Expresiones regulares y conversiones

Teoría de Lenguajes

DC-UBA

April 14, 2024

Plan de la clase

- Expresiones regulares
- Conversión de AF a ER
- Conversión de ER a AF
- Ejercicio integrador

Expresiones regulares

Expresiones regulares

- Las expresiones regulares se utilizan para *denotar* o *definir* **lenguajes regulares**, al igual que los autómatas finitos.
- Son utilizadas por muchas herramientas como grep, awk, sed y editores de texto, y están implementadas en muchos lenguajes de programación. Vamos a ver una versión más simple.
- Ejemplo clásico: `*.txt`.

Ejemplos motivadores

Ejemplos motivadores:

- Cadenas de as y bs de longitud mayor o igual que 1:

$$(a|b)^+$$

- Números naturales sin ceros no significativos con signo opcional:

$$0 \,\Big|\, (- \mid \lambda)(1|\ldots|9)(0|1|\ldots|9)^*$$

¿Cómo se definen?

Son expresiones regulares:

- $\emptyset$
- λ
- a (para cada $a \in \Sigma$)

Además, si S y R son expresiones regulares, entonces también los son:

- $L(R).L(S)$ (Concatenación)
- $R|S$ (Unión)
- R^* (Clausura de Kleene, o informalmente estrella)
- R^+ (Clausura positiva, o informalmente "más")

Lenguaje denotado

Cada expresión regular e va a denotar un lenguaje regular $L(e)$.
Sean $a \in \Sigma$ un símbolo, R, S expresiones regulares

e	**Nombre**	$L(e)$
$\emptyset$	Vacío	$\emptyset$
λ	Cadena vacía	$\{\lambda\} = \Lambda$
a	Símbolo	$\{a\}$
$R.S$	Concatenación	$L(R).L(S)$
$R\|S$	Unión	$L(R) \cup L(S)$
R^*	Clausura de Kleene	$(L(R))^*$
R^+	Clausura positiva	$L(R.R^*)$

Definición (Equivalencia de ERs)

Dos expresiones regulares e y e' se dicen **equivalentes** si $L(e) = L(e')$. Lo notamos como $e = e'$.

Precedencia

Podemos poner paréntesis para hacer explícito cómo agrupar los operadores, pero ¿cómo interpretamos estas expresiones?

- $1.2^* = 1.(2^*)$ o $(1.2)^*$
- $0.1|2 = (0.1)|2$ o $0.(1|2)$
- $0|1^* = (0|1)^*$ o $0|(1^*)$

Precedencia

La precedencia de los operadores es $1) * 2) \ . \ 3) \ |$

¿Qué es la precedencia?

Nos dice cuales operadores se aplican primero. Por ejemplo para expresiones aritméticas, $\times$ tiene mayor precedencia que $+$, por lo que $1 + 2 \times 3 = 1 + (2 \times 3)$.

Ejercicio 0

Precedencia

La precedencia de los operadores es 1) $*$ 2) . 3) $|$

Ejercicio 0

¿Cómo se interpreta $0|0.1^*$?

$(0 \mid (\ 0\ .\ (1^*)))$ y se suele escribir como $0|01^*$

Ejercicio 1

Sea $\Sigma = \{0, 1\}$, dar una expresión regular que defina:

a. Cadenas terminadas en 01:

$$(((0|1)^*).0).1 = (0|1)^*01$$

b. Cadenas que tienen como subcadena a 000:

$$(((((0|1)^*).0).0).0).((0|1)^*) = (0|1)^*000(0|1)^*$$

Asociatividad

La unión y concatenación son asociativas:

$$(R|S)|T = R|(S|T) = R|S|T$$
$$(R.S).T = R.(S.T) = R.S.T$$

c. Cadenas con 0s y 1s alternados:

$$(0|\lambda)(10)^*(1|\lambda)$$

d. Complemento del primero (Cadenas que no terminan en 01):

$$(0|1)^*(00|10|11) \mid 0|1|\lambda = (0|1)^*(0|11) \mid 1|\lambda$$

Aclaraciones

- No vamos a pedir que demuestren rigurosamente que la ER genera el lenguaje pedido, pero **tiene que haber una justificación convincente de por qué lo hace**.
- El poder expresivo de las ER y AF son el mismo (lenguajes regulares), pero **a veces que un formalismo es más intuitivo para un lenguaje que otro**. Por ejemplo, el ejercicio de cantidad par de 0s (subitem del ejercicio 1 de la práctica de expresiones regulares) no es tan sencillo con ERs como con AFs.
- También es más cómodo trabajar complemento, iniciales/finales/subcadenas, reversa, etc. de un lenguaje sobre AFs que sobre ERs.[1]
- Recordemos que **aceptar** (o denotar) un lenguaje L se refiere *exactamente* a L, y no alguno incluido o otro que lo incluya.

[1] **Spoiler**: Útil para el ejercicio integrador del final

Otras propiedades

- **Conmutatividad de $|$**

$$u|v = v|u$$

(notar que . no es conmutativo)

- **Distributividad de . respecto de $|$**

$$u(v|w) = uv|uw$$
$$(u|v)w = uw|vw$$

- **Elemento neutro de .**: $u.\lambda = \lambda.u = u$
- **Elemento absorbente de .**: $u.\emptyset = \emptyset.u = \emptyset$
- **Elemento neutro de $|$**: $u|\emptyset = \emptyset|u = u$

Extensiones que suelen aparecer

Muchos lenguajes de programación implementan expresiones regulares y suelen agregar otras características a parte de los operadores que vimos. Si R es una expresión regular, n, m naturales:

- `R?` $= R|\lambda$
- `R{n}` $= R^n$
- `R{n, m}` $= R^n|R^{n+1}|\ldots|R^m$
- `()` $= \lambda$
- `.` $=$ cualquier carácter
- `^` $=$ comienzo de línea
- `$` $=$ fin de línea
- `\` para escapear carácteres especiales

Si quieren jugar un poco visiten `https://regexr.com/`

Extensiones que suelen aparecer

Conjuntos de caracteres:

- `[abc]` $= a|b|c$
- `[a - z]` $= a|\ldots|z$
- `^` al comienzo de la lista indica el complemento,
  `[^abc]` = cualquier cáracter menos a, b o c.

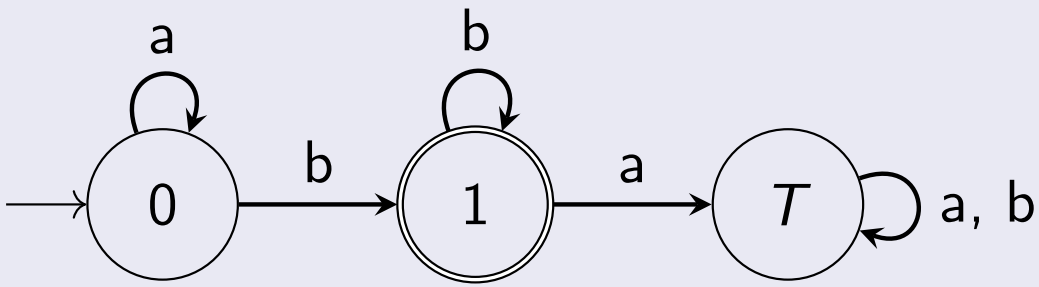
Hay clases de carácteres predefinidas: (en POSIX) *alnum*, *alpha*, *digit*, *space*, ...

Conversión de AF a ER

Ejercicio 2

Convertir el siguiente autómata a expresión regular

$$A = \langle \{0, 1\}, \{a, b\}, \delta, 0, \{1\} \rangle$$



A *ojímetro*: $a^* b^+$

¿Pero si es más complicado? Vamos a proponer un **método** más mecánico

Repaso autómatas finitos

Sea $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFD, donde

- Q es el conjunto de estados
- Σ es el alfabeto
- $\delta : Q \times \Sigma \to Q$ es la función de transición
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales.

A partir de δ se define la relación $\vdash$ entre configuraciones instantáneas (elementos de $Q \times \Sigma^*$):

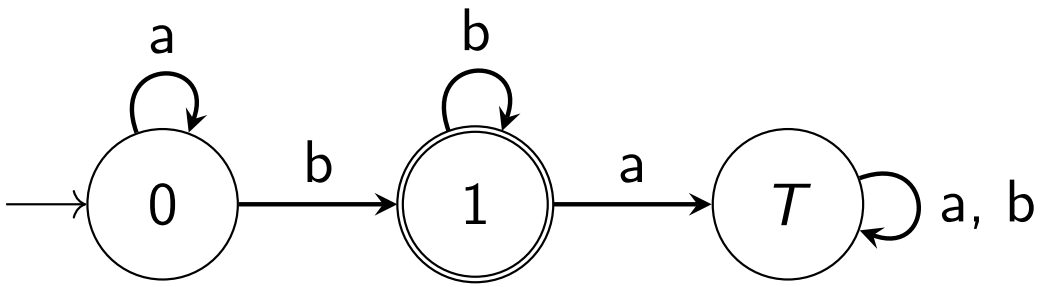
$$(q_i, a\alpha) \vdash (q_j, \alpha) \iff \delta(q_i, a) = q_j.$$

Y el lenguaje aceptado por el autómata se define como

$$L(A) = \{\alpha \in \Sigma^* \mid \exists q_f \in F : (q_0, \alpha) \vdash^* (q_f, \lambda)\}.$$

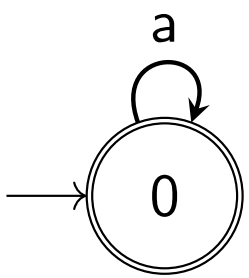
Intuición

- Tenemos L, el conjunto de cadenas aceptadas partiendo de q_0.
- Vamos a pensar en L_i = **el lenguaje aceptado partiendo del estado** q_i.
- Queremos denotar cada uno con una *expresión regular*.
- De esa forma, la ER que denote $L_0 = L(A)$ va a denotar el lenguaje que acepta el autómata.



Lenguaje aceptado por un estado

¿Cómo podemos expresar el lenguaje aceptado por un estado con una ER?
Por ejemplo, si tenemos el siguiente autómata



Genera o bien λ (base, por ser final), o a y una cadena que acepte L_0 (recursivo). Podemos expresarlo con una ecuación:

$$L_0 = a.L_0 \mid \lambda$$

Contamos con un resultado que nos brinda una forma de obtener una ER cerrada para este tipo de ecuaciones: El **Lema de Arden**

Lema de Arden

Lema (Arden)

Sea R un lenguaje, α, β expresiones regulares sobre Σ,

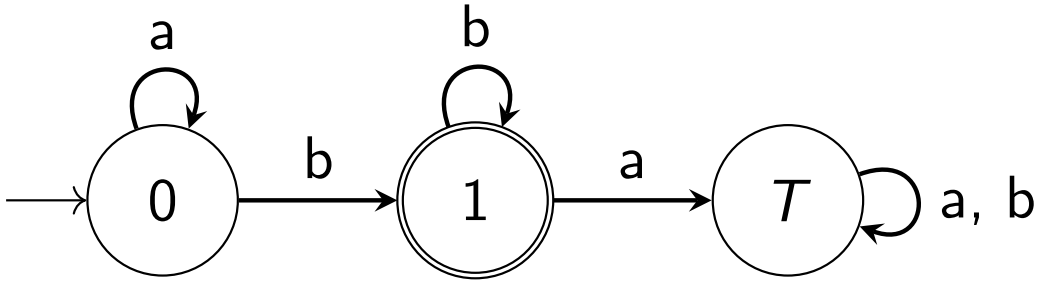
$$\text{Si } R = \alpha.R \mid \beta \text{ y } \lambda \notin L(\alpha)^a, \text{ entonces } R = \alpha^*.\beta$$

[a] Si no se cumple $\lambda \notin L(\alpha)$, la solución podría no ser única ni regular.

Intuición: α es el caso recursivo y β el caso base.
Usando Arden podemos resolver la ecuación anterior, con $\alpha = a$, $\beta = \lambda$ y $\lambda \notin L(a) = \{a\}$

$$\begin{aligned} L_0 &= a.L_0 \mid \lambda \\ &= a^*.\lambda \\ &= a^* \end{aligned}$$

Volvamos al autómata



Si escribimos estas *ecuaciones* para todos los estados, nos queda el siguiente sistema

$$L_0 = a.L_0 \mid b.L_1$$
$$L_1 = a.L_T \mid b.L_1 \mid \lambda$$
$$L_T = a.L_T \mid b.L_T \mid \emptyset$$

Si lo resolvemos, vamos a tener una expresión cerrada que denote $L_0 = L(A)$.

Resolviendo el sistema

Para resolver estos sistemas de ecuaciones, en general

- Elegimos una ecuación, la pasamos a la forma:

$$\alpha.R \mid \beta$$

- Usamos Arden para obtener una expresión cerrada.
- Reemplazamos en todas las ecuaciones que tengan dependencias.
- Repetir hasta conseguir L_0.

$$L_0 = a.L_0 \mid b.L_1$$
$$L_1 = a.L_T \mid b.L_1 \mid \lambda$$
$$L_T = a.L_T \mid b.L_T \mid \emptyset$$

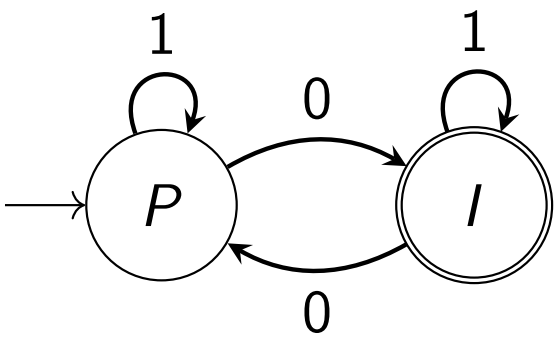
$$L_T = a.L_T \mid b.L_T \mid \emptyset = (a|b).L_T \mid \emptyset = (a|b)^*.\emptyset = \emptyset$$
$$L_1 = a.L_T \mid b.L_1 \mid \lambda = b.L_1 \mid \lambda = b^*.\lambda = b^*$$
$$L_0 = a.L_0 \mid b.L_1 = a.L_0 \mid b.b^* = a^*b^+$$

de esta forma, obtuvimos la ER a^*b^+ que denota $L_0 = L(A)$ y es la

Ejercicio 3

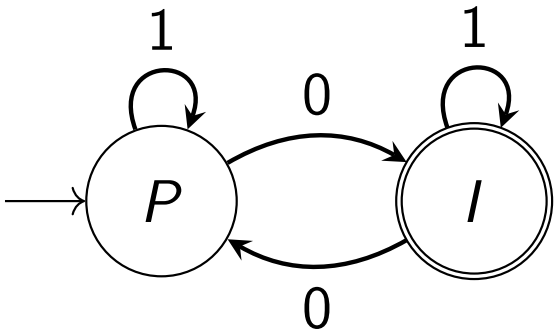
Veamos un ejemplo un poco más complicado. ¿Qué lenguaje reconoce el siguiente autómata?

$$A_3 = \langle \{P, I\}, \{0, 1\}, \delta, P, \{I\} \rangle$$



Cantidad impar de ceros. Ya no es trivial dar una ER. Usemos el método que vimos recién para convertirlo de forma mecánica.

Resolución



$$L_P = 1.L_P \mid 0.L_I$$
$$L_I = 1.L_I \mid 0.L_P \mid \lambda$$

Hay una dependencia mútua entre ambos, pero se puede resolver despejando sin mucho problema.

$$\begin{aligned}
L_I &= 1^*(0.L_p \mid \lambda) = 1^*0.L_P \mid 1^* && (\lambda \notin L(1), \text{Arden}) \\
L_P &= 1.L_P \mid 0.(1^*0.L_P \mid 1^*) && (\text{Reemplazando } L_I) \\
&= (1 \mid 01^*0).L_P \mid 01^* && (\text{Factor común}) \\
&= (1 \mid 01^*0)^*01^* && (\lambda \notin L(1 \mid 01^*0), \text{Arden})
\end{aligned}$$

Formalización - Generalización de lenguaje aceptado

De la misma forma que definimos el lenguaje L como el conjunto de cadenas aceptadas por el autómata partiendo del estado q_0,

$$L(A) = \{\alpha \in \Sigma^* \mid \exists q_f \in F : (q_0, \alpha) \vdash^* (q_f, \lambda)\},$$

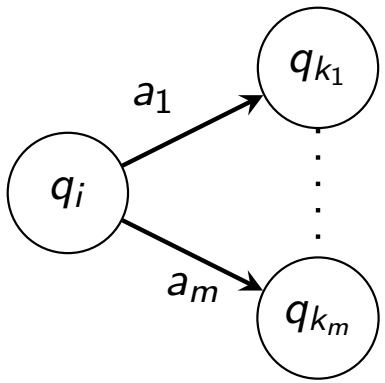
para cada estado q_i podemos definir un lenguaje L_i que corresponda al conjunto de cadenas aceptadas partiendo de él

$$L_i = \{\alpha \in \Sigma^* \mid \exists q_f \in F : (q_i, \alpha) \vdash^* (q_f, \lambda)\}.$$

Formalización - Ecuaciones

Queremos una forma de caracterizar el lenguaje generado por cada estado q_i con una expresión regular. Consideremos las siguientes igualdades para un estado q_i y $\Sigma = \{a_1, \ldots, a_m\}$

$$\begin{aligned} \delta(q_i, a_1) &= q_{k_1} \\ \delta(q_i, a_2) &= q_{k_2} \\ &\vdots \\ \delta(q_i, a_m) &= q_{k_m} \end{aligned}$$



A partir de ellas, podemos expresar el lenguaje L_i mediante la siguiente ecuación

$$L_i = a_1.L_{k_1} \mid a_2.L_{k_2} \mid \cdots \mid a_m.L_{k_m} \mid \epsilon(L_i)$$

donde

$$\epsilon(L) = \begin{cases} \emptyset & \text{si } \lambda \notin L \\ \{\lambda\} & \text{si } \lambda \in L \end{cases}$$

Formalización - Ecuaciones entre lenguajes

Es complicado dar un método general para resolver ecuaciones entre lenguajes. Pero para la conversión nos va a alcanzar con que tengan la forma $L = \alpha L \mid \beta$ donde α, β son expresiones regulares sobre Σ.

Lema (Arden)

Sea R un lenguaje, α, β expresiones regulares sobre Σ,

$$\text{Si } R = \alpha.R \mid \beta \text{ y } \lambda \notin L(\alpha), \text{ entonces } R = \alpha^*.\beta$$

Si no se cumple la hipótesis $\lambda \notin L(\alpha)$, la solución podría no ser única. Incluso no regular.

Resumen conversión AF a ER

Conversión AF $\rightarrow$ ER

Para hacer la conversión de AF a ER queremos encontrar una expresión cerrada que denote $L_0 = L$. Para ello,

- Planteamos el sistema de ecuaciones

$$L_i = a_1.L_{k_1} \mid \cdots \mid a_m.L_{k_m} \mid \epsilon(L_i)$$

  Donde $\epsilon(L_i) = \lambda$ si L_i es final y $\emptyset$ sino.
- Lo resolvemos llevando las ecuaciones a una forma que nos permita aplicar el Teorema de Arden.
- Terminamos al encontrar una expresión cerrada para L_0

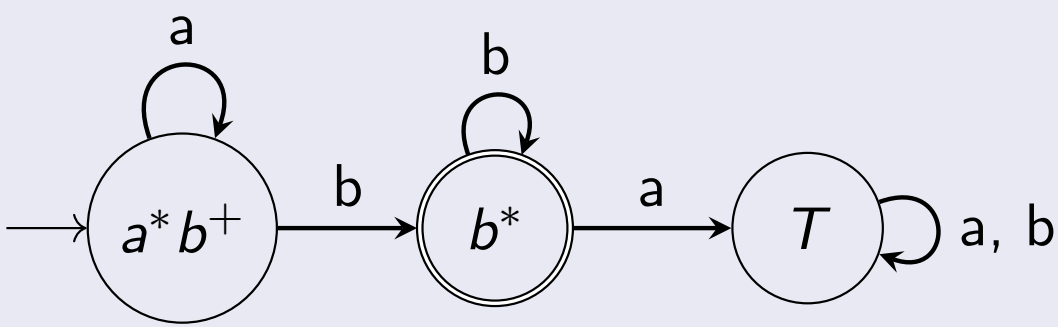
Conversión de ER a AF

ER $\rightarrow$ AF - Intuición

- Queremos formular un método para el camino inverso: dada una expresión regular, construir un autómata que acepte el mismo lenguaje.
- La idea será interpretar a las ER como *estados* que nos indican qué cadenas se aceptan partiendo de ellos. Las transiciones corresponderán a hacerlas "consumir" un símbolo del alfabeto, dando como resultado otra ER que acepta lo que resta.

Ejemplo intuitivo - Conversión a^*b^+

Si estamos parados sobre a^*b^+ y consumimos una a, seguimos aceptando a^*b^+. Pero si consumimos una b, nos resta por consumir b^*. Aplicando el razonamiento para todos los estados,



Derivada de un lenguaje

Para formalizar la noción de "consumir" un símbolo, vamos a definir las **derivadas**. Comencemos definiéndolas sobre *lenguajes*.

Derivada de un lenguaje

Sea L un lenguaje sobre un alfabeto Σ, $a \in \Sigma$,

$$\partial a(L) = \{\alpha \mid \alpha \in \Sigma^* \text{ que cumplen } a\alpha \in L\}$$

Por ejemplo, si $\Sigma = \{a, b, c\}$ y $L = \{abc, cbe, a\}$

- $\partial a(L) = \{bc, \lambda\}$
- $\partial c(L) = \{be\}$
- $\partial a(\partial c(L)) = \emptyset$

Derivada de una expresión regular

- De la misma manera que definimos la derivada de un lenguaje, podemos definirla para una expresión regular.
- La idea va a ser que si R es una ER, $\partial a(R)$ es otra ER que denota $\partial a(L(R))$ (igual que antes, pero sobre el lenguaje generado por la expresión regular).
- Dada una expresión regular R, la derivada $\partial a(R)$ **nos dice a cuál ER transiciona luego de consumir** a.
- Vamos a definirla recursivamente sobre la estructura de la expresión regular.

Estructura ERs (Repaso)

Sean $a \in \Sigma$ un símbolo, R, S expresiones regulares

Definición

e	**Nombre**	$L(e)$
$\emptyset$	Vacío	$\emptyset$
λ	Cadena vacía	$\{\lambda\} = \Lambda$
a	Símbolo	$\{a\}$
$R.S$	Concatenación	$L(R).L(S)$
$R\|S$	Unión	$L(R) \cup L(S)$
R^*	Clausura de Kleene	$(L(R))^*$
R^+	Clausura positiva	$L(R.R^*)$

Derivada de una expresión regular

Derivada de una ER según su estructura

Para $a \in \Sigma$,

$$\partial a(\emptyset) = \emptyset \qquad \partial a(b) = \begin{cases} \lambda & \text{si } a = b \\ \emptyset & \text{si } a \neq b \end{cases}$$
$$\partial a(\lambda) = \emptyset$$

Si R, S son expresiones regulares,

$$\partial a(R|S) = \partial a(R) \mid \partial a(S)$$
$$\partial a(R.S) = \partial a(R).S \mid \epsilon(R).\partial a(S)$$
$$\partial a(R^*) = \partial a(R).R^*$$

donde $\epsilon(R) = \begin{cases} \emptyset & \text{si } \lambda \notin L(R) \\ \lambda & \text{si } \lambda \in L(R) \end{cases}$

Ejercicio 3 - Derivadas de ERs

Calcular las siguientes derivadas

- $\partial a(a) = \lambda$
- $\partial a(a|b) = \partial a(a)|\partial a(b) = \lambda|\emptyset = \lambda$
- $\partial a((a|b).c) = \partial a(a|b).c \mid \epsilon(a|b).\partial a(c)$
  $= \lambda.c \mid \emptyset|\emptyset$
  $= c$
- $\partial a(a^*) = \partial a(a).a^* = \lambda.a^* = a^*$
- $\partial a(a^+) = \partial a(a.a^*)$
  $= \partial a(a).a^* \mid \epsilon(a).\partial a(a^*)$
  $= \lambda.a^* \mid \emptyset.\partial a(a^*)$
  $= a^*$
- $\partial a(b.a^{20}) = \partial a(b).a^{20} \mid \epsilon(b).\partial a(a^20)$
  $= \emptyset.a^{20} \mid \emptyset.\partial a(a^20)$
  $= \emptyset$

¿Por qué las derivadas nos dan las transiciones?

Recordemos las igualdades que habíamos planteado antes. Para un estado q_i y $\Sigma = \{a_1, \ldots, a_m\}$, teníamos ecuaciones de la forma

$$\delta(q_i, a_j) = q_{k_j}$$

que nos permitían expresar el lenguaje L_i como

$$L_i = a_1.L_{k_1} \mid a_2.L_{k_2} \mid \cdots \mid a_m.L_{k_m} \mid \epsilon(L_i) \tag{1}$$

es decir, que está formado por las cadenas que empiezan con a_j y siguen con alguna cadena de L_{k_j}. Esto lo podemos expresar mediante una derivada

$$\partial a_j(L_i) = L_{k_j}$$

$$\partial a(L) = \{\alpha \mid a\alpha \in L\} \text{ donde } a \in \Sigma, \alpha \in \Sigma^*$$

Reemplazando en 1,

$$L_i = a_1.\partial a_1(L_i) \mid a_2.\partial a_2(L_i) \mid \cdots \mid a_m.\partial a_m(L_i) \mid \epsilon(L_i)$$

¿Por qué las derivadas nos dan las transiciones?

Como conocemos la expresión regular R que define el lenguaje, también tenemos la que define $L_0 = L(R)$. Luego, como

$$\partial a_j(L_0) = L_{k_j},$$

entonces

$$\begin{aligned} L_0 &= a_1.L_{k_1} \mid a_2.L_{k_2} \mid \cdots \mid a_m.L_{k_m} \mid \epsilon(L_0) \\ &= a_1.\partial a_1(L_0) \mid a_2.\partial a_2(L_0) \mid \cdots \mid a_m.\partial a_m(L_0) \mid \epsilon(L_0). \end{aligned}$$

De esa forma, si derivamos e con respecto a cada símbolo $a_j \in \Sigma$, obtendremos la expresión regular que define el lenguaje aceptado por el autómata partiendo del estado $\delta(q_0, a_j)$ (que puede ser el mismo L_0 u otro lenguaje L_k).
Repitiendo este proceso, iremos obteniendo una cantidad finita de expresiones regulares, que iremos tomando como estados del autómata.

Ejercicio 4

Ejercicio 4 - a^*b^+

Dar un autómata finito que reconozca el lenguaje $L_4 = L(a^*b^+)$ usando el método de las derivadas.

Veamos como obtendríamos más metódicamente lo que antes hicimos a ojo de forma intuitiva

- $\partial a(L_0) = \partial a(a^*b^+)$
  $= \partial a(a^*).b^+ \mid \epsilon(a^*).\partial a(b^+)$
  $= a^*.b^+ \mid \lambda.\emptyset = a^*b^+ = L_0$
- $\partial b(L_0) = \partial b(a^*b^+)$
  $= \partial b(a^*).b^+ \mid \epsilon(a^*).\partial b(b^+)$
  $= \emptyset.b^+ \mid \lambda.b^*$
  $= b^* = \boldsymbol{L_1}$
- $\partial a(L_1) = \partial a(b^*) = \emptyset = \boldsymbol{L_T}$
- $\partial b(L_1) = \partial b(b^*) = b^* = L_1$
- $\partial a(L_T) = \partial b(L_T) = \emptyset = L_T$

Armado del autómata

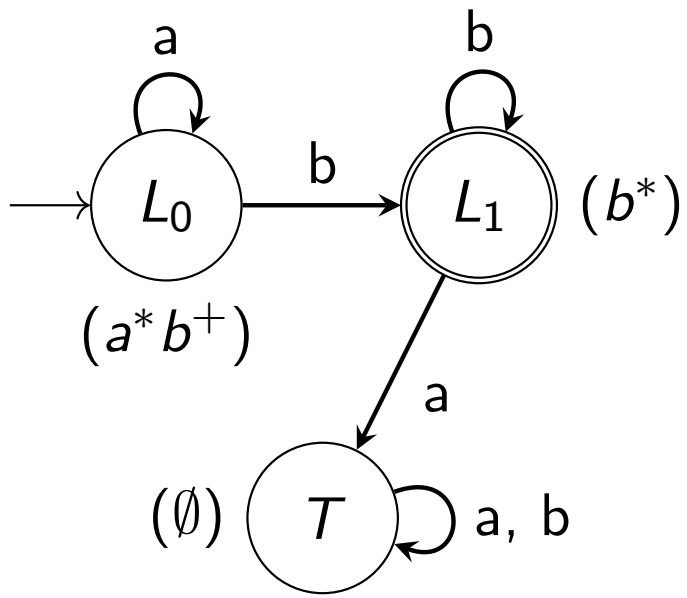
Una vez que tenemos las derivadas calculadas, podemos construir el autómata correspondiente:

- Los estados son las expresiones regulares L_i,
- $\delta(L_i, a) = L_j$ si $\partial a(L_i) = L_j$,
- Son estados finales los L_i tales que $\lambda \in L_i$.

Para $L_0 = a^* b^+$ tenemos

L_i	$=$	∂a	∂b
L_0	$a^* b^+$	L_0	L_1
L_1	b^*	L_T	L_1
L_T	$\emptyset$	L_T	L_T

L_1 es el único final pues $\lambda \in L_1$.



Resumen conversión ER $\rightarrow$ AF

Conversión ER $\rightarrow$ AF

Para hacer la conversión de una ER R a AF,

- Tomamos como el estado inicial a R
- Para cada estado nuevo (que será una ER), lo hacemos transicionar por todos los símbolos del alfabeto usando las derivadas.
- Tomamos como estados finales aquellas ER que acepten λ ($\lambda \in L(Q)$)

Ejercicio integrador

Ejercicio integrador

Ejercicio integrador

En caso de que exista, dar una expresión regular que denote $L((12|2)^*(1|\lambda))^c$. Si no existe, probarlo.

Observaciones:

- ¿Es regular? ¡Sí! Ya vimos que los lenguajes regulares son cerrados por complemento (como el lenguaje es regular, existe un autómata y sabemos conseguir su complemento algorítmicamente).
- ¿Cómo conseguimos el complemento de una expresión regular? Más fácil es complementar un autómata.

La estrategia va a ser la siguiente:

1. Convertir la ER $(12|2)^*(1|\lambda)$ a AFD.
2. Complementarlo.
3. Convertir el resultado a ER.

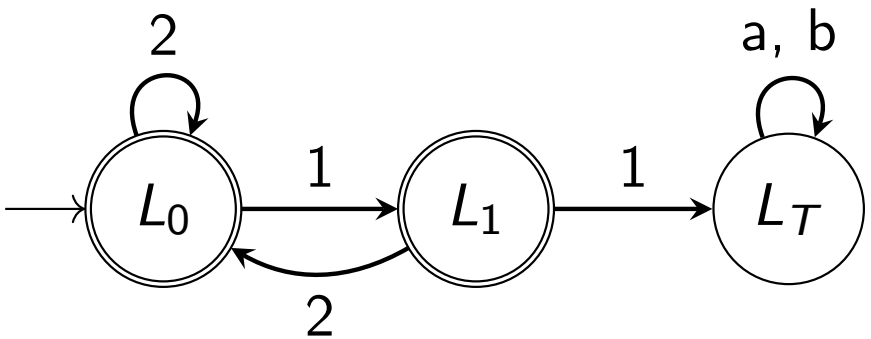
1 - Conversión a AFD

Tenemos $L_0 = (12|2)^*(1|\lambda)$

- $\partial 1(L_0) = \partial 1((12|2)^*(1|\lambda))$
  $= \partial 1((12|2)^*).(1|\lambda) \mid \epsilon((12|2)^*).\partial 1((1|\lambda))$
  $= \partial 1(12|2)(12|2)^*(1|\lambda) \mid \lambda.\partial 1((1|\lambda))$
  $= (\partial 1(12)|\partial 1(2)).(12|2)^*(1|\lambda) \mid \partial 1(1) \mid \partial 1(\lambda)$
  $= 2(12|2)^*(1|\lambda)|\lambda = \boldsymbol{L_1}$
- $\partial 2(L_0) = \partial 2((12|2)^*(1|\lambda))$
  $= \partial 2((12|2)^*)(1|\lambda) \mid \epsilon((12|2)^*).\partial 2((1|\lambda))$
  $= \partial 2((12|2))(12|2)^*(1|\lambda) \mid \emptyset$
  $= \lambda(12|2)^*(1|\lambda)$
  $= (12|2)^*(1|\lambda) = L_0$
- $\partial 1(L_1) = \partial 1(2(12|2)^*(1|\lambda)|\lambda) = \emptyset = \boldsymbol{L_T}$
- $\partial 2(L_1) = \partial 2(2(12|2)^*(1|\lambda)|\lambda) = (12|2)^*(1|\lambda) = L_0$
- $\partial 1(L_T) = \partial 2(L_T) = \emptyset = L_T$

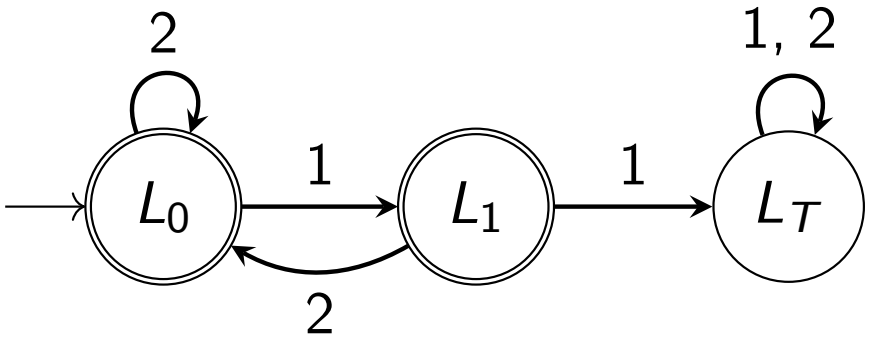
L_i	$=$	$\partial 1$	$\partial 2$	$\lambda \in ?$
L_0	$(12\|2)^*(1\|\lambda)$	L_1	L_0	Si
L_1	$2(12\|2)^*(1\|\lambda)\|\lambda$	L_T	L_0	Si
L_T	$\emptyset$	L_T	L_T	No

$$A = \langle \{L_0, L_1, L_T\}, \{1, 2\}, \delta, 0, \{L_0, L_1\} \rangle$$



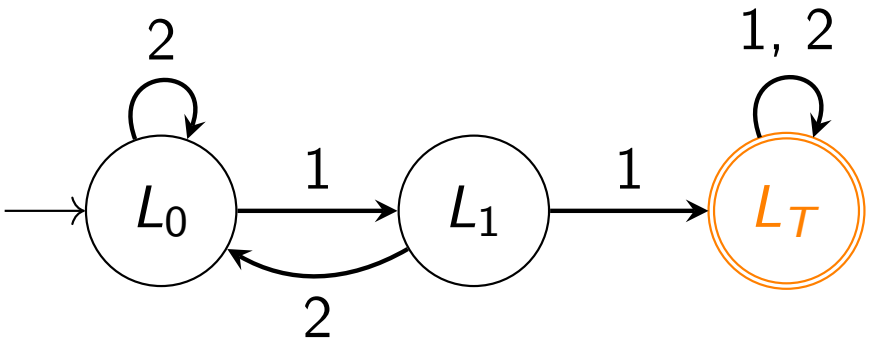
2 - Complemento del AFD

$$A = \langle \{L_0, L_1, L_T\}, \{1, 2\}, \delta, 0, \{L_0, L_1\} \rangle$$



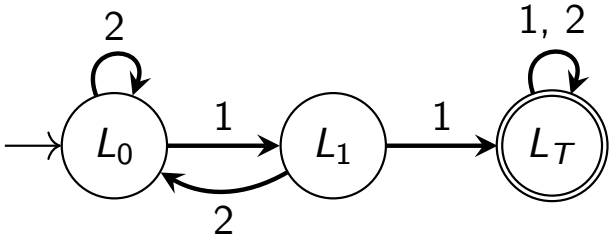
Para complementar un AFD completo, hacemos $F' = Q \setminus F$.

$$A^c = \langle \{L_0, L_1, L_T\}, \{1, 2\}, \delta, 0, \{L_T\} \rangle$$



3 - Conversión a ER

$A^c = \langle \{L_0, L_1, L_T\}, \{1, 2\}, \delta, 0, \{L_T\} \rangle$



$$L_0 = 1.L_1 \mid 2.L_0 \mid \emptyset$$
$$L_1 = 1.L_T \mid 2.L_0 \mid \emptyset$$
$$L_T = 1.L_T \mid 2.L_T \mid \lambda$$

Si $R = \alpha.R \mid \beta$ y $\lambda \notin L(\alpha)$, entonces $R = \alpha^*.\beta$

- $L_T = (1|2).L_T \mid \lambda = (1|2)^*.\lambda = (1|2)^* \quad (\lambda \notin (1|2))$
- $L_1 = 1.L_T \mid 2.L_0 = 1(1|2)^* \mid 2.L_0$
- $L_0 = 1.L_1 \mid 2.L_0$
  $= 1(1(1|2)^* \mid 2.L_0) \mid 2.L_0$
  $= 11(1|2)^* \mid 12.L_0 \mid 2.L_0$
  $= (12|2).L_0 \mid 11(1|2)^* \quad (\lambda \notin (12|2))$
  $= (12|2)^*11(1|2)^*$

Terminando

Ej 2 - 2c2021 r1p

En caso de que exista, dar una expresión regular que denote $L((12|2)^*(1|\lambda))^c$. Si no existe, probarlo.

Estrategia:

1. Convertir la ER $(12|2)^*(1|\lambda)$ a AFD.
2. Complementarlo.
3. Convertir el resultado a ER.

Respuesta

Existe una expresión regular que denota $L((12|2)^*(1|\lambda))^c$ y es

$$(12|2)^*11(1|2)^*$$

Conclusiones

Vimos,

- Expresiones regulares, ejemplos y ejercicios
- Pasaje de AF a ER (Método de ecuaciones + Arden)
- Pasaje de ER a AF (Método de derivadas)
- Un ejercicio integrador

Ya pueden hacer toda la Práctica 4 (Expresiones Regulares).

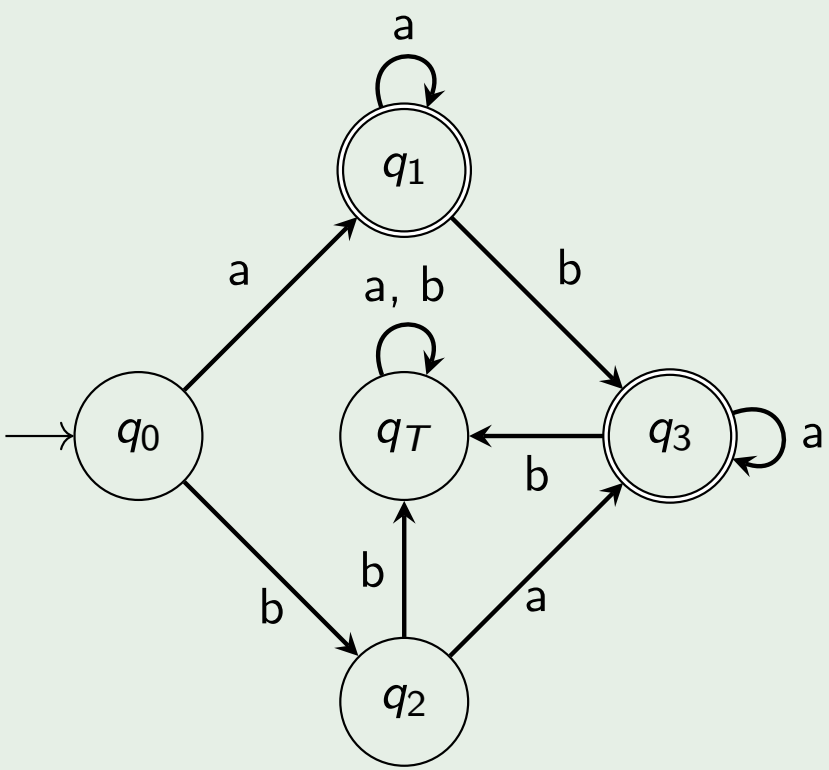
Anexo: Ejemplo extra

Ejemplo extra

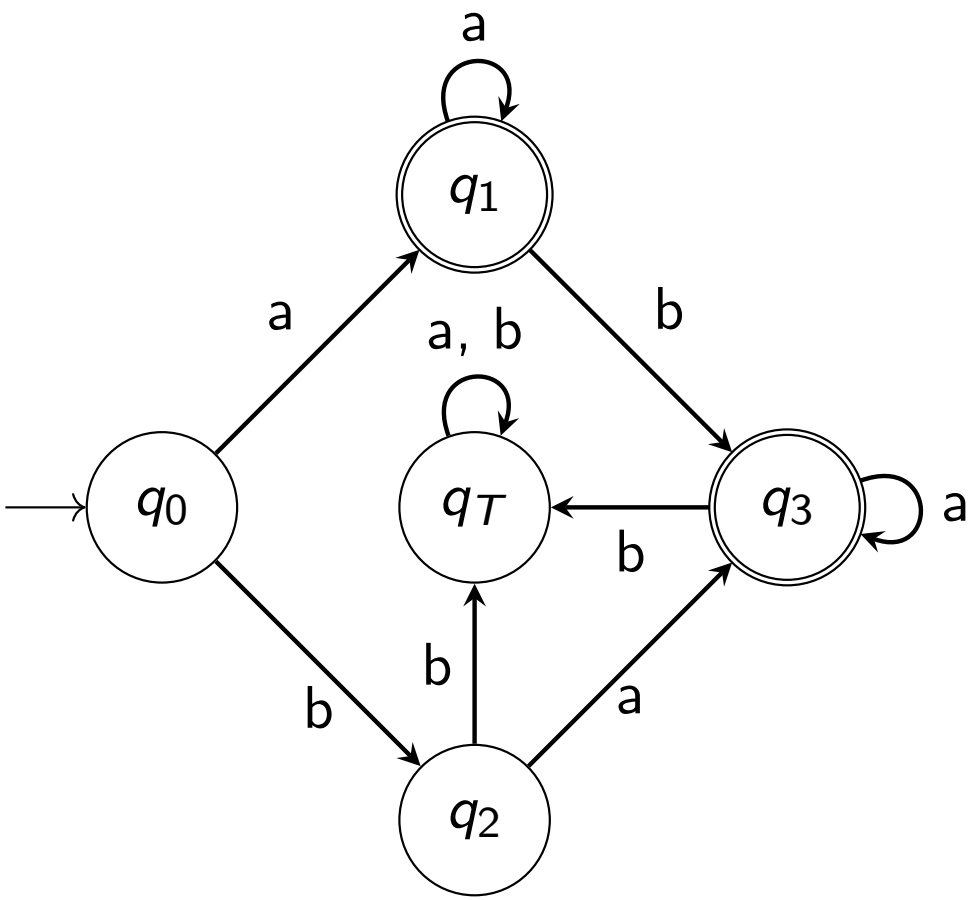
Ejercicio extra - Ida y vuelta

Convertir el siguiente autómata a una expresión regular mediante el método de las ecuaciones y de vuelta a autómata mediante el método de las derivadas.

$$A = \langle \{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_1, q_3\} \rangle$$

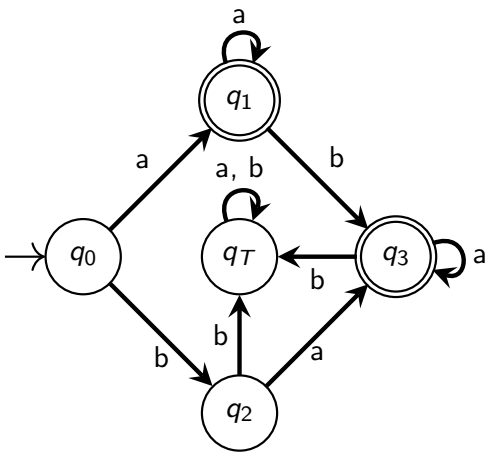


AF $\rightarrow$ ER: Ecuaciones



$$
\begin{aligned}
L_0 &= a.L_1 \mid b.L_2 \mid \emptyset \\
L_1 &= a.L_1 \mid b.L_3 \mid \lambda \\
L_2 &= a.L_3 \mid b.L_T \mid \emptyset \\
L_3 &= a.L_3 \mid b.L_T \mid \lambda \\
L_T &= a.L_T \mid b.L_T \mid \emptyset
\end{aligned}
$$

AF $\rightarrow$ ER: Resolución



- $L_T = a.L_T|b.L_T|\emptyset = (a|b).L_T|\emptyset$
  y como $\lambda \notin L(a|b) = \{a, b\}$, entonces $L_T = (a|b)^*.\emptyset = \emptyset$
- $L_3 = a.L_3|b.L_T|\lambda = a.L_3|b.\emptyset|\lambda = a.L_3|\lambda$
  y como $\lambda \notin \{a\}$, entonces $L_3 = a^*.\lambda = a^*$
- $L_2 = a.L_3|b.L_T|\emptyset = a.a^*|\emptyset = a^+$
- $L_1 = a.L_1|b.L_3|\lambda = a.L_1|(b.a^*|\lambda)$
  y como $\lambda \notin \{a\}$, entonces $L_1 = a^*.(b.a^* \mid \lambda)$
- $L_0 = a.L_1|b.L_2|\emptyset = a.a^*.(b.a^* \mid \lambda) \mid b.a^+$
  $= a^+(ba^* \mid \lambda) \mid ba^+$

ER $\rightarrow$ AF

Vamos con la vuelta, de la ER $a^+(ba^*|\lambda)|ba^+$ a AF.
Partimos de $L_0 = a^+(ba^+|\lambda)|ba^+$

- $\partial a(L_0) = \partial a(a^+(ba^*|\lambda)|ba^+)$
  $= \partial a(a^+(ba^*|\lambda)) \mid \partial a(ba^+)$
  $= \partial a(a^+)(ba^*|\lambda) \mid \epsilon(a^+)\partial a(ba^*|\lambda) \mid \emptyset$
  $= a^*(ba^*|\lambda) \mid \emptyset.\partial a(ba^*|\lambda)$
  $= a^*(ba^*|\lambda) = \boldsymbol{L_1}$
- $\partial b(L_0) = \partial b(a^+(ba^*|\lambda)|ba^+)$
  $= \partial b(a^+(ba^*|\lambda)) \mid \partial b(ba^+)$
  $= \emptyset \mid \partial b(b).a^+|\epsilon(b).\partial b(a^+)$
  $= \lambda.a^+ \mid \emptyset.\partial b(a^+)$
  $= a^+ = \boldsymbol{L_2}$

ER $\rightarrow$ AF

- $\partial a(L_1) = \partial a(a^*(ba^*|\lambda))$
  $= \partial a(a^*).(ba^*|\lambda) \mid \epsilon(a^*).\partial a(ba^*|\lambda)$
  $= a^*.(ba^*|\lambda) \mid \lambda.(\partial a(ba^*)|\partial a(\lambda))$
  $= a^*(ba^*|\lambda) \mid (\emptyset|\emptyset)$
  $= a^*(ba^*|\lambda) = L_1$
- $\partial b(L_1) = \partial b(a^*(ba^*|\lambda))$
  $= \partial b(a^*).(ba^*|\lambda) \mid \epsilon(a^*).\partial b(ba^*|\lambda)$
  $= \emptyset \mid \lambda.(\partial b(b).a^*|\epsilon(b).\partial b(a^*) \mid \partial b(\lambda))$
  $= (\lambda.a^*|\emptyset.\partial b(a^*) \mid \emptyset)$
  $= a^* = \boldsymbol{L_3}$
- $\partial a(L_2) = \partial a(a^+) = a^* = L_3$
- $\partial b(L_2) = \partial b(a^+) = \emptyset = \boldsymbol{L_T}$
- $\partial a(L_3) = \partial a(a^*) = a^* = L_3$
- $\partial b(L_3) = \partial b(a^*) = \emptyset = L_T$
- $\partial a(L_T) = \partial b(L_T) = \emptyset = L_T$

ER $\rightarrow$ AF

Construyamos el autómata

L_i	$=$	∂a	∂b	$\lambda \in ?$
L_0	$a^+(ba^*\|\lambda)\|ba^+$	L_1	L_2	No
L_1	$a^*(ba^*\|\lambda)$	L_1	L_3	Si
L_2	a^+	L_3	L_T	No
L_3	a^*	L_3	L_T	Si
L_T	$\emptyset$	L_T	L_T	No