguibles por alguna entrada

Algoritmo de minimización de estados

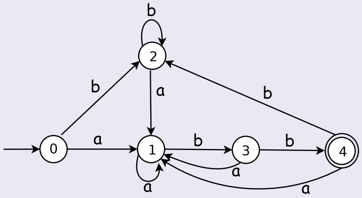
- Al comienzo del algoritmo la partición tiene dos conjuntos:
  - Estados de aceptación
  - Estados que no son de aceptación
- Dado un conjunto de estados, dentro de una partición, se examina su comportamiento para un determinado símbolo del alfabeto:
  - Si para un símbolo $\sigma \in \Sigma$ todos los estados dentro de un conjunto transitan a estados de un mismo conjunto, el conjunto de partida se puede conservar
  - En caso contrario, hay que particionar el conjunto inicial
- Este proceso de división se repetirá hasta que no sea necesario dividir ningún conjunto de la partición considerada

Algoritmo de minimización de estados

1. Construir una partición Π del conjunto de estados Q del DFA original: $\Pi = \{F, Q - F\}$
2. **REPEAT**
   a. Sea $\Pi = G_1 \cup G_2 \cup G_3 ... \cup G_n$ la partición actual.
      Particione cada conjunto G_i en subconjuntos de modo que $s, t \in Q$ están en el mismo subconjunto $\Leftrightarrow \forall a \in \Sigma$ hay transiciones $s \rightarrow s'$ y $t \rightarrow t'$ cumpliendo que s' y t' están en el mismo subconjunto G_j
   b. Combine los subconjuntos resultantes en una nueva partición Π'
3. **UNTIL** $\Pi == \Pi'$
4. Construir el DFA mínimo con un estado por cada subconjunto de estados en la partición final

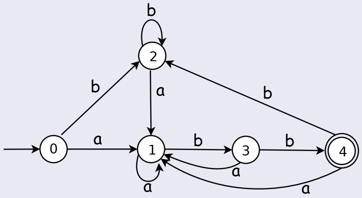
Algoritmo de minimización de estados

Ejemplo



Algoritmo de minimización de estados

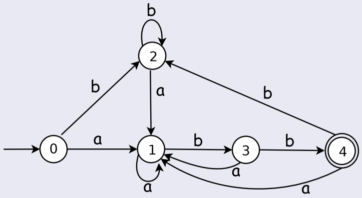
Ejemplo



- La primera partición contiene dos conjuntos de estados; el que contiene el estado de aceptación y el que contiene al resto de estados: $\Pi = \{\{4\}, \{0, 1, 2, 3\}\}$.

Algoritmo de minimización de estados

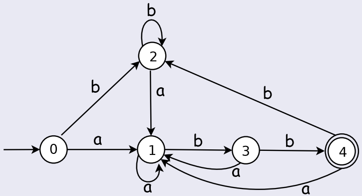
Ejemplo



- La primera partición contiene dos conjuntos de estados; el que contiene el estado de aceptación y el que contiene al resto de estados: $\Pi = \{\{4\}, \{0, 1, 2, 3\}\}$.
- En el paso 2 del algoritmo, se considera el conjunto $\{4\}$ de la partición, pero este conjunto no puede ser particionado porque sólo contiene un estado.

Algoritmo de minimización de estados

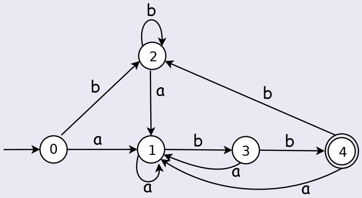
Ejemplo



- La primera partición contiene dos conjuntos de estados; el que contiene el estado de aceptación y el que contiene al resto de estados: $\Pi = \{\{4\}, \{0, 1, 2, 3\}\}$.
- En el paso 2 del algoritmo, se considera el conjunto $\{4\}$ de la partición, pero este conjunto no puede ser particionado porque sólo contiene un estado.
- Se pasa al siguiente conjunto de la partición, $\{0, 1, 2, 3\}$, y se recorre el bucle en el que se consideran cada uno de los símbolos del alfabeto.

Algoritmo de minimización de estados

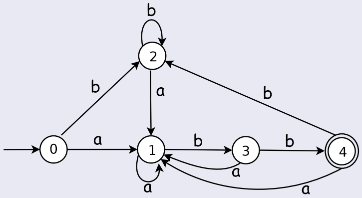
Ejemplo



- Para el símbolo a, todos los estados del conjunto $\{0, 1, 2, 3\}$ tienen una transición al estado 1, con lo cual todos los estados permanecen en el mismo conjunto.

Algoritmo de minimización de estados

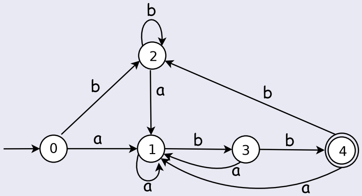
Ejemplo



- Para el símbolo a, todos los estados del conjunto $\{0, 1, 2, 3\}$ tienen una transición al estado 1, con lo cual todos los estados permanecen en el mismo conjunto.
- Para el símbolo b, los estados $0, 1$ y 2 transitan a estados del conjunto $\{0, 1, 2, 3\}$ mientras que el estado 3 transita a 4 con esa entrada.

Algoritmo de minimización de estados

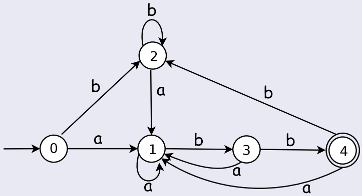
Ejemplo



- Para el símbolo a, todos los estados del conjunto $\{0, 1, 2, 3\}$ tienen una transición al estado 1, con lo cual todos los estados permanecen en el mismo conjunto.
- Para el símbolo b, los estados $0, 1$ y 2 transitan a estados del conjunto $\{0, 1, 2, 3\}$ mientras que el estado 3 transita a 4 con esa entrada.
- Por lo tanto, hay que dividir el conjunto y actualizar la partición: $\Pi = \{\{4\}, \{3\}, \{0, 1, 2\}\}$.

Algoritmo de minimización de estados

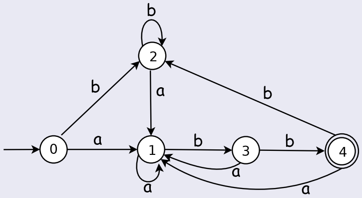
Ejemplo



- Ahora se trata de refinar el conjunto $\{0, 1, 2\}$ (el resto de conjuntos tiene un único estado y no se puede dividir más).

Algoritmo de minimización de estados

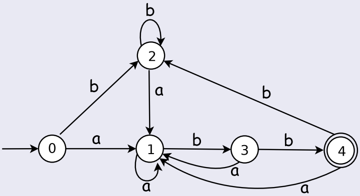
Ejemplo



- Ahora se trata de refinar el conjunto {0, 1, 2} (el resto de conjuntos tiene un único estado y no se puede dividir más).
- Con el símbolo a no se producen divisiones, pero para la entrada b los estados 0 y 2 transitan a 2 mientras que el 1 transita al 3.

Algoritmo de minimización de estados

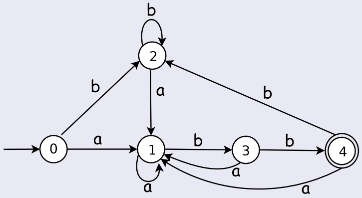
Ejemplo



- Ahora se trata de refinar el conjunto {0, 1, 2} (el resto de conjuntos tiene un único estado y no se puede dividir más).
- Con el símbolo a no se producen divisiones, pero para la entrada b los estados 0 y 2 transitan a 2 mientras que el 1 transita al 3.
- Hay que dividir el conjunto $\{0, 1, 2\}$ dando lugar a una nueva partición: $\Pi = \{\{4\}, \{3\}, \{1\}, \{0, 2\}\}$.

Algoritmo de minimización de estados

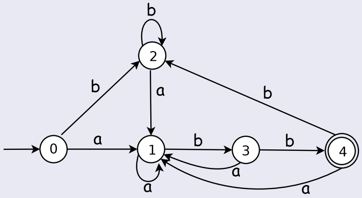
Ejemplo



- Al intentar dividir el conjunto $\{0, 2\}$ se observa que no es necesario puesto que con a ambos estos transitan a 1 y con entrada b ambos transitan al estado 2.

Algoritmo de minimización de estados

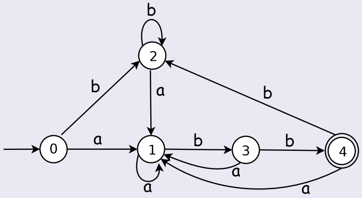
Ejemplo



- Al intentar dividir el conjunto $\{0, 2\}$ se observa que no es necesario puesto que con a ambos estos transitan a 1 y con entrada b ambos transitan al estado 2.
- El algoritmo termina puesto que no quedan conjuntos que puedan ser particionados: $\Pi = \{\{4\}, \{3\}, \{1\}, \{0, 2\}\}$.

Algoritmo de minimización de estados

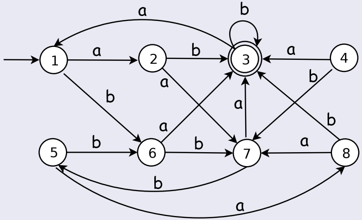
Ejemplo



- Al intentar dividir el conjunto $\{0, 2\}$ se observa que no es necesario puesto que con a ambos estos transitan a 1 y con entrada b ambos transitan al estado 2.
- El algoritmo termina puesto que no quedan conjuntos que puedan ser particionados: $\Pi = \{\{4\}, \{3\}, \{1\}, \{0, 2\}\}$.
- El DFA mínimo tendrá cuatro estados.

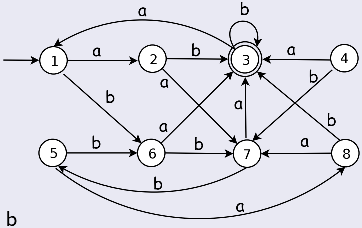
Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA



Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA



Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Inicialmente: $\Pi_0 = \{G_0, G_1\} = \{\{3\}, \{1,2,4,5,6,7,8\}\}$

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Inicialmente: $\Pi_0 = \{G_0, G_1\} = \{\{3\}, \{1, 2, 4, 5, 6, 7, 8\}\}$
- Puesto que el primer conjunto G_0 de la partición no se puede refinar, nos centramos en $G_1 = \{1, 2, 4, 5, 6, 7, 8\}$:

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Inicialmente: $\Pi_0 = \{G_0, G_1\} = \{\{3\}, \{1, 2, 4, 5, 6, 7, 8\}\}$
- Puesto que el primer conjunto G_0 de la partición no se puede refinar, nos centramos en $G_1 = \{1, 2, 4, 5, 6, 7, 8\}$:
  - Para el símbolo a, desde los estados $\{1, 2, 5, 8\}$ se transita a estados de G_1, mientras que desde los estados $\{4, 6, 7\}$ se transita a estados de G_0.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Inicialmente: $\Pi_0 = \{G_0, G_1\} = \{\{3\}, \{1, 2, 4, 5, 6, 7, 8\}\}$
- Puesto que el primer conjunto G_0 de la partición no se puede refinar, nos centramos en $G_1 = \{1, 2, 4, 5, 6, 7, 8\}$:
  - Para el símbolo a, desde los estados $\{1, 2, 5, 8\}$ se transita a estados de G_1, mientras que desde los estados $\{4, 6, 7\}$ se transita a estados de G_0.
  - Para el símbolo b, desde los estados $\{1, 4, 5, 6, 7\}$ se transita a estados de G_1, mientras que desde los estados $\{2, 8\}$ se transita a estados de G_0.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Inicialmente: $\Pi_0 = \{G_0, G_1\} = \{\{3\}, \{1, 2, 4, 5, 6, 7, 8\}\}$
- Puesto que el primer conjunto G_0 de la partición no se puede refinar, nos centramos en $G_1 = \{1, 2, 4, 5, 6, 7, 8\}$:
  - Para el símbolo a, desde los estados $\{1, 2, 5, 8\}$ se transita a estados de G_1, mientras que desde los estados $\{4, 6, 7\}$ se transita a estados de G_0.
  - Para el símbolo b, desde los estados $\{1, 4, 5, 6, 7\}$ se transita a estados de G_1, mientras que desde los estados $\{2, 8\}$ se transita a estados de G_0.
  - Por lo tanto, $\Pi_1 = \{G_0, G_1, G_2, G_3\} = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6, 7\}\}$.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Siendo $\Pi_1 = \{\{3\}, \{1,5\}, \{2,8\}, \{4,6,7\}\}$, los conjuntos G_0, G_1 y G_2 no se pueden refinar más, así que nos centramos en $G_3 = \{4,6,7\}$:

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Siendo $\Pi_1 = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6, 7\}\}$, los conjuntos G_0, G_1 y G_2 no se pueden refinar más, así que nos centramos en $G_3 = \{4, 6, 7\}$:
  - Para el símbolo a, desde todos los estados $\{4, 6, 7\}$ se transita al estado de G_0.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Siendo $\Pi_1 = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6, 7\}\}$, los conjuntos G_0, G_1 y G_2 no se pueden refinar más, así que nos centramos en $G_3 = \{4, 6, 7\}$:
  - Para el símbolo a, desde todos los estados $\{4, 6, 7\}$ se transita al estado de G_0.
  - Para el símbolo b, desde los estados $\{4, 6\}$ se transita a estados de G_3, mientras que desde el estado $\{7\}$ se transita a un estado de G_1.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

Estado/Entrada	a	b
1	2	6
2	7	3
3	1	3
4	3	7
5	8	6
6	3	7
7	3	5
8	7	3

- Siendo $\Pi_1 = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6, 7\}\}$, los conjuntos G_0, G_1 y G_2 no se pueden refinar más, así que nos centramos en $G_3 = \{4, 6, 7\}$:
  - Para el símbolo a, desde todos los estados $\{4, 6, 7\}$ se transita al estado de G_0.
  - Para el símbolo b, desde los estados $\{4, 6\}$ se transita a estados de G_3, mientras que desde el estado $\{7\}$ se transita a un estado de G_1.
  - Por lo tanto, $\Pi_2 = \{G_0, G_1, G_2, G_3, G_4\} = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6\}, \{7\}\}$.

Algoritmo de minimización de estados

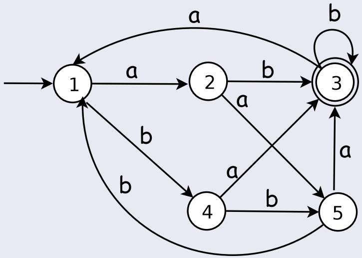
Ejercicio: minimizar el siguiente DFA

- Cuando $\Pi = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6\}, \{7\}\}$ no se pueden hacer más particiones.
- El DFA mínimo tiene cinco estados.

Algoritmo de minimización de estados

Ejercicio: minimizar el siguiente DFA

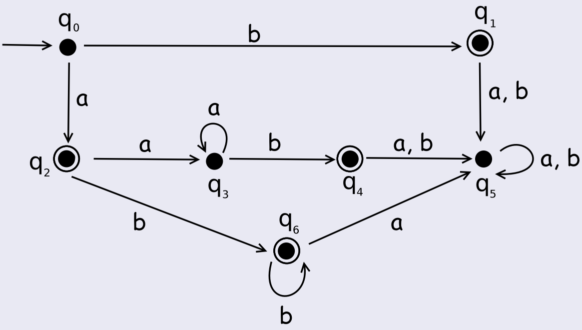
- Cuando $\Pi = \{\{3\}, \{1, 5\}, \{2, 8\}, \{4, 6\}, \{7\}\}$ no se pueden hacer más particiones.
- El DFA mínimo tiene cinco estados.



Autómata finito no determinista (AFN - NFA)

Para algunos lenguajes relativamente sencillos, tenemos DFAs no tan sencillos

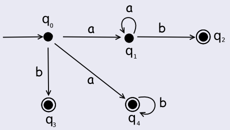
Ejemplo: $a*b|ab*$



Autómata finito no determinista (AFN - NFA)

Si se permite que desde un estado se realicen cero, una o más transiciones mediante el mismo símbolo de entrada, se dice que el autómata finito es *no determinista*

Ejemplo: $a{*}b|ab{*}$



- No asigna un estado siguiente a los pares estado-entrada (q_4, a), (q_3, a), (q_3, b), (q_2, a), y (q_2, b)
- Existe más de un estado siguiente para el par (q_0, a)

Autómata finito no determinista (AFN - NFA)

Definición

Formalmente, un *autómata finito no determinista* M es una colección de cinco elementos:

- Un alfabeto de entrada Σ
- Una colección finita de estados Q
- Un estado inicial s
- Un conjunto F de estados finales o de aceptación
- Una función de transición $\delta : (Q \times \Sigma) \rightarrow 2^Q$

Autómata finito no determinista (AFN - NFA)

Definición

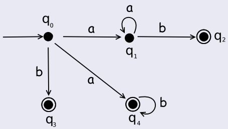
Formalmente, un *autómata finito no determinista* M es una colección de cinco elementos:

- Un alfabeto de entrada Σ
- Una colección finita de estados Q
- Un estado inicial s
- Un conjunto F de estados finales o de aceptación
- Una función de transición $\delta : (Q \times \Sigma) \to 2^Q$

- Téngase en cuenta que $\emptyset \in 2^Q$ y también $Q \in 2^Q$
- Dado un estado y un símbolo, el NFA transita a un **conjunto de estados**
- No existe nada en el modelo que determine la elección: el comportamiento es no-determinista

Autómata finito no determinista (AFN - NFA)

Notación



El NFA del ejemplo anterior se representa mediante $M = (Q, \Sigma, s, F, \delta)$, donde:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_2, q_3, q_4\}$

δ	a	b
q_0	$\{q_1, q_4\}$	$\{q_3\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_4\}$

Lenguaje aceptado por un NFA

Función de transición extendida

Si $X \subseteq Q$, vamos a interpretar $\hat{\delta}(X, w)$ como el conjunto de estados:

$$\{p \mid q \in X \text{ y } p \in \hat{\delta}(q, w)\}$$

- $\hat{\delta}(X, w)$ es el conjunto de todos los estados siguientes a los que se puede llegar desde estados de X con la entrada w

Lenguaje aceptado por un NFA

Función de transición extendida

Si $X \subseteq Q$, vamos a interpretar $\hat{\delta}(X, w)$ como el conjunto de estados:

$$\{p \mid q \in X \text{ y } p \in \hat{\delta}(q, w)\}$$

- $\hat{\delta}(X, w)$ es el conjunto de todos los estados siguientes a los que se puede llegar desde estados de X con la entrada w

Ejemplo: $w = ab$

$$\hat{\delta}(q_0, ab) = \delta(\delta(q_0, a), b) = \delta(\{q_1, q_4\}, b) = \{q_2\} \cup \{q_4\} = \{q_2, q_4\}$$

Lenguaje aceptado por un NFA

Sea M un NFA definido por $M \equiv (Q, \Sigma, \delta, q_0, F)$

Definición

El *lenguaje aceptado* por el NFA M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Lenguaje aceptado por un NFA

Sea M un NFA definido por $M \equiv (Q, \Sigma, \delta, q_0, F)$

Definición

El *lenguaje aceptado* por el NFA M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

- Para determinar si una cadena pertenece a $L(M)$ se debe encontrar en el diagrama de transiciones del NFA un camino que termine en un estado de aceptación cuando acabe de consumir toda la cadena
- Para afirmar que una cadena no está en $L(M)$ se deben agotar todas las formas posibles de recorrer el diagrama de transiciones para dicha cadena
- Dos NFAs M_1 y M_2 son *equivalentes* si $L(M_1) = L(M_2)$

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

- Σ es el alfabeto de entrada del autómata

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

- Σ es el alfabeto de entrada del autómata
- Q es el conjunto de estados del autómata (finito, $Q \neq \emptyset, Q \cap \Sigma = \emptyset$)

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

- Σ es el alfabeto de entrada del autómata
- Q es el conjunto de estados del autómata (finito, $Q \neq \emptyset, Q \cap \Sigma = \emptyset$)
- F es el conjunto de estados finales o de aceptación ($F \subseteq Q$, podría ser vacío)

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

- Σ es el alfabeto de entrada del autómata
- Q es el conjunto de estados del autómata (finito, $Q \neq \emptyset, Q \cap \Sigma = \emptyset$)
- F es el conjunto de estados finales o de aceptación ($F \subseteq Q$, podría ser vacío)
- q_0 es el estado inicial del autómata o estado de arranque

NFA con ε-transiciones

Definición

Una $\varepsilon - transicion$ es una transición entre estados que no consume ningún símbolo de la entrada

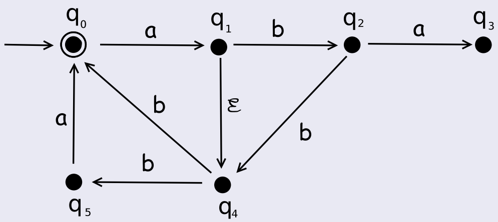
Definición

Un *autómata finito no determinista con* $\varepsilon - transiciones$ se define como una tupla $(\Sigma, Q, F, q_0, \delta)$, donde:

- Σ es el alfabeto de entrada del autómata
- Q es el conjunto de estados del autómata (finito, $Q \neq \emptyset, Q \cap \Sigma = \emptyset$)
- F es el conjunto de estados finales o de aceptación ($F \subseteq Q$, podría ser vacío)
- q_0 es el estado inicial del autómata o estado de arranque
- δ es la función de transición del autómata ($\delta : Q \times (\Sigma \cup \{\varepsilon)\} \to 2^Q, \delta(q, a) \subseteq Q$)

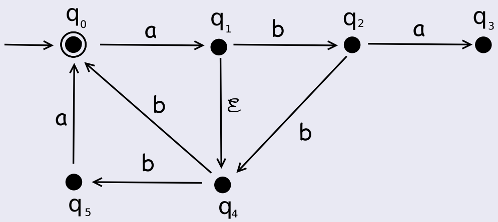
NFA con ε-transiciones

Ejemplo



NFA con ε-transiciones

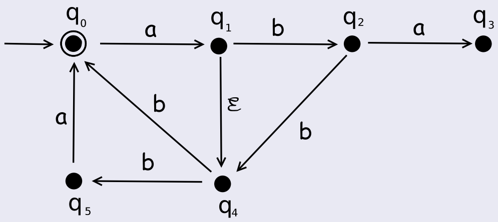
Ejemplo



δ	a	b	ε
q_0	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\{q_2\}$	$\{q_4\}$
q_2	$\{q_3\}$	$\{q_4\}$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_0, q_5\}$	$\emptyset$
q_5	$\{q_0\}$	$\emptyset$	$\emptyset$

NFA con ε-transiciones

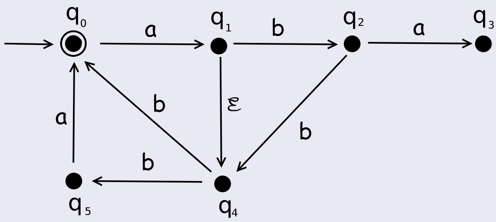
Ejemplo: ¿a qué estados siguientes podríamos llegar si...?



- q_0 es el estado actual y a es la entrada:

NFA con ε-transiciones

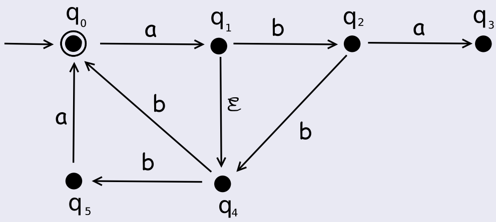
Ejemplo: ¿a qué estados siguientes podríamos llegar si...?



- q_0 es el estado actual y a es la entrada: $\{q_1, q_4\}$

NFA con ε-transiciones

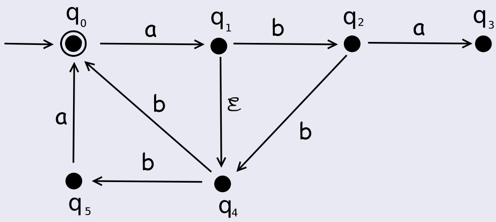
Ejemplo: ¿a qué estados siguientes podríamos llegar si...?



- q_0 es el estado actual y a es la entrada: $\{q_1, q_4\}$
- q_1 es el estado actual y b es la entrada:

NFA con ε-transiciones

Ejemplo: ¿a qué estados siguientes podríamos llegar si...?



- q_0 es el estado actual y a es la entrada: $\{q_1, q_4\}$
- q_1 es el estado actual y b es la entrada: $\{q_0, q_2, q_5\}$

Extensión de la función transición

Se pretende conseguir que $\hat{\delta}(q, w)$ sean todos los estados $p \in Q$ tales que se pueda recorrer el camino dirigido de q a p usando transiciones etiquetadas con los símbolos de w e incluyendo quizás transiciones etiquetadas con ε

Extensión de la función transición

Se pretende conseguir que $\hat{\delta}(q, w)$ sean todos los estados $p \in Q$ tales que se pueda recorrer el camino dirigido de q a p usando transiciones etiquetadas con los símbolos de w e incluyendo quizás transiciones etiquetadas con ε

Definición: Sea $q \in Q$

$\varepsilon - clausura(q) = \{p \mid p$ es accesible desde q sin consumir símbolos de la entrada$\}$
La $\varepsilon - clausura(q)$ es el conjunto de estados que incluye a q y todos los estados alcanzables desde q con ε-transiciones

Extensión de la función transición

Se pretende conseguir que $\hat{\delta}(q, w)$ sean todos los estados $p \in Q$ tales que se pueda recorrer el camino dirigido de q a p usando transiciones etiquetadas con los símbolos de w e incluyendo quizás transiciones etiquetadas con ε

Definición: Sea $q \in Q$

$\varepsilon - clausura(q) = \{p \mid p$ es accesible desde q sin consumir símbolos de la entrada$\}$
La $\varepsilon - clausura(q)$ es el conjunto de estados que incluye a q y todos los estados alcanzables desde q con ε-transiciones

Observación

Cualquier estado es accesible desde sí mismo sin consumir ningún símbolo de la entrada

Extensión de la función transición

Se pretende conseguir que $\hat{\delta}(q, w)$ sean todos los estados $p \in Q$ tales que se pueda recorrer el camino dirigido de q a p usando transiciones etiquetadas con los símbolos de w e incluyendo quizás transiciones etiquetadas con ε

Definición: Sea $q \in Q$

$\varepsilon - clausura(q) = \{p \mid p$ es accesible desde q sin consumir símbolos de la entrada$\}$
La $\varepsilon - clausura(q)$ es el conjunto de estados que incluye a q y todos los estados alcanzables desde q con ε-transiciones

Observación

Cualquier estado es accesible desde sí mismo sin consumir ningún símbolo de la entrada

Definición: Sea $A \subseteq Q$

$$\varepsilon - clausura(A) = \bigcup_{q \in A} \varepsilon - clausura(q)$$

Extensión de la función transición

Extensión de δ a conjuntos de estados

Sea $R \subseteq Q$. Se define $\delta(R, a) = \bigcup_{q \in R} \delta(q, a)$

$$\hat{\delta}(q, w) = \begin{cases} \varepsilon - clausura(q) & \text{si } w = \varepsilon \\ \bigcup_{r \in \delta(\varepsilon - clausura(q), a)} \hat{\delta}(r, x) & \text{si } w = ax, a \in \Sigma, x \in \Sigma^* \end{cases}$$

Extensión de la función transición

Extensión de δ a conjuntos de estados

Sea $R \subseteq Q$. Se define $\delta(R, a) = \bigcup_{q \in R} \delta(q, a)$

$$\hat{\delta}(q, w) = \begin{cases} \varepsilon - clausura(q) & \text{si } w = \varepsilon \\ \bigcup_{r \in \delta(\varepsilon - clausura(q), a)} \hat{\delta}(r, x) & \text{si } w = ax, a \in \Sigma, x \in \Sigma^* \end{cases}$$

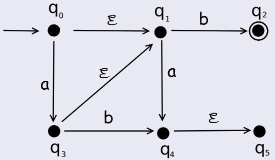
Finalmente se extiende $\hat{\delta}$ para operar sobre conjuntos de estados:

$$\hat{\delta}(R, w) = \bigcup_{q \in R} \hat{\delta}(q, w)$$

- En este caso, $\hat{\delta}(q, a)$ no tiene porqué coincidier con $\delta(q, a)$

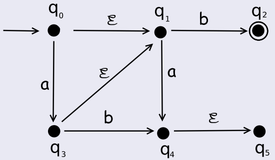
Extensión de la función transición

Ejemplo



Extensión de la función transición

Ejemplo



Consideremos la cadena $w = a$ y calculemos $\hat{\delta}(q_0, a)$:

- $\varepsilon - clausura(q_0) = \{q_0, q_1\}$
- $\delta(\varepsilon - clausura(q_0), a) = \delta(\{q_0, q_1\}, a) = \{q_3, q_4\}$
- $\varepsilon - clausura(\{q_3, q_4\}) = \{q_1, q_3, q_4, q_5\}$
- Observamos que $\hat{\delta}(q_0, a) = \{q_1, q_3, q_4, q_5\}$ mientras que $\delta(q_0, a) = \{q_3\}$
- $\hat{\delta}(q_0, a) \cap F = \{q_1, q_3, q_4, q_5\} \cap \{q_2\} = \emptyset$ y por lo tanto $w = a$ es rechazada por el autómata

Lenguaje reconocido por un NFA con ε-transiciones

$\varepsilon - clausura(\delta(\varepsilon - clausura(q), a))$
Es el conjunto de estados accesibles desde q tomando primero $\varepsilon - transiciones$, luego una transición con símbolo a y, finalmente, $\varepsilon - transiciones$

Lenguaje reconocido por un NFA con ε-transiciones

$\varepsilon - clausura(\delta(\varepsilon - clausura(q), a))$
Es el conjunto de estados accesibles desde q tomando primero $\varepsilon - transiciones$, luego una transición con símbolo a y, finalmente, $\varepsilon - transiciones$

Definición: Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un NFA con $\varepsilon - transiciones$

El lenguaje reconocido por M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Lenguaje reconocido por un NFA con ε-transiciones

$\varepsilon - clausura(\delta(\varepsilon - clausura(q), a))$
Es el conjunto de estados accesibles desde q tomando primero $\varepsilon - transiciones$, luego una transición con símbolo a y, finalmente, $\varepsilon - transiciones$

Definición: Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un NFA con $\varepsilon - transiciones$

El lenguaje reconocido por M es:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Equivalencia entre NFAs con $\varepsilon - transiciones$

Dos NFAs con $\varepsilon - transiciones$ M_1 y M_2 son equivalentes si $L(M1) = L(M2)$

Relación entre NFAs y DFAs

- Un DFA es un caso particular de NFA:
  - La clase de los lenguajes aceptados por NFAs incluye los lenguajes regulares
- Dado un NFA, siempre es posible definir un DFA (equivalente) que reconozca el mismo lenguaje

Relación entre NFAs y DFAs

- Un DFA es un caso particular de NFA:
  - La clase de los lenguajes aceptados por NFAs incluye los lenguajes regulares
- Dado un NFA, siempre es posible definir un DFA (equivalente) que reconozca el mismo lenguaje
- El fundamento de la construcción que justifica que todo NFA puede ser simulado por un DFA consiste en asociar conjuntos de estados del NFA con estados individuales del DFA

Relación entre NFAs y DFAs

- Un DFA es un caso particular de NFA:
  - La clase de los lenguajes aceptados por NFAs incluye los lenguajes regulares
- Dado un NFA, siempre es posible definir un DFA (equivalente) que reconozca el mismo lenguaje
- El fundamento de la construcción que justifica que todo NFA puede ser simulado por un DFA consiste en asociar conjuntos de estados del NFA con estados individuales del DFA
- Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un NFA, el **algoritmo de construcción de subconjuntos** permite hallar un DFA $M' \equiv (Q', \Sigma, \delta', q_0', F')$ de modo que $L(M) = L(M')$

Algoritmo de construcción de subconjuntos

1. $A = q'_0 = \varepsilon - clausura(q_0)$;
2. Desmarcar(A);
3. $Q' = \{A\}$;
4. **WHILE** ($\exists T \in Q'$ && !marcado(T)) **DO**
   1. marcar(T);
   2. **FOR** all $a \in \Sigma$ **DO**

      $R = \varepsilon - clausura(\delta(T, a))$;

      $\delta'(T, a) = R$;

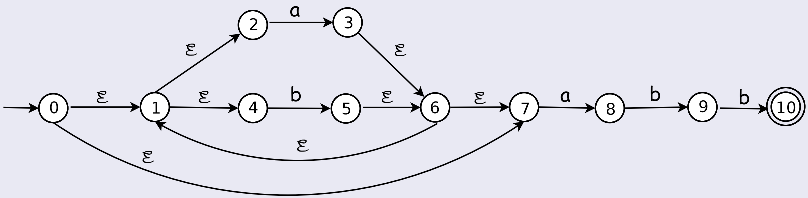
      IF ($R \notin Q'$) **THEN**

      $Q' = Q' \cup \{R\}$;

      desmarcar(R);

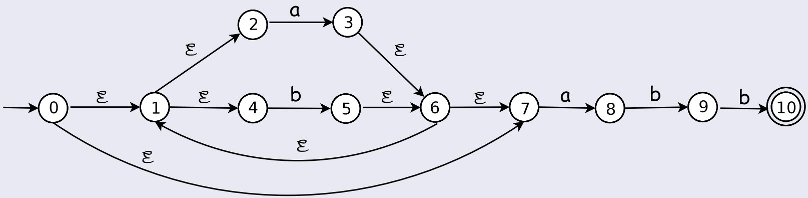
Algoritmo de construcción de subconjuntos

Ejemplo



Algoritmo de construcción de subconjuntos

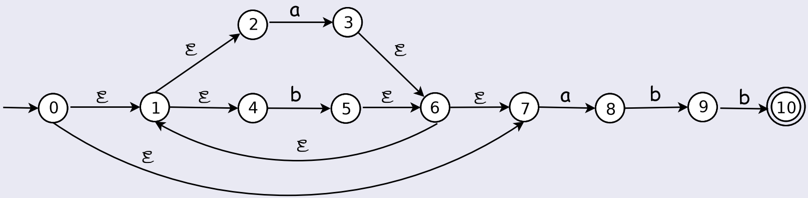
Ejemplo



1. $A = \varepsilon - clausura(0) = \{0, 1, 2, 4, 7\} = q_0'$

Algoritmo de construcción de subconjuntos

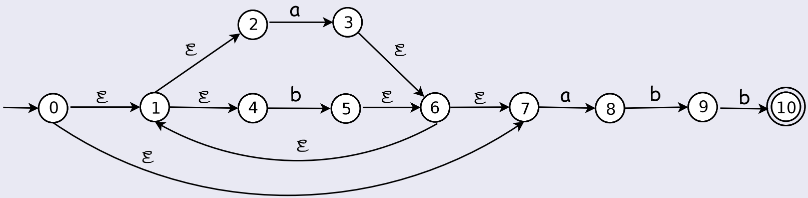
Ejemplo



1. $A = \varepsilon - clausura(0) = \{0, 1, 2, 4, 7\} = q'_0$
2. $\delta(A, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B \rightarrow \delta'(A, a) = B$

Algoritmo de construcción de subconjuntos

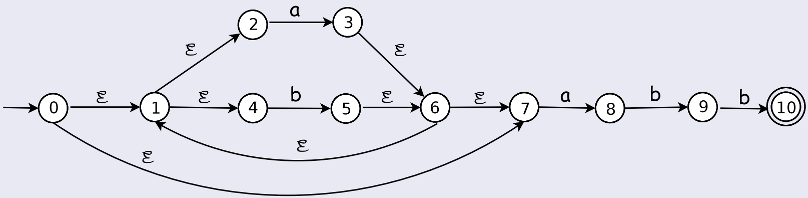
Ejemplo



1. $A = \varepsilon - clausura(0) = \{0, 1, 2, 4, 7\} = q'_0$
2. $\delta(A, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B \rightarrow \delta'(A, a) = B$
3. $\delta(A, b) = \{5\}$
   $\varepsilon - clausura(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C \rightarrow \delta'(A, b) = C$

Algoritmo de construcción de subconjuntos

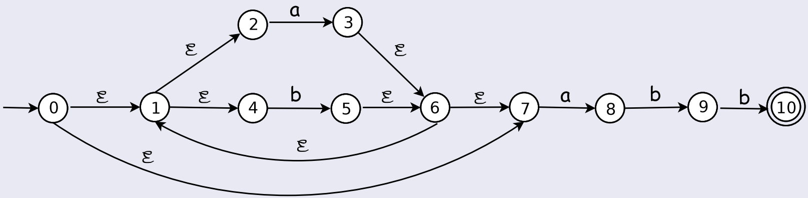
Ejemplo



1. $A = \varepsilon - clausura(0) = \{0, 1, 2, 4, 7\} = q'_0$
2. $\delta(A, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B \rightarrow \delta'(A, a) = B$
3. $\delta(A, b) = \{5\}$
   $\varepsilon - clausura(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C \rightarrow \delta'(A, b) = C$
4. $\delta(B, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = B \rightarrow \delta'(B, a) = B$

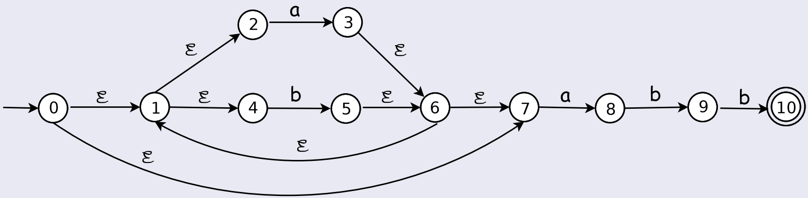
Algoritmo de construcción de subconjuntos

Ejemplo



Algoritmo de construcción de subconjuntos

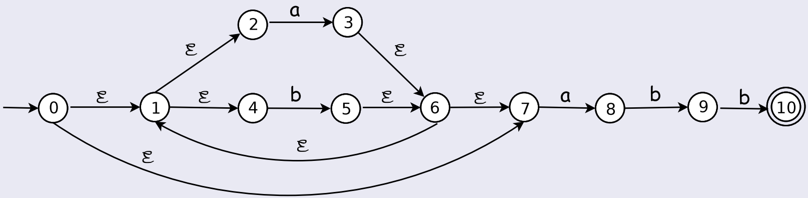
Ejemplo



5. $\delta(B, b) = \{5, 9\}$
   $\varepsilon - clausura(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\} = D \rightarrow \delta'(B, b) = D$

Algoritmo de construcción de subconjuntos

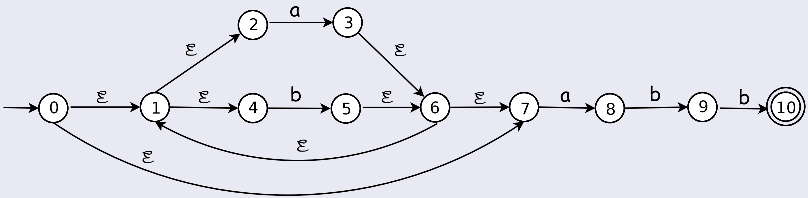
Ejemplo



5. $\delta(B,b)=\{5,9\}$
   $\varepsilon - clausura(\{5,9\}) = \{1,2,4,5,6,7,9\} = D \rightarrow \delta'(B,b) = D$
6. $\delta(C,a)=\{3,8\}$
   $\varepsilon - clausura(\{3,8\}) = B \rightarrow \delta'(C,a) = B$

Algoritmo de construcción de subconjuntos

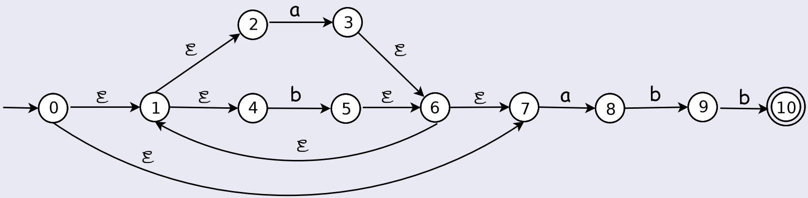
Ejemplo



5. $\delta(B, b) = \{5, 9\}$
   $\varepsilon - clausura(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\} = D \rightarrow \delta'(B, b) = D$
6. $\delta(C, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = B \rightarrow \delta'(C, a) = B$
7. $\delta(C, b) = \{5\}$
   $\varepsilon - clausura(\{5\}) = C \rightarrow \delta'(C, b) = C$

Algoritmo de construcción de subconjuntos

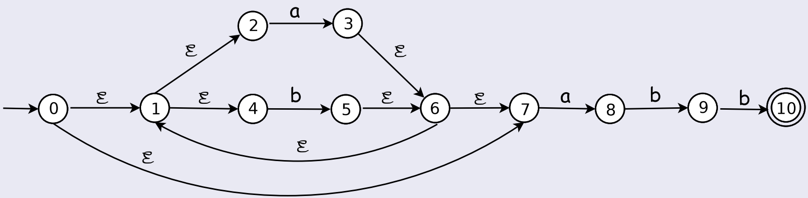
Ejemplo



5. $\delta(B,b) = \{5,9\}$
   $\varepsilon - clausura(\{5,9\}) = \{1,2,4,5,6,7,9\} = D \to \delta'(B,b) = D$
6. $\delta(C,a) = \{3,8\}$
   $\varepsilon - clausura(\{3,8\}) = B \to \delta'(C,a) = B$
7. $\delta(C,b) = \{5\}$
   $\varepsilon - clausura(\{5\}) = C \to \delta'(C,b) = C$
8. $\delta(D,a) = \{3,8\}$
   $\varepsilon - clausura(\{3,8\}) = B \to \delta'(D,a) = B$

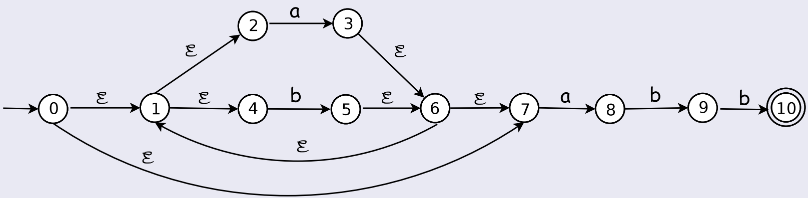
Algoritmo de construcción de subconjuntos

Ejemplo



Algoritmo de construcción de subconjuntos

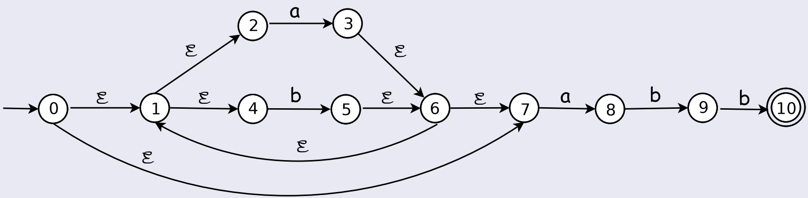
Ejemplo



9. $\delta(D, b) = \{5, 10\}$
   $\varepsilon - clausura(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\} = E \rightarrow \delta'(D, b) = E$

Algoritmo de construcción de subconjuntos

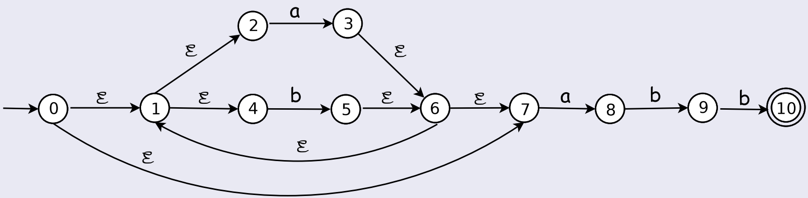
Ejemplo



9. $\delta(D, b) = \{5, 10\}$
   $\varepsilon - clausura(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\} = E \rightarrow \delta'(D, b) = E$
10. $\delta(E, a) = \{3, 8\}$
   $\varepsilon - clausura(\{3, 8\}) = B \rightarrow \delta'(E, a) = B$

Algoritmo de construcción de subconjuntos

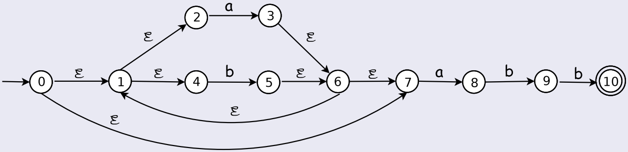
Ejemplo



9. $\delta(D, b) = \{5, 10\}$
   $\varepsilon - clausura(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\} = E \rightarrow \delta'(D, b) = E$
10. $\delta(E, a) = \{3, 8\}$
    $\varepsilon - clausura(\{3, 8\}) = B \rightarrow \delta'(E, a) = B$
11. $\delta(E, b) = \{5\}$
    $\varepsilon - clausura(\{5\}) = C \rightarrow \delta'(E, b) = C$

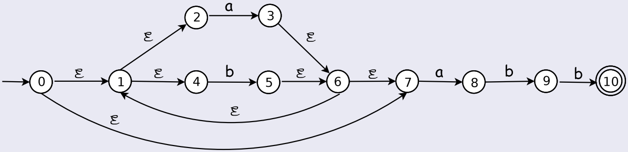
Algoritmo de construcción de subconjuntos

NFA original

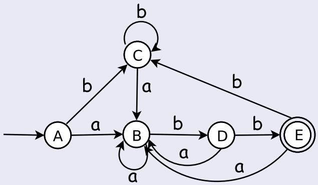


Algoritmo de construcción de subconjuntos

NFA original

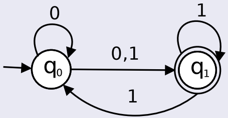


DFA resultante



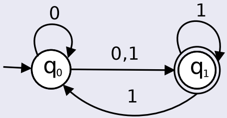
Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$



Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

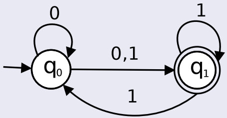


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

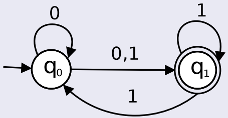


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

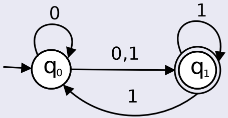


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

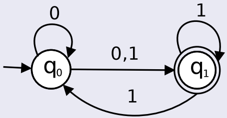


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

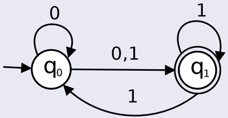


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

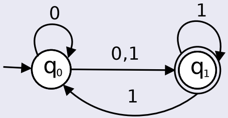


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

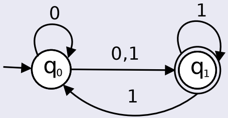


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$
- $\delta'(\{q_1\}, 1) = \delta(q_1, 1) = \{q_0, q_1\}$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$

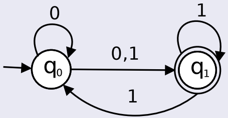


Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$
- $\delta'(\{q_1\}, 1) = \delta(q_1, 1) = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$

Algoritmo de construcción de subconjuntos

Aplicar el algoritmo a un NFA sin $\varepsilon - transiciones$



Construyendo el DFA equivalente

- $M' \equiv (Q', \{0,1\}, \delta', \varepsilon - clausura(q_0), F')$
- $Q' = 2^Q = \{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$
- $\delta'(\{q_1\}, 1) = \delta(q_1, 1) = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$

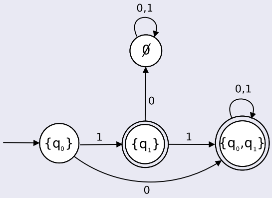
Algoritmo de construcción de subconjuntos

- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$
- $\delta'(\{q_1\}, 1) = \delta(q_1, 1) = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$

Algoritmo de construcción de subconjuntos

- $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$
- $\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_1\}$
- $\delta'(\{q_1\}, 0) = \delta(q_1, 0) = \emptyset$
- $\delta'(\{q_1\}, 1) = \delta(q_1, 1) = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta'(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$

El DFA resultante es:



Algoritmo de construcción de subconjuntos

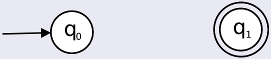
- En la práctica muchos estados del DFA no son accesibles desde q_0
- Por ello, para construir el DFA equivalente se comienza con $q_0 = \varepsilon - clausura(q_0)$ y se van añadiendo nuevos estados al DFA sólo si aparecen como resultado de alguna transición desde un estado existente

Indice

Lenguajes regulares y NFAs

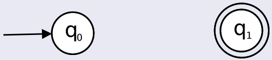
Lenguajes regulares y NFAs

$\emptyset$

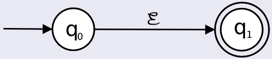


Lenguajes regulares y NFAs

$\emptyset$

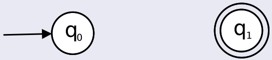


$\{\varepsilon\}$

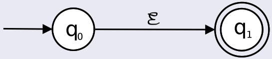


Lenguajes regulares y NFAs

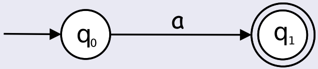
$\emptyset$



$\{\varepsilon\}$

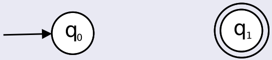


Lenguaje unitario

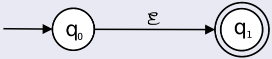


Lenguajes regulares y NFAs

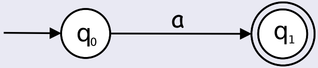
$\emptyset$



$\{\varepsilon\}$



Lenguaje unitario



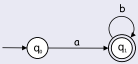
Sean R y S expresiones regulares

Supondremos que existen NFAs que reconocen los lenguajes $L(R)$ y $L(S)$

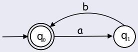
Lenguajes regulares y NFAs

Lenguajes regulares y NFAs

Sea R una expresión regular

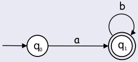


Sea S una expresión regular

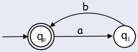


Lenguajes regulares y NFAs

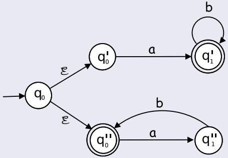
Sea R una expresión regular



Sea S una expresión regular



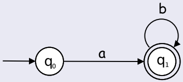
$R|S$



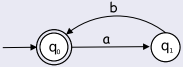
Lenguajes regulares y NFAs

Lenguajes regulares y NFAs

Sea R una expresión regular

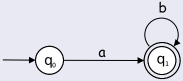


Sea S una expresión regular

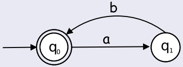


Lenguajes regulares y NFAs

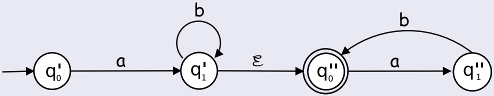
Sea R una expresión regular



Sea S una expresión regular

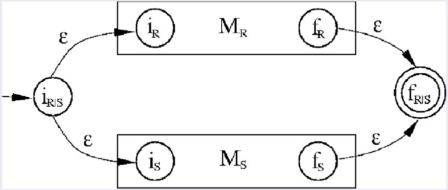


$R \cdot S$



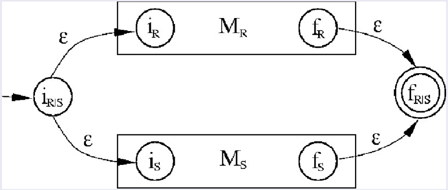
Lenguajes regulares y NFAs

Construcción de Thompson: Disyunción

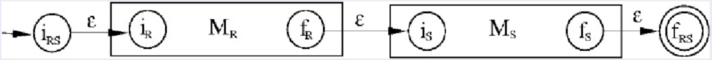


Lenguajes regulares y NFAs

Construcción de Thompson: Disyunción

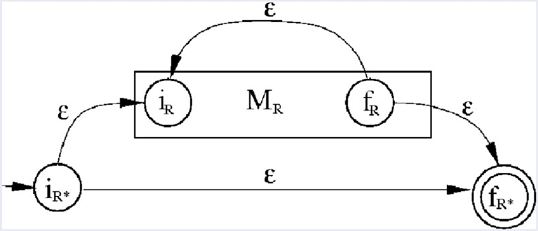


Construcción de Thompson: Concatenación



Lenguajes regulares y NFAs

Construcción de Thompson: Asterisco



Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

- $M(R)$ tiene a lo sumo el doble de estados que de símbolos y operadores que hay en R

Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

- $M(R)$ tiene a lo sumo el doble de estados que de símbolos y operadores que hay en R
- Esto se debe a que en cada paso de la construcción se crean a lo sumo dos nuevos estados

Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

- $M(R)$ tiene a lo sumo el doble de estados que de símbolos y operadores que hay en R
- Esto se debe a que en cada paso de la construcción se crean a lo sumo dos nuevos estados
- $M(R)$ tiene un único estado de aceptación

Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

- $M(R)$ tiene a lo sumo el doble de estados que de símbolos y operadores que hay en R
- Esto se debe a que en cada paso de la construcción se crean a lo sumo dos nuevos estados
- $M(R)$ tiene un único estado de aceptación
- El estado de aceptación no tiene transiciones salientes

Características de los autómatas

Para una expresión regular R, la construcción de Thompson produce un NFA $M(R)$ con las siguientes características:

- $M(R)$ tiene a lo sumo el doble de estados que de símbolos y operadores que hay en R
- Esto se debe a que en cada paso de la construcción se crean a lo sumo dos nuevos estados
- $M(R)$ tiene un único estado de aceptación
- El estado de aceptación no tiene transiciones salientes
- Cada estado de $M(R)$ tiene a lo sumo dos transiciones salientes

Teorema de Kleene

Lema

Sea M un autómata finito, existe una expresión regular R tal que $L(M) = L(R)$

Teorema de Kleene

Lema

Sea M un autómata finito, existe una expresión regular R tal que $L(M) = L(R)$

Demostración

Se deduce en un sentido, de la Construcción de Thompson (dada una expresión regular R se puede construir un NFA M tal que $L(M) = L(R)$) y en el otro sentido, del Lema de Arden, que permite hallar una expresión regular R que represente al lenguaje $L(M)$ que reconoce un autómata finito M

Teorema de Kleene

Lema

Sea M un autómata finito, existe una expresión regular R tal que $L(M) = L(R)$

Demostración

Se deduce en un sentido, de la Construcción de Thompson (dada una expresión regular R se puede construir un NFA M tal que $L(M) = L(R)$) y en el otro sentido, del Lema de Arden, que permite hallar una expresión regular R que represente al lenguaje $L(M)$ que reconoce un autómata finito M

Teorema de Kleene

Un lenguaje es regular $\Leftrightarrow$ es aceptado por un autómata finito

Lema del Bombeo

Lema

Sea $L \subseteq \Sigma^*$ un lenguaje regular infinito.
Entonces, existe una constante n, natural y positiva, tal que para cualquier cadena $z \in L$, con $|z| \geq n$, se verifica que:

Lema del Bombeo

Lema

Sea $L \subseteq \Sigma^*$ un lenguaje regular infinito.
Entonces, existe una constante n, natural y positiva, tal que para cualquier cadena $z \in L$, con $|z| \geq n$, se verifica que:

1. $z = uvw$, con u, v, $w \in \Sigma^*$
2. $uv^iw \in L$, $\forall i \geq 0$
3. $|v| \geq 1$ (v contiene al menos un símbolo)
4. $|uv| \leq n$

Lema del Bombeo

Lema

Sea $L \subseteq \Sigma^*$ un lenguaje regular infinito.
Entonces, existe una constante n, natural y positiva, tal que para cualquier cadena $z \in L$, con $|z| \geq n$, se verifica que:

1. $z = uvw$, con u, v, $w \in \Sigma^*$
2. $uv^iw \in L$, $\forall i \geq 0$
3. $|v| \geq 1$ (v contiene al menos un símbolo)
4. $|uv| \leq n$

Ademas, $n \leq |Q|$ siendo Q el conjunto de estados del DFA mínimo que reconoce L

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA
- Sea $z \in L$ tal que $z = a_1 a_2 ... a_m$, con $m \geq n$ ($|z| = m \geq n$)

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA
- Sea $z \in L$ tal que $z = a_1 a_2 ... a_m$, con $m \geq n$ ($|z| = m \geq n$)
- Puesto que $z \in L = L(M)$, el DFA acepta la cadena z: proceso de reconocimiento de una secuencia de m símbolos

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA
- Sea $z \in L$ tal que $z = a_1 a_2 ... a_m$, con $m \geq n$ ($|z| = m \geq n$)
- Puesto que $z \in L = L(M)$, el DFA acepta la cadena z: proceso de reconocimiento de una secuencia de m símbolos
- Al reconocer la cadena z, M pasa por $|\{q_{i_0}, q_{i_1}, ..., q_{i_{m-1}}, q_{i_m}\}| = m + 1$ estados

Lema del Bombeo

Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA
- Sea $z \in L$ tal que $z = a_1 a_2 ... a_m$, con $m \geq n$ ($|z| = m \geq n$)
- Puesto que $z \in L = L(M)$, el DFA acepta la cadena z: proceso de reconocimiento de una secuencia de m símbolos
- Al reconocer la cadena z, M pasa por $|\{q_{i_0}, q_{i_1}, ..., q_{i_{m-1}}, q_{i_m}\}| = m + 1$ estados
- Puesto que M sólo tiene n estados y $m \geq n$, en esa secuencia tiene que haber algún estado repetido (por el que se pasa más de una vez)

Lema del Bombeo

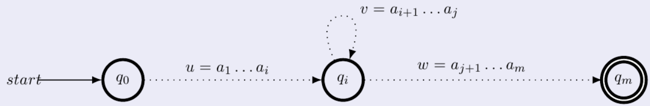
Demostración

- Puesto que L es regular, existe un DFA mínimo $M \equiv (Q, \Sigma, \delta, q_0, F)$ que reconoce L: $L(M) = L$
- Sea $n = |Q|$ el número de estados de ese DFA
- Sea $z \in L$ tal que $z = a_1 a_2 ... a_m$, con $m \geq n$ ($|z| = m \geq n$)
- Puesto que $z \in L = L(M)$, el DFA acepta la cadena z: proceso de reconocimiento de una secuencia de m símbolos
- Al reconocer la cadena z, M pasa por $|\{q_{i_0}, q_{i_1}, ..., q_{i_{m-1}}, q_{i_m}\}| = m + 1$ estados
- Puesto que M sólo tiene n estados y $m \geq n$, en esa secuencia tiene que haber algún estado repetido (por el que se pasa más de una vez)
- Luego habrá dos estados en esa secuencia que son iguales: $q_{i_k} = q_{i_j}$

Lema del Bombeo

Demostración

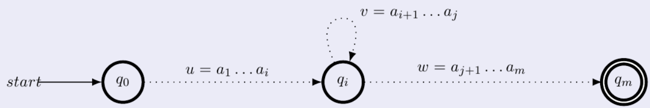
- Al haber dos estados iguales en la secuencia, la situación sería (se dibujan sólo el estado inicial, el final y el repetido):



Lema del Bombeo

Demostración

- Al haber dos estados iguales en la secuencia, la situación sería (se dibujan sólo el estado inicial, el final y el repetido):



- Si llamamos:

$u = a_1 a_2 ... a_{i-1} a_i$

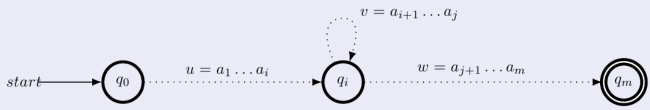
$v = a_{i+1} a_{i+2} ... a_{j-1} a_j$

$w = a_{j+1} a_{j+2} ... a_{m-1} a_m$

Lema del Bombeo

Demostración

- Al haber dos estados iguales en la secuencia, la situación sería (se dibujan sólo el estado inicial, el final y el repetido):



- Si llamamos:

  $u = a_1 a_2 \ldots a_{i-1} a_i$

  $v = a_{i+1} a_{i+2} \ldots a_{j-1} a_j$

  $w = a_{j+1} a_{j+2} \ldots a_{m-1} a_m$

- Resulta que $z = uvw$ (Condición 1)

Lema del Bombeo

Demostración

Y también se cumple:

- $\delta(q_i, v) = q_i = q_j$ y, por tanto, $z' = a_1 a_2 ... a_i a_{j+1} ... a_m \in L$

Lema del Bombeo

Demostración

Y también se cumple:

- $\delta(q_i, v) = q_i = q_j$ y, por tanto, $z' = a_1 a_2 ... a_i a_{j+1} ... a_m \in L$
- $uv^i w \in L$, $\forall i \geq 0$ (Condición 2)

Lema del Bombeo

Demostración

Y también se cumple:

- $\delta(q_i, v) = q_i = q_j$ y, por tanto, $z' = a_1a_2...a_ia_{j+1}...a_m \in L$
- $uv^iw \in L$, $\forall i \geq 0$ (Condición 2)
- $|v| \geq 1$ (Condición 3) puesto que al no haber $\varepsilon - transiciones$ se necesita al menos un símbolo de z para pasar de q_{i_k} a q_{i_j}

Lema del Bombeo

Demostración

Y también se cumple:

- $\delta(q_i, v) = q_i = q_j$ y, por tanto, $z' = a_1 a_2 ... a_i a_{j+1} ... a_m \in L$
- $uv^i w \in L$, $\forall i \geq 0$ (Condición 2)
- $|v| \geq 1$ (Condición 3) puesto que al no haber $\varepsilon - transiciones$ se necesita al menos un símbolo de z para pasar de q_{i_k} a q_{i_j}
- $|uv| \leq n$ (Condición 4), puesto que en el mejor de los casos podremos leer $n-1$ símbolos de z sin que se repita ningún estado, pero al leer el n-ésimo símbolo, es seguro que se repetirá algún estado

Lema del Bombeo

Utilidad del lema

El *Lema del Bombeo* (LB, Pumping Lemma) es una condición necesaria, pero no suficiente:

- Si un lenguaje es regular, el Lema del Bombeo se cumple, pero es posible que un L no regular cumpla el lema
- Por ello, el Lema del Bombeo se utiliza para demostrar que un $L \subseteq \Sigma^*$ **NO es regular**
- Por el contrario, el Lema del Bombeo NO puede utilizarse para demostrar que un lenguaje SÍ sea regular: *Si un lenguaje cumple el LB, es posible que sea regular o que no lo sea*
- Para demostrar que un L no cumple el LB se procede por contradicción: se supone que el lema se cumple y se intenta llegar a una contradicción

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular
2. Se elige una cadena z que cumpla $z \in L$ y también $|z| \geq n$

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular
2. Se elige una cadena z que cumpla $z \in L$ y también $|z| \geq n$
3. Puesto que no se conoce n, se han de considerar todas las posibilidades

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular
2. Se elige una cadena z que cumpla $z \in L$ y también $|z| \geq n$
3. Puesto que no se conoce n, se han de considerar todas las posibilidades
4. Se divide la cadena z en tres partes: $z = uvw$, cumpliendo que: $|uv| \leq n$, $|v| \geq 1$

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular
2. Se elige una cadena z que cumpla $z \in L$ y también $|z| \geq n$
3. Puesto que no se conoce n, se han de considerar todas las posibilidades
4. Se divide la cadena z en tres partes: $z = uvw$, cumpliendo que: $|uv| \leq n$, $|v| \geq 1$
5. Se trata de hallar un valor para $i = 0, 1, \ldots$ tal que $uv^i w \notin L$

Lema del Bombeo

Pasos para demostrar que un L no cumple el Lema del Bombeo

1. Identificar el lenguaje L que se pretende demostrar que no es regular
2. Se elige una cadena z que cumpla $z \in L$ y también $|z| \geq n$
3. Puesto que no se conoce n, se han de considerar todas las posibilidades
4. Se divide la cadena z en tres partes: $z = uvw$, cumpliendo que: $|uv| \leq n$, $|v| \geq 1$
5. Se trata de hallar un valor para $i = 0, 1, \ldots$ tal que $uv^i w \notin L$

Basta con demostrar que, para cualquier n, se puede encontar una cadena z y que, para cualquier forma de descomponer $z = uvw$ existe un i tal que $uv^i w \notin L$

Lema del Bombeo. Ejemplo

$L = \{0^i 1^i \mid i \geq 0\}$

1. Se fija la constante del lema, n (no se conoce su valor, pero es fijo)
2. Se elige $z = 0^n 1^n$ que cumple $z \in L$ y también $|z| = 2n \geq n$
3. Se tienen que cumplir todas las condiciones que plantea el lema, y hay que analizar todas las posibles descomposiciones de z Puesto que $|uv| \leq n$ las cadenas v y u sólo pueden contener símbolos "0". Todas las posibilidades son:
   - $u = 0^p$
   - $v = 0^q$
   - $w = 0^{n-p-q} 1^n$

   Cumpliéndose que $p + q \leq n$ y también $1 \leq q \leq n$
4. Si se elige $i = 2$ en la condición 3 del Lema del Bombeo:
   ¿$z' = uv^2w = 0^p 0^{2q} 0^{n-p-q} 1^n \in L$?
   ¿$p + 2q + n - p - q = n$? $\rightarrow$ ¿$q + n = n$?
   Sólo es cierto si $q = 0$, lo que es falso por hipótesis
5. Hemos alcanzado la contradiccion: $z' \notin L \Rightarrow L$ no es regular

Lema del Bombeo. Ejemplo

$L = \{$Palíndromos en $\{0, 1\}\}$

1. Se fija la constante del lema, n
2. Se elige $z = 0^n 1 0^n$ que cumple $z \in L$ y también $|z| = 2n + 1 \geq n$.
3. Se tienen que cumplir todas las condiciones que plantea el lema. Puesto que $|uv| \leq n$ la cadena v sólo puede contener símbolos "0". Todas las posibilidades son:
   - $u = 0^p$
   - $v = 0^q$
   - $w = 0^{n-p-q}\ 1\ 0^n$

   Cumpliéndose que $p + q \leq n$ y también $1 \leq q \leq n$.
4. Si se elige $i = 2$ en la condición 3 del Lema del Bombeo:
   ¿$z' = uv^2w = 0^p 0^{2q} 0^{n-p-q}\ 1\ 0^n \in L$?
   ¿$p + 2q + n - p - q = n$? $\rightarrow$ ¿$q + n = n$?
   Sólo es cierto si $q = 0$, lo que es falso por hipótesis
5. Se ha llegado a una contradiccion: $z' \notin L \Rightarrow L$ no es regular

Lema del Bombeo. Ejemplo

$L = \{ww \mid w \in L((a|b)*)\}$

1. Se fija la constante del lema, n (no se conoce su valor, pero es fijo)
2. Consideremos $z = a^n b a^n b$ que cumple $z \in L$ y también $|z| = 2n + 2 \geq n$
3. Se tienen que cumplir todas las condiciones que plantea el lema, y hay que analizar todas las posibles descomposiciones de z. Puesto que $|uv| \leq n$ la cadena v sólo puede contener símbolos "a". Todas las posibilidades son:
   - $u = a^p$
   - $v = a^q$
   - $w = a^{n-p-q} b a^n b$

   Cumpliéndose que $p + q \leq n$ y también $1 \leq q \leq n$
4. Si se elige $i = 2$ en la condición 3 del Lema del Bombeo:
   ¿$z' = uv^2w = a^p a^{2q} a^{n-p-q} b a^n b \in L$?
   ¿$p + 2q + n - p - q = n$? $\rightarrow$ ¿$q + n = n$?
   Sólo es cierto si $q = 0$, lo que es falso por hipótesis
5. Se ha llegado a una contradiccion: $z' \notin L \Rightarrow L$ no es regular

Lema del Bombeo. Ejemplo

$L = \{w \mid w \in L((a|b)*) \wedge w$ tiene un número par de $aes\}$

1. Se fija la constante del lema, n
2. Consideremos $z = a^{2n}b$ que cumple $z \in L$ y también $|z| = 2n + 1 \geq n$
3. Se tienen que cumplir todas las condiciones que plantea el lema Puesto que $|uv| \leq n$ la cadena v sólo puede contener símbolos "a". Todas las posibilidades son:
    - $u = a^p$
    - $v = a^q$
    - $w = a^{2n-p-q}b$

   Cumpliéndose que $p + q \leq n$ y también $1 \leq q \leq n$
4. ¿$z' = uv^iw = a^pa^{iq}a^{2n-p-q}b \in L$?
   ¿$p + iq + 2n - p - q = 2n + q(i - 1) = par$?
   Siempre que q sea par $z' = uv^iw \in L \; \forall i \geq 0$
5. Hay que acreditar que no hay ninguna forma de descomponer $w = xyz$ que cumpla con las condiciones del Lema del Bombeo.
   Si existe alguna descomposición $z' = uv^iw$ posible, que cumpla con las condiciones, el lema se cumple

Lema del Bombeo. Ejemplo

$L = \{w \mid w \in L((a|b)*) \wedge w$ tiene un número par de $aes\}$

- Se ha fallado al demostrar que L no cumple el Lema del Bombeo, pero **¿significa eso que L es regular?**:

Lema del Bombeo. Ejemplo

$L = \{w \mid w \in L((a|b)*) \land w$ tiene un número par de $aes\}$

- Se ha fallado al demostrar que L no cumple el Lema del Bombeo, pero **¿significa eso que L es regular?**:
  - No: es posible que no se haya elegido una cadena z adecuada para hacer la demostración
  - El no ser capaz de diseñar un NFA que reconozca un L dado tampoco demuestra que ese lenguaje no sea regular

Lema del Bombeo. Ejemplo

$L = \{w \mid w \in L((a|b)*) \wedge w$ tiene un número par de $aes\}$

- Se ha fallado al demostrar que L no cumple el Lema del Bombeo, pero **¿significa eso que L es regular?**:
  - No: es posible que no se haya elegido una cadena z adecuada para hacer la demostración
  - El no ser capaz de diseñar un NFA que reconozca un L dado tampoco demuestra que ese lenguaje no sea regular
- Entonces, **¿cómo se puede demostrar que un lenguaje L es regular?**:

Lema del Bombeo. Ejemplo

$L = \{w \mid w \in L((a|b)*) \wedge w$ tiene un número par de $aes\}$

- Se ha fallado al demostrar que L no cumple el Lema del Bombeo, pero **¿significa eso que L es regular?**:
  - No: es posible que no se haya elegido una cadena z adecuada para hacer la demostración
  - El no ser capaz de diseñar un NFA que reconozca un L dado tampoco demuestra que ese lenguaje no sea regular
- Entonces, **¿cómo se puede demostrar que un lenguaje L es regular?**:
  - Creando un NFA o un DFA que reconozca L o una expresión regular que lo represente

Lema del Bombeo

$L = \{a^i b^j c^k \mid (i = 0) \vee (j = k)\}$ no es regular, pero cumple el LB

Veamos que se cumplen las condiciones del LB para $n = 2$.
Hay 2 posibilidades:

1. $i = 0$. En este caso, las cadenas de L son de la forma $z = b^j c^k$. Se puede descomponer $z = uvw$ tomando:
   - $u = \varepsilon$
   - v es el primer símbolo de z
   - w es z salvo su primer símbolo
   - De este modo se cumple:
     1. $|uv| = |v| = 1 \leq n = 2$
     2. $|v| = 1 \geq 1$
     3. $uv^s w \in L \; \forall s \geq 0$ (Secuencia de b's seguidas de c's)

Lema del Bombeo

$L = \{a^i b^j c^k \mid (i = 0) \vee (j = k)\}$ no es regular, pero cumple el LB

2. Si $i \neq 0$, las cadenas de L son de la forma $z = a^i b^j c^j$
   En este caso se puede descomponer $z = uvw$ tomando:
   - $u = \varepsilon$
   - $v = a$ es el primer símbolo de z (como $i \neq 0$ es seguro que z comienza por a)
   - w es z salvo su primer símbolo
   - De este modo se cumple:
     1. $|uv| = |v| = 1 \leq n = 2$
     2. $|v| = 1 \geq 1$
     3. $uv^s w \in L \ \forall s \geq 0$ (Secuencia de a's seguidas de b's seguidas de c's, siendo el número de b's igual al número de c's)

$L(M)$ no es vacío

Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un autómata finito con $|Q| = k$ estados

Teorema

$L(M) \neq \emptyset \Leftrightarrow M$ acepta una cadena de longitud menor que k

$L(M)$ no es vacío

Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un autómata finito con $|Q| = k$ estados

Teorema

$L(M) \neq \emptyset \Leftrightarrow M$ acepta una cadena de longitud menor que k

Demostración

1. $L(M) \neq \emptyset \Leftarrow M$ acepta una cadena de longitud menor que k
   - Si M acepta una cadena w de longitud $|w| < k$, entonces es obvio que $L(M) \neq \emptyset$, puesto que la cadena aceptada pertenece a $L(M)$
2. $L(M) \neq \emptyset \Rightarrow M$ acepta una cadena de longitud menor que k
   - Sea $z \in L(M)$, entonces:
     - Si $|z| < k$, el teorema se cumple
     - Si $|z| \geq k$, entonces $z = uvw$ (aplicando el Lema del Bombeo) y $z' = uv^iw \in L(M)\ \forall i \geq 0$
     - Puesto que $|v| \geq 1, z'' = uw \in L(M)$ y $|z''| < |z|$
       Resulta un mecanismo que permite disminuir (reiteradamente) la longitud de z hasta que su longitud sea inferior a k, cumpliendo así el teorema

$L(M)$ es infinito

Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un autómata finito con $|Q| = k$ estados

Teorema

$L(M)$ es infinito $\Leftrightarrow$ M acepta una cadena de longitud n, $k \leq n < 2k$

$L(M)$ es infinito

Sea $M \equiv (Q, \Sigma, \delta, q_0, F)$ un autómata finito con $|Q| = k$ estados

Teorema

$L(M)$ es infinito $\Leftrightarrow$ M acepta una cadena de longitud n, $k \leq n < 2k$

Demostración

1. ($\Leftarrow$) Si $z \in L(M)$ con $|z| \geq k$, por el Lema del Bombeo, $z = uvw$ y $\forall\ i \geq 0$, $z' = uv^iw \in L(M)$, luego $L(M)$ es infinito
2. ($\Rightarrow$) Si $L(M)$ es infinito, entonces no todas las cadenas de $L(M)$ tendrán longitud inferior a k y, por lo tanto, $\exists z \in L(M)$ cumpliendo $|z| \geq k$
   - Si $|z| < 2k$, el teorema queda demostrado
   - Si $|z| \geq 2k$, $z = uvw$ cumpliendo $1 \leq |v| \leq k$. Como se sabe que $2k \leq |uvw| = |z|$ y $|v| \leq k$, entonces $|uw| \geq k$. Igual que anteriormente, si $|uw| < 2k$ el teorema queda demostrado, y si no, se aplica este proceso de disminución de la longitud de la cadena hasta hallar una cuya longitud esté entre k y $2k - 1$

Propiedades de los lenguajes regulares

Ya se ha estudiado que...

El conjunto de los lenguajes regulares es cerrado bajo las operaciones de unión, concatenación y cierre de Kleene

Propiedades de los lenguajes regulares

Ya se ha estudiado que...

El conjunto de los lenguajes regulares es cerrado bajo las operaciones de unión, concatenación y cierre de Kleene

Teorema

Los lenguajes regulares son cerrados respecto a la complementación

Propiedades de los lenguajes regulares

Ya se ha estudiado que...

El conjunto de los lenguajes regulares es cerrado bajo las operaciones de unión, concatenación y cierre de Kleene

Teorema

Los lenguajes regulares son cerrados respecto a la complementación

Demostración

Sea L un lenguaje regular.
Entonces, existe un DFA, $M \equiv (Q, \Sigma, \delta, q_0, F)$ tal que $L(M) = L$

- Consideremos el DFA $M' \equiv (Q, \Sigma, \delta, q_0, Q - F)$:
  - Si $w \in L(M) \Rightarrow w \notin L(M')$
  - Si $w \notin L(M) \Rightarrow w \in L(M')$
- Es decir, M' reconoce $\Sigma^* - L$, el complementario de L, y por tanto, $\overline{L}$ es regular

Propiedades de los lenguajes regulares

Ya se ha estudiado que...

El conjunto de los lenguajes regulares es cerrado bajo las operaciones de unión, concatenación y cierre de Kleene

Teorema

Los lenguajes regulares son cerrados respecto a la complementación

Demostración

Sea L un lenguaje regular.
Entonces, existe un DFA, $M \equiv (Q, \Sigma, \delta, q_0, F)$ tal que $L(M) = L$

- Consideremos el DFA $M' \equiv (Q, \Sigma, \delta, q_0, Q - F)$:
  - Si $w \in L(M) \Rightarrow w \notin L(M')$
  - Si $w \notin L(M) \Rightarrow w \in L(M')$
- Es decir, M' reconoce $\Sigma^* - L$, el complementario de L, y por tanto, $\overline{L}$ es regular

NOTA: Esta demostración no es válida para NFAs

Propiedades de los lenguajes regulares

Teorema

Los lenguajes regulares son cerrados respecto a la operación de intersección

Propiedades de los lenguajes regulares

Teorema

Los lenguajes regulares son cerrados respecto a la operación de intersección

Demostración

$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$

Puesto que L_1 y L_2 son lenguajes regulares y los lenguajes regulares son cerrados respecto a unión y complementación, queda demostrado que el lenguaje intersección $L_1 \cap L_2$ también es un lenguaje regular

Propiedades de los lenguajes regulares

$L = \{ww^i \mid w \in \{a|b\}^*\}$ no es regular

- Sea $L_1 = \{a^n b^{2k} a^n \mid n, k \geq 0\}$
  Mediante el LB se puede demostrar (¡hágalo!) que L_1 no es regular

Propiedades de los lenguajes regulares

$L = \{ww^i \mid w \in \{a|b\}^*\}$ no es regular

- Sea $L_1 = \{a^n b^{2k} a^n \mid n, k \geq 0\}$
  Mediante el LB se puede demostrar (¡hágalo!) que L_1 no es regular
- Sea $L_2 = \{a^n b^k a^m \mid n, k, m \geq 0\} = L(a{*}b{*}a{*})$
  Evidentemente L_2 es regular, porque se describe con una expresión regular

Propiedades de los lenguajes regulares

$L = \{ww^i \mid w \in \{a|b\}^*\}$ no es regular

- Sea $L_1 = \{a^n b^{2k} a^n \mid n, k \geq 0\}$
  Mediante el LB se puede demostrar (¡hágalo!) que L_1 no es regular
- Sea $L_2 = \{a^n b^k a^m \mid n, k, m \geq 0\} = L(a{*}b{*}a{*})$
  Evidentemente L_2 es regular, porque se describe con una expresión regular
- Se verifica que $L_1 = L \cap L_2$
  Si L fuera regular, $L_1 = L \cap L_2$ también lo sería, de modo que se concluye que L no es regular

Esta demostración ilustra el uso (ingenioso) de las propiedades de cierre para demostrar que un lenguaje no es regular

IMPORTANTE

Estas transparencias se utilizan ÚNICAMENTE como guía para el profesorado durante las clases.

Estas transparencias NO son un material completo y autocontenido para el uso del alumnado.