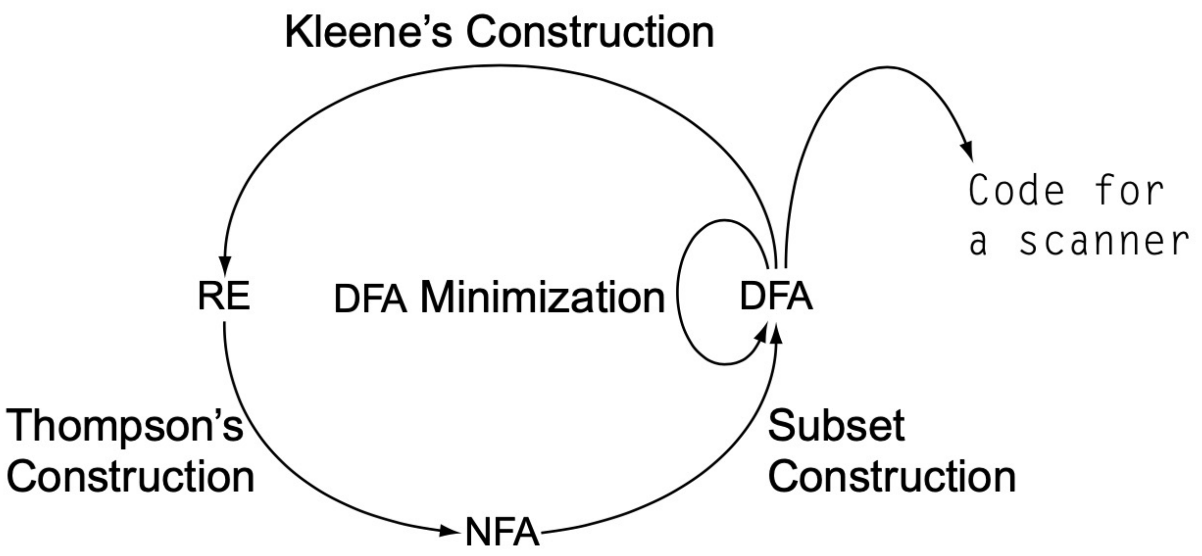
TRANSFORMACIONES ENTRE EXPRESIONES REGULARES Y AUTÓMATAS FINITOS

Prof. Maureen Murillo
Teoría de la Computación
Escuela de Ciencias de la Computación e Informática
Universidad de Costa Rica

Presentación basada e imágenes tomadas de: Cooper and Torczon, Engineering a Compiler

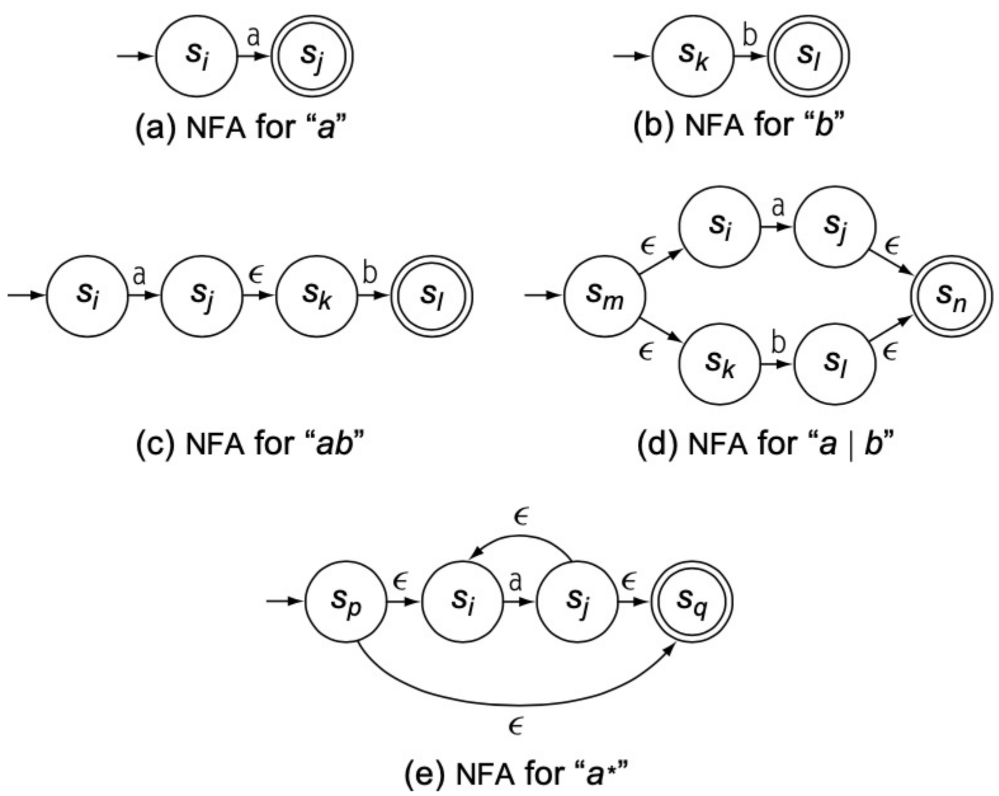
EQUIVALENCIA ENTRE REGEX, NFA Y DFA

Alternativa: algoritmo de eliminación de estados



■ **FIGURE 2.3** The Cycle of Constructions.

CONSTRUCCIÓN DE THOMPSON: DE REGEX A NFA

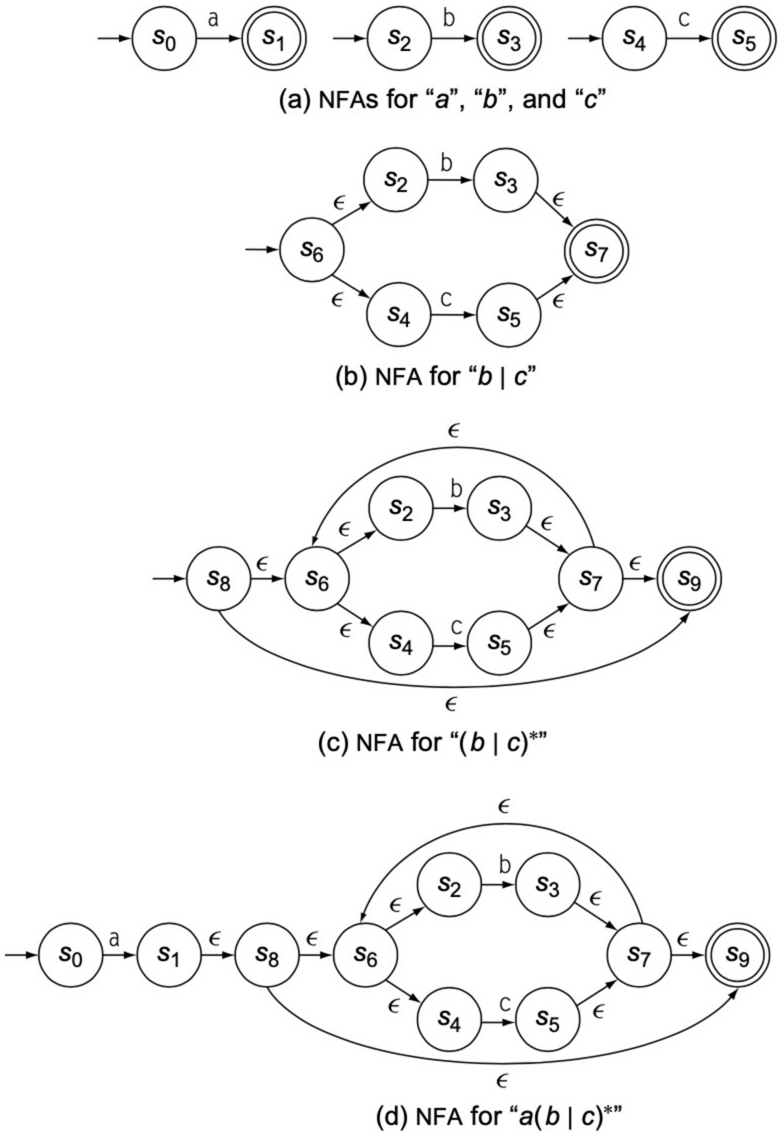


■ **FIGURE 2.4** Trivial NFAs for Regular Expression Operators.

Caso trivial

CONSTRUCCIÓN DE THOMPSON: EJEMPLO

Caso: a(b|c)*



■ **FIGURE 2.5** Applying Thompson's Construction to $a(b|c)^*$.

CONSTRUCCIÓN POR SUBCONJUNTOS: DE NFA A DFA

```
q0 ← ε-closure({n0});
Q ← q0;
WorkList ← {q0};

while (WorkList ≠ Ø ) do
    remove q from WorkList;

    for each character c ∈ Σ do
        t ← ε-closure(Delta(q,c));
        T[q,c] ← t;

        if t ∉ Q then
            add t to Q and to WorkList;
    end;

end;
```

■ **FIGURE 2.6** The Subset Construction.

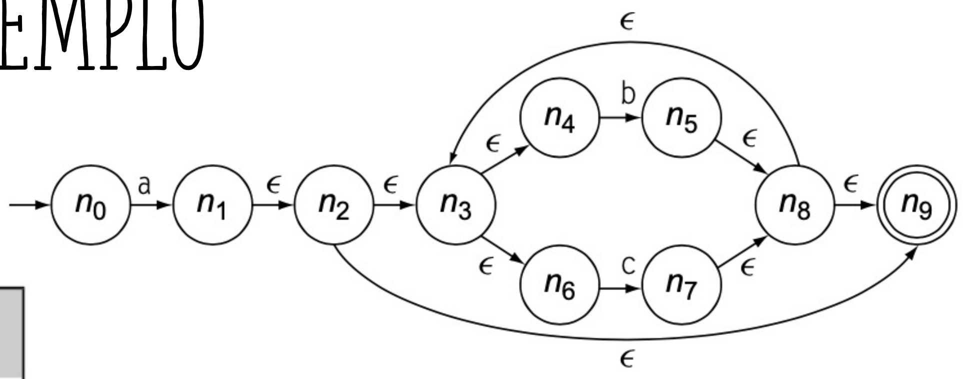
Algoritmo

CONSTRUCCIÓN POR SUBCONJUNTOS: EJEMPLO

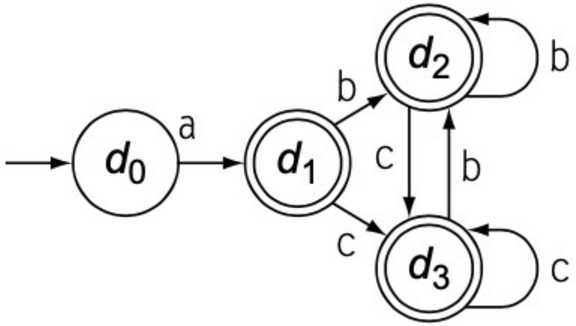
Caso: a(b|c)*

Set Name	DFA States	NFA States	ϵ-closure(Delta(**q**,*))		
			a	b	c
q_0	d_0	n_0	$\{n_1, n_2, n_3, n_4, n_6, n_9\}$	– none –	– none –
q_1	d_1	$\{n_1, n_2, n_3, n_4, n_6, n_9\}$	– none –	$\{n_5, n_8, n_9, n_3, n_4, n_6\}$	$\{n_7, n_8, n_9, n_3, n_4, n_6\}$
q_2	d_2	$\{n_5, n_8, n_9, n_3, n_4, n_6\}$	– none –	q_2	q_3
q_3	d_3	$\{n_7, n_8, n_9, n_3, n_4, n_6\}$	– none –	q_2	q_3

(b) Iterations of the Subset Construction



(a) NFA for "a(b | c)*" (With States Renumbered)



(a) Resulting DFA

ALGORITMO DE HOPCROFT: MINIMIZACIÓN DE DFA

```
T ← {D_A,  {D − D_A} };
P ← Ø

while (P ≠ T) do
    P ← T;
    T ← Ø;

    for each set p ∈ P do
        T ← T ∪ Split(p);
    end;

end;
```

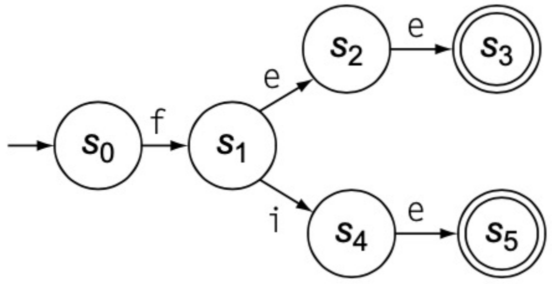
```
Split(S) {
    for each c ∈ Σ do
        if c splits S into s1 and s2
            then return {s1,s2};
    end;

    return S;
}
```

■ **FIGURE 2.9** DFA Minimization Algorithm.

Algoritmo

ALGORITMO DE HOPCROFT: EJEMPLO 1

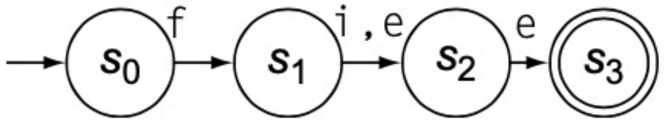


(a) DFA for "*fee* | *fie*"

La idea es detectar cuando dos estados son equivalentes

Step	Current Partition	Examines		
		Set	Char	Action
0	$\{\{s_3,s_5\},\{s_0,s_1,s_2,s_4\}\}$	—	—	—
1	$\{\{s_3,s_5\},\{s_0,s_1,s_2,s_4\}\}$	$\{s_3,s_5\}$	*all*	*none*
2	$\{\{s_3,s_5\},\{s_0,s_1,s_2,s_4\}\}$	$\{s_0,s_1,s_2,s_4\}$	e	*split* $\{s_2,s_4\}$
3	$\{\{s_3,s_5\},\{s_0,s_1\},\{s_2,s_4\}\}$	$\{s_0,s_1\}$	f	*split* $\{s_1\}$
4	$\{\{s_3,s_5\},\{s_0\},\{s_1\},\{s_2,s_4\}\}$	*all*	*all*	*none*

(b) Critical Steps in Minimizing the DFA

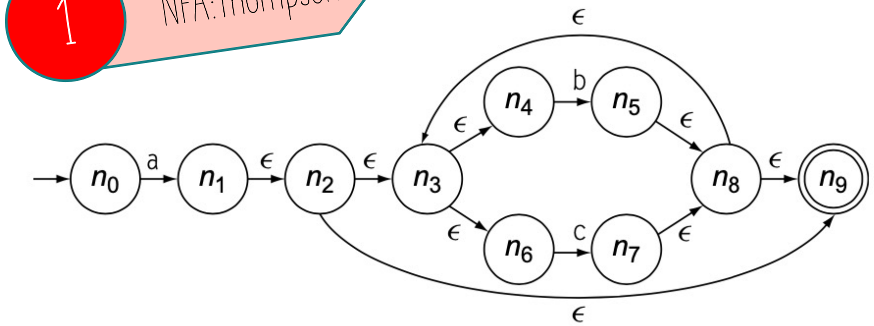


(c) The Minimal DFA (States Renumbered)

ALGORITMO DE HOPCROFT: EJEMPLO 2

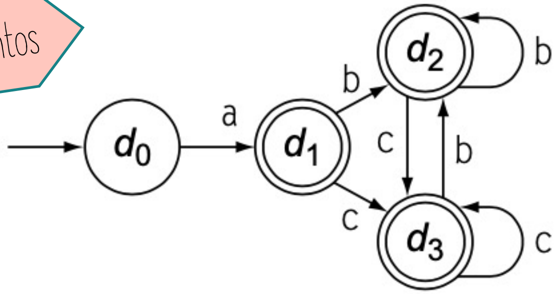
Caso: $a(b|c)^*$

1 NFA:Thompson



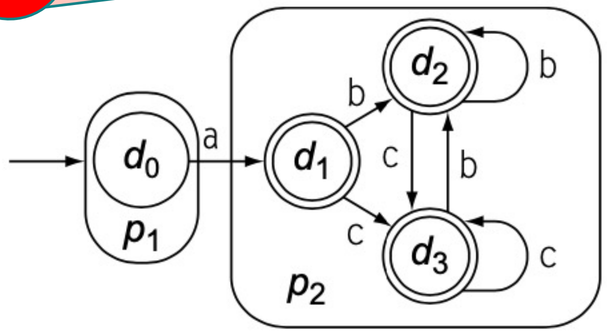
(a) NFA for “$a(b \mid c)^*$” (With States Renumbered)

2 DFA:Subconjuntos



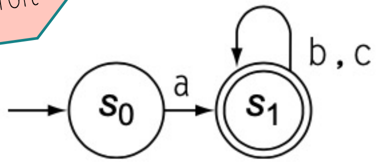
(a) Original DFA

3A Minimización:Hopcroft



(b) Initial Partition

3B Minimización:Hopcroft



CONSTRUCCIÓN DE KLEENE: DE DFA A REGEX

for i = 0 to $|D|-1$

  for j = 0 to $|D|-1$

    $R_{ij}^{-1} = \{\, a \mid \delta(d_i, a) = d_j \}$

    if (i = j) then

      $R_{ij}^{-1} = R_{ij}^{-1} \mid \{\, \epsilon \,\}$

for k = 0 to $|D|-1$

  for i = 0 to $|D|-1$

    for j = 0 to $|D|-1$

      $R_{ij}^{k} = R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1} \mid R_{ij}^{k-1}$

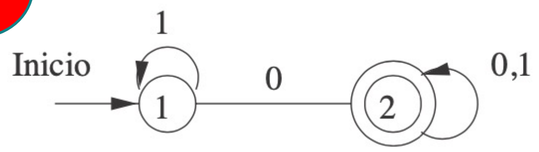
$L = |_{s_j \in D_A} \quad R_{0j}^{|D|-1}$

■ **FIGURE 2.18** Deriving a Regular Expression from a DFA.

Algoritmo

CONSTRUCCIÓN DE KLEENE: EJEMPLO

1



DFA que acepta todas las cadenas que tienen al menos un 0

2

$R_{11}^{(0)}$	$\varepsilon + \mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$(\varepsilon + \mathbf{0} + \mathbf{1})$

3

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)} (R_{11}^{(0)})^* R_{1j}^{(0)}$$

	Por sustitución directa	Simplificada
$R_{11}^{(1)}$	$\varepsilon + \mathbf{1} + (\varepsilon + \mathbf{1})(\varepsilon + \mathbf{1})^*(\varepsilon + \mathbf{1})$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0} + (\varepsilon + \mathbf{1})(\varepsilon + \mathbf{1})^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\varepsilon + \mathbf{1})^*(\varepsilon + \mathbf{1})$	$\emptyset$
$R_{22}^{(1)}$	$\varepsilon + \mathbf{0} + \mathbf{1} + \emptyset(\varepsilon + \mathbf{1})^*\mathbf{0}$	$\varepsilon + \mathbf{0} + \mathbf{1}$

Figura 3.5. Expresiones regulares para caminos que sólo pueden pasar a través del estado 1.

4

$$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} (R_{22}^{(1)})^* R_{2j}^{(1)}$$

	Por sustitución directa	Simplificada
$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\varepsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\varepsilon + \mathbf{0} + \mathbf{1})^*(\varepsilon + \mathbf{0} + \mathbf{1})$	$\mathbf{1}^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset + (\varepsilon + \mathbf{0} + \mathbf{1})(\varepsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$\emptyset$
$R_{22}^{(2)}$	$\varepsilon + \mathbf{0} + \mathbf{1} + (\varepsilon + \mathbf{0} + \mathbf{1})(\varepsilon + \mathbf{0} + \mathbf{1})^*(\varepsilon + \mathbf{0} + \mathbf{1})$	$(\mathbf{0} + \mathbf{1})^*$

Figura 3.6. Expresiones regulares para los caminos que puede pasar por cualquier estado.

5

Resultado final: unión de todas las expresiones en las que el primer estado sea el estado inicial y el segundo estado sea el estado de aceptación. En este caso R^2_{12}, o sea, $1^*0(0|1)^*$

Ejemplo 3.5 tomado de: Hopcroft, Motwani and Ullman; Automata Theory, Languages, and Computation

ELIMINACIÓN DE ESTADOS: DE DFA A REGEX

El algoritmo de Kleene para transformar un DFA en una Regex resulta costoso

IDEA: considerar expresiones regulares sobre las aristas, y sacar los estados intermedios mientras mantenemos coherentes las etiquetas de las aristas

ELIMINACIÓN DE ESTADOS: EJEMPLO

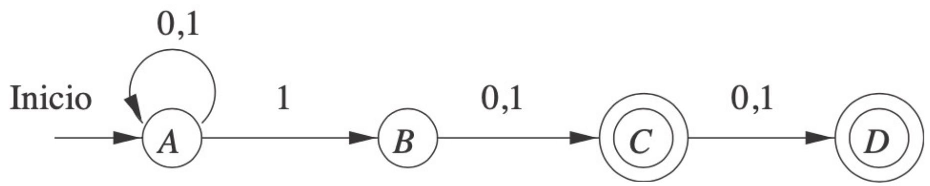


Figura 3.11. Un AFN que acepta cadenas que tienen un 1 a dos o tres posiciones respecto del final.

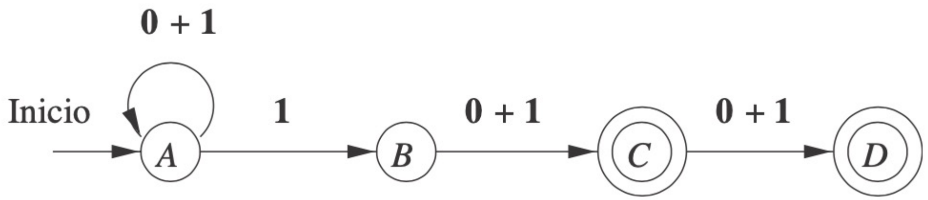


Figura 3.12. El autómata de la Figura 3.11 con expresiones regulares como etiquetas.

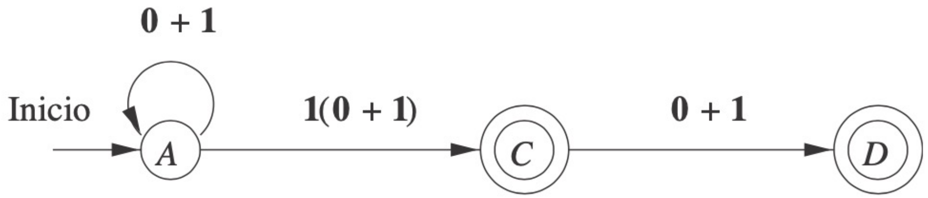


Figura 3.13. Eliminación del estado *B*.

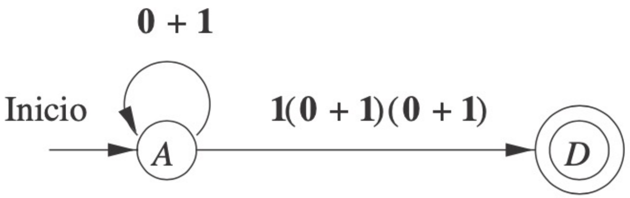


Figura 3.14. Autómata de dos estados con los estados *A* y *D*.

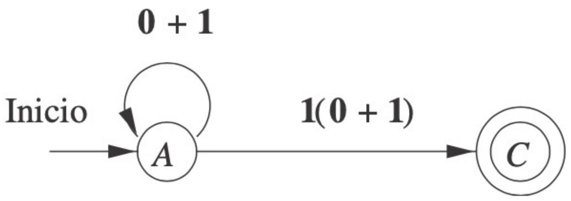


Figura 3.15. Autómata de dos estados resultado de la eliminación de *D*.

Resultado final: unir los dos últimos autómatas

(0|1)*1(0|1)(0|1) | (0|1)*1(0|1)

Ejemplo 3.6 tomado de: Hopcroft, Motwani and Ullman; Automata Theory, Languages, and Computation

Fin