Autómatas y Lenguajes Formales
Nota 04. Expresiones Regulares*

Noé Salomón Hernández S.

1. Expresiones Regulares

En aritmética podemos usar las operaciones + y × para construir expresiones tales como

$$(5 + 3) \times 4.$$

Similarmente, podemos usar operadores regulares para construir expresiones describiendo lenguajes, las cuales son llamadas **expresiones regulares**. Por ejemplo,

$$(0 + 1)0^*.$$

El valor de la expresión aritmética es el número 32. El valor de una expresión regular es un lenguaje. En este caso es el lenguaje que consiste de todas las cadenas comenzando en 0 o 1, seguidas de cualquier número de 0s. Conseguimos este lenguaje al descomponer la expresión en sus partes. Primero, los símbolos 0 y 1 son nombres cortos para los lenguajes $\{0\}$ y $\{1\}$, y el operador + representa la operación $\cup$. Así, $(0 + 1)$ significa $(\{0\} \cup \{1\})$. El valor de esta parte es el lenguaje $\{0, 1\}$. La parte 0^* significa $\{0\}^*$, y su valor es el lenguaje de las cadenas que son secuencias de 0s, incluyendo a la secuencia vacía. Segundo, como el símbolo × en álgebra, el símbolo de la concatenación $\cdot$ es a menudo implícito en expresiones regulares. Así, $(0 + 1)0^*$ es el nombre corto para $(0 + 1) \cdot 0^*$. La concatenación junta las cadenas de las dos partes para obtener el valor de la expresión completa.

Las expresiones regulares son una notación algebraica para definir lenguajes que toman una perspectiva declarativa, a diferencia de la perspectiva computacional en un autómata; además, tienen un papel importante en aplicaciones de las ciencias de la computación. En aplicaciones que manejan texto, los usuarios pueden requerir buscar cadenas que satisfagan ciertos patrones. Las expresiones regulares proveen un método poderoso para describir tales patrones. El patrón en cuestión puede representar el nombre de una variable, el nombre de un archivo, el nombre de un directorio, expresiones numéricas en un lenguaje de programación, el texto en una búsqueda, el formato para un correo electrónico o dirección IP. La idea es que con una expresión regular se capturan patrones de texto para múltiples propósitos. Utilidades en UNIX como `awk` y `grep`, lenguajes de programación como PYTHON, y editores de texto proveen mecanismos para la descripción de patrones usando expresiones regulares o mecanismos similares.

*Esta nota se basa en libro de M. Sipser. *Introduction to the Theory of Computation*, en las notas del profesor Favio Miranda, y en los textos del profesor Rajeev Motwani, los cuales pueden encontrar aquí.

Ejemplo 1.1 Otro de ejemplo de expresión regular es

$$(0+1)^*.$$

Comienza con el lenguaje $(0+1)$ y aplica la operación *. El valor de esta expresión es el lenguaje que consiste de todas las cadenas de 0s y 1s. Si $\Sigma = \{0,1\}$, podemos escribir Σ como la forma corta de la expresión regular $(0+1)$. Si Σ es un alfabeto, la expresión regular Σ describe el lenguaje que consiste de todas las cadenas de longitud 1 sobre dicho alfabeto, y Σ^* describe el lenguaje que consiste de todas las cadenas sobre el alfabeto Σ. Similarmente, Σ^*1 es el lenguaje que contiene todas las cadenas que terminan en 1. El lenguaje $(0\Sigma^*) + (\Sigma^*1)$ consiste de todas las cadenas que, o bien empiezan con 0, o terminan con 1.

Definición 1.2 Expresiones regulares. Decimos que R es una expresión regular si R es:

I a para alguna a en el alfabeto Σ;

II ε;

III $\varnothing$;

IV $(R_1 + R_2)$, donde R_1 y R_2 son expresiones regulares;

V $(R_1 \cdot R_2)$, donde R_1 y R_2 son expresiones regulares; ó

VI (R_1^*), donde R_1 es una expresión regular.

En los incisos I y II, las expresiones regulares a y ε representan los lenguajes $\{a\}$ y $\{\varepsilon\}$, respectivamente. En el inciso III, $\varnothing$ representa el lenguaje vacío. En los incisos IV, V y VI, las expresiones representan los lenguajes obtenidos al tomar la unión o la concatenación de los lenguajes que simbolizan R_1 y R_2, o al aplicar el operador estrella al lenguaje simbolizado por R_1, respectivamente.

Definición 1.3 Los lenguajes generados por expresiones regulares son llamados **lenguajes regulares**.

No debemos confundir las expresiones regulares ε y $\varnothing$. La expresión ε representa el lenguaje que contiene a una única cadena, a saber, la cadena vacía; mientras que $\varnothing$ representa el lenguaje que no contiene ninguna cadena.

En las expresiones regulares se tiene la siguiente **precedencia**: la operación estrella se realiza primero, luego la concatenación, y finalmente la suma o unión. De manera que los paréntesis en la expresión pueden ser omitidos, a menos que se desee **cambiar la precedencia anterior**.

Por conveniencia, denotamos a RR^* como R^+. En otras palabras, mientras que R^* tiene a todas las cadenas que resultan de la concatenación, posiblemente nula, de cadenas de R, el lenguaje R^+ tiene todas las cadenas que resultan de concatenar al menos una vez las cadenas de R. Así, $R^+ + \varepsilon = R^*$. También denotamos R^k al resultado de concatenar k veces R consigo misma.

Cuando deseamos distinguir entre una expresión regular R y el lenguaje que ésta describe, escribimos $L(R)$ como el lenguaje de R.

Dar una expresión regular es un ejercicio de creatividad e ingenio. Algunos ejemplos se muestran a continuación:

Ejemplo 1.4

a. En la expresión regular 01^*, el operador estrella de Kleene se aplica al 1 únicamente pues dicho operador es el de mayor precedencia. Así, 01^* es la expresión regular que concatena 0 con 1^*, generando las cadenas binarias que inician con un 0 y enseguida tienen cero o más 1's, es decir, $L(01^*) = \{0, 01, 011, 0111, \ldots\}$.

b. En la expresión regular $(01)^*$ la estrella de Kleene se aplica a la cadena 01, así $(01)^*$ representa a todas las secuencias de cero o más repeticiones de la cadena 01. $L\big((01)^*\big) = \{\varepsilon, 01, 0101, 010101, 01010101, \ldots\}$.

Ejemplo 1.5 Encuentre expresiones regulares para cada uno de los lenguajes siguientes. El alfabeto en todos los casos es $\{0, 1\}$.

a. $\{w \mid w$ comienza con un 1 y termina con un $0\}$ $\quad \mathbf{1(0+1)^*0}$

b. $\{w \mid w$ tiene al menos longitud 3 y su tercer símbolo es un $0\}$ $\quad \mathbf{(0+1)(0+1)0(0+1)^*}$

c. $\{w \mid w$ comienza con 0 y tiene longitud impar, o comienza con 1 y tiene longitud par$\}$
$\mathbf{0(00+01+10+11)^* + 1(0+1)(00+01+10+11)^*}$

d. $\{w \mid w$ es cualquier cadena a excepción de 11 y 111$\}$
$\varepsilon + \mathbf{(0+1)}(\varepsilon + \mathbf{0}) + \mathbf{01} + \mathbf{(0+1)(0+1)0} + \mathbf{001} + \mathbf{011} + \mathbf{101} + \mathbf{(0+1)(0+1)(0+1)(0+1)^+}$

e. $\{w \mid$ cada posición impar de w es un $1\}$ $\quad \mathbf{(10+11)^*}(\varepsilon + \mathbf{1})$

Ejemplo 1.6 Encuentre una expresión regular para el lenguaje de las cadenas en $\{a, b\}^*$ con un número impar de as. Tales cadenas tienen al menos una a, y las as adicionales pueden ser agrupadas en pares. Además, puede haber un número arbitrario de bs.

$b^*ab^*(ab^*a)^*b^*$	✗	Porque el repetir ab^*a no permite bs entre la segunda a en una repetición y la primer a de la siguiente repetición. De manera que la cadena $abaa\underline{b}aba$ **no** es generada por esta expresión regular.
$b^*a(b^*ab^*ab^*)^*$	✗	Porque **no** permite cadenas de una sola a que terminen en b; por ejemplo, $babb$.
$b^*ab^*(ab^*ab^*)^*$	✓	
$b^*a(b^*ab^*a)^*b^*$	✓	

Ejemplo 1.7 Sea L el lenguaje sobre $\{a, b\}^*$ conteniendo a las cadenas x para las cuales $n_a(x)$ y $n_b(x)$ son ambos pares. Por ejemplo, $bbabbaaa \in L$ y $abaa \notin L$. Encuentre una expresión regular para L.

Vamos a encontrar una primera expresión regular R que genere las cadenas más sencillas o básicas que cumplan la especificación del lenguaje L, para luego aplicar la estrella de Kleene a R con lo que se generarán cadenas más complejas que también cumplan con estar en L. R producirá cadenas de longitud par pues el número de a's y b's es par, así al calcular R^* resultará en cadenas de longitud par (hay que recordar que un número par, más un número par, resulta en un número par). En R está aa y bb. Al analizar las cadenas ab o ba se aprecia que no tienen un número par de a's y b's, para que eso se cumpla después de ab o ba se debe tener ab o ba; por ejemplo, $abba$ y $baab$, ya cumplen con lo que se pide, pero también $abbbaaba$ y $baaaaaab$. Por lo que si la cadena

inicia con ab o ba, le puede seguir cualquier secuencia (incluida la secuencia vacía) de cadenas aa y bb, terminando con ab o ba para así llegar a cadenas que cumplan la especificación; lo anterior no es más que $(ab + ba)(aa + bb)^*(ab + ba)$.

Con esto se tiene que R es $aa + bb + (ab + ba)(aa + bb)^*(ab + ba)$, lo cual genera las cadenas más sencillas o básicas para el lenguaje L. Para obtener una expresión regular que genere todas las cadenas en L simplemente hacemos R^*, es decir, $\big(aa + bb + (ab + ba)(aa + bb)^*(ab + ba)\big)^*$.

Ejemplo 1.8 Una dirección MAC (media access control) es un identificador único asignado a interfaces de red para comunicaciones sobre el segmento físico de la red.

El formato estándar (IEEE 802) para imprimir direcciones MAC-48 en una forma amigable al humano es dar seis grupos de dos dígitos hexadecimales (0 a 9 ó A a F), separados por un guión (por ejemplo, 01-23-45-67-89-AB).

¿Cómo verificar que una cadena `inputString` corresponde a una dirección MAC-48? En PYTHON se tiene lo siguiente:

```
def isMAC48Address(inputString):
    return bool(re.match(('^' + '[\dA-F]{2}-' * 6)[:-1] + '$', inputString))
```

La función `bool([x])` regresa un valor booleano al convertir `x` mediante el procedimiento de prueba de verdad estándar. En dicho procedimiento los objetos `None`, `False`, `0`, `[]`, entre otros, se consideran falsos. Para reconocer expresiones regulares, la función que estamos usando tiene firma `re.match(pattern, string, flags=0)`, como trabaja es que si cero o más caracteres al comienzo de la cadena `string` corresponden a la expresión regular `pattern`, se regresa un objeto de emparejamiento que la función `bool` evalúa a verdadero. Se regresa `None` si la cadena no tiene una correspondencia con la expresión regular. Para más detalles consultar aquí. La expresión regular que le pasamos es `('^' + '[\dA-F]{2}-' * 6)[:-1] + '$'`, como `'+'` es la concatenación de cadenas, podemos descomponer la expresión en dos partes. Primero, `('^' + '[\dA-F]{2}-' * 6)[:-1]` que indica con `'^'` el inicio de la cadena y con `'[\dA-F]{2}-' * 6` seis grupos de dos dígitos hexadecimales que terminan con guión; hay que notar que se tiene un último guión en el sexto grupo, con `[:-1]` no se toma ese último guión. Segundo, `'$'` identifica el fin de la cadena. Por lo que la expresión regular antes mencionada reconoce el patrón para direcciones MAC-48 en PYTHON.

Ejemplo 1.9 Digamos que estamos a cargo de un proyecto web en `php` de gran tamaño que no desarrollamos nosotros. No tenemos bien claro como funciona pero hay un bug que debemos corregir. Investigamos como corregirlo y nos damos cuenta que hay que reemplazar cierto patrón, con su argumento, por otro patrón conservando el mismo argumento, pero ese reemplazo es en cientos de presencias del patrón original en distintos archivos, ¿cómo le hacemos? Usamos expresiones regulares que capturen el patrón a reemplazar y también usamos expresiones regulares para definir el nuevo patrón. La modificación consiste en convertir los caracteres que vengan en `$_GET['var']` mediante la función `htmlspecialchars`, incrementado así la seguridad de un sitio evitando que inyecten código[1]. Un comando que emplea una expresión regular para capturar el patrón a cambiar y realiza el reemplazo en todo el código del proyecto web es:

```
find . -type f -exec sed -i -e
  "s/\$_GET\[\([^]]*\)\]/htmlspecialchars(\$_GET[\1],ENT_QUOTES,'utf-8')/g" {} \;
```

[1] Ataque conocido como *cross site scripting*. Otra función a utilizar para evitar estos ataques es `strip_tags`, que se puede combinar con `htmlspecialchars` como en `$proj= htmlspecialchars(strip_tags($_GET['proj']),ENT_QUOTES,'utf-8');`.

La expresión regular `[^]]*` captura el argumento de `$_GET[...]` que deseamos conservar, ya que `^` representa cualquier símbolo diferente del corchete que cierra, lo cual se escribe entre corchetes, obteniendo `[^]]`. Luego, `[^]]*` reconoce la secuencia de caracteres que conforman el argumento de `$_GET[...]`. Finalmente, la expresión regular `\$_GET\[\([^]]*\)\]` es reemplazada por `htmlspecialchars(\$_GET[\1],ENT_QUOTES,'utf-8')`, donde `\1` hace referencia a `[^]]*`, que representa el argumento en cuestión que se desea conservar.

En el proyecto web, también es aconsejable escapar los caracteres presentes en la instrucción `$_POST['var']` de `php`.

1.1. Expresiones Regulares y Lenguajes de Programación

Un identificador en el lenguaje de programación C es una cadena de longitud al menos 1 que contiene únicamente letras, dígitos y guiones bajo ("_") y no comienza con un dígito.

Sea $l = a + b + c + \ldots + x + y + z + A + B + C + \ldots + X + Y + Z$, y $d = 0 + 1 + \ldots + 8 + 9$. Una expresión regular para los identificadores del lenguaje C es

$$(l + \_)(l + d + \_)^*.$$

El siguiente código en C es válido. Al ejecutarse imprime `'Adding two underscore variables: 4'`.

```c
#include <stdio.h>
int main()
{
   int _ = 3;
   int __ = 1;
   printf("Adding two underscore variables: %d\n",_+__);
   return 0;
}
```

Ahora buscamos una expresión regular para describir el lenguaje de las *literales* numéricas tales como 14, +1, -12, 14.3, -.99, 16., 3E14, -1.00E2, 4.1E-1 y .3E+2. Tales expresiones pueden o no empezar con un signo de más o menos; tendrá uno o más digitos decimales, y posiblemente un punto decimal, y puede o no terminar con una subexpresión comenzando con E. Si se tiene tal expresión, la porción después de la E puede o no comenzar con un signo de más o menos, y tendrá uno o más digitos decimales.

Sea s una variable que representa al signo de más, al signo de menos o a ε; y sea p una variable que denota al punto decimal. Una expresión regular para el lenguaje de las literales numéricas es:

$$s(dd^* + dd^*pd^* + pdd^*)(\varepsilon + \mathrm{E}sdd^*).$$

En algunos lenguajes de programación, no se les permite a las literales numéricas contener un punto decimal a menos que haya al menos un digito decimal en ambos lados del punto. Una expresión regular que incorpore lo anterior es:

$$(sdd^* + sdd^*pdd^*)(\varepsilon + \mathrm{E}sdd^*).$$

2. Equivalencia entre expresiones regulares

Definición 2.1 Dadas dos expresiones regulares α y β decimos que son equivalentes y escribimos $\alpha = \beta$ si y sólo si α y β generan el mismo lenguaje, es decir, si y sólo si $L(\alpha) = L(\beta)$.

Se tienen las siguientes equivalencias entre expresiones regulares que el lector puede verificar.

1. Asociatividad: $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \qquad \alpha(\beta\gamma) = (\alpha\beta)\gamma$
2. Conmutatividad: $\alpha + \beta = \beta + \alpha$
3. Distributividad: $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \qquad (\beta + \gamma)\alpha = \beta\alpha + \gamma\alpha$
4. Elemento identidad: $\alpha + \varnothing = \alpha \qquad \alpha\varepsilon = \alpha$
5. Elemento neutro: $\alpha\varnothing = \varnothing$
6. Idempotencia: $\alpha + \alpha = \alpha \qquad (\alpha^*)^* = \alpha^*$
7. Propiedades de la cerradura de Kleene:
   - 7.1. $\varepsilon^* = \varepsilon \qquad \varnothing^* = \varepsilon$
   - 7.2. $\alpha\alpha^* = \alpha^*\alpha \qquad \alpha^* = \alpha^*\alpha^* \qquad \alpha^* = \varepsilon + \alpha\alpha^*$
   - 7.3. $(\alpha + \beta)^* = (\alpha^* + \beta^*)^* \qquad (\alpha + \beta)^* = (\alpha^*\beta^*)^* = (\alpha^*\beta)^*\alpha^*$
   - 7.4. $(\alpha + \beta)^* = \alpha^*(\beta\alpha^*)^*$
   - 7.5. $(\alpha\beta)^* = \varepsilon + \alpha(\beta\alpha)^*\beta \qquad \alpha(\beta\alpha)^* = (\alpha\beta)^*\alpha$
8. Propiedades condicionales:
   - 8.1. Si $L(\beta) \subseteq L(\alpha)$, entonces $\alpha + \beta = \alpha$.
   - 8.2. Si $L(\alpha^*) \subseteq L(\beta^*)$, entonces $(\alpha + \beta)^* = \beta^*$.

Ejemplo 2.2 Dadas dos expresiones regulares $R = bc + ac^*ac + ac^*c + a$ y $S = (b + ac^*a)c + ac^*$, ¿representan R y S el mismo lenguaje?

Sí, ya que se tienen las siguientes equivalencias:

$$\begin{aligned}
R &= bc + ac^*ac + ac^*c + a \\
&= (b + ac^*a)c + a(c^*c + \varepsilon) && \text{por distributividad} \\
&= (b + ac^*a)c + a(cc^* + \varepsilon) && \text{propiedad 7.2. } (\alpha\alpha^* = \alpha^*\alpha) \\
&= (b + ac^*a)c + a(\varepsilon + cc^*) && \text{por conmutatividad} \\
&= (b + ac^*a)c + ac^* && \text{por propiedad 7.2. } (\alpha^* = \varepsilon + \alpha\alpha^*) \\
&= S
\end{aligned}$$

Ejemplo 2.3 Demuestre que $\Big((a+b)a^*b+b^*a\Big)^* = \Big(\big((a+b)a^*b\big)^*\big(b^*a\big)^*\Big)^*$. Lo anterior se obtiene al aplicar la propiedad 7.3. $(\alpha + \beta)^* = (\alpha^*\beta^*)^*$, con $\alpha = (a + b)a^*b$ y $\beta = b^*a$.

Ejemplo 2.4 Demuestre que las expresiones regulares $R = (a^*(b+c)^* + b^*)^*$ y $S = (a+b+c)^*$ son equivalentes.

$$\begin{aligned} S &= (a+b+c)^* \\ &= (a^*(b+c)^*)^* && \text{propiedad 7.3. } ((\alpha+\beta)^* = (\alpha^*\beta^*)^*) \\ &= (b^* + a^*(b+c)^*)^* && \text{propiedad 8.2. con } \alpha = b^* \text{ y } \beta = a^*(b+c)^* \\ &= (a^*(b+c)^* + b^*)^* && \text{por conmutatividad} \\ &= R \end{aligned}$$

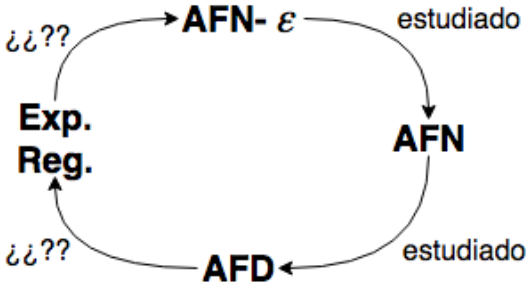
Ejercicios

- Encuentre una expresión regular para el conjunto de representaciones binarias de enteros que son potencia de 4. Las representaciones generadas por la expresión regular deben ser únicas, es decir, no se deben aceptar representaciones con ceros a la izquierda del primer 1.
- Considere las expresiones regulares $R = 0^* + 1^*$ y $S = 01^* + 10^* + 1^*0 + (0^*1)^*$.
  - Encuentre una cadena en $L(R) \setminus L(S)$.
  - Encuentre una cadena en $L(R) \cap L(S)$.
  - Encuentre una cadena en $\{0,1\}^*$ que no pertenezca a $L(R)$ ni a $L(S)$.
- ¿Son las expresiones regulares $R = \varepsilon + (0+1)^*1$ y $S = (0^*1)^*$ equivalentes?
- ¿Son las expresiones regulares $R = (0+1)^*1 + 0^*$ y $S = (1+0)(0^*1)^*$ equivalentes?

3. Relación entre las expresiones regulares y los autómatas finitos

Hemos visto la relación entre autómatas finitos, ahora nos preguntamos como se relacionan éstos con las expresiones regulares.

Tenemos lo siguiente:



3.1. Teorema de Kleene I

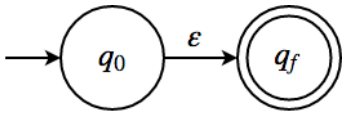
Teorema 3.1 **[ER $\Rightarrow$ AF]** Para cualquier lenguaje regular $L(R)$ definido por la expresión regular R, existe un AFN-ε N tal que $L(R) = L(N)$.

Demostración. Para la demostración utilizaremos los AFNs-ε *nobles*, que únicamente tienen un estado final. Un resultado sencillo es demostrar que todo AFN-ε puede convertirse a un AFN-ε *noble*, pues es posible generar un nuevo estado final al que se llega desde los estados finales originales con movimientos ε, y tales estados dejan de ser finales.

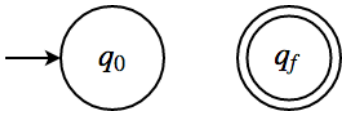
La demostración del teorema es por inducción estructural sobre R.

Base. Es sencillo obtener un AFN-ε *noble* para ε, $\varnothing$ y a, con $a \in \Sigma$. Los autómatas que buscamos son:

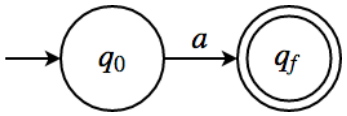
- N_ε



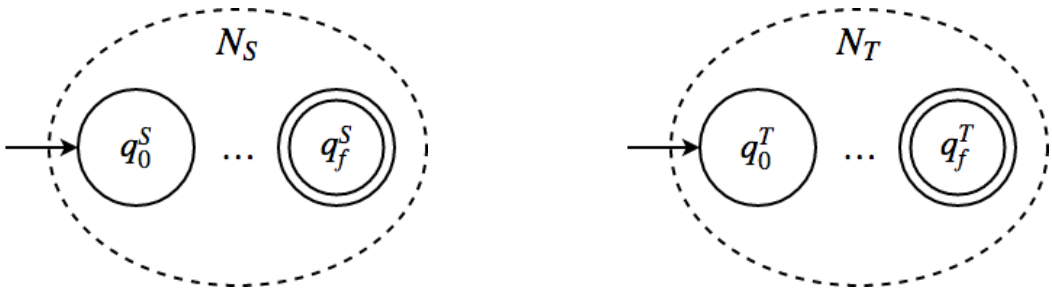
- $N_\varnothing$



- N_a

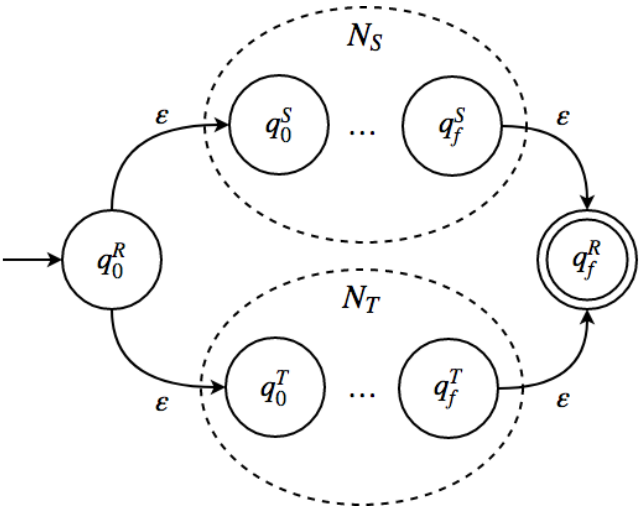


Hipótesis de inducción. Suponemos que tenemos AFNs-ε *nobles* N_S y N_T que reconocen los lenguajes $L(S)$ y $L(T)$, respectivamente, con S y T expresiones regulares. Los diagramas de estado para N_S y N_T son:



Paso inductivo. Por demostrar que existe un AFN-ε *noble* para el lenguaje $L(R)$, donde R puede ser $S + T$, $S \cdot T$ y S^*.

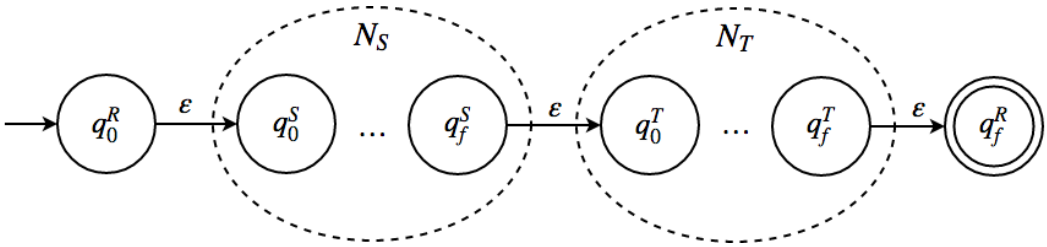
- $R = S + T$. Definimos N_R como:



Notemos que q_f^S y q_f^T dejan de ser estados finales. Además,

$$\begin{aligned} L(N_R) &= L(N_S) \cup L(N_T) \\ &\overset{\text{HI}}{=} L(S) \cup L(T) \\ &= L(S + T) \\ &= L(R). \end{aligned}$$

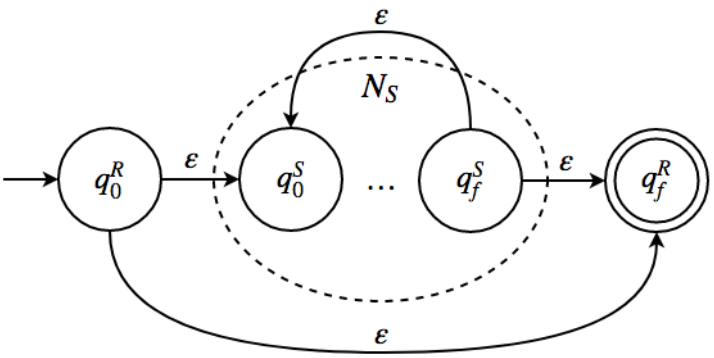
- $R = S \cdot T$. Definimos N_R como:



Se tiene que:

$$\begin{aligned} L(N_R) &= L(N_S) \cdot L(N_T) \\ &\overset{\text{HI}}{=} L(S) \cdot L(T) \\ &= L(S \cdot T) \\ &= L(R). \end{aligned}$$

- $R = S^*$. Definimos N_R como:



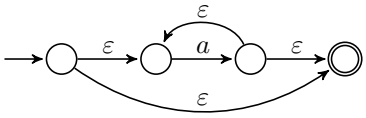
Se tiene que:

$$\begin{aligned} L(N_R) &= L(N_S)^* \\ &\overset{\text{HI}}{=} L(S)^* \\ &= L(S^*) \\ &= L(R). \end{aligned}$$

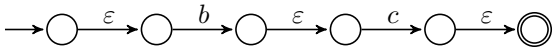
⊣

Ejemplo 3.2 Empleando el método arriba descrito construya un $AFN-\varepsilon$ *noble* para la expresión regular $a^* + bc$.

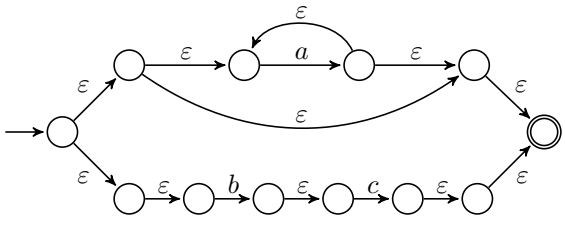
El $AFN-\varepsilon$ *noble* para a^* es,



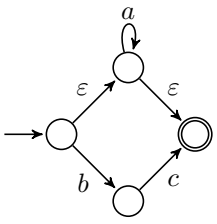
El $AFN-\varepsilon$ *noble* para bc es,



Por lo que el $AFN-\varepsilon$ *noble* para $a^* + bc$ es,



Lo simplificamos como,



3.2. Teorema de Kleene II

Teorema 3.3 [**AF** $\Rightarrow$ **ER**] Dado un autómata finito M existe una expresión regular α tal que $L(M) = L(\alpha)$.

Demostración. Usaremos el método de ecuaciones características empleando el lema de Arden. $\dashv$

3.2.1. Ecuaciones de lenguajes

Definición 3.4 Sean $A, B \subseteq \Sigma^*$ dos lenguajes y X una variable. Una *ecuación lineal derecha* para X es una expresión de la forma:

$$X = AX + B$$

Análogamente, una *ecuación lineal izquierda* es una expresión de la forma

$$X = XA + B$$

Aquí, el símbolo + denota la unión de conjuntos.

3.2.2. Lema de Arden

Lema 3.5 *Sean $A, B \subseteq \Sigma^*$ dos lenguajes y $X = AX + B$ una ecuación lineal derecha.*

1. *A^*B es una solución de la ecuación.*
2. *Si L es otra solución, entonces $A^*B \subseteq L$, i.e. A^*B es la solución mínima.*
3. *Si $\varepsilon \notin A$, entonces A^*B es la única solución.*

Demostración.

1. $A(A^*B) + B \overset{\text{Asoc.}}{=} (AA^*)B + B \overset{\text{Distr.}}{=} (AA^* + \varepsilon)B \overset{\text{Prop. 7.2.}}{=} A^*B$
2. Si L es otra solución, entonces $L = AL + B$. Demostraremos por inducción matemática sobre n que

$$L = A^{n+1}L + \sum_{0 \le i \le n} A^i B \qquad \forall n \in \mathbb{N}$$

Base $n = 0$

$$\begin{aligned} L &\overset{L \text{ es sol.}}{=} AL + B \\ &= AL + A^0 B \\ &= A^{0+1}L + \sum_{0 \le i \le 0} A^i B \end{aligned}$$

Hipótesis de Inducción Suponemos que para $n = k$ se cumple que $L = A^{k+1}L + \sum_{0 \le i \le k} A^i B$

Paso inductivo Demostrar que se vale para $n = k+1$, es decir, $L = A^{(k+1)+1}L + \sum_{0 \leq i \leq k+1} A^i B$

$$\begin{aligned} L &\overset{\text{HI}}{=} A^{k+1}L + \sum_{0 \leq i \leq k} A^i B \\ &\overset{L \text{ es sol.}}{=} A^{k+1}(AL + B) + \sum_{0 \leq i \leq k} A^i B \\ &= A^{(k+1)+1}L + A^{k+1}B + \sum_{0 \leq i \leq k} A^i B \\ &= A^{(k+1)+1}L + \sum_{0 \leq i \leq k+1} A^i B \end{aligned}$$

Por consiguiente, $\forall n \in \mathbb{N},\ L = A^{n+1}L + \sum_{0 \leq i \leq n} A^i B$. Luego, $\forall n \in \mathbb{N},\ A^n B \subseteq L$. Así, $A^*B \subseteq L$.

3. Sea L una solución. Veremos que $L = A^*B$. Por el punto 2, $A^*B \subseteq L$, falta ver que $L \subseteq A^*B$. Sea $u \in L$ con $|u| = m$. También por el punto 2, L puede ser descrito como $L = A^{m+1}L + \sum_{0 \leq i \leq m} A^i B$, de modo que $u \in A^{m+1}L + \sum_{0 \leq i \leq m} A^i B$. Como $\varepsilon \notin A$, la cadena más pequeña que puede estar en A es de tamaño al menos 1, por lo que en A^{m+1} la cadena más pequeña es de tamaño al menos $m+1$. Luego, $u \notin A^{m+1}L$ ya que las cadenas en $A^{m+1}L$ son de tamaño al menos $m+1$ y $|u| = m$. Entonces $u \in \sum_{0 \leq i \leq m} A^i B$, esto implica que $u \in A^*B$. Así que $L \subseteq A^*B$; por lo tanto $L = A^*B$, es decir, A^*B es la única solución.

⊣

3.2.3. El sistema de ecuaciones de un AFD

Dado un AFN definimos un sistema de ecuaciones de lenguajes, cuya solución, dada por el lema de Arden, nos llevará a una expresión regular para el lenguaje correspondiente.

Definición 3.6 Dado un AFN $N = (Q, \Sigma, \delta, q_0, F)$ tal que $Q = \{q_0, \ldots, q_n\}$. Definimos los conjuntos siguientes:

- El conjunto de cadenas que se aceptan desde el estado q_i, para cualquier $0 \leq i \leq n$:

$$L_i = \{w \in \Sigma^* \mid \widehat{\delta}(q_i, w) \cap F \neq \varnothing\}$$

- El conjunto de símbolos de Σ tal que existe una transición del estado q_i al estado q_j, para cualesquiera $0 \leq i, j \leq n$

$$X_{i,j} = \{a \in \Sigma \mid q_j \in \delta(q_i, a)\}$$

- El conjunto auxiliar Y_i que indica si ε es aceptada desde q_i

$$Y_i = \begin{cases} \{\varepsilon\} & \text{si } q_i \in F \\ \varnothing & \text{en otro caso} \end{cases}$$

Observe que L_0 es el lenguaje generado por N, i.e., $L_0 = L(N)$.

No es sencillo calcular directamente los conjuntos L_i; sin embargo, es posible hacerlo resolviendo el sistema de ecuaciones de lenguajes dado por la siguiente proposición.

Proposición 3.7 *Los conjuntos L_i, $X_{i,j}$ y Y_i recién definidos se relacionan como sigue:*

$$L_i = \sum_{j=0}^{n} X_{i,j} L_j + Y_i, \qquad 0 \leq i \leq n$$

Demostración. Recordemos una variante de la $\widehat{\delta}$: $\boxed{\widehat{\delta}(q, \varepsilon) = q \text{ y } \widehat{\delta}(q, aw) = \bigcup_{p_i \in \delta(q,a)} \widehat{\delta}(p_i, w)}$.

$\subseteq$) Sea $u \in L_i$. Analizaremos los casos para u:

- $u = \varepsilon \in L_i$, i.e. $\widehat{\delta}(q_i, \varepsilon) = \{q_i\} \cap F \neq \varnothing$, así $q_i \in F$. Por lo que $Y_i = \{\varepsilon\}$. Luego $u \in Y_i$, lo que implica que $u \in \sum_{j=0}^{n} X_{i,j} L_j + Y_i$.
- $u = aw \in L_i$, i.e. $\widehat{\delta}(q_i, aw) = \bigcup_{p_j \in \delta(q_i,a)} \widehat{\delta}(p_j, w) \cap F \neq \varnothing$. De modo que para alguna $q_k \in \delta(q_i, a)$ con $0 \leq k \leq n$ se tiene $\widehat{\delta}(q_k, w) \cap F \neq \varnothing$. Entonces $a \in X_{i,k}$ y $w \in L_k$. Por lo tanto, $aw \in X_{i,k} L_k$, de donde tenemos que $u \in \sum_{j=0}^{n} X_{i,j} L_j + Y_i$.

$\supseteq$) Sea $u \in \sum_{j=0}^{n} X_{i,j} L_j + Y_i$. Se tienen los siguientes casos:

- $u \in Y_i$, $u = \varepsilon$ porque es el único elemento que puede tener $Y_i = \{\varepsilon\}$. Así, $q_i \in F$, como q_i es final, entre las cadenas que se aceptan desde q_i está ε, i.e., $u \in L_i$.
- $u \in \sum_{j=0}^{n} X_{i,j} L_j$. Así u se puede descomponer en $u = aw$ con $a \in X_{i,k}$ y $w \in L_k$ para alguna k, $0 \leq k \leq n$. Vemos que $q_k \in \delta(q_i, a)$ y $\widehat{\delta}(q_k, w) \cap F \neq \varnothing$. Por lo que

  $$\widehat{\delta}(q_i, aw) = \bigcup_{p_j \in \delta(q_i,a)} \widehat{\delta}(p_j, w) \cap F \neq \varnothing,$$

  es decir, $\widehat{\delta}(q_i, u) \cap F \neq \varnothing$. Por consiguiente, $u \in L_i$.

$\dashv$

La relación proporcionada genera el llamado sistema de ecuaciones características de una AFN. Para resolver estas ecuaciones basta aplicar el lema de Arden. Dado un AFN N, para obtener una expresión regular para $L(N)$ se procede como sigue:

- Construir los conjuntos $X_{i,j}$ y Y_i.
- Resolver en orden descendiente las ecuaciones características para $L_n, L_{n-1}, \ldots, L_0$ mediante el lema de Arden. Es decir, resolver primero L_n hasta llegar a L_0.

- La solución para L_0 genera una expresión regular para $L(N)$.

Ejemplo 3.8 Mediante el lema de Arden encuentre una expresión regular para el autómata de la Figura 1:

Los conjuntos $X_{i,j} = \{a \in \Sigma \,|\, q_j \in \delta(q_i, a)\}$ y $Y_i = \begin{cases} \{\varepsilon\} & \text{si } q_i \in F \\ \varnothing & \text{en otro caso} \end{cases}$, con $0 \leq i \leq n$, son los siguientes. Observe que para nuestro ejemplo $n = 2$.

$$\begin{array}{lll} X_{0,0} = \{a\} & X_{1,0} = \{a\} & X_{2,0} = \{a\} \\ X_{0,1} = \{b\} & X_{1,1} = \varnothing & X_{2,1} = \{b\} \\ X_{0,2} = \varnothing & X_{1,2} = \{b\} & X_{2,2} = \varnothing \\ Y_0 = \{\varepsilon\} & Y_1 = \{\varepsilon\} & Y_2 = \varnothing \end{array}$$

Sabemos que $L_i = \sum_{j=0}^{n} X_{i,j} L_j + Y_i$. De modo que las ecuaciones características son:

$$L_2 = X_{2,0}L_0 + X_{2,1}L_1 + X_{2,2}L_2 + Y_2 = aL_0 + bL_1$$

$$L_1 = X_{1,0}L_0 + X_{1,1}L_1 + X_{1,2}L_2 + Y_1 = aL_0 + bL_2 + \varepsilon$$

$$L_0 = X_{0,0}L_0 + X_{0,1}L_1 + X_{0,2}L_2 + Y_0 = aL_0 + bL_1 + \varepsilon$$

Por el lema de Arden, la solución a $X = AX + B$ es $X = A^*B$. Aplicamos este lema para solucionar las ecuaciones L_i del índice mayor al menor.

Tomando $L_2 = aL_0 + bL_1$ y acoplándolo con $L_2 = AL_2 + B$, tenemos que $A = \varnothing$ y $B = aL_0 + bL_1$, de manera que la solución es $L_2 = A^*B = \varnothing^*(aL_0 + bL_1) = aL_0 + bL_1$.

Sustituimos $L_2 = aL_0 + bL_1$ en la ecuación para L_1. Obtenemos:

$$L_1 = aL_0 + b(aL_0 + bL_1) + \varepsilon = bbL_1 + (a + ba)L_0 + \varepsilon,$$

acoplando $L_1 = bbL_1 + (a + ba)L_0 + \varepsilon$ con $L_1 = AL_1 + B$, tenemos que $A = bb$ y $B = (a + ba)L_0 + \varepsilon$. Por el lema de Arden, $L_1 = A^*B = (bb)^*((a + ba)L_0 + \varepsilon)$.

Finalmente, sustituimos L_1 en L_0. Llegamos a

$$L_0 = aL_0 + b\Big((bb)^*((a + ba)L_0 + \varepsilon)\Big) + \varepsilon = \Big(a + b(bb)^*(a + ba)\Big)L_0 + \Big(b(bb)^* + \varepsilon\Big),$$

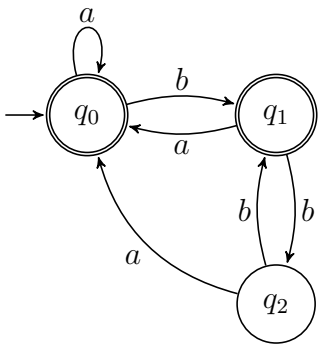


Figura 1: Autómata finito del cual obtener una expresión regular que genere el mismo lenguaje.

acoplando lo anterior con $L_0 = AL_0 + B$, tenemos que $A = a + b(bb)^*(a + ba)$ y $B = b(bb)^* + \varepsilon$. Por el lema de Arden, la solución es

$$L_0 = \Big(a + b(bb)^*(a + ba)\Big)^* \Big(b(bb)^* + \varepsilon\Big).$$

Esta es la expresión regular para el autómata finito de la Figura 1. Se puede simplificar aplicando las equivalencias de la Sección 2.

Ejemplo 3.9 Mediante el lema de Arden encuentre una expresión regular para el autómata de la Figura 2

Los conjuntos $X_{i,j}$ y Y_i, con $0 \leq i \leq n$, son los siguientes. Observe que para nuestro ejemplo $n = 2$.

$$\begin{array}{lll} X_{0,0} = \varnothing & X_{1,0} = \varnothing & X_{2,0} = \{b\} \\ X_{0,1} = \{a, b\} & X_{1,1} = \{a\} & X_{2,1} = \{a\} \\ X_{0,2} = \varnothing & X_{1,2} = \{b\} & X_{2,2} = \varnothing \\ Y_0 = \varnothing & Y_1 = \{\varepsilon\} & Y_2 = \varnothing \end{array}$$

Vemos que $X_{0,1} = \{a, b\}$ puede escribirse como $X = (a+b)$. También sabemos que $L_i = \sum_{j=0}^{n} X_{i,j}L_j + Y_i$. De modo que las ecuaciones características son:

$$L_2 = X_{2,0}L_0 + X_{2,1}L_1 + X_{2,2}L_2 + Y_2 = bL_0 + aL_1$$

$$L_1 = X_{1,0}L_0 + X_{1,1}L_1 + X_{1,2}L_2 + Y_1 = aL_1 + bL_2 + \varepsilon$$

$$L_0 = X_{0,0}L_0 + X_{0,1}L_1 + X_{0,2}L_2 + Y_0 = (a + b)L_1$$

Por el lema de Arden, la solución a $X = AX+B$ es $X = A^*B$. Aplicamos este lema para solucionar las ecuaciones L_i del índice mayor al menor.

Tomando $L_2 = bL_0+aL_1$ y acoplándolo con $L_2 = AL_2+B$, tenemos que $A = \varnothing$ y $B = bL_0+aL_1$, de manera que la solución es $L_2 = A^*B = \varnothing^*(bL_0 + aL_1) = bL_0 + aL_1$.

Sustituimos $L_2 = bL_0 + aL_1$ en la ecuación para L_1. Obtenemos:

$$L_1 = aL_1 + b(bL_0 + aL_1) + \varepsilon = (a + ba)L_1 + bbL_0 + \varepsilon,$$

acoplando $L_1 = (a+ba)L_1 + bbL_0 + \varepsilon$ con $L_1 = AL_1 + B$, tenemos que $A = a + ba$ y $B = bbL_0 + \varepsilon$. Por el lema de Arden, $L_1 = A^*B = (a + ba)^*(bbL_0 + \varepsilon)$.

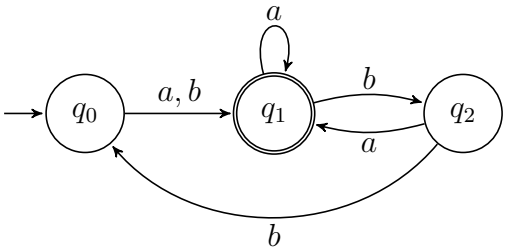


Figura 2: Autómata finito del que se busca una expresión regular que genere el mismo lenguaje.

Finalmente, sustituimos L_1 en L_0. Llegamos a

$$L_0 = (a+b)\Big((a+ba)^*(bbL_0+\varepsilon)\Big) = \Big((a+b)(a+ba)^*bb\Big)L_0 + \Big((a+b)(a+ba)^*\Big),$$

acoplando lo anterior con $L_0 = AL_0+B$, tenemos que $A = (a+b)(a+ba)^*bb$ y $B = (a+b)(a+ba)^*$. Por el lema de Arden, la solución es

$$L_0 = \Big((a+b)(a+ba)^*bb\Big)^*\Big((a+b)(a+ba)^*\Big).$$

Esta es la expresión regular buscada.

Hemos comenzado a manipular expresiones de la teoría de lenguajes formales de un modo algebraico. En efecto, las leyes algebraicas que gobiernan como + y $\cdot$ funcionan se parecen mucho a la manera en la que + y $\times$ funcionan en aritmética (aunque la ley conmutativa no se satisface para $\cdot$). ¿Qué hay acerca de *? En aritmética tenemos la siguiente igualdad:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$$

válida para cualquier $|x| < 1$. Dicha ecuación sugiere que la operación * en teoría de lenguajes formales es el análogo al operador *casi* inverso $x \mapsto \frac{1}{1-x}$. Es decir, la solución A^*B para la ecuación $X = AX + B$ corresponde a la solución numérica $\frac{b}{1-a}$ de la ecuación $x = ax + b$ sobre los números reales.