INF155 - Informática Teórica

Ayudantía #2

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Motivación

Las expresiones regulares son una manera simple de ***denotar*** lenguajes. Ahora, sería ideal contar con un artefacto que nos permita ***reconocer*** un lenguaje, es decir, le damos una palabra y determinamos si pertenece (o no) a un lenguaje dado. La máquina abstracta más simple que nos permite responder dicha pregunta se conoce como autómata, y la estudiaremos a continuación.

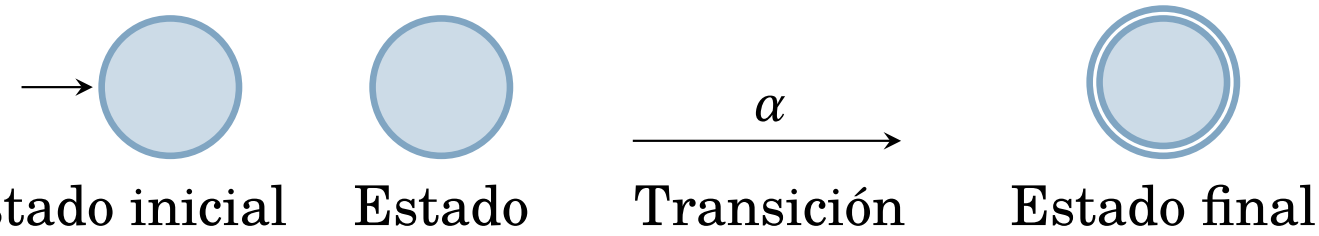
DFA - Autómata Finito Determinista

Un *autómata finito determinista* (DFA) $M = (Q, \Sigma, \delta, q_0, F)$ se compone de:

$$
\begin{aligned}
Q &: \text{Conjunto finito de estados.} \\
\Sigma &: \text{Alfabeto.} \\
\delta &: \textit{Función de transición}, \delta : Q \times \Sigma \to Q \\
q_0 &: \textit{Estado inicial}, q_0 \in Q \\
F &: \textit{Estados finales}, F \subseteq Q.
\end{aligned}
$$

Definición: Un estado *muerto* es un estado **no final** del cual el autómata no sale con ningún símbolo.

Notación gráfica y consideraciones

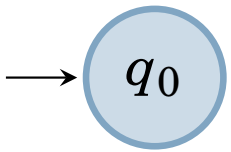


- Los estados muertos NO se dibujan (para simplificar el dibujo).
- Sobre la transición va un único símbolo.
- Cada estado debe llevar una etiqueta que lo identifique del resto.
- El estado inicial es ÚNICO (no hay 2 o más).
- Un DFA puede tener múltiples estados finales.

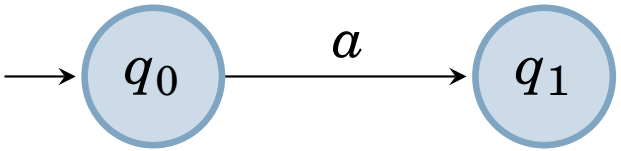
DFA - Ejemplo

Consideremos el alfabeto $\Sigma = \{a, b, c\}$. Supongamos que tenemos la expresión regular $R = a(b \mid c)^*$. Construiremos un DFA que nos permita reconocer $\mathscr{L}(R)$.

1. Dibujamos el estado inicial y lo llamamos q_0:

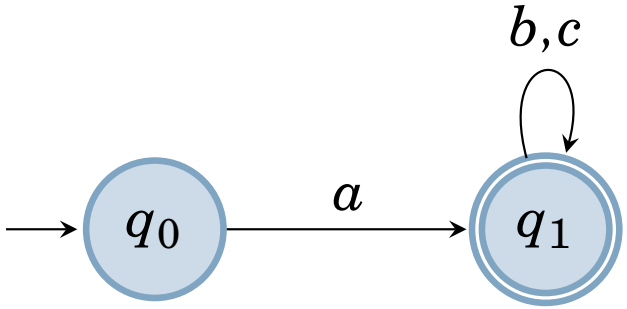


2. Notamos que todas las palabras de $\mathscr{L}(R)$ comienzan con una a, así que añadimos un nuevo estado (al que llamaremos q_1, y al cual llegaremos desde q_0 consumiendo el símbolo a).



DFA - Ejemplo

3. Notamos que $(b \mid c)$ tiene una estrella de Kleene (o sea, puede repetirse 0 o más veces). Por lo tanto, una vez que leemos el símbolo a del inicio de la palabra no es necesario cambiar a otro estado. Es más, podríamos no leer ningún símbolo b o c, y tendríamos una palabra válida en $\mathscr{L}(R)$, por lo que q_1 es el estado final de nuestro autómata. Así, nuestro dibujo quedaría así:



DFA - Ejemplo

Dibujamos el estado muerto (en rojo) solo con fines ilustrativos:

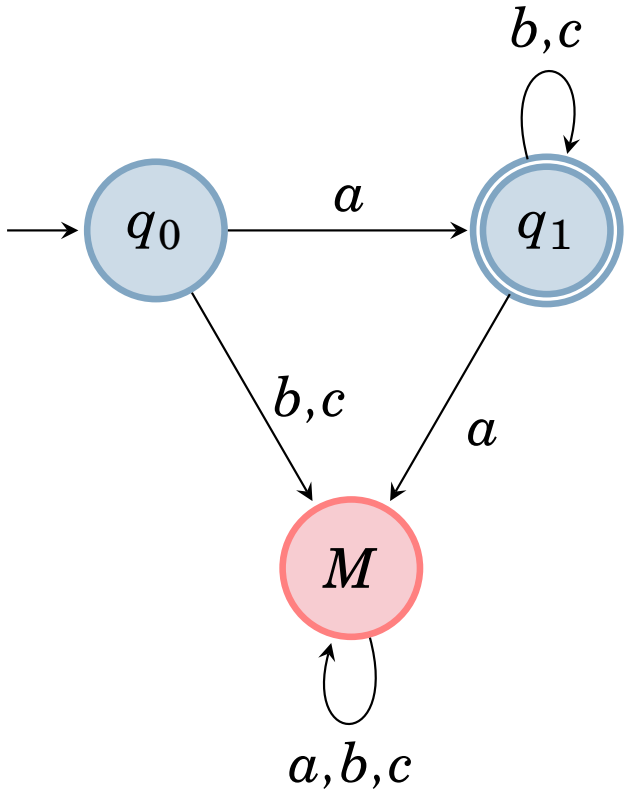


Tabla de transición

Todo autómata tiene una función de transición asociada. Una forma de visualizar dicha función es mediante una **tabla de transición**. La tabla de nuestro autómata es la siguiente:

Estado	**a**	**b**	**c**	**Tipo**
q_0	q_1	M	M	Inicial
q_1	M	q_1	q_1	Final
M	M	M	M	Muerto

NFA - Autómata Finito No Determinista

Un *autómata finito no determinista* (NFA) $M = (Q, \Sigma, \delta, q_0, F)$ se compone de:

Q : Conjunto finito de estados.

Σ : Alfabeto.

δ : *Función de transición*, $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ (conjunto potencia).

q_0 : *Estado inicial*, $q_0 \in Q$

F : *Estados finales*, $F \subseteq Q$.

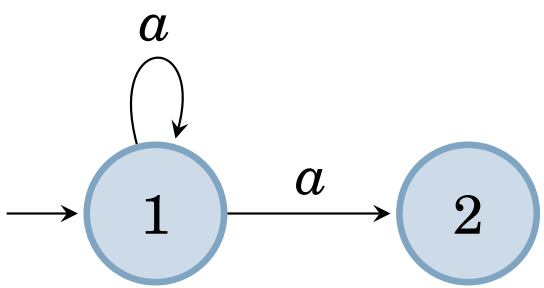
No determinismo

La idea del no-determinismo significa que el autómata no siempre tiene definido completamente el camino a seguir, por lo que debe tomar decisiones. Así la computación es exitosa si existe una secuencia de decisiones que lleven al autómata a un estado de aceptación.

En palabras sencillas: **el autómata "adivina" cuál es la decisión correcta, y siempre tomará el camino más corto que lo lleve a ella**.

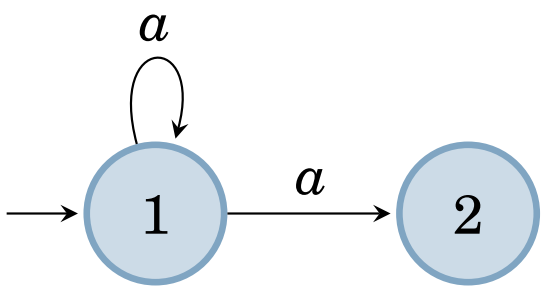
DFA vs NFA

La principal diferencia entre un NFA y un DFA es que el primero permite transiciones ε. Pero hay diferencias que no son tan obvias a primera vista. Observemos el siguiente autómata: ¿es un NFA o un DFA?



DFA vs NFA

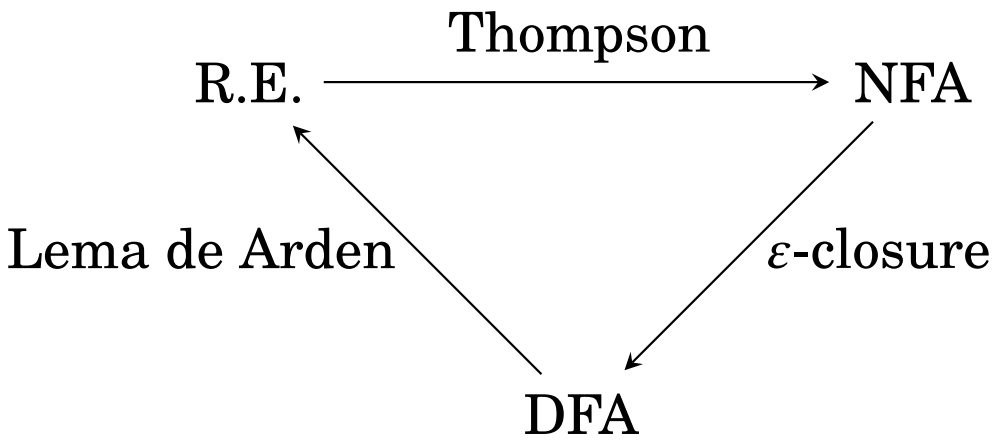
La principal diferencia entre un NFA y un DFA es que el primero permite transiciones ε. Pero hay diferencias que no son tan obvias a primera vista. Observemos el siguiente autómata: ¿es un NFA o un DFA?



Notemos que el autómata, al encontrarse en el estado 1 y recibir un símbolo a, debe decidir si moverse al estado 2 o mantenerse en el mismo estado. Por lo tanto, corresponde a un NFA.

Relación entre DFA, NFA y RE

Hasta el momento tenemos 3 herramientas para describir lenguajes: REs, DFAs y NFAs. La relación entre ellas (y cómo transformar unas en otras) se representa en el siguiente diagrama:



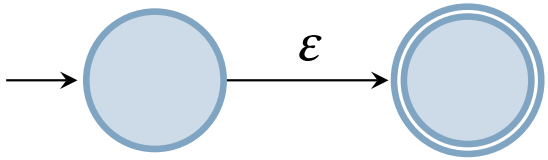
Construcción de Thompson

La construcción explícita de un NFA que acepta el lenguaje denotado por una RE se conoce como *construcción de Thompson*. Esta construcción es recursiva, y entrega como resultado un autómata con un único estado final.
A continuación, veremos las reglas para construir los NFAs.

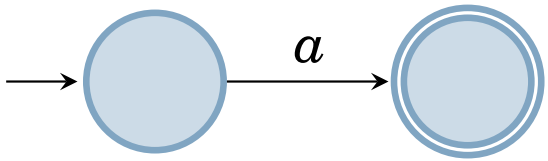
Reglas de construcción de Thompson

Para el lenguaje vacío $\varnothing$:

Para la palabra vacía ε:

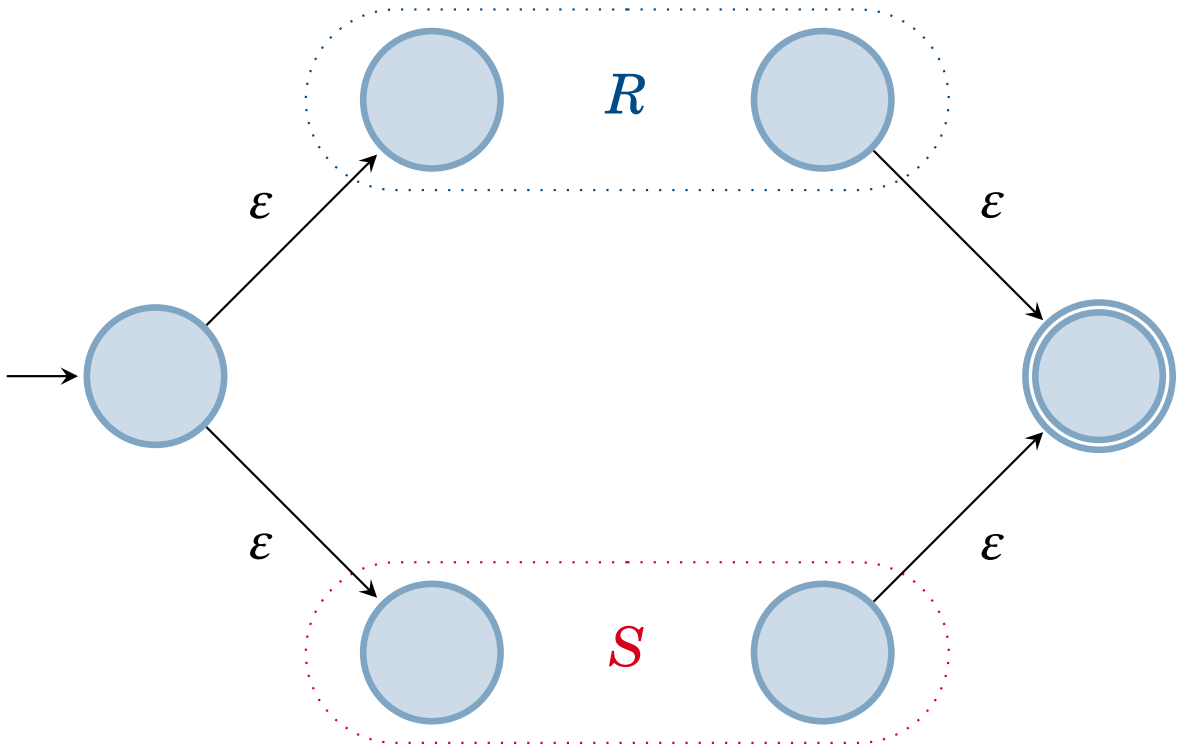


Para un símbolo $a \in \Sigma$:



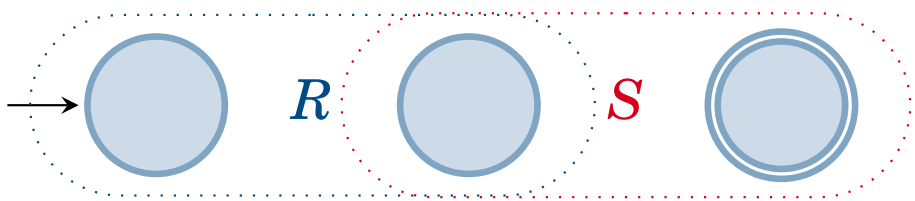
Reglas de construcción de Thompson

Alternancia entre dos expresiones regulares $(R) \mid (S)$:



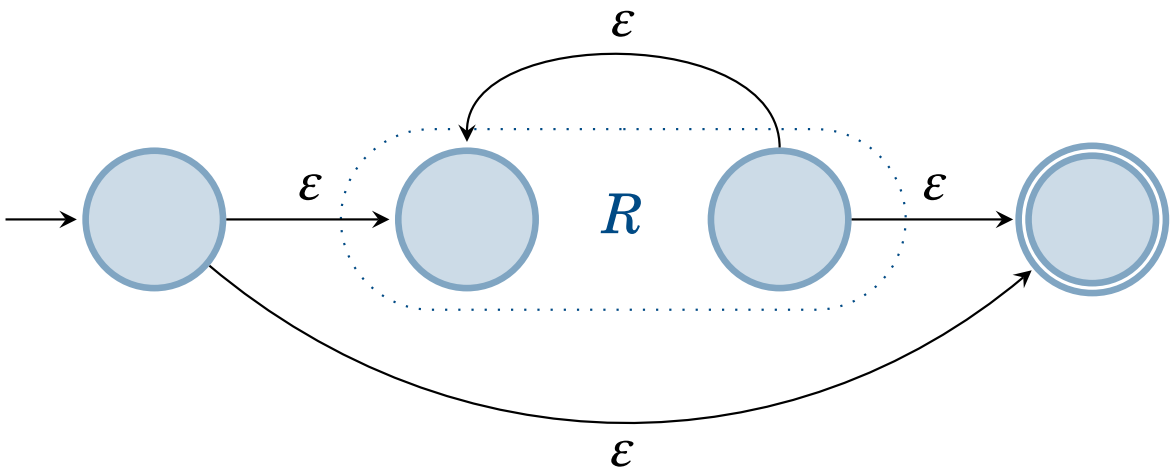
Reglas de construcción de Thompson

Concatenación de dos expresiones regulares $(R) \cdot (S)$:



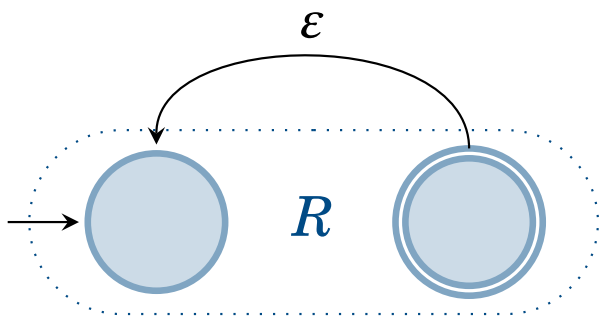
Reglas de construcción de Thompson

Estrella de Kleene de una expresión regular $(R)^*$:



Reglas de construcción de Thompson

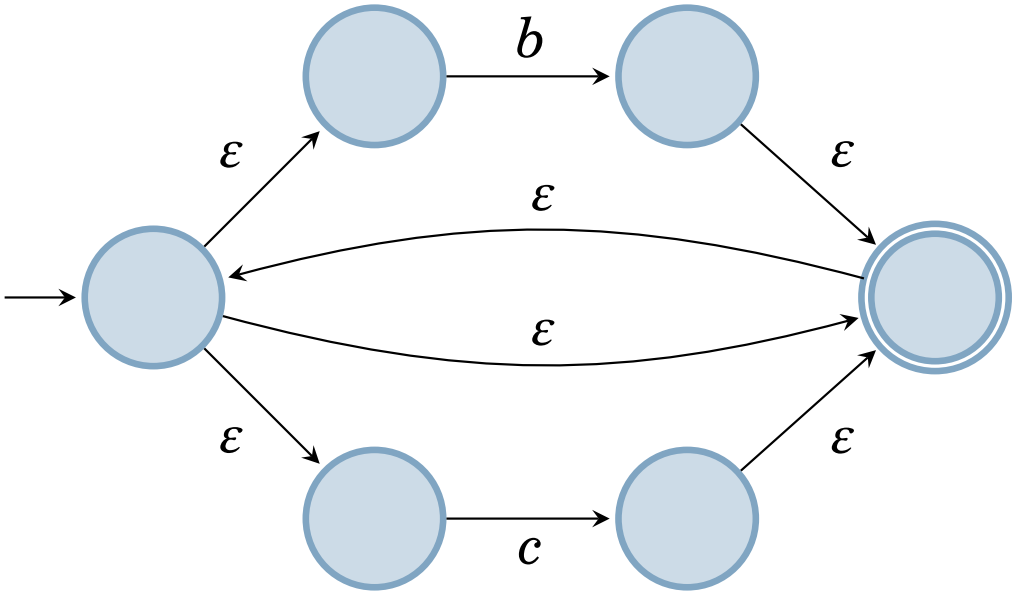
Kleene Plus de una expresión regular $(R)^+ = (R) \cdot (R)^*$:



Reglas de construcción de Thompson - Ejemplo

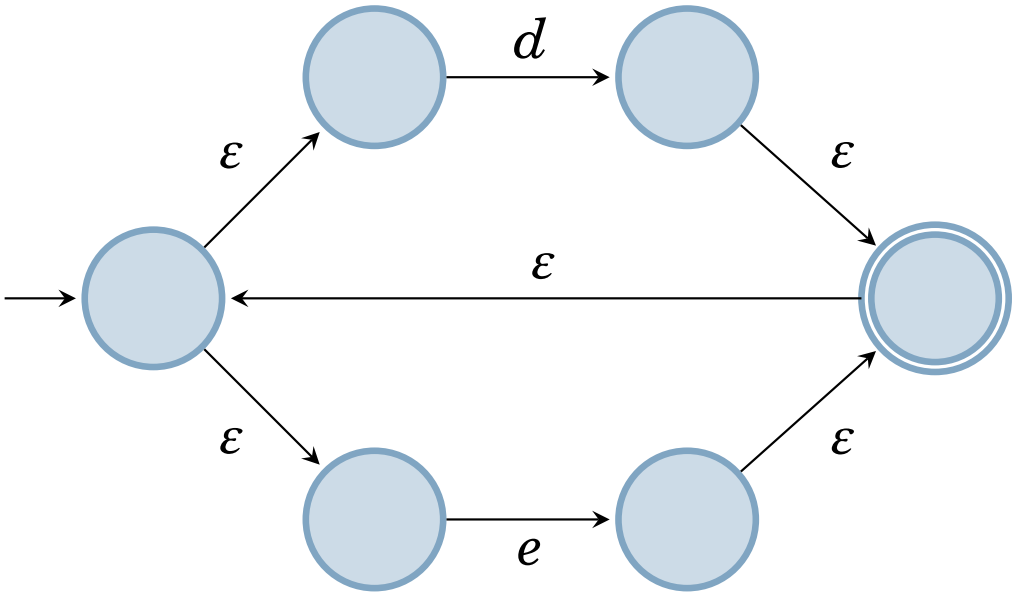
Sea $R = a((b \mid c)^* \mid (d \mid e)^+)^* f$ una RE. Construyamos el NFA asociado por partes:

1. $(b \mid c)^*$



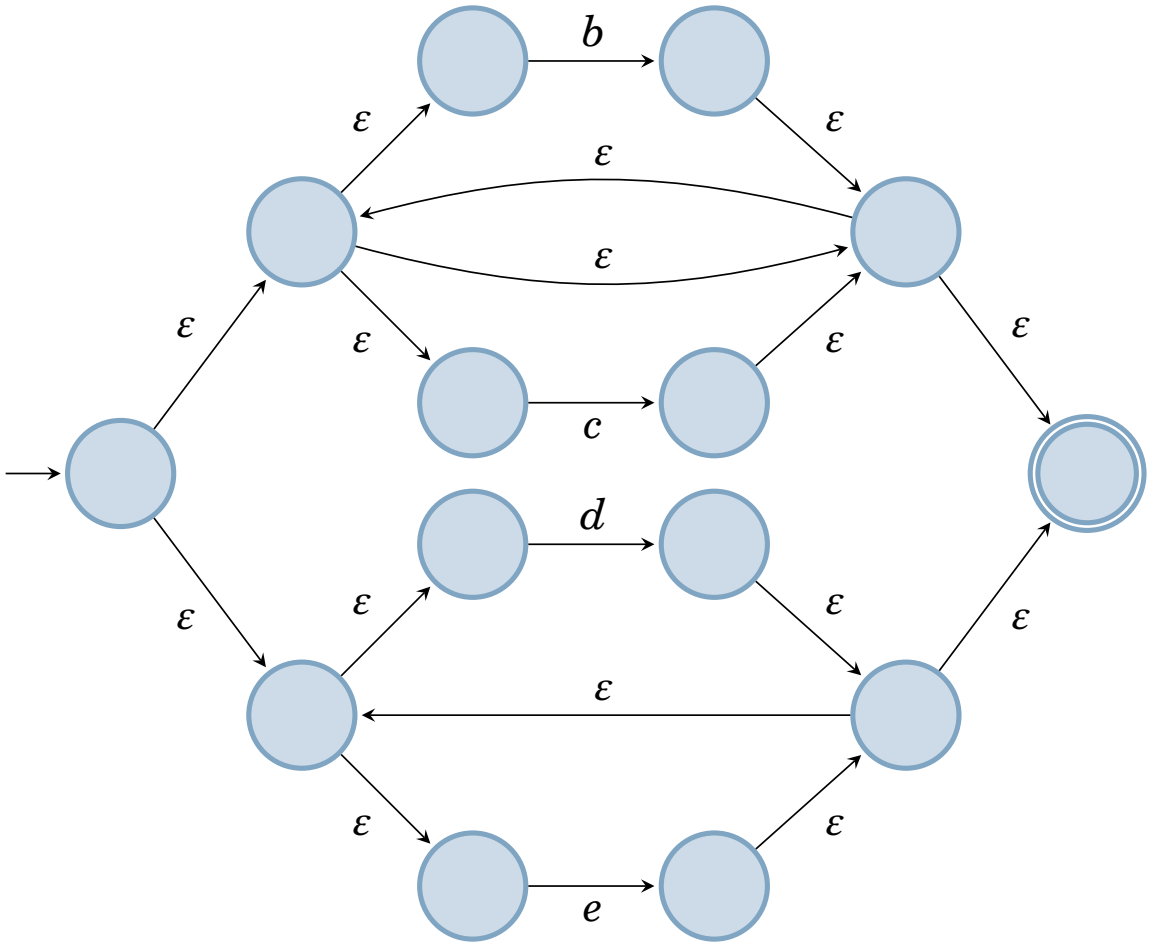
Reglas de construcción de Thompson - Ejemplo

2. $(d \mid e)^+$



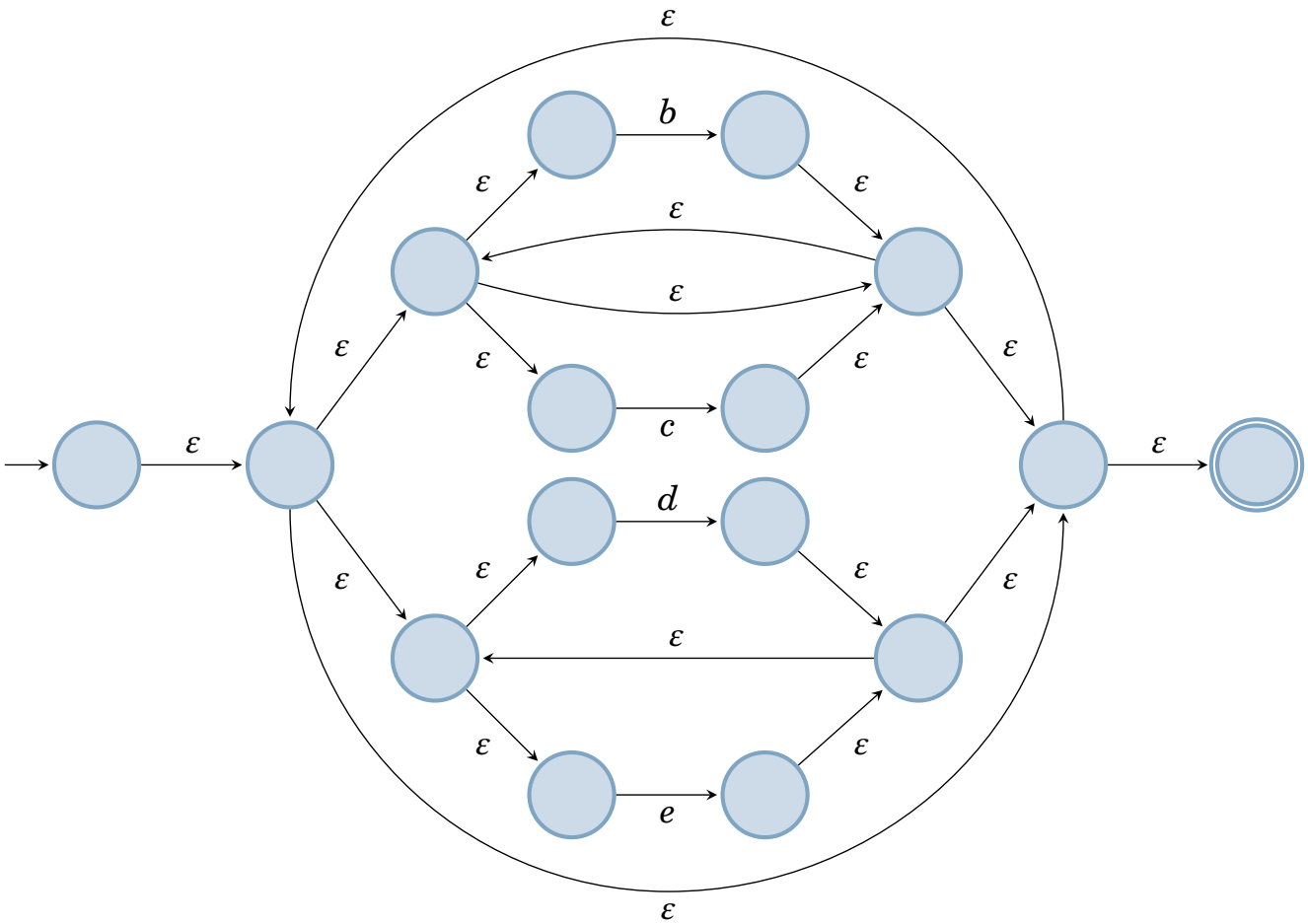
Reglas de construcción de Thompson - Ejemplo

3. $(b \mid c)^* \mid (d \mid e)^+$



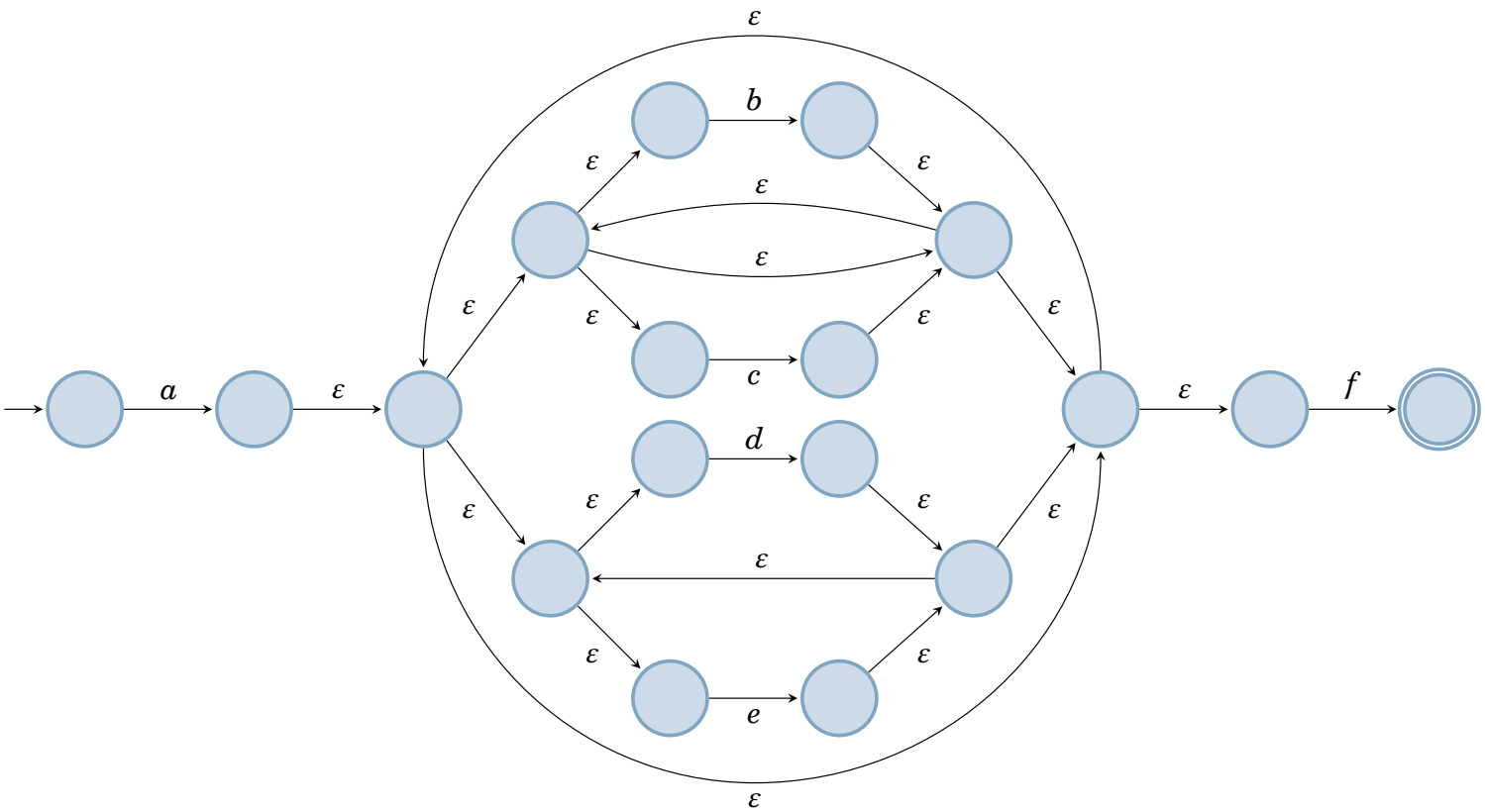
Reglas de construcción de Thompson - Ejemplo

4. $((b \mid c)^* \mid (d \mid e)^+)^*$

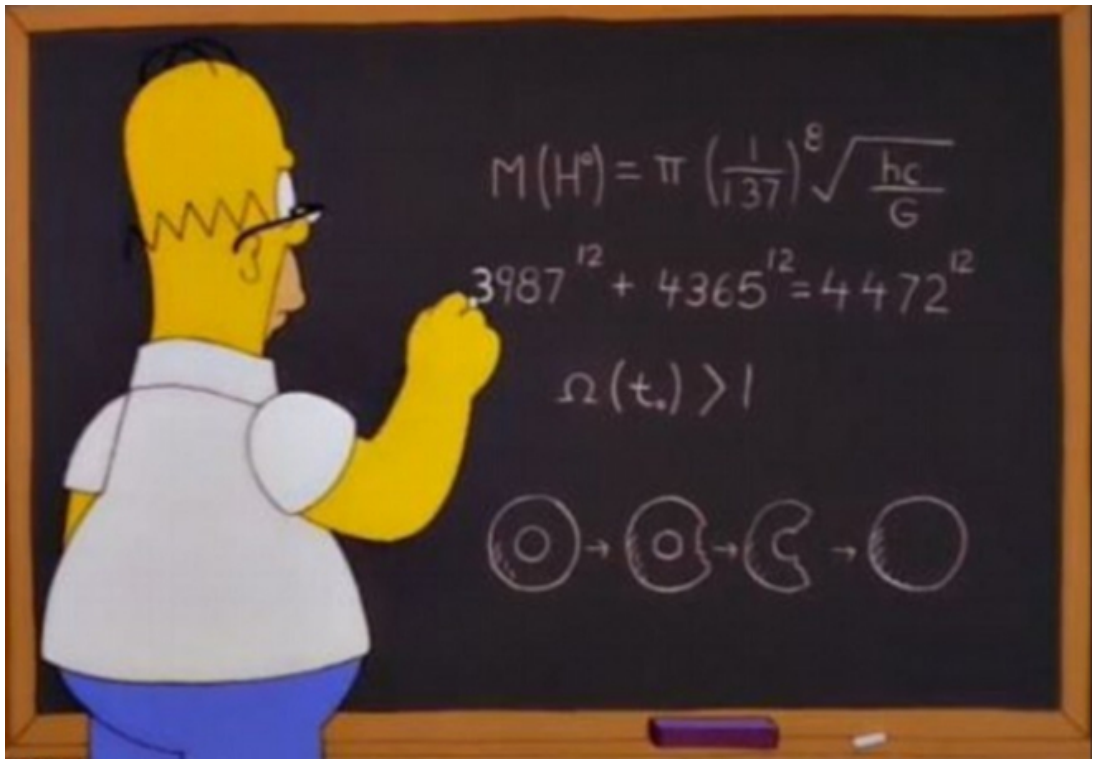


Reglas de construcción de Thompson - Ejemplo

5. Finalmente, $R = a((b \mid c)^* \mid (d \mid e)^+)^* f$



Ejercicios


$M(H^0) = \pi \left(\frac{1}{137}\right)^8 \sqrt{\frac{hc}{G}}$
$3987^{12} + 4365^{12} = 4472^{12}$
$\Omega(t_0) > 1$

Ejercicio 1

Construya los autómatas que reconocen a los siguientes lenguajes sobre el alfabeto $\Sigma = \{a, b, c\}$:

a) $\mathscr{L}_a$ = Todas las palabras en las que aparecen las a antes de las c

b) $\mathscr{L}_b$ = Todas las palabras que contienen a lo más dos a

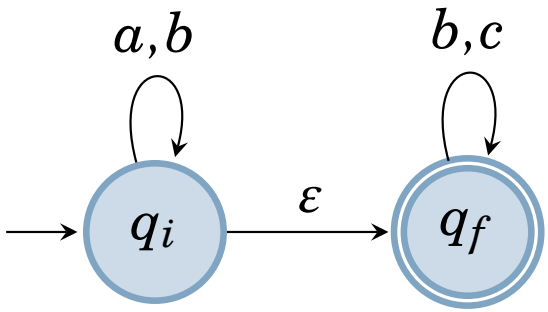
c) $\mathscr{L}_c$ = Todas las palabras en donde si aparecen a, estas aparecen en pares o tripletas pero no solas, esto es aa o aaa, y finalizan con bc o bb

Solución Ejercicio 1

Es ideal primero encontrar las expresiones regulares que denotan a los lenguajes para tener un mayor entendimiento en la construcción de los autómatas, y luego construirlos.

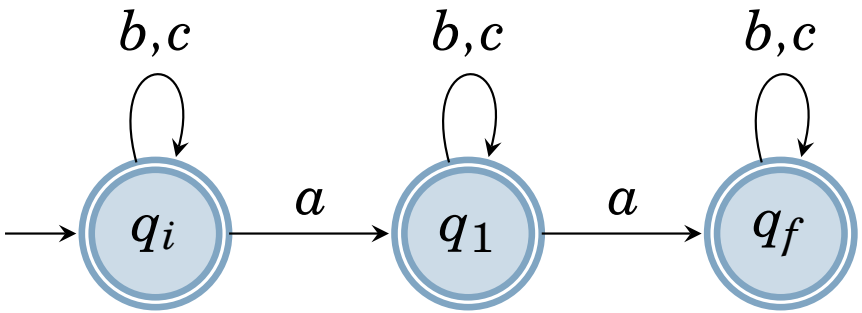
Solución Ejercicio 1 a

La expresión regular que denota al lenguaje $\mathscr{L}_a$ es $(a \mid b)^*(b \mid c)^*$ y el autómata que lo acepta es el siguiente:



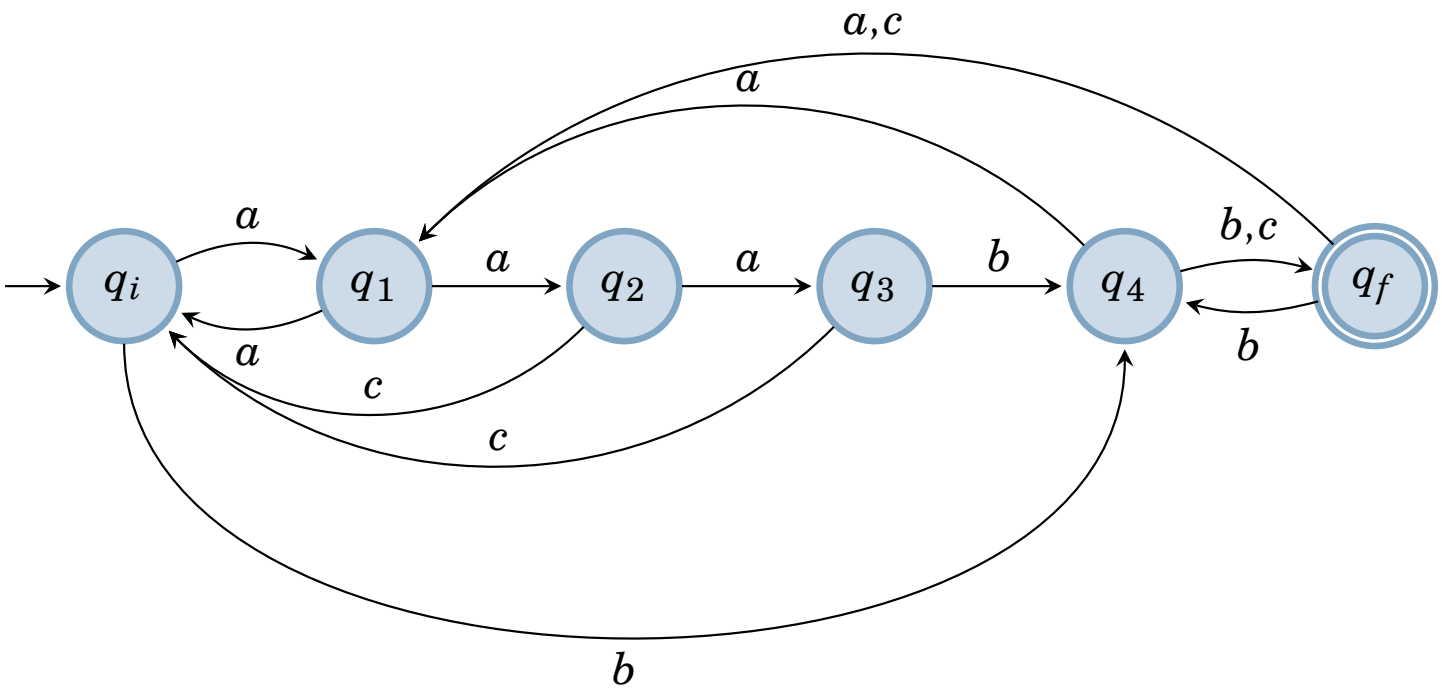
Solución Ejercicio 1 b

La expresión regular que denota al lenguaje $\mathscr{L}_b$ es $(b \mid c)^* \mid (b \mid c)^* a (b \mid c)^* \mid (b \mid c)^* a (b \mid c)^* a (b \mid c)^*$ y el autómata que lo acepta es el siguiente:



Solución Ejercicio 1 c

La expresión regular que denota al lenguaje $\mathscr{L}_c$ es $(aa \mid aaa \mid b \mid c)b(b \mid c)$ y el autómata que lo acepta es el siguiente:



Ejercicio 2

Lisa Simpson está ayudando a su hermana menor, Maggie, a entender DFA's y NFA's pero se dio cuenta que su hermana no comprende elementos poco gráficos, por lo que Lisa le pide ayuda a usted, alumno de Informática Teórica, para construir las siguientes definiciones textuales de DFA y NFA a su representación gráfica.

a) $M_a = (\{q_i, q_1, q_2, q_3, q_f\}, \{a, b, c\}, \delta_a, q_i, \{q_f\})$
b) $M_b = (\{q_i, q_1, q_{f_1}, q_{f_2}\}, \{a, b, c\}, \delta_b, q_i, \{q_{f_1}, q_{f_2}\})$

En donde δ_a y δ_b se definen a continuación

Ejercicio 2 - Transiciones

- $\delta_a =$

Estado	a	b	c
q_i	q_1	q_3	q_3
q_1	q_1	q_2	q_3
q_2	q_3	q_3	q_f
q_3	q_3	q_3	q_3
q_f	q_3	q_f	q_3

- $\delta_b =$

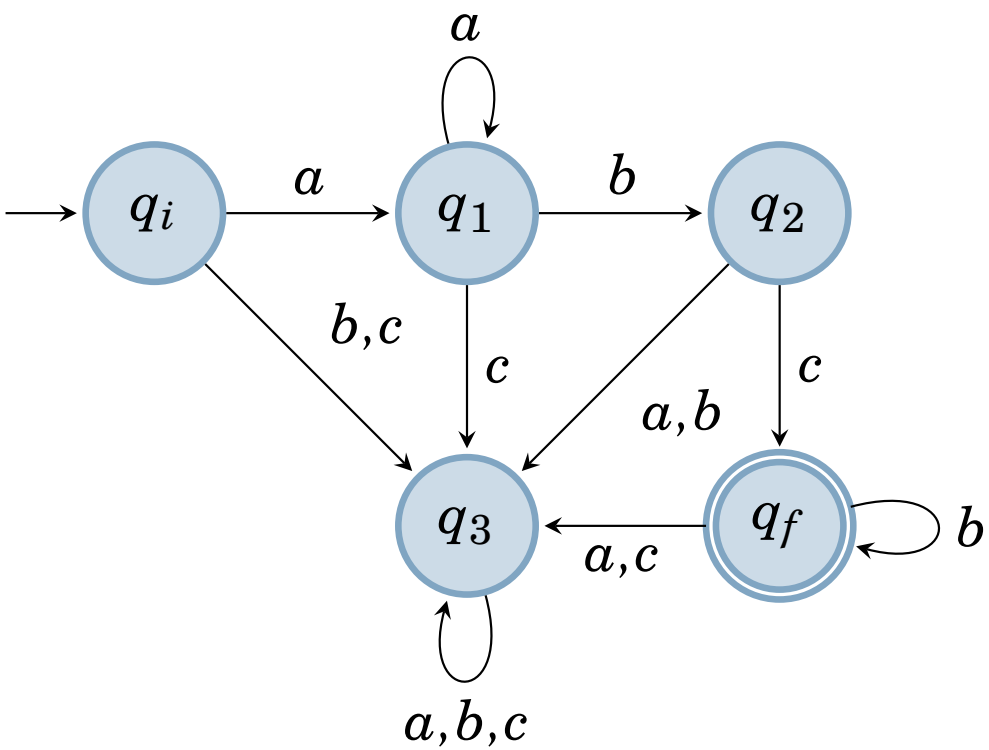
Estado	a	b	c
q_i	$\{q_i, q_{f_2}\}$		$\{q_1\}$
q_1	$\{q_1\}$	$\{q_{f_2}\}$	$\{q_{f_1}\}$
q_{f_1}	$\{q_{f_2}\}$	$\{q_1\}$	$\{q_i\}$
q_{f_2}			

Solución Ejercicio 2

Como ya tenemos la definición de los autómatas M_1 y M_2, esto es, sabemos todos sus estados, alfabeto, estado inicial, estados finales y función de transición, solo nos queda representarlos en dibujo.

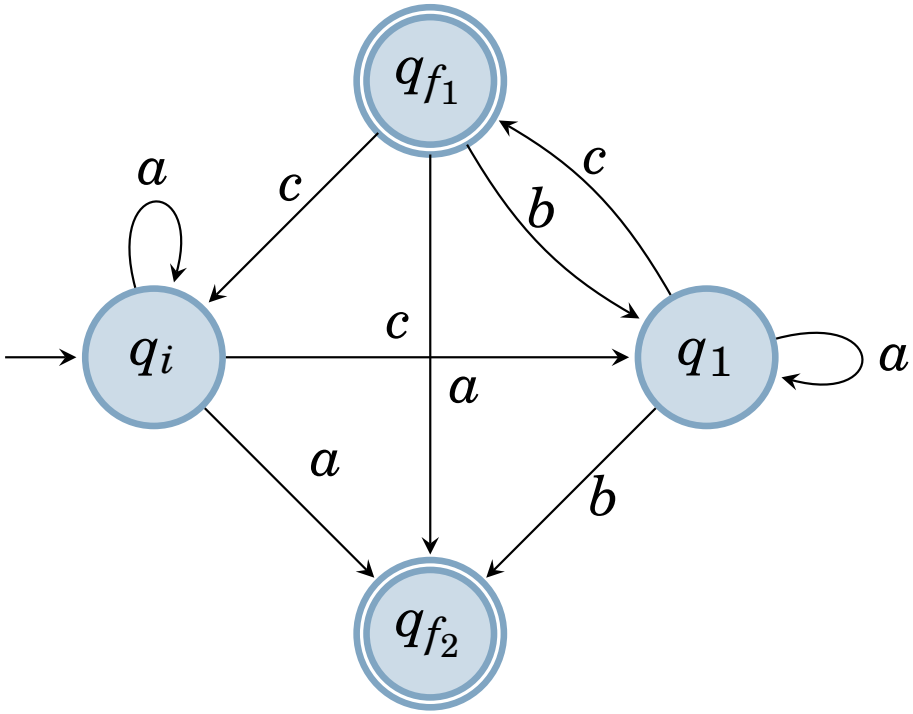
Solución Ejercicio 2 a

Si analizamos la función de transición σ_a, notaremos que estamos frente a un DFA. Luego, solo nos queda dibujar los 5 estados $(q_i, q_1, q_2, q_3, q_f)$ y su función de transición a través de arcos entre los estados.



Solución Ejercicio 2 b

Al igual que el caso anterior, analizamos la función de transición σ_b y notamos que estamos frente a un NFA. Luego, solo nos queda dibujar los 4 estados $(q_i, q_1, q_{f_1}, q_{f_2})$ y su función de transición a través de arcos entre los estados.



Ejercicio 3

Bob Patiño ha estado planificando una venganza contra su hermano Cecilio, y para esto ha desarrollando una expresión regular que es capaz de reconocer correos electrónicos a través de la red local en la cual se ubica. El problema es que su computador solo es capaz de reconocer autómatas mas no expresiones regulares, por lo que Bob le pide ayuda a usted para que construya el autómata utilizando las reglas de construcción de Thompson y así él poder ejecutar su plan.

La expresión regular (sobre $\Sigma = \{\alpha, @, .\}$) que construyó Bob es:

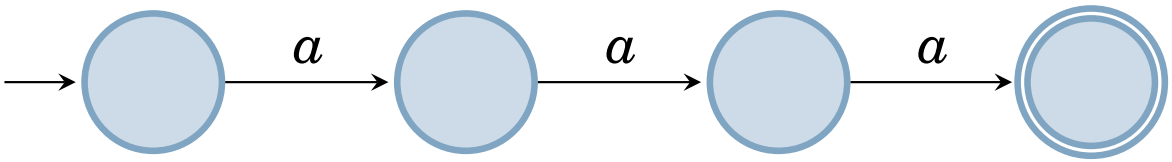
$$\alpha^3\alpha^*@\alpha^3\alpha^*(.|\alpha^2|\alpha^3)^+$$

Solución Ejercicio 3

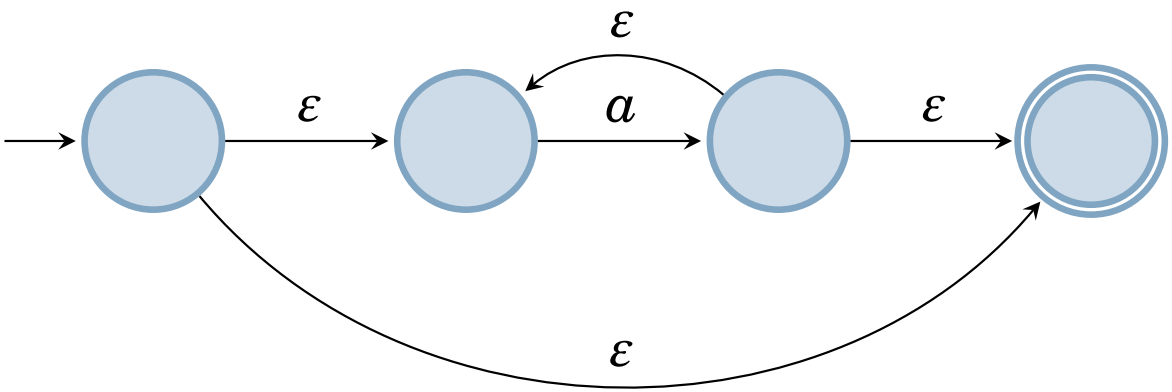
Primero construiremos las representaciones de segmentes individuales de la expresión regular, para finalmente unirlas y generar el NFA que reconoce el lenguaje representado por la expresión regular

Solución Ejercicio 3

- $a^3 =$

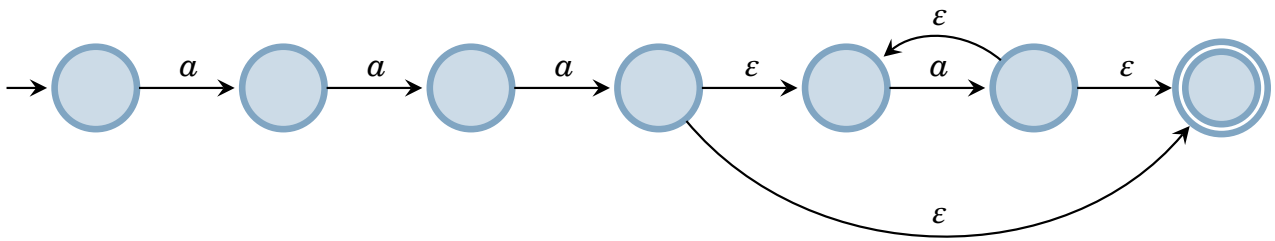


- $a^* =$



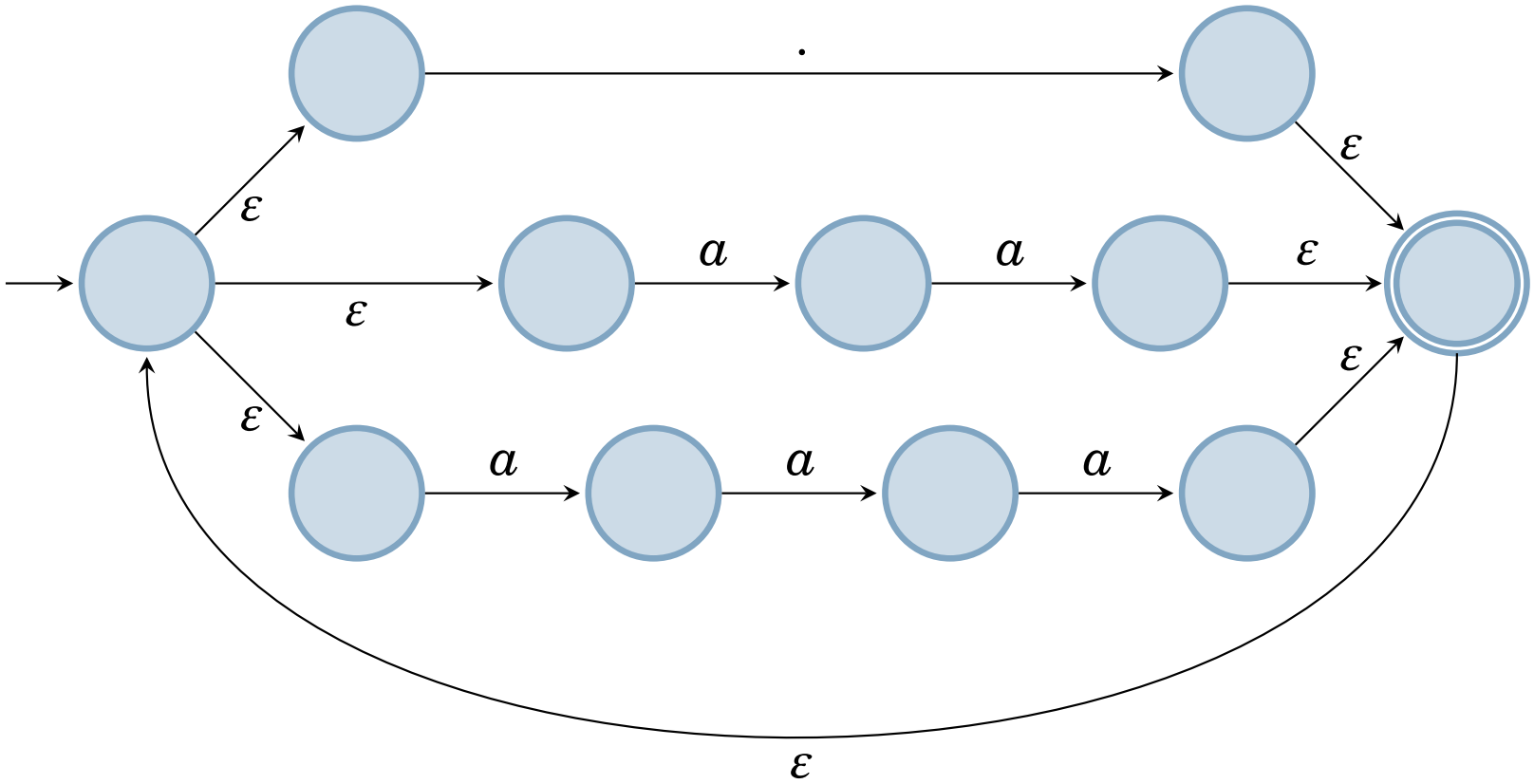
Solución Ejercicio 3

- $a^3a^* =$



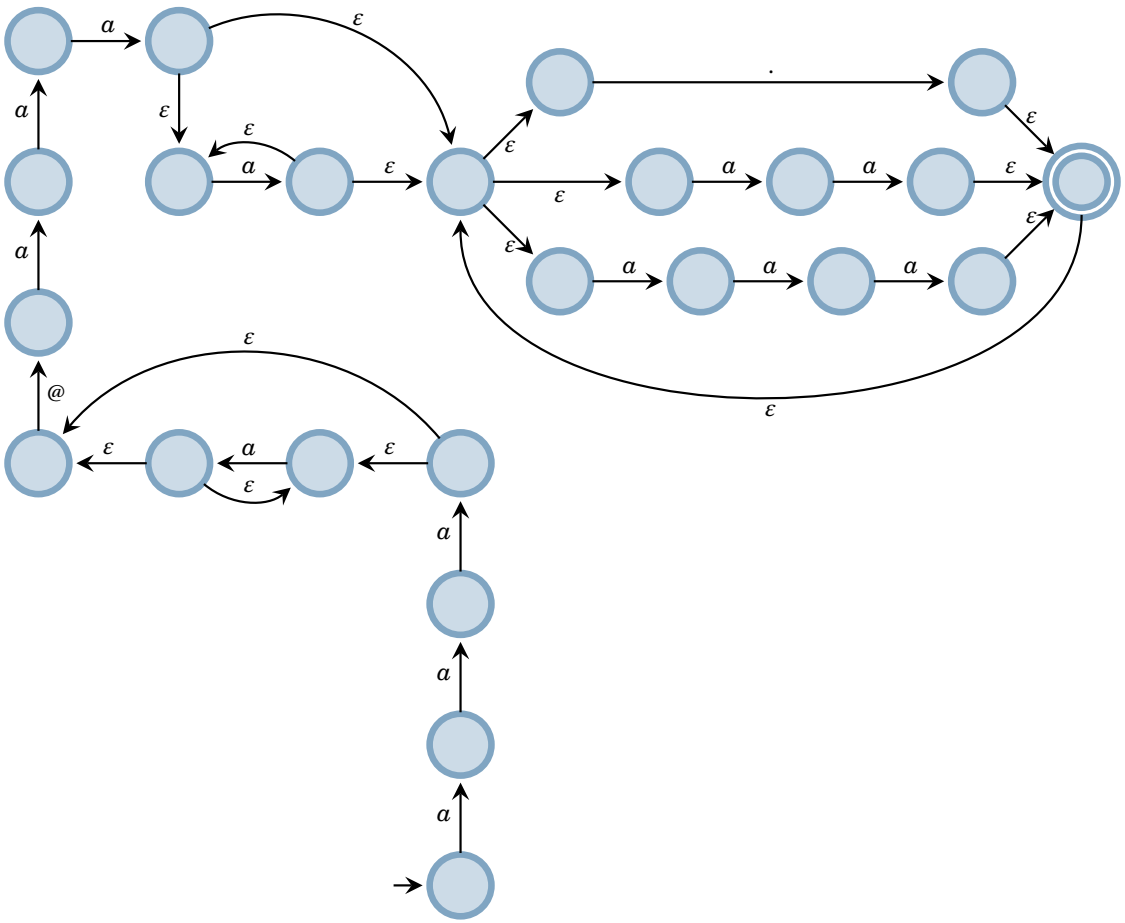
Solución Ejercicio 3

- $(. \mid a^2 \mid a^3)^+ =$



Solución Ejercicio 3

Una vez construido los segmentos, nos queda unirlos en un único NFA:



Ejercicio 4 - Opcional

Dado el lenguaje $\mathscr{L} = \{a^n b^n : n \geq 0\}$, intente construir un autómata finito que lo reconozca. ¿Se puede? Concluya.

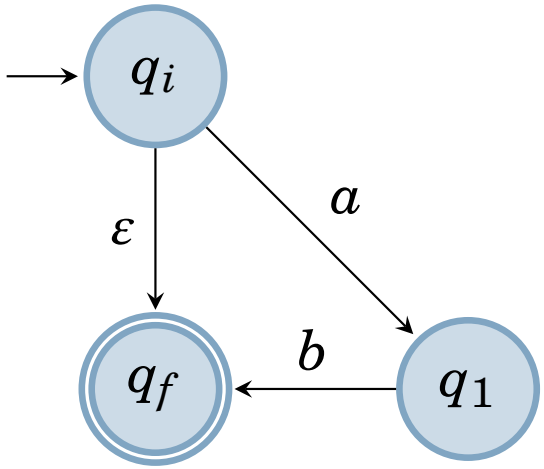
Solución Ejercicio 4

Para tratar de construir un autómata capaz de reconocer al lenguaje $\mathscr{L}$, es de gran ayuda comprender como se construyen sus palabras.

Notar que la palabra mínima es ε, luego la sucede ab, después $aabb$, $aaabbb$, $aaaabbbb$, ... Parece que se inyectasen ab's en el centro de la palabra. Esta idea es muy importante para conceptos que veremos más adelante en la asignatura.

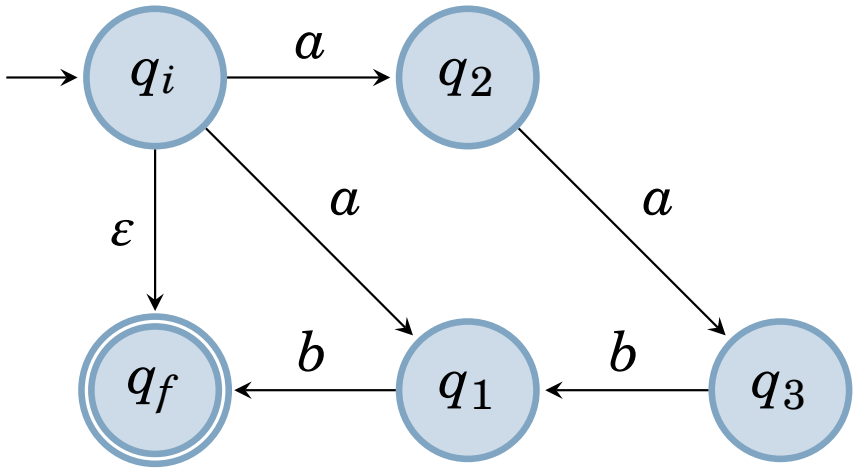
Solución Ejercicio 4

Nuestra primera aproximación de NFA que acepta parcialmente el lenguaje es:



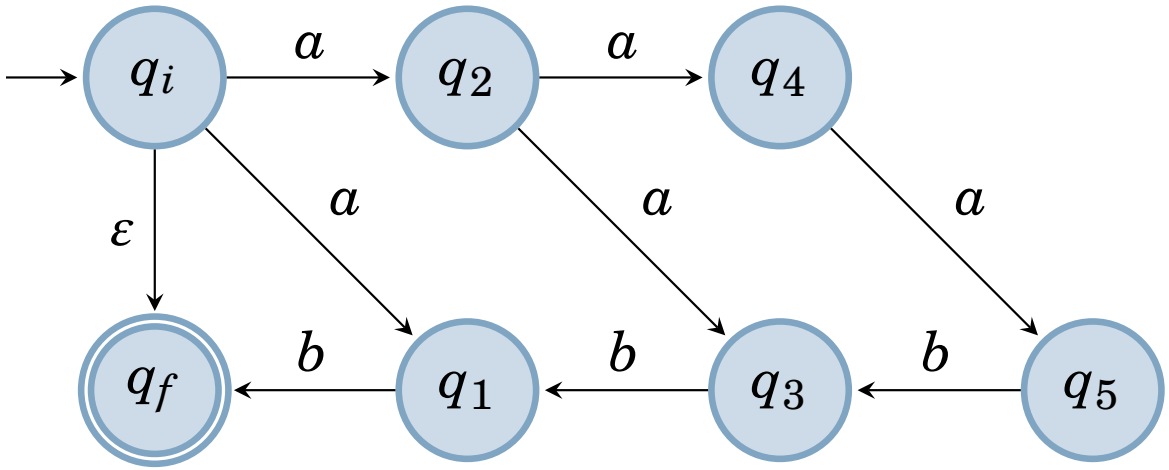
Claramente, esta aproximación solo reconoce ε y ab.
¡Extendámoslo!

Solución Ejercicio 4



Ahora, a diferencia de antes, también aceptamos *aabb*. *Keep bombing*

Solución Ejercicio 4



Con esta versión se añade $aaabbb$, ¿Podemos seguir agregando 2 estados hasta lograr que para todo n tal que $a^n b^n$ sea aceptado?

Solución Ejercicio 4 - Conclusiones

La respuesta es un tajante ***NO***. Añadir infinitos estados rompe el principio de estos autómatas, ser un autómata finito.
La primera conclusión que podemos extraer es que los NFA (y por tanto DFA) no son lo suficientemente potentes para aceptar este tipo de lenguajes. Pero ¿por qué?

Solución Ejercicio 4 - Conclusiones

Respuesta simple: los NFa y DFA son *stateless*, lo que significa que no recuerdan lo que han leído. Pero entonces ¿Existe algún artefacto que sí sea capaz de recordar lo leído, y por tanto aceptar este tipo de lenguajes? Sí, estos artefactos son conocidos como PDA. Lo verán en clases con el profesor, así que no se pierda otro capítulo de Informática Teórica.

Solución 4 - Resumen

- Los NFA / DFA no tienen la capacidad de reconocer cualquier lenguaje. Solo son capaces de reconocer *lenguajes regulares*
- Los NFA / DFA / regex no tienen memoria. No recuerdan lo que han leído previamente (Ojo: las regex de *Lenguajes de programación* no son regex puras, estas son las famosas `POSIX BRE / ERE`)
- Antes de intentar construir un NFA / DFA, si el lenguaje parece sospechoso de no ser regular, es buena idea demostrarlo a través del *lema del bombeo*

¿Dudas?

INF155 - Informática Teórica

Ayudantía #2

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA