Expresiones regulares y conversiones

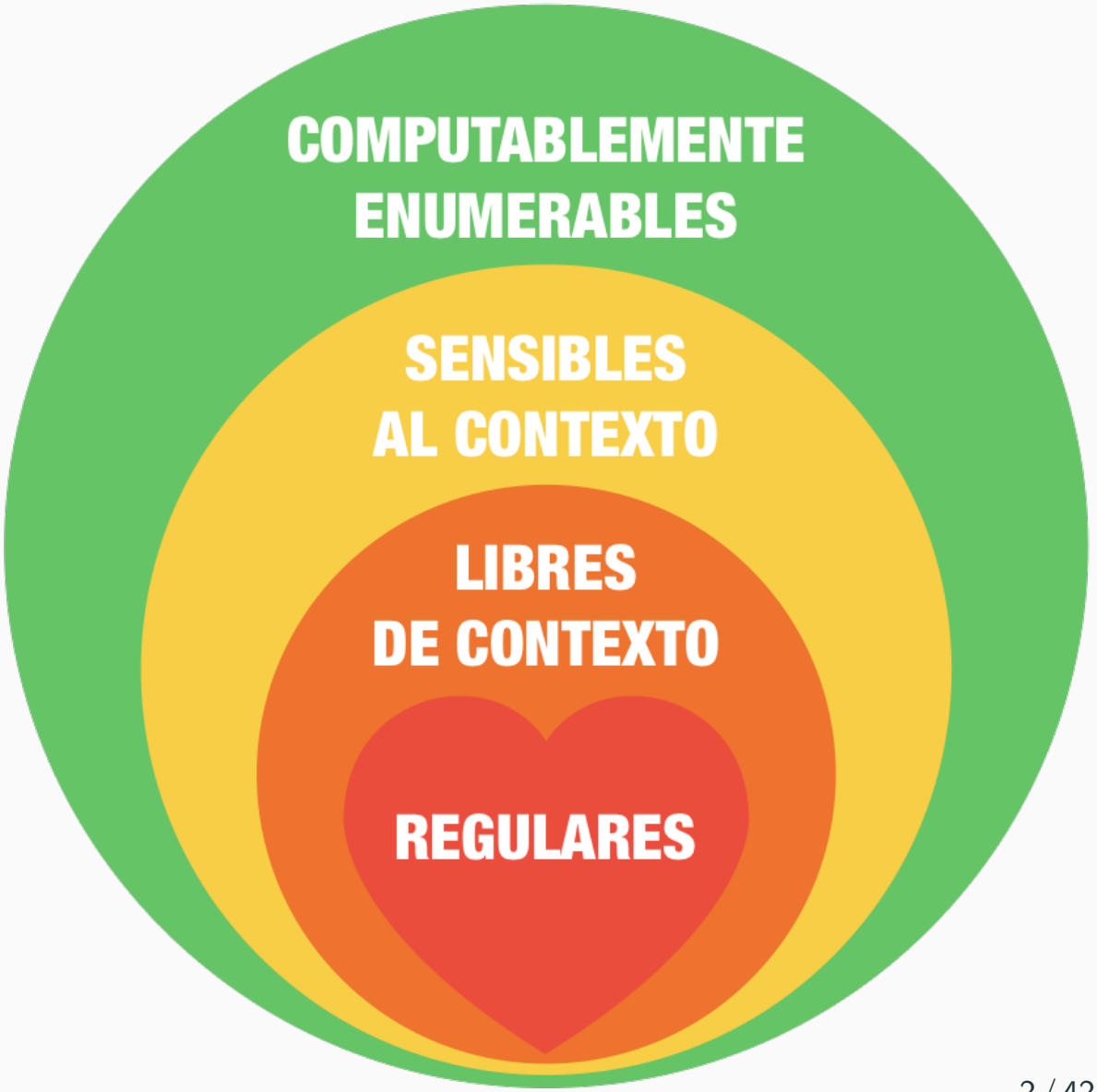
Lenguajes Formales, Autómatas y Computabilidad

DC-UBA

23 de abril 2025

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 **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:

$$0 \;\Big|\; (- \mid \lambda)(1|\cdots|9)(0|1|\cdots|9)^*$$

¿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:

- $(R.S)$ 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	$L(R.R^*)$

Notación

Por simplicidad usaremos directamente E para referirnos al lenguaje denotado por E, en vez de $L(E)$.

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 la concatenación son asociativas:

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

Ejercicio 1

c. Cadenas con 0s y 1s alternados:

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

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

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

e. Cadenas con cantidad par de 0s:

$$(01^*0|1)^* \text{ equivalentemente } 1^*(01^*01^*)^*$$

Aclaraciones

- 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 no es tan sencillo con ERs como con AFs.
- También algunas operaciones como complemento, iniciales/finales/subcadenas, reversa, etc. de un lenguaje son mas fáciles de hacer sobre AFs que sobre ERs.
- Recordemos que **aceptar** (o denotar) un lenguaje L se refiere *exactamente* a L, y no alguno incluido o otro que lo incluya.

Otras propiedades

- **Conmutatividad de la unión**

$$R|S = S|R$$

(notar que la concatenación **no** es conmutativa)

- **Distributividad de la concatenación respecto de la unión**

$$R(S|T) = RS|RT$$
$$(R|S)T = RT|ST$$

- **Elemento neutro para la concatenación**:

$$R.\lambda = \lambda.R = R$$

- **Elemento absorbente para la concatenación**:

$$R.\emptyset = \emptyset.R = \emptyset$$

- **Elemento neutro para la unión**:

$$R|\emptyset = \emptyset|R = R$$

Extensiones que suelen aparecer

Muchos lenguajes de programación implementan expresiones regulares y suelen agregar otras características además 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.

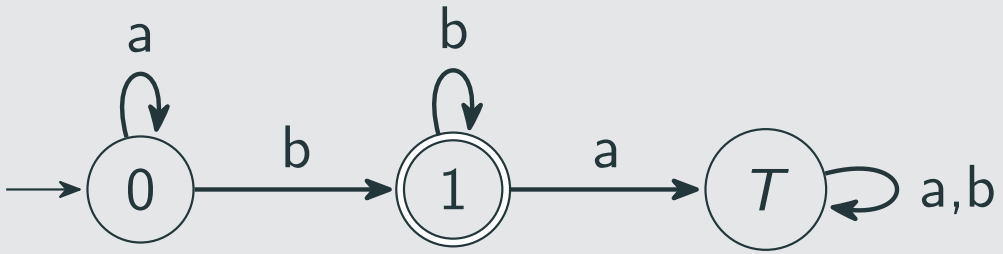
Hay clases de carácteres predefinidas: (en POSIX) *alnum*, *alpha*, *digit*, *space*, ...

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 $AFND - \lambda$, donde

- Q es el conjunto de estados
- Σ es el alfabeto
- $\delta : Q \times \Sigma \cup \{\lambda\} \to \mathcal{P}(Q)$ 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 $Q \times \Sigma^*$):

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

$$(q_i, \alpha) \vdash (q_j, \alpha) \iff q_j \in \delta(q_i, \lambda).$$

Y el lenguaje aceptado por el autómata se define como

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

Autómatas finitos generalizados

Sea $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un autómata finito generalizado, donde

- Q es el conjunto de estados
- Σ es el alfabeto
- $\delta : Q \times E \to \mathcal{P}(Q)$ es la función de transición, con E una expresión regular sobre Σ
- $q_0 \in Q$ 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 $Q \times \Sigma^*$):

$$(q_i, \omega\alpha) \vdash (q_j, \alpha) \iff q_j \in \delta(q_i, E) \text{ y } \omega \in E.$$

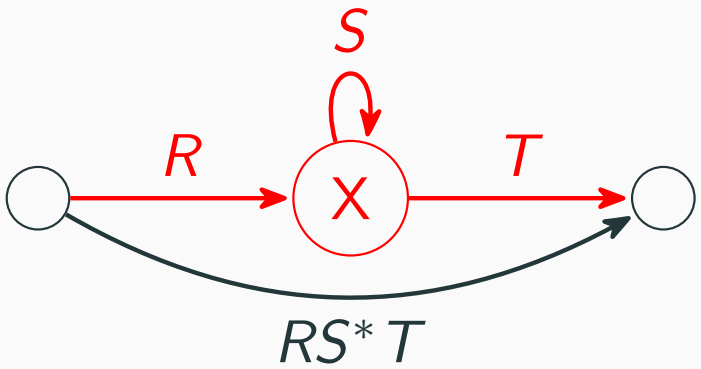
Y el lenguaje aceptado por el autómata se define como

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

Intuición

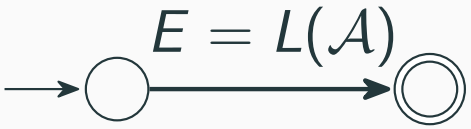
- Queremos llegar a un autómata generalizado con solo dos estados: uno inicial y uno final. Con solo una transición y que vaya del inicial al final.

- Para eso vamos a necesitar eliminar estados pero manteniendo el lenguaje aceptado. Eliminemos el estado X:

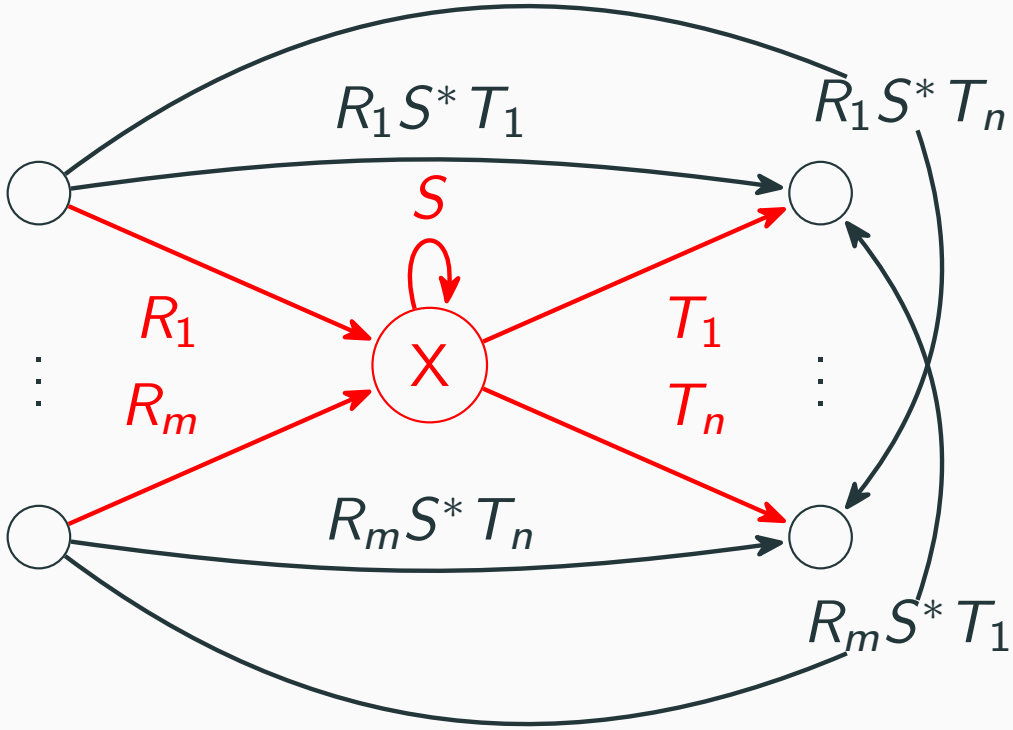


Intuición

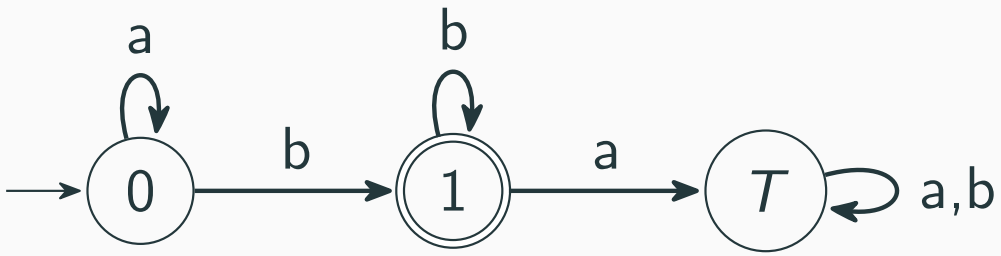
- Queremos llegar a un autómata generalizado con solo dos estados: uno inicial y uno final. Con solo una transición y que vaya del inicial al final.



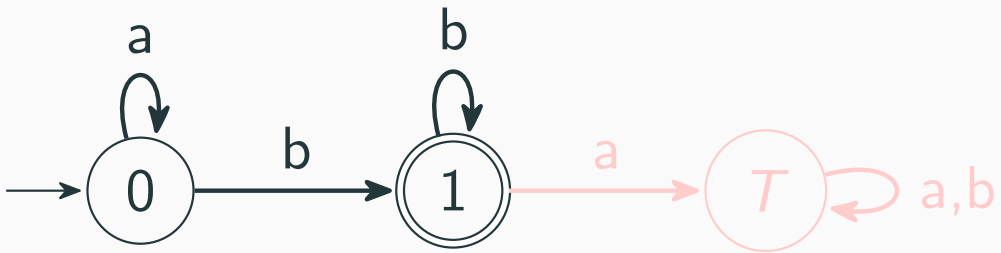
- Para eso vamos a necesitar eliminar estados pero manteniendo el lenguaje aceptado. Eliminemos el estado X:

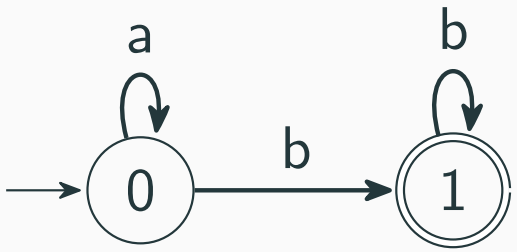


Volvamos al autómata



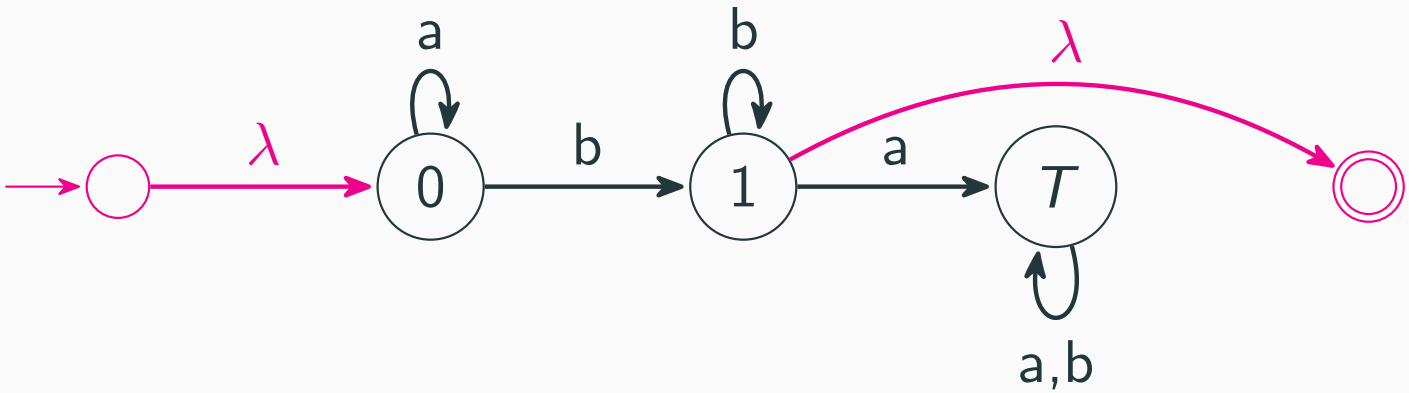
Eliminamos el estado T





¡No quedo con una sola transición!

Automata normalizado

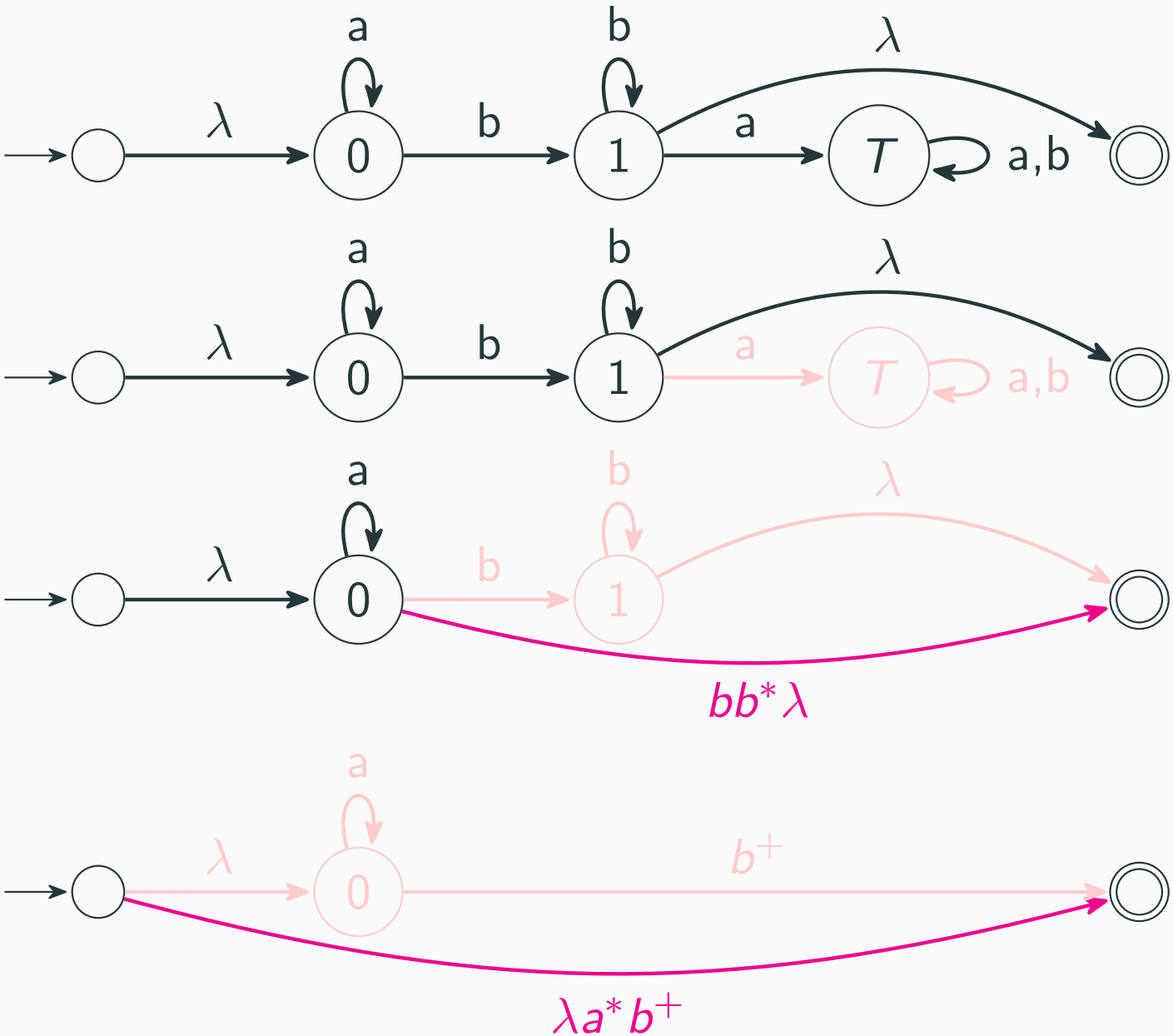


Importante

Necesitamos empezar con un autómata que tenga un único estado final sin transiciones salientes. Además, el estado inicial no debe tener transiciones entrantes (ni ser final).
Para eso podemos normalizar el autómata antes de empezar con la eliminación:

- Agregando un nuevo estado inicial solamente con una transición λ al estado inicial original.
- Agregando un nuevo estado final con transiciones λ desde los estados finales originales, los cuales dejan de ser finales.

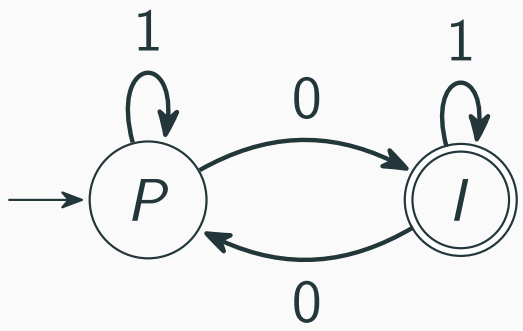
Eliminando estados



Ejercicio 3

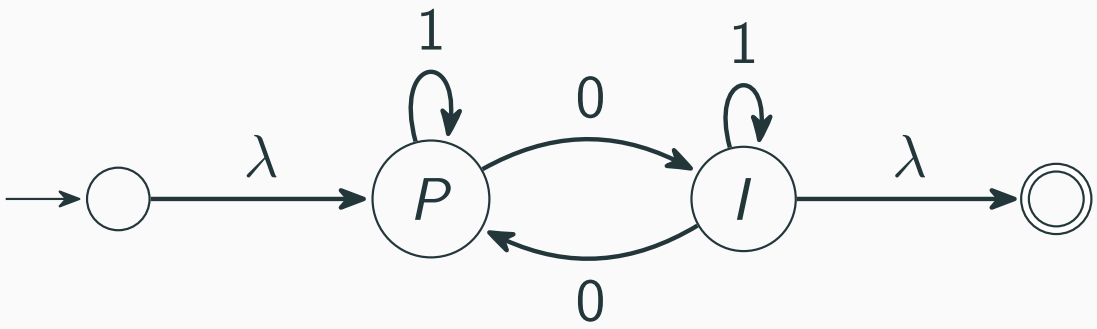
Veamos un ejemplo un poco más complicado. ¿Qué lenguaje reconoce el siguiente autómata?

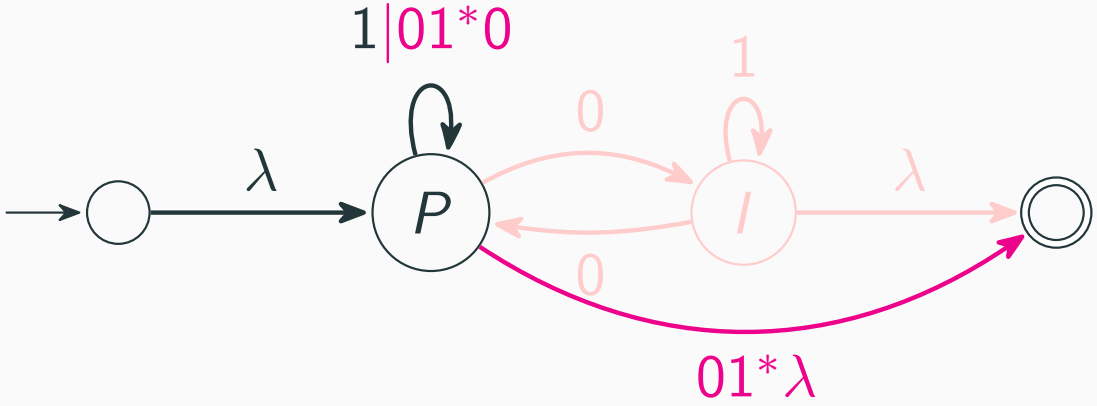
$$A_3 = \langle \{P, I\}, \{0, 1\}, \delta, P, \{I\} \rangle$$

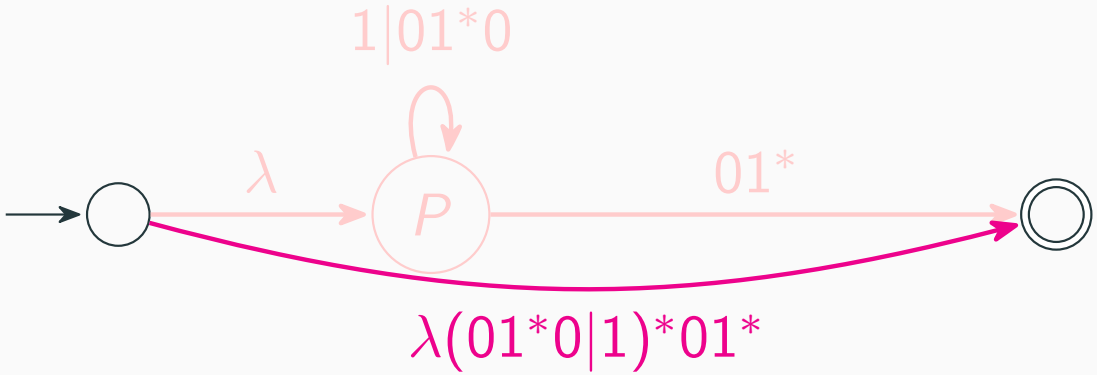


Cantidad impar de ceros. Ya no es trivial dar una ER. Usemos el método que vimos recién para convertirlo de forma mecánica.

Resolución

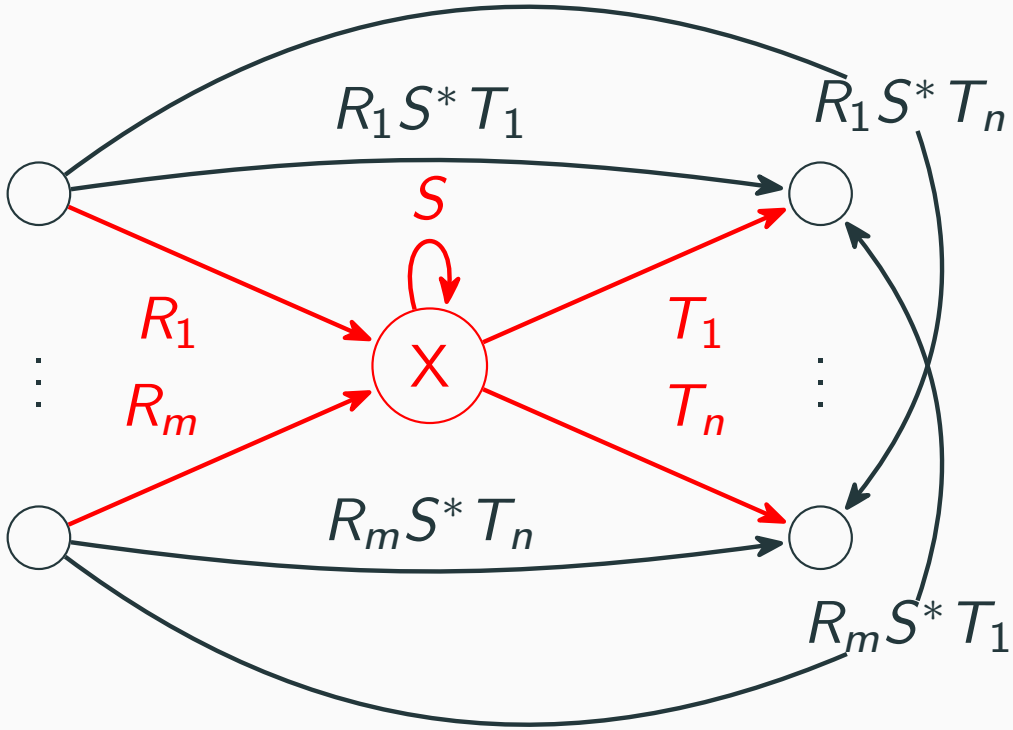






Resumen conversión AF a ER

- Normalizamos el autómata.
- Eliminamos todos los estados salvo el final y el inicial, aplicando la siguiente equivalencia.



En rojo las transiciones originales y el estado a eliminar.
En negro las transiciones que debemos agregar para mantener los caminos que pasaban por X.

- Terminamos con una única transición con la ER equivalente.

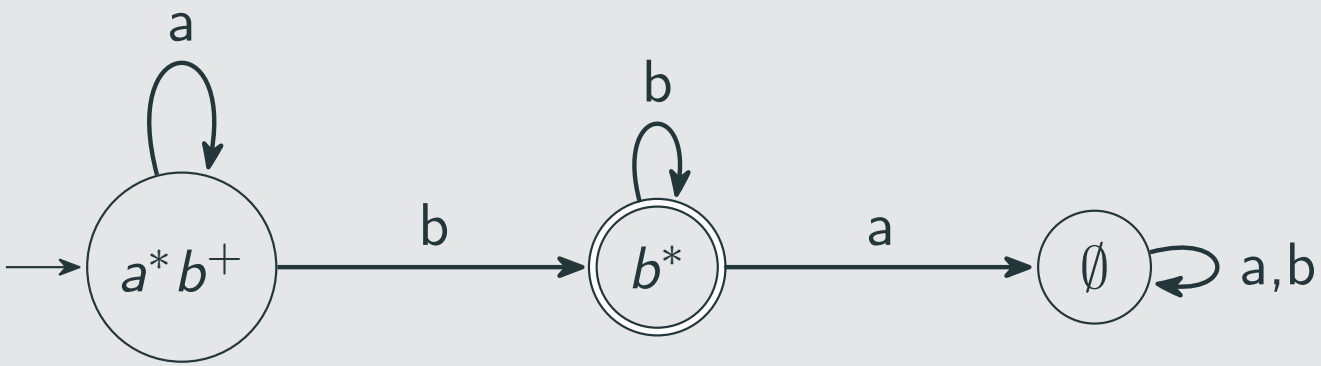
Conversión de ER a AF

ER $\rightarrow$ AF - Intuición

- Queremos dar 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, obtenemos



Cociente de un lenguaje

Para formalizar la noción de *consumir* un símbolo, vamos a definir el **cociente** de un *lenguaje*.

Cociente de un lenguaje

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

$$a^{-1}(L) = \{\alpha \mid \alpha \in \Sigma^* \text{ que cumplen } a\alpha \in L\}$$

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

- $a^{-1}(L) = \{bc, \lambda\}$
- $c^{-1}(L) = \{be\}$
- $a^{-1}(c^{-1}(L)) = \emptyset$

Derivada de una expresión regular

- De la misma manera que definimos el cociente para un lenguaje, podemos definir las derivadas para una expresión regular.
- La idea va a ser que si R es una ER, $\partial a(R)$ es otra ER que denota $a^{-1}L(R)$ (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 y lenguaje denotado (Repaso)

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	$L(R.R^*)$

Notación

Por simplicidad usaremos directamente E para referirnos al lenguaje denotado por E, en vez de $L(E)$.

Derivada de una expresión regular

Sea E una expresión regular sobre Σ y un símbolo $a \in \Sigma$. La derivada de E respecto a a ($\partial a(E)$), también es una expresión regular sobre Σ y se define recursivamente usando las siguientes formulas:

$$\partial a(\emptyset) = \emptyset$$

$$\partial a(\lambda) = \emptyset$$

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

Si R, S son expresiones regulares,

$$\partial a(R|S) = \partial a(R) \mid \partial a(S)$$

$$\partial a(R^*) = \partial a(R).R^*$$

$$\partial a(R.S) = \begin{cases} \partial a(R).S & \text{si } \lambda \notin R \\ \partial a(R).S \mid \partial a(S) & \text{si } \lambda \in R \end{cases}$$

Clausura positiva

No es necesario definir la $\partial a(R^+)$ ya que $R^+ = R.R^*$ por lo tanto $\partial a(R^+) = \partial a(R.R^*) = \partial a(R).R^*$

Ejercicio 3 - Derivadas de ERs

Calcular las siguientes derivadas

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

Ejercicio 4

Ejercicio 4 - a^*b^+

Dar un autómata finito que reconozca el lenguaje $E_0 = a^*b^+$ usando el método de las derivadas.

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

- $\partial a(E_0) = \partial a(a^*b^+)$
  $= \partial a(a^*).b^+ \mid \partial a(b^+)$
  $= a^*.b^+ \mid \emptyset = a^*b^+ = E_0$
- $\partial b(E_0) = \partial b(a^*b^+)$
  $= \partial b(a^*).b^+ \mid \partial b(b^+)$
  $= \emptyset.b^+ \mid b^*$
  $= b^* = \boldsymbol{E_1}$
- $\partial a(E_1) = \partial a(b^*) = \emptyset = \boldsymbol{E_T}$
- $\partial b(E_1) = \partial b(b^*) = b^* = E_1$
- $\partial a(E_T) = \partial b(E_T) = \emptyset = E_T$

Armado del autómata

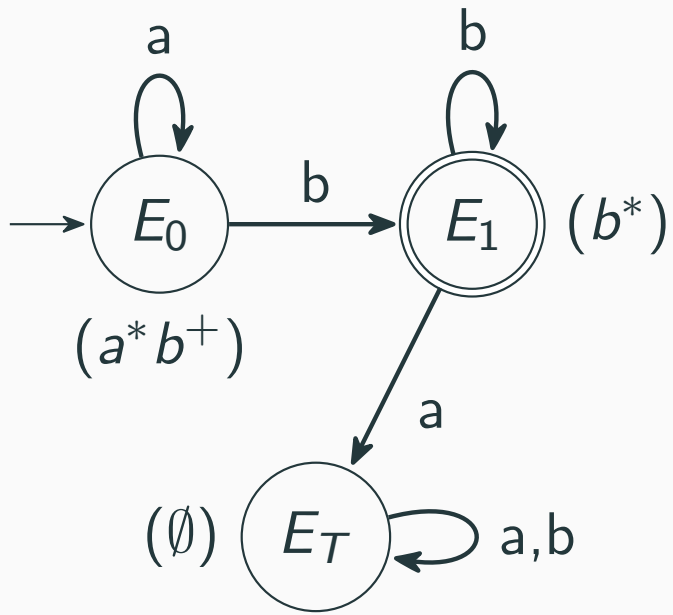
Una vez que tenemos las derivadas calculadas, podemos construir el autómata correspondiente:

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

Para $E_0 = a^* b^+$ tenemos

E_i	$=$	∂a	∂b
E_0	$a^* b^+$	E_0	E_1
E_1	b^*	E_T	E_1
E_T	$\emptyset$	E_T	E_T

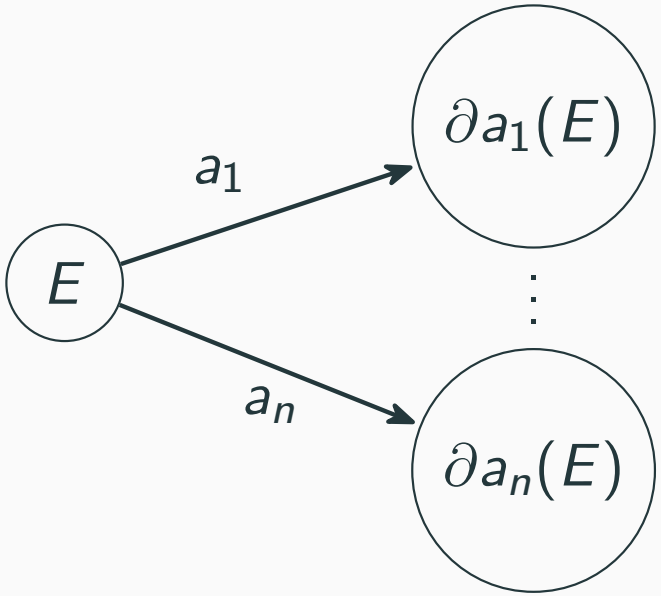
E_1 es el único final pues λ solo pertenece a E_1.



Resumen conversión ER $\rightarrow$ AF

Para hacer la conversión de una expresion regular E_0 a AF,

- Agregamos el estado inicial E_0.
- Para cada estado que agregamos (que se corrsponderá con una ER), lo hacemos transicionar para cada símbolos del alfabeto al estado que corresponda su derivada para ese simbolo.



- Tomamos como estados finales los estados cuya ER tenga a λ

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

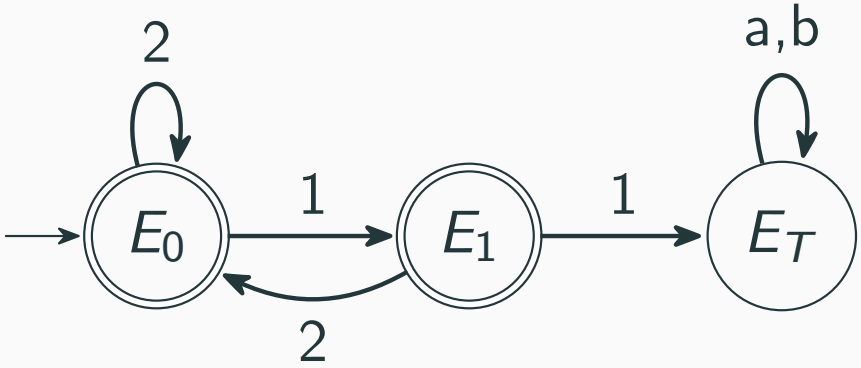
Tenemos $E_0 = (12|2)^*(1|\lambda)$

- $\partial 1(E_0) = \partial 1((12|2)^*(1|\lambda))$
  $= \partial 1((12|2)^*).(1|\lambda) \mid \partial 1((1|\lambda))$
  $= \partial 1(12|2)(12|2)^*(1|\lambda) \mid \partial 1((1|\lambda))$
  $= (\partial 1(12)|\partial 1(2)).(12|2)^*(1|\lambda) \mid \partial 1(1) \mid \partial 1(\lambda)$
  $= 2(12|2)^*(1|\lambda)|\lambda = \boldsymbol{E_1}$
- $\partial 2(E_0) = \partial 2((12|2)^*(1|\lambda))$
  $= \partial 2((12|2)^*)(1|\lambda) \mid \partial 2((1|\lambda))$
  $= \partial 2((12|2))(12|2)^*(1|\lambda) \mid \emptyset$
  $= \lambda(12|2)^*(1|\lambda)$
  $= (12|2)^*(1|\lambda) = E_0$
- $\partial 1(E_1) = \partial 1(2(12|2)^*(1|\lambda)|\lambda) = \emptyset = \boldsymbol{E_T}$
- $\partial 2(E_1) = \partial 2(2(12|2)^*(1|\lambda)|\lambda) = (12|2)^*(1|\lambda) = E_0$
- $\partial 1(E_T) = \partial 2(E_T) = \emptyset = E_T$

1 - Construcción del AFD

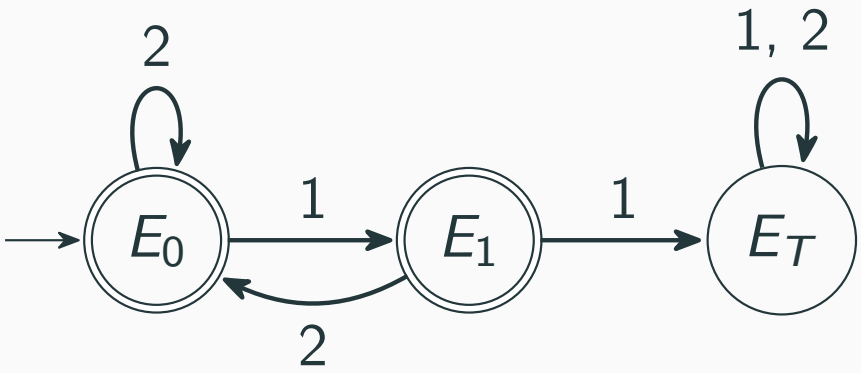
E_i	$=$	$\partial 1$	$\partial 2$	$\lambda \in ?$
E_0	$(12\|2)^*(1\|\lambda)$	E_1	E_0	Si
E_1	$2(12\|2)^*(1\|\lambda)\|\lambda$	E_T	E_0	Si
E_T	$\emptyset$	E_T	E_T	No

$$A = \langle \{E_0, E_1, E_T\}, \{1, 2\}, \delta, 0, \{E_0, E_1\} \rangle$$



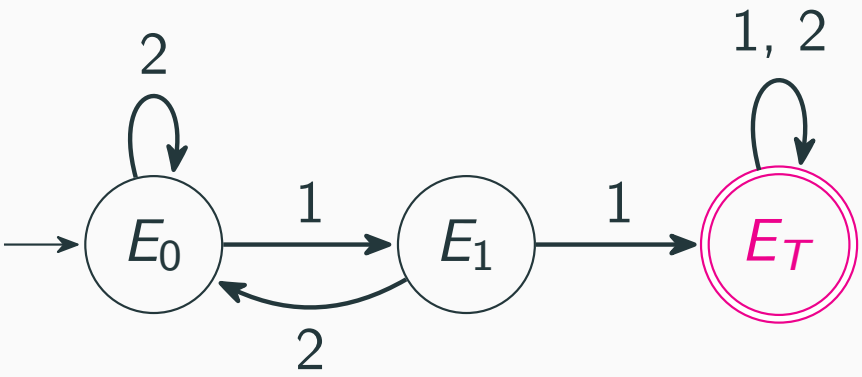
2 - Complemento del AFD

$$A = \langle \{E_0, E_1, E_T\}, \{1, 2\}, \delta, 0, \{E_0, E_1\} \rangle$$

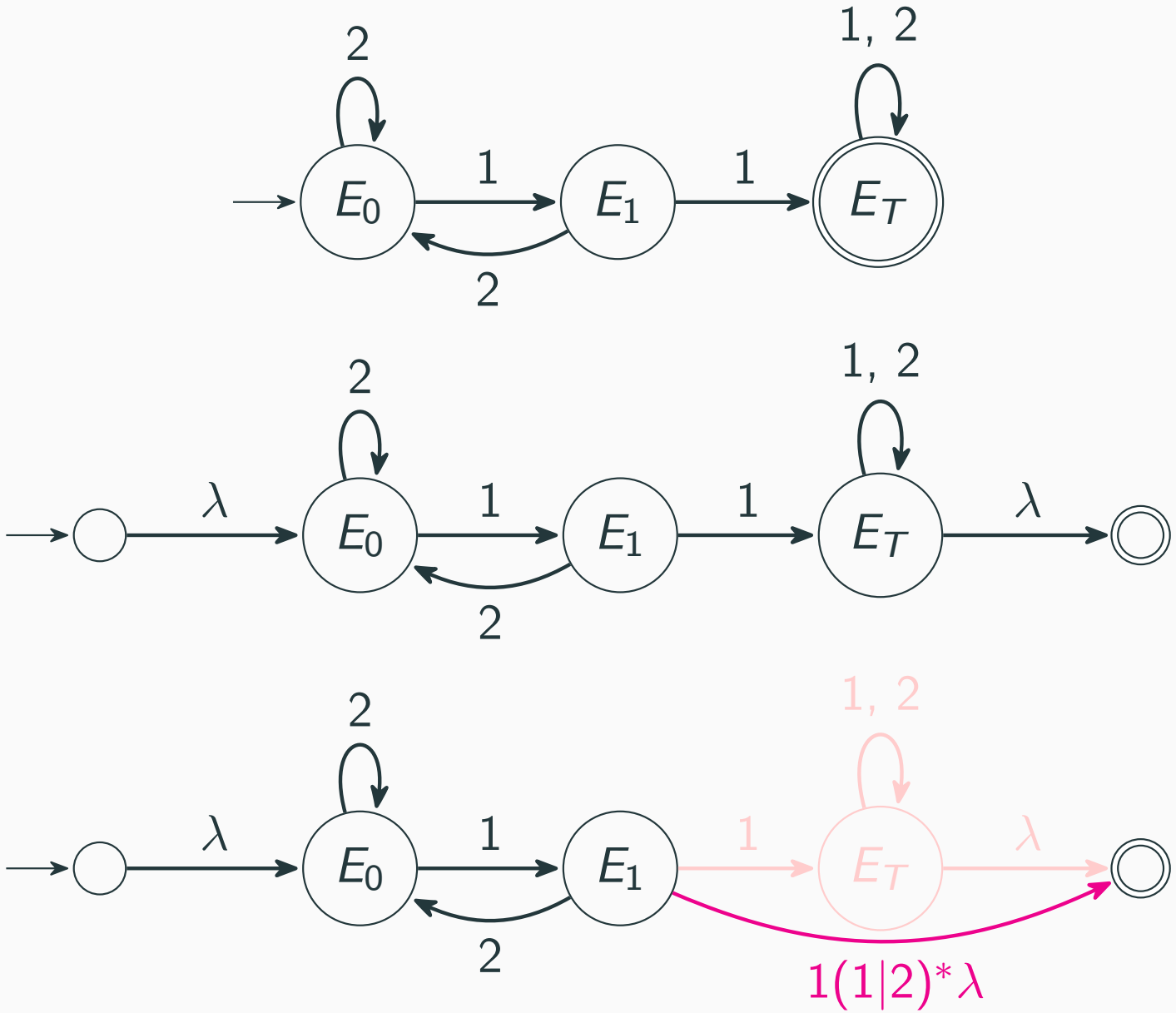


Para complementar un AFD completo, hacemos $F' = Q \setminus F$.

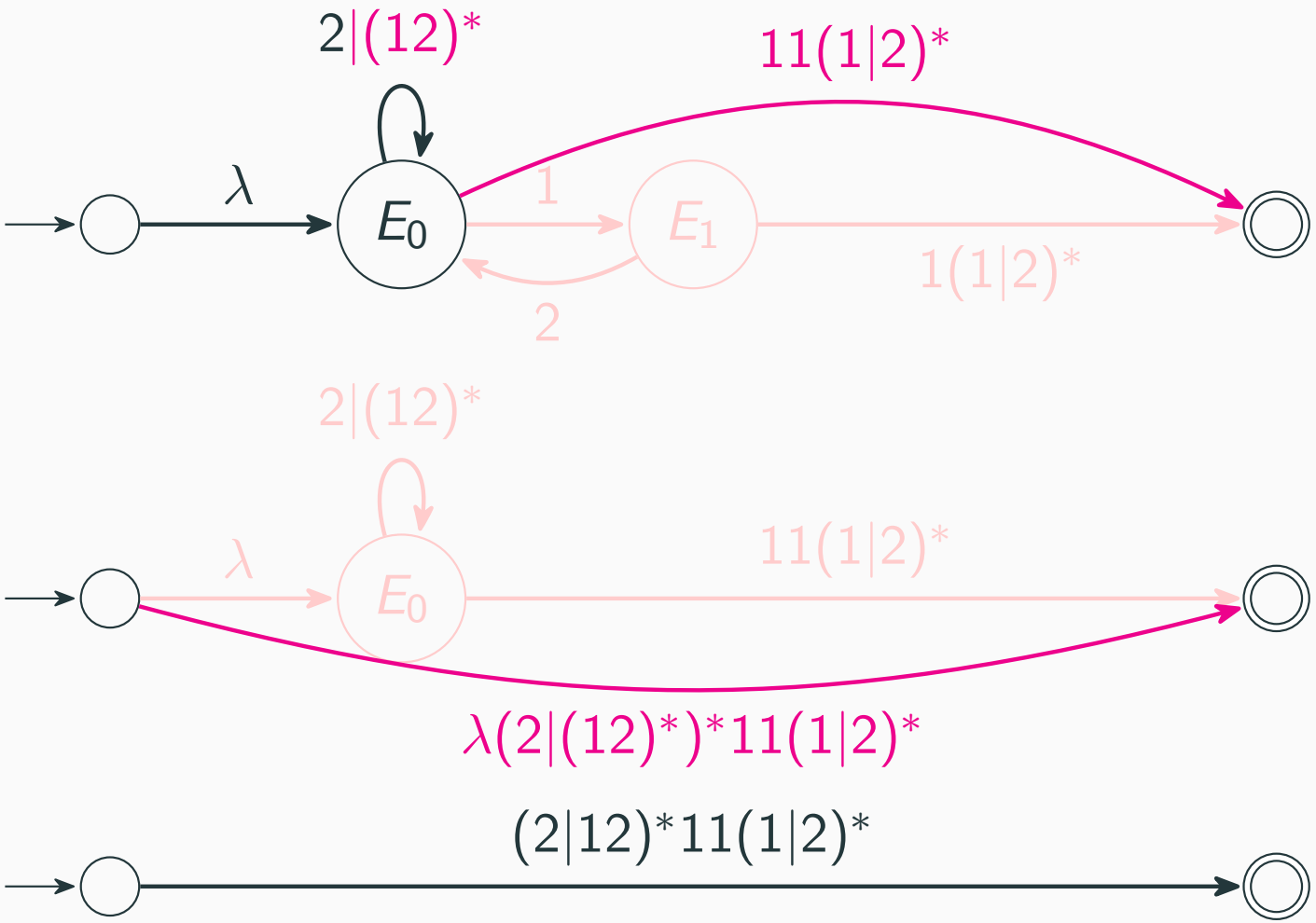
$$A^c = \langle \{E_0, E_1, E_T\}, \{1, 2\}, \delta, 0, \{E_T\} \rangle$$



3 - Conversión a ER i



3 - Conversión a ER ii



Terminando

Ejercicio

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 $L((2|12)^*(1|\lambda))^c$ y es

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

Conclusiones

Vimos,

- Expresiones regulares, ejemplos y ejercicios
- Pasaje de AF a ER (Método de eliminación de estados)
- Pasaje de ER a AF (Método de derivadas)
- Un ejercicio integrador

Ya pueden hacer toda la Práctica de Expresiones Regulares.