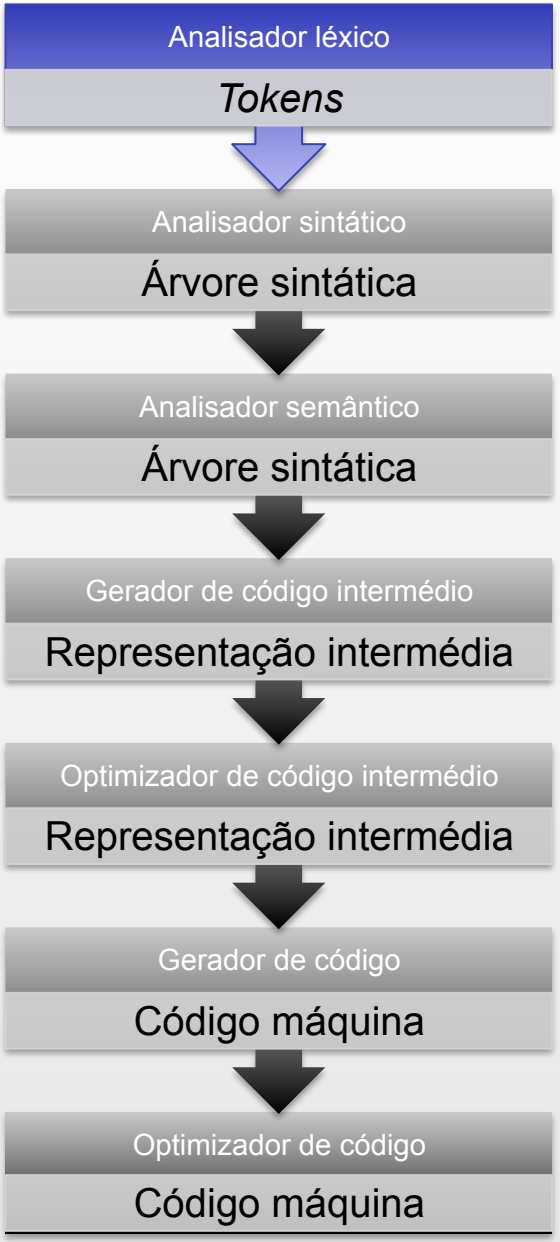
Compiladores
ANÁLISE LEXICAL

Etapa:
Analisador léxico

Também conhecido como *scanner*.



TOKENS

Tokens

A análise lexical transforma uma sequência de caracteres numa sequência de *tokens* ou símbolos terminais.

Na maior parte das linguagens, espaços em branco (incluindo tabulações e quebras de linha), assim como comentários, são irrelevantes para a linguagem, pelo que é nesta fase que nos livramos deles.

```
For i:=0 To len-1 Do
    Begin
        If array[i]<>10 Then
            array[i] := 0; (* apaga *)
    End;
```

‘for’, ‘i’, ‘:=’, ‘0’, ‘to’, ‘len’, ‘-’, ‘1’, ‘do’, ‘begin’, ‘if’, ‘array’, ‘[‘, ‘i’, ‘]’, ‘<>’, ’10’, ‘then’, ‘array’, ‘[‘, ‘i’, ‘]’, ‘:=’, ‘0’, ‘;’, ‘end’, ‘;’

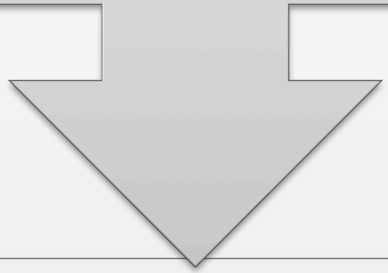
Tokens são números

Para facilidade de utilização, estes *tokens* são definidos como constante numéricas (enums ou #defines, em C).

Precisamos de generalizar (abstrair) certos *tokens:* identificadores (nomes de variáveis ou funções), inteiros, reais e *strings*.

No entanto, ao fazê-lo não podemos perder o valor subjacente, original.

```
For i:=0 To len-1 Do
    Begin
        If array[i]<>10 Then
            array[i] := 0; (* apaga *)
    End;
```



FOR, IDENT(‘i’), ATRIBUI, INT(0), TO, IDENT(‘len’), SUB, INT(1), DO, BEGIN, IF, IDENT(‘array’), VECT_ESQ, IDENT(‘i’), VECT_DIR, CMP_DIF, INT(10), THEN, IDENT(‘array’), VECT_ESQ, IDENT(‘i’), VECT_DIR, ATRIBUI, INT(0), PVIRG, END, PVIRG

Atributos

Cada *token* tem assim um atributo associado (que pode ser vazio), dependendo da forma como especifico a linguagem.

Para um *token* como um identificador, o seu atributo pode ser uma *string*, ou um apontador para a tabela de símbolos (mais detalhes mais tarde).

Os *tokens* serão importantes na análise sintática.

Os atributos serão importantes na análise semântica.

Token	Atributo
FOR	
IDENT	‘i’
ATRIBUI	
INT	0
TO	
IDENT	‘len’
SUB	
INT	1
DO	
BEGIN	
IF	
IDENT	‘array’
VECT_ESQ	
IDENT	‘i’
…	…

Definições

Token – Padrão – Lexema

Exemplo:

Token: `WHILE`

Padrão *(RE):* `while`

Na *string* de código-fonte:

```
if(1) while(a<10);
```

O lexema para este *token* será:

```
while
```

Exemplo:

Token: `INT`

Padrão *(RE):* `[0-9]+`

Na *string* de código-fonte:

```
if(1) while(a<10);
```

Os lexemas para este *token* serão:

```
1
10
```

Token

- O par com o nome do *token* e o seu atributo opcional.

Padrão

- Uma regra que define que lexemas podem estar associados a um *token*.

Lexema

- *String* no código-fonte que é aceite/encontrada por um padrão e então representa uma instância de um *token*.

LINGUAGENS

Definições

Símbolo – Alfabeto – *String* – **ε** – Linguagem – **L**

Concatenação

(das *strings* $\mathbf{s}_1$ e $\mathbf{s}_2$), representada por $\mathbf{s}_1\mathbf{s}_2$, é a *string* formada por acrescentar a *string* $\mathbf{s}_2$ ao final de $\mathbf{s}_1$.

"**Exponenciação**"

(da *string* **s**, **i** vezes), representada por $\mathbf{s}^i$, é a concatenação de **s** consigo mesma **i** vezes (e.g.: $\mathbf{s}^3 = \mathbf{sss}$). $\mathbf{s}^0$ é **ε**.

Símbolo

- Caractere ou sua abstração (e.g.: ε).

Alfabeto

- Conjunto finito de símbolos (e.g.: letras minúsculas).

String

- Sequência finita de símbolos do alfabeto.

ε

- Lê-se "épsilon". Representa uma *string* de tamanho zero (i.e., vazia).

Linguagem (L)

- Conjunto finito de *strings*.

Operações em linguagens: Concatenação

Todas as combinações de concatenação de todas as *strings* de cada linguagem.

$$\mathbf{L}_1\mathbf{L}_2$$

$$= \{ \mathbf{s}_1\mathbf{s}_2 \text{ com } \mathbf{s}_1 \in \mathbf{L}_1 \text{ e } \mathbf{s}_2 \in \mathbf{L}_2 \}$$

Exemplo, se:

$\mathbf{L}_1$ = {0,1,2,3,4,5,6,7,8,9}

$\mathbf{L}_2$ = {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}

Então $\mathbf{L}_1\mathbf{L}_2$ = todas as sequências de 2 caracteres onde o 1º é um digito numérico e o 2º uma letra maiúscula.

Operações em linguagens: União

Todas as *strings* de ambas as linguagens.

$$\mathbf{L}_1|\mathbf{L}_2$$

$$= \{ \mathbf{s} \text{ com } \mathbf{s} \in \mathbf{L}_1 \text{ ou } \mathbf{s} \in \mathbf{L}_2 \}$$
$$= \mathbf{L}_1 \cup \mathbf{L}_2$$

Exemplo, se:

$\mathbf{L}_1$ = {0,1,2,3,4,5,6,7,8,9}

$\mathbf{L}_2$ = {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}

Então $\mathbf{L}_1|\mathbf{L}_2$ = conjunto de todos os dígitos numéricos e letras maiúsculas.

Operações em linguagens: Fecho de Kleene

Todas as combinações de zero ou mais concatenações de *strings* da linguagem.

$$\mathbf{L}^* = \bigcup_{\mathbf{i}=0..\infty} \mathbf{L}^{\mathbf{i}}$$

Exemplo, se:

L = {0,1,2,3,4,5,6,7,8,9}

Então **L*** = todos os números inteiros sem sinal, incluindo ε.

Operações em linguagens: Fecho positivo

Todas as combinações de uma ou mais concatenações de *strings* da linguagem.

$$\mathbf{L}^{+} = \bigcup_{\mathbf{i}=1..\infty} \mathbf{L}^{\mathbf{i}}$$

Exemplo, se:

L = {0,1,2,3,4,5,6,7,8,9}

Então $\mathbf{L}^{+}$ = todos os números inteiros sem sinal.

Expressões Regulares

PADRÕES

Definições

Expressão Regular – **L**(**RE**)

RE é uma expressão regular *(Regular Expression)*.

Expressão Regular

- Notação (sequência de caracteres) que define/representa uma linguagem.

L(RE)

- Notação para a linguagem que a expressão regular **RE** define (ou seja, é o universo de lexemas possível para **RE**).

Épsilon

ε representa a *string* vazia.

$$\varepsilon$$

$$= \textit{string}\ \text{vazia}$$

$$\mathbf{L}(\varepsilon) = \{\}$$

Exemplo:

ε

é a expressão regular que representa a *string* vazia.

Caracteres

Qualquer símbolo **s** do alfabeto representa-se a si mesmo.

Na prática isto só pode acontecer se o símbolo for um caractere que não é usado na notação de expressões regulares em si.

s

= símbolo **s**

L(**s**) = {**s**}

Exemplo:

a

é a expressão regular que representa a *string* “a”.

Operações: Concatenação

Concatenação da linguagem que a primeira expressão regular representa, com a linguagem que a segunda expressão regular representa.

$$\mathbf{RE_1RE_2}$$

$$= \text{a concatenação de } \mathbf{L(RE_1)} \text{ e } \mathbf{L(RE_2)}$$

$$\mathbf{L(RE_1RE_2) = L(RE_1)L(RE_2)}$$

Exemplo:

`ana`

é a expressão regular que representa a *string* "`ana`".

Operações: União, "ou"

União da linguagem que a primeira expressão regular representa, com a linguagem que a segunda expressão regular representa.

$$\mathbf{RE_1|RE_2}$$

$$= \text{a união de } \mathbf{L(RE_1)} \text{ e } \mathbf{L(RE_2)}$$

$$\mathbf{L(RE_1|RE_2) = L(RE_1)|L(RE_2)}$$

Exemplo:

```
ana|pedro
```

é a expressão regular que representa quer a *string* "`ana`" quer a *string* "`pedro`".

Operações:
Fecho de Kleene

Concatenação (repetição) de 0 ou mais vezes da linguagem que a expressão regular representa.

RE*

= concatenação 0 ou mais vezes de **L**(**RE**)

L(**RE***) = **L**(**RE**)*

Exemplo:

`ana*`

é a expressão regular que representa as *strings* “an”, “ana”, “anaa”, “anaaa”, etc.

Operações: Fecho positivo

Concatenação (repetição) de 1 ou mais vezes da linguagem que a expressão regular representa.

Note-se que nem todos os autores definem este fecho em expressões regulares, já que **RE**$^+$ = (**RE**)(**RE***).

$$\mathbf{RE}^+$$

$$= \text{concatenação 1 ou mais vezes de } \mathbf{L(RE)}$$

$$\mathbf{L(RE}^+\mathbf{)} = \mathbf{L(RE)}^+$$

Exemplo:

`ana+`

é a expressão regular que representa as *strings* "ana", "anaa", "anaaa", etc.

Operações: Opcional

União de ε com a linguagem que a expressão regular representa.

Note-se que nem todos os autores definem esta operação em expressões regulares, já que **RE**? = (**RE**|ε).

RE?

= união de **ε** com **L**(**RE**)

L(**RE**?) = **L**(**RE**)|**L**(**ε**)

Exemplo:

`ana?`

é a expressão regular que representa quer a *string* "`an`" quer a *string* "`ana`".

Parêntesis

As expressões regulares podem ser agrupadas com parêntesis sem isso alterar a linguagem que representam.

$$(\mathbf{RE})$$

$$\mathbf{L}((\mathbf{RE})) = \mathbf{L}(\mathbf{RE})$$

Exemplo:

```
(ana)+
```

é a expressão regular que representa as *strings* “ana”, “anaana”, “anaanaana”, etc.

Classes de caracteres

Notação abreviada para representar qualquer um de um conjunto de símbolos da linguagem.

[abcd**]** representa um destes três símbolos.

[a-d**]** representa o mesmo.

Note-se que nem todos os autores definem esta operação em expressões regulares, já que **[**abc**]** = (a|b|c).

[...]

= conjunto dos símbolos definidos

L([...]) = {...}

Exemplo:

```
[a-zA-Z]
```

é a expressão regular que representa todas as *strings* que têm um único caractere alfabético.

Exercício

1. Indique quais as expressões regulares que representam as seguintes linguagens.

   L(*letra*)|L(*digito*)

   L(*digito*)+L(*letra*)*

2. Explique por suas palavras que linguagens representam estas expressões regulares.

   `0(0|1)*0`

   `((ε|0)1*)*`

   `(0|1)*0(0|1)[01]`

AUTÓMATOS FINITOS

Autómatos Finitos (FA)

São máquinas de estados que implementam expressões regulares.

NFA = *Nondeterministic Finite Automata* (autómatos finitos não determinísticos).

DFA = *Deterministic Finite Automata* (autómatos finitos determinísticos).

Componentes para definição de um NFA

- **Conjunto finito de estados**
  - Inclui estado inicial
  - Inclui estados finais
- **Conjunto finito de símbolos de entrada**
- **Função/tabela de transição**
  - Mapeia pares (estado, símbolo) com um conjunto de estados

Tipicamente um símbolo de entrada é um caracter aceite na linguagem. **ε** nunca pertence aos símbolos de entrada.

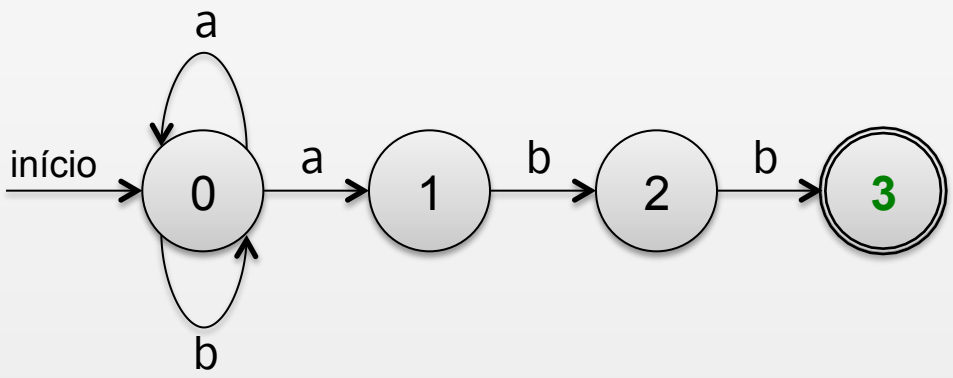
Na prática isto irá significar que um símbolo de entrada é um caracter ASCII.

ε faz parte dos símbolos mapeados na função/tabela de transição.

Exemplo de NFA

para a expressão regular:

`(a|b)*abb`



Conjunto de estados: {0,1,2,3}

Estado inicial: 0

Estados finais: {3}

Símbolos de entrada: {a,b}

Função e tabela de transição:

(0,a) = {0,1}

(0,b) = {0}

(1,b) = {2}

(2,b) = {3}

	a	b	ε
0	{0,1}	{0}	∅
1	∅	{2}	∅
2	∅	{3}	∅
3	∅	∅	∅

Aceitação de um FA

Definição (aplica-se a NFA e DFA).

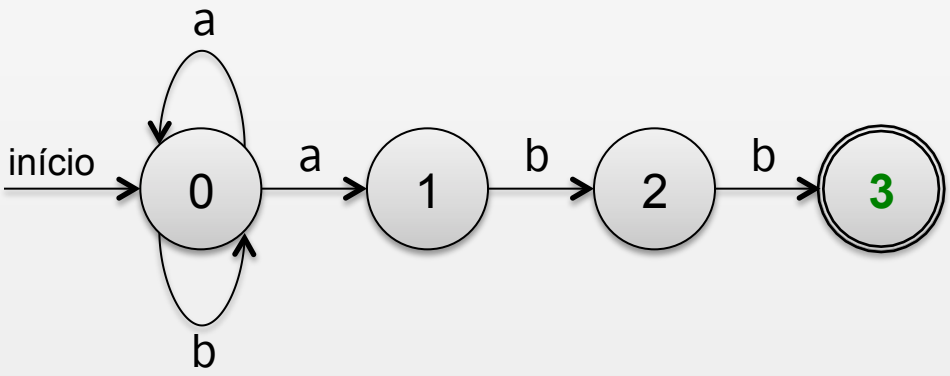
Um FA **aceita** uma *string* de entrada **s** se existir **algum** caminho no diagrama de transições, do início até algum estado final, seguindo sequencialmente os símbolos em **s**.

Num NFA, em qualquer instante é possível seguir transições **ε**.

Exemplos de aceitação

para a expressão regular:

`(a|b)*abb`



String de entrada: **baabb**
Sequência de estados: {0} **b➔** {0} **a➔** {0,1} **a➔** {0,1} **b➔** {0,2} **b➔** {0,**3**}
Aceite.

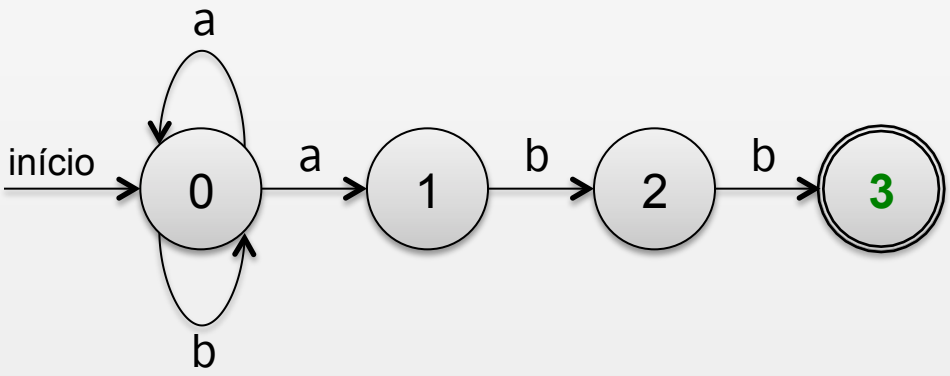
String de entrada: **aabbb**
Sequência de estados: {0} **a➔** {0,1} **a➔** {0,1} **b➔** {0,2} **b➔** {0,**3**} **b➔** {0}
Não aceite.

Vamos ver estas transições em detalhe

Exemplos de aceitação

para a expressão regular:

`(a|b)*abb`



{0,2} **b➔** Função de transição:
(0,b) = {0}
(2,b) = {**3**}
União = {0,**3**}

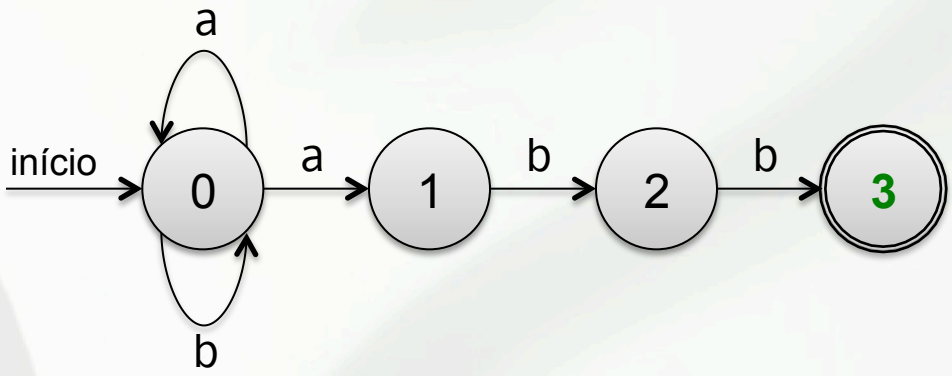
Temos um estado final, mas não terminámos a *string* de entrada.

{0,**3**} **b➔** Função de transição:
(0,b) = {0}
(**3**,b) = ∅
União = {0}

Resultado: {0} = não aceite.

Exercício

Dado o seguinte NFA, avalia a aceitação (indicando a sequência de estados) para as *strings* de entrada indicadas abaixo.

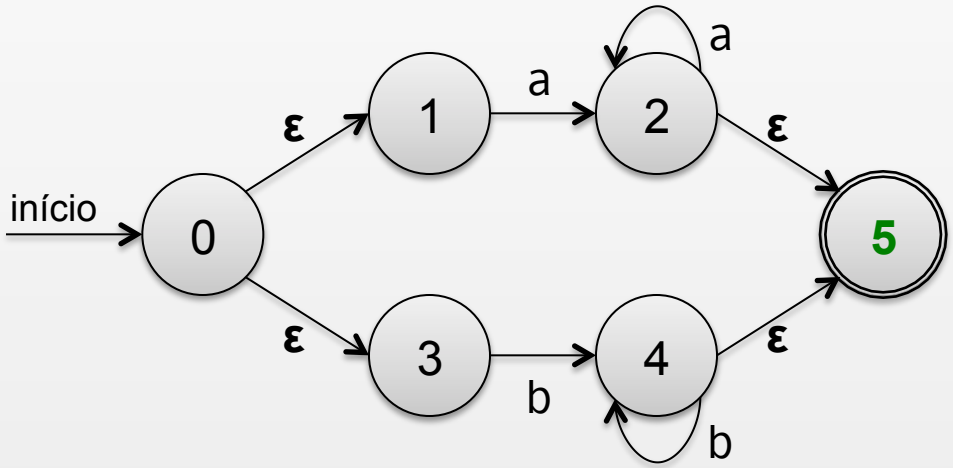


String de entrada: `abab`
String de entrada: `bbb`
String de entrada: `baaabb`
String de entrada: `abb`

Outro exemplo de NFA

para a expressão regular:

aa*|bb*



Conjunto de estados: {0,1,2,3,4,5}

Estado inicial: 0

Estados finais: {**5**}

Símbolos de entrada: {a,b}

Função de transição:

(0,**ε**) = {1,3}

(1,a) = {2}

(2,a) = {2}

(2,**ε**) = {**5**}

(3,b) = {4}

(4,b) = {4}

(4,**ε**) = {**5**}

Fecho- ε

Para simular um NFA com transições com **ε**, precisamos de conhecer o conceito de **fecho-ε(E)**, onde **E** é um conjunto de estados do NFA.

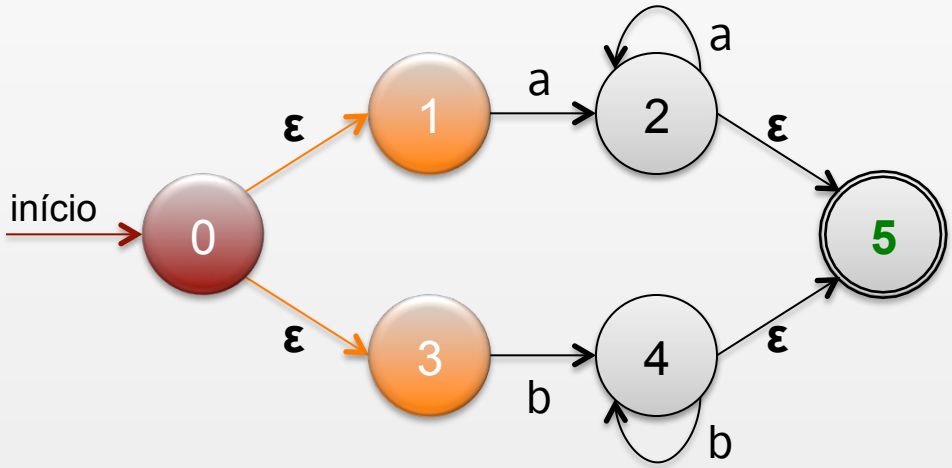
Então para simular um NFA, aplico inicialmente **fecho-ε({0})**, e também aplico **fecho-ε(E)** depois de determinar os estados **E** para onde posso transitar com cada símbolo de entrada.

Conjunto de estados do NFA onde se pode chegar a partir de qualquer estado em **E** seguindo apenas transições **ε**.
Inclui **E**.

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



Primeiro exemplo.

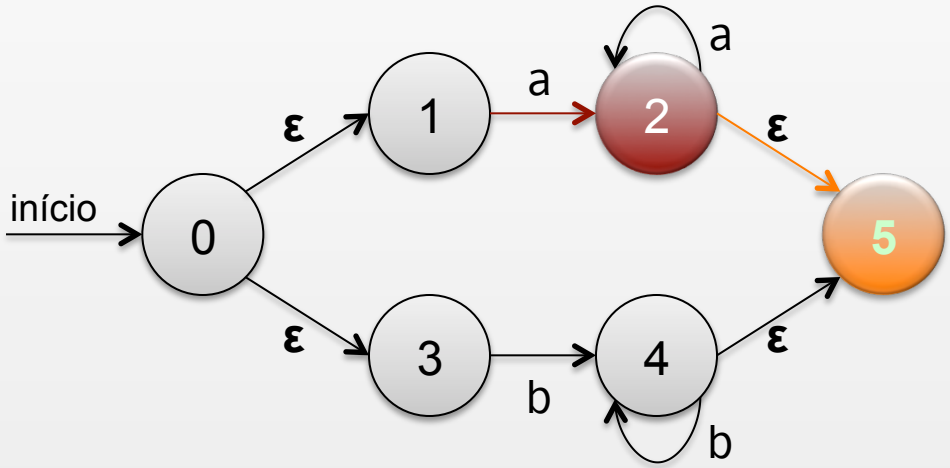
String de entrada: **aaa**

Sequência de estados inicial: **fecho-ε({0})** = {0,1,3}

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



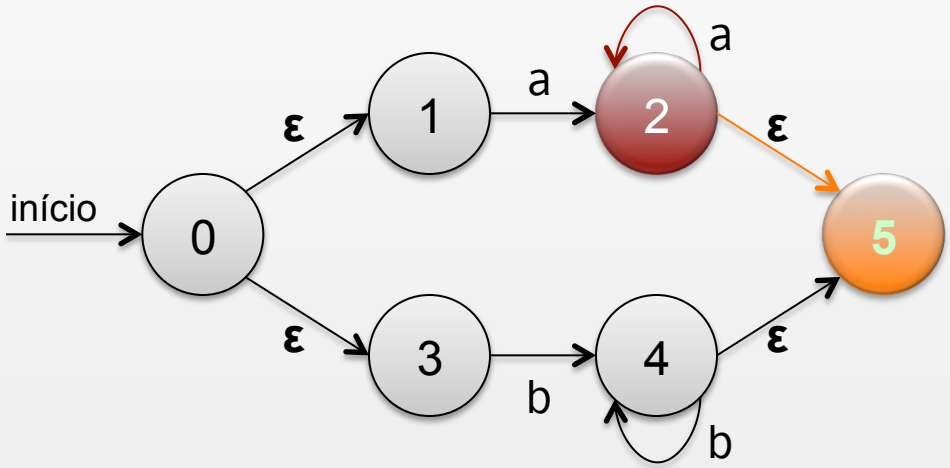
String de entrada: aaa

Sequência de estados: {0,1,3} a➔ fecho-ε({2}) = {2,5}

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



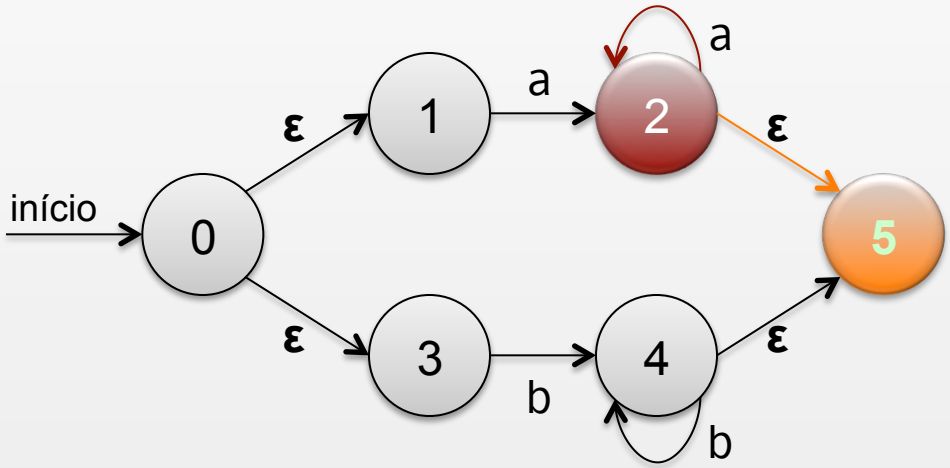
String de entrada: aaa

Sequência de estados: {0,1,3} a➔ {2,**5**} a➔ **fecho-ε({2})** = {2,**5**}

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



String de entrada: aaa

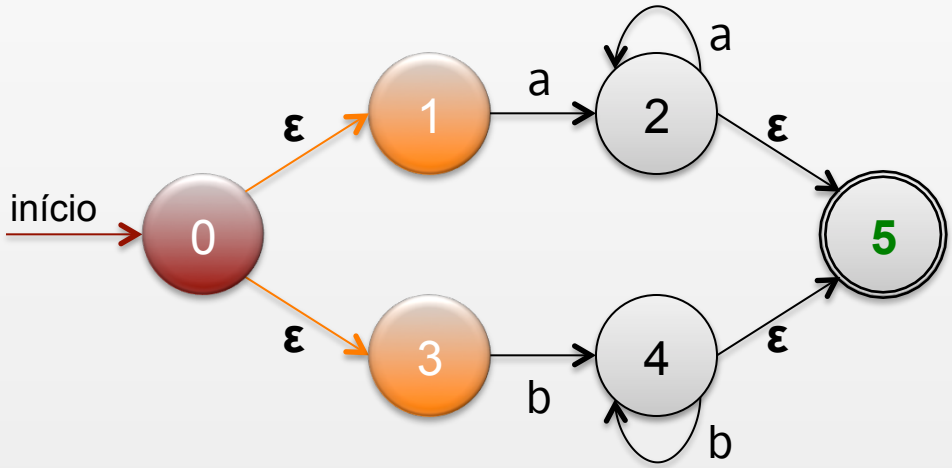
Sequência de estados: {0,1,3} a➔ {2,**5**} a➔ {2,**5**} a➔ **fecho-ε({2})** = {2,**5**}

Aceite.

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



Segundo exemplo.

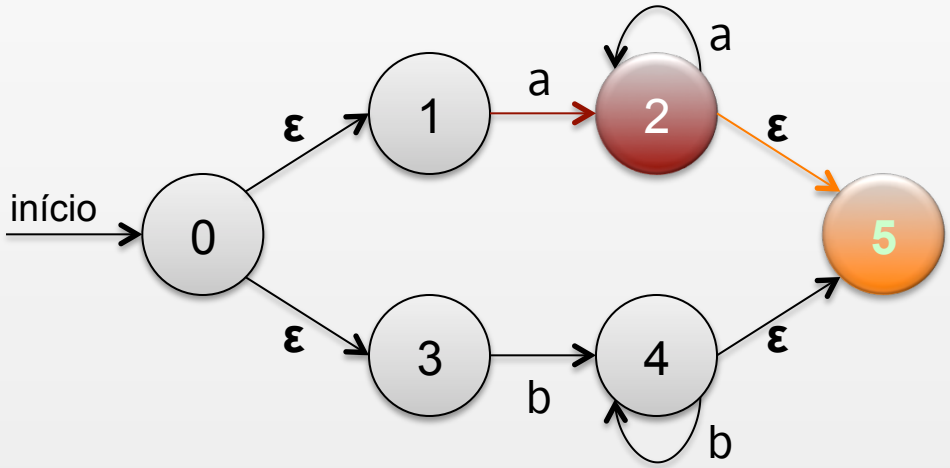
String de entrada: **abbb**

Sequência de estados inicial: **fecho-ε({0})** = {0,1,3}

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



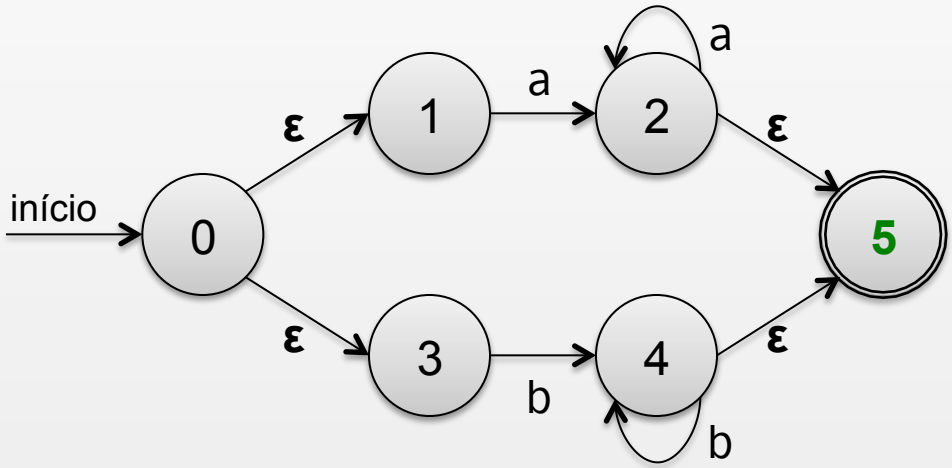
String de entrada: **abbb**

Sequência de estados: {0,1,3} **a➔** **fecho-ε({2})** = {2,**5**}

Outros exemplos de aceitação

para a expressão regular:

aa*|bb*



String de entrada: abbb

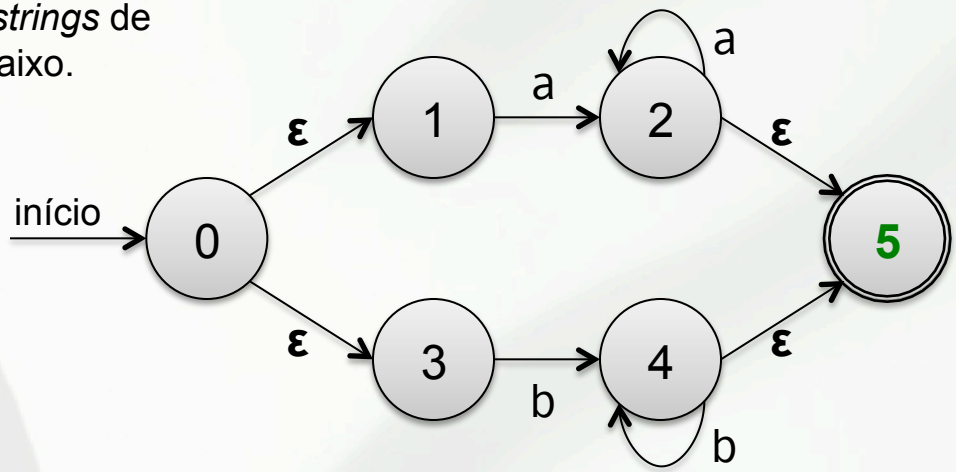
Sequência de estados: {0,1,3} a➔ {2,**5**} b➔ **fecho-ε(∅)** = ∅

Como o resultado é o conjunto vazio (∅), os símbolos seguintes vão dar como resultado também ∅.

Não aceite.

Exercício

Dado o seguinte NFA, avalia a aceitação (indicando a sequência de estados) para as *strings* de entrada indicadas abaixo.



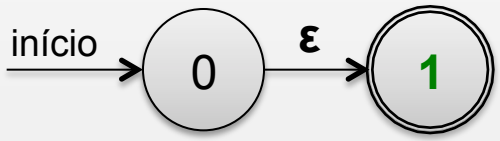
String de entrada: `abab`

String de entrada: `bbb`

Transformar uma expressão regular *(RE)* num NFA

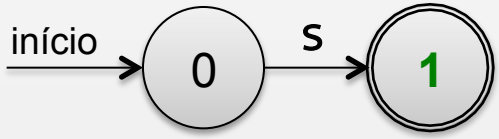
CONSTRUÇÃO DE THOMPSON

Para Épsilon (ε)



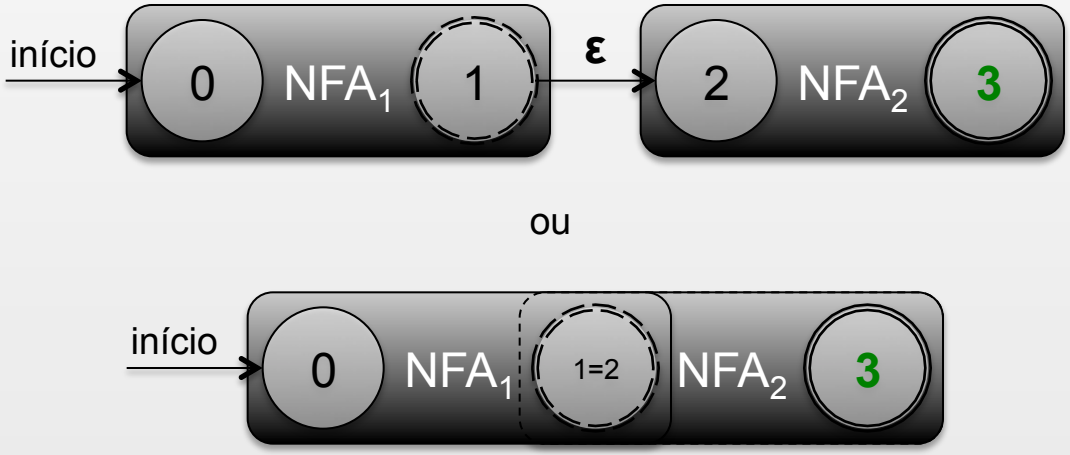
Para qualquer símbolo de entrada

Chamemos a esse símbolo, s.



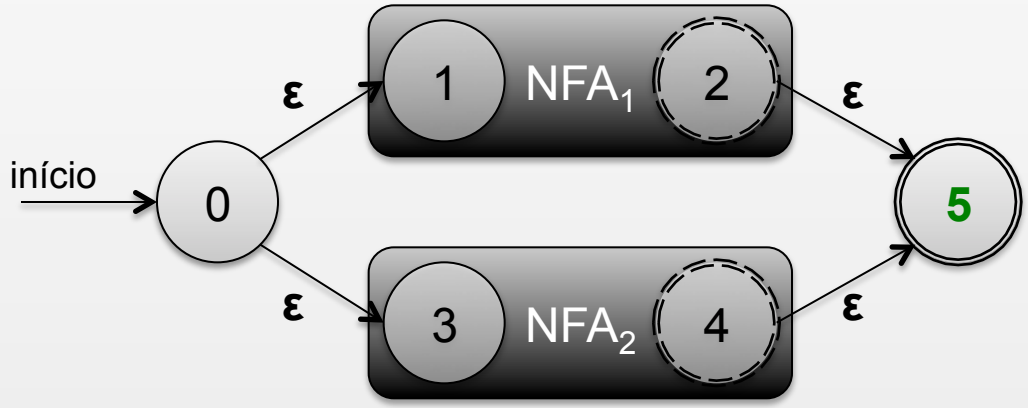
Para RE_1RE_2

Sendo NFA_1 o NFA correspondente a RE_1 (a 1ª expressão regular) e NFA_2 o NFA correspondente a RE_2.



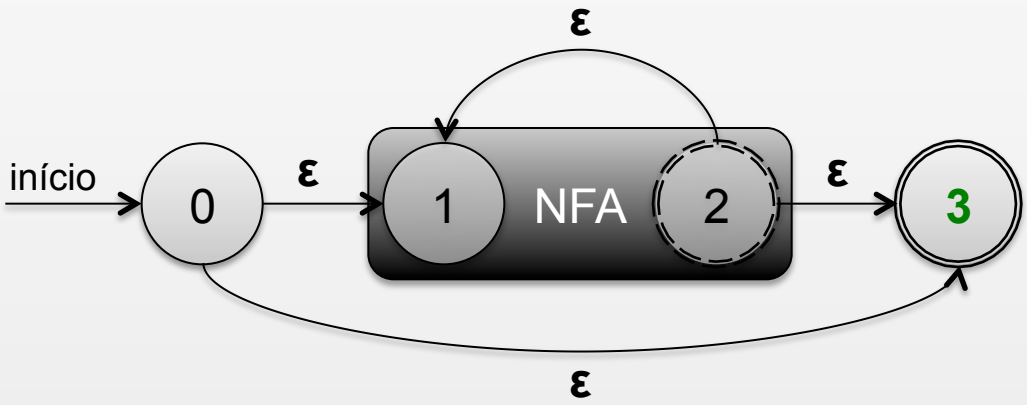
Para $RE_1|RE_2$

Sendo NFA_1 o NFA correspondente a RE_1 (a 1ª expressão regular) e NFA_2 o NFA correspondente a RE_2.



Para RE*

Sendo NFA o NFA correspondente
a RE (a expressão regular).



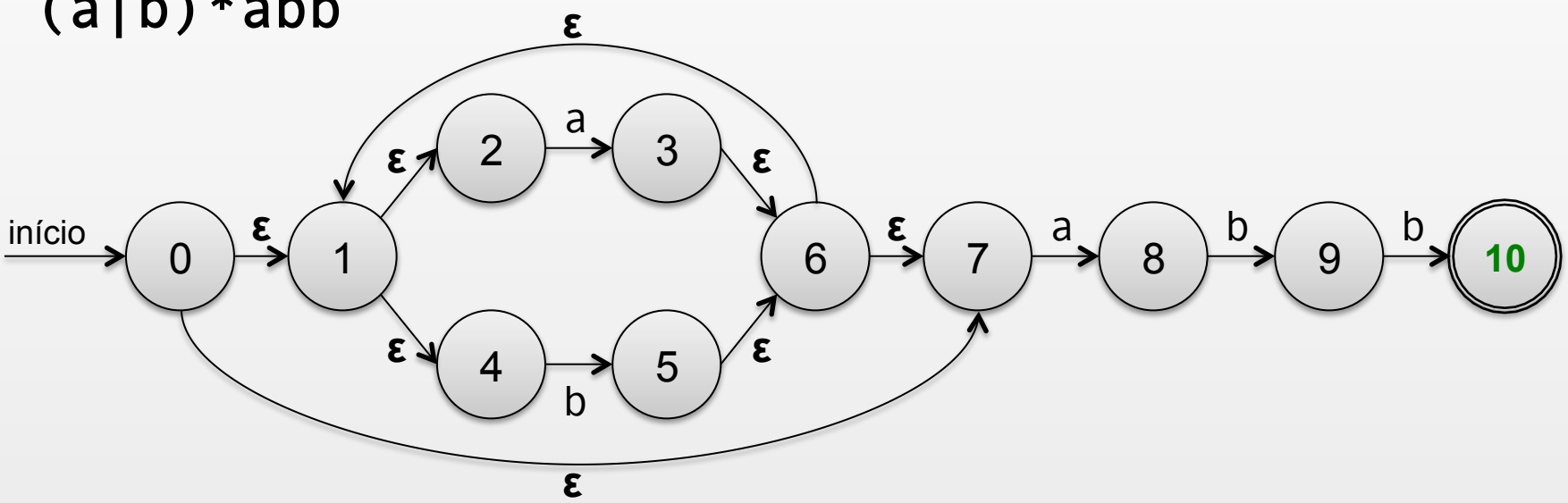
Questão:

Como transformo este NFA num que se aplique a “RE+”?

Exemplo

para a expressão regular:

(a|b)*abb



Exercício

Converte a expressão regular indicada abaixo num NFA.

```
D*.D|D.D*
```

Aqui o conjunto de símbolos de entrada é: { ‘D’, ’.’ }

AUTÓMATOS FINITOS DETERMINÍSTICOS

Distinção entre NFA e DFA

Definições para um DFA.

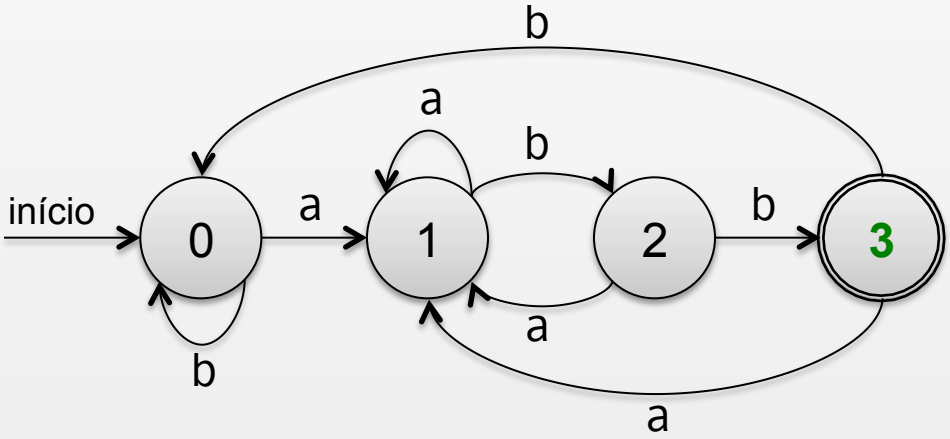
- **Não existem transições ε**
- **Em qualquer estado não existe mais de uma transição que sai com base no mesmo símbolo**

Com estas exigências eu tenho **apenas um caminho** que posso percorrer no diagrama de estados para uma dada *string* de entrada.

Exemplo de DFA

para a expressão regular:

`(a|b)*abb`



Conjunto de estados: {0,1,2,3}

Estado inicial: 0

Estados finais: {3}

Símbolos de entrada: {a,b}

Função e tabela de transição:

(0,a) = 1 (2,a) = 1

(0,b) = 0 (2,b) = 3

(1,a) = 1 (3,a) = 1

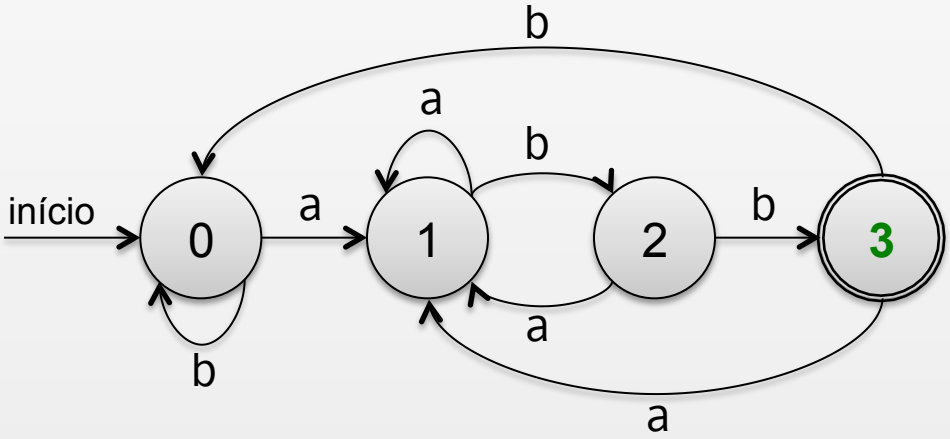
(1,b) = 2 (3,b) = 0

	a	b
0	1	0
1	1	2
2	1	3
3	1	0

Exemplos de aceitação

para a expressão regular:

`(a|b)*abb`



String de entrada: `baabb`

Sequência de estados: 0 **b➔** 0 **a➔** 1 **a➔** 1 **b➔** 2 **b➔** **3**

Aceite.

String de entrada: `aabbb`

Sequência de estados: 0 **a➔** 1 **a➔** 1 **b➔** 2 **b➔** **3** **b➔** 0

Não aceite.

Exercício

Discussão: Que diferenças vê num DFA ao nível de:

1. Função/tabela de transições
2. Facilidade de avaliar a aceitação
3. Facilidade de desenhar o diagrama de estados

Transformar um NFA num DFA

CONSTRUÇÃO DE SUBCONJUNTOS

Definições

O algoritmo de conversão irá construir uma tabela de transição **TabTrans** para o DFA. Inicialmente esta tabela está vazia.

Nesta tabela, cada estado do DFA será equivalente a um conjunto de estados do NFA original.

Durante a construção da tabela **TabTrans** iremos ter os estados do DFA “marcados” ou “desmarcados” numa tabela auxiliar **TabEstados**.

	Definição
e	Um qualquer estado do NFA.
E	Um qualquer conjunto de estados do NFA.
s	Um qualquer símbolo de entrada.
TabTrans	Tabela de transições do DFA gerado.
TabEstados	Tabela de estados do DFA gerado, usada como auxiliar no algoritmo.
fecho-ε(e)	Mesmo que **fecho-ε({e})** (veja abaixo).
fecho-ε(E)	Conjunto de estados do NFA onde se pode chegar a partir de qualquer estado em **E** seguindo apenas transições **ε**. Inclui **E**.
move(E,s)	Conjunto de estados no NFA onde se pode chegar a partir de qualquer estado em **E** seguindo apenas transições para o símbolo **s**.
U	Conjunto de estados possíveis após se receber o símbolo **s** estando em qualquer dos estados **E**. Auxiliar no algoritmo.

Algoritmo

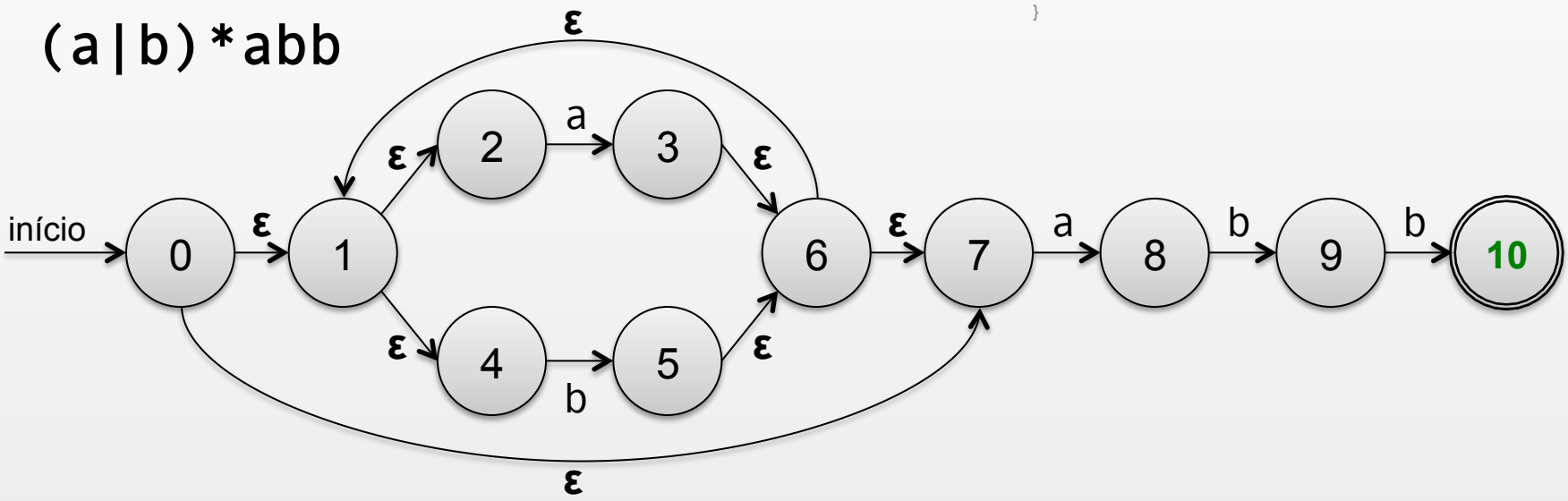
adicionar **fecho-ε(0)** como um estado desmarcado a **TabEstados**;
while(existe um estado desmarcado (**E**) em **TabEstados**)
{
marcar **E**;
for(cada símbolo de entrada **s**)
{
U = fecho-ε(move(E,s));
if(**U** não está em **TabEstados**)
adicionar **U** como um estado desmarcado a **TabEstados**;
TabTrans[E,s] = U;
}
}

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



TabEstados e **TabTrans**:

DFA	NFA (E)	a	b

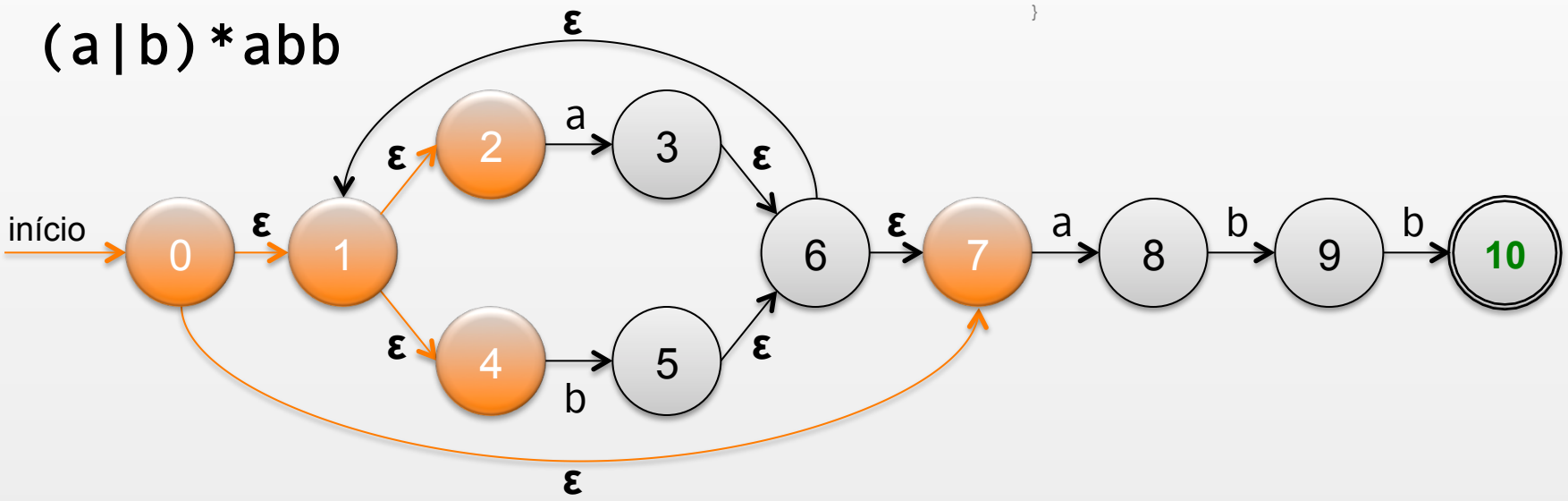
Inicialmente, tabelas vazias.

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



TabEstados e **TabTrans**:

	{0,1,2,4,7}

DFA	NFA (E)	a	b

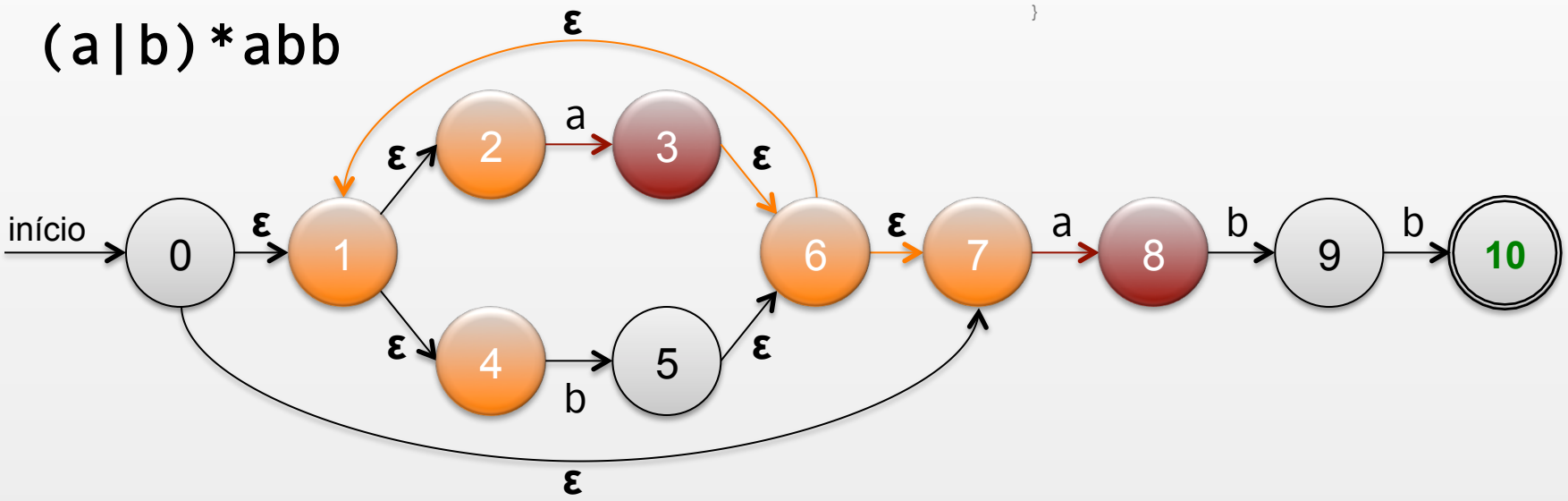
Adicionado **fecho-ε(0)** a **TabEstados**.

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)*abb$



E = {0,1,2,4,7}.

Marcou-se o estado **E**.

U = **fecho-ε(move(E,a))**.

Adicionado **U** a **TabEstados**.

Adicionado **U** a **TabTrans[E,a]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
	{1,2,3,4,6,7,8}

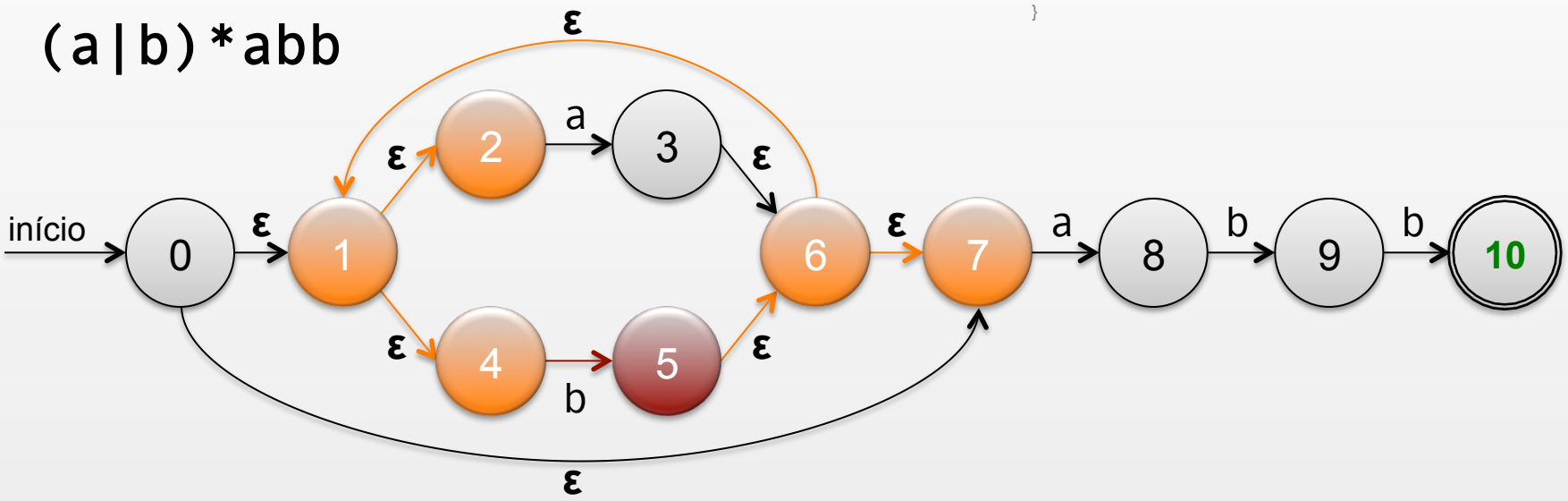
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	
B	{1,2,3,4,6,7,8}		

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)*abb$



U = **fecho-ε(move(E,b))**.
Adicionado **U** a **TabEstados**.
Adicionado **U** a **TabTrans[E,b]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
	{1,2,3,4,6,7,8}
	{1,2,4,5,6,7}

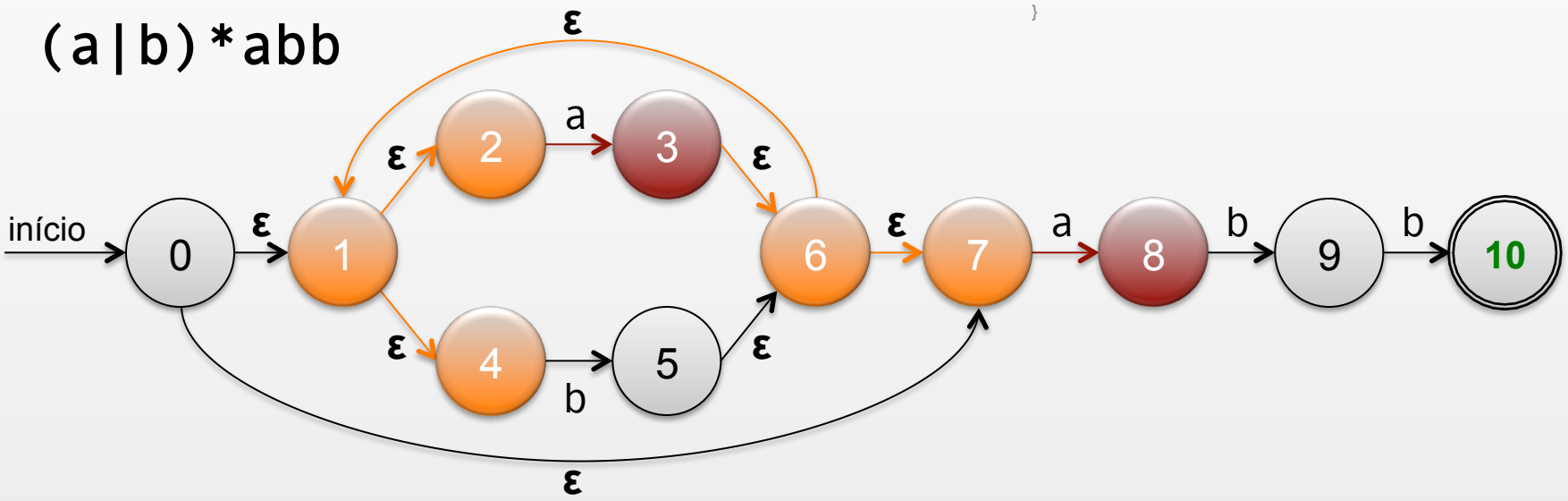
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}		
C	{1,2,4,5,6,7}		

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



E = {1,2,3,4,6,7,8}.
Marcou-se o estado **E**.
U = **fecho-ε(move(E,a))**.
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,a]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
	{1,2,4,5,6,7}

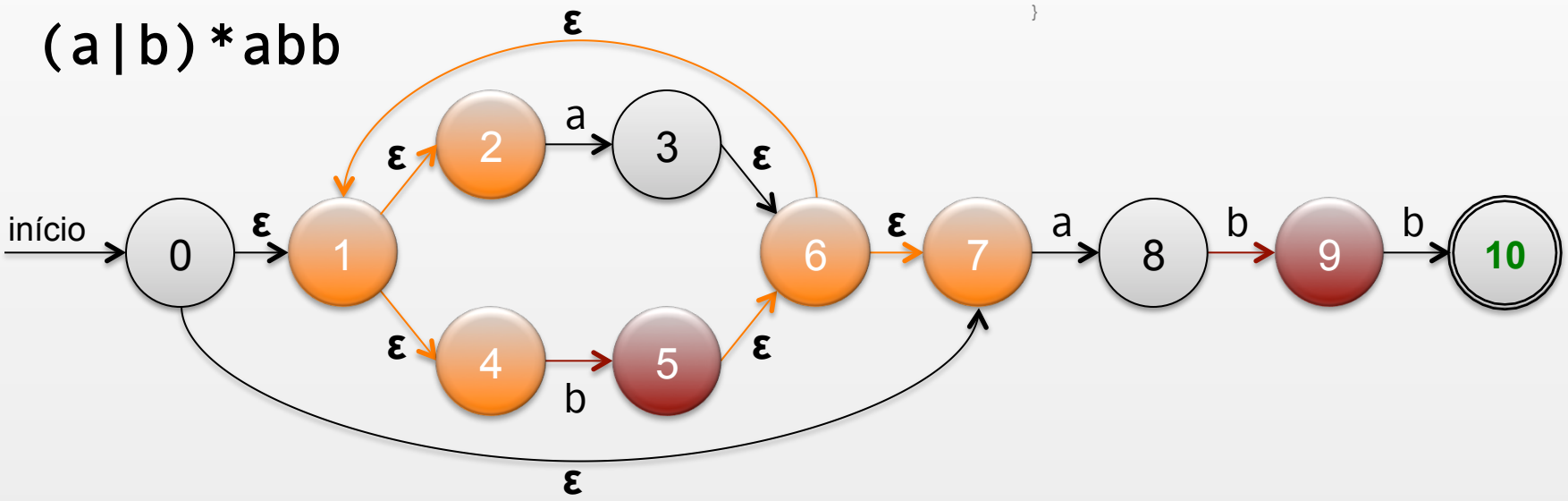
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	
C	{1,2,4,5,6,7}		

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)*abb$



TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
	{1,2,4,5,6,7}
	{1,2,4,5,6,7,9}

DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}		
D	{1,2,4,5,6,7,9}		

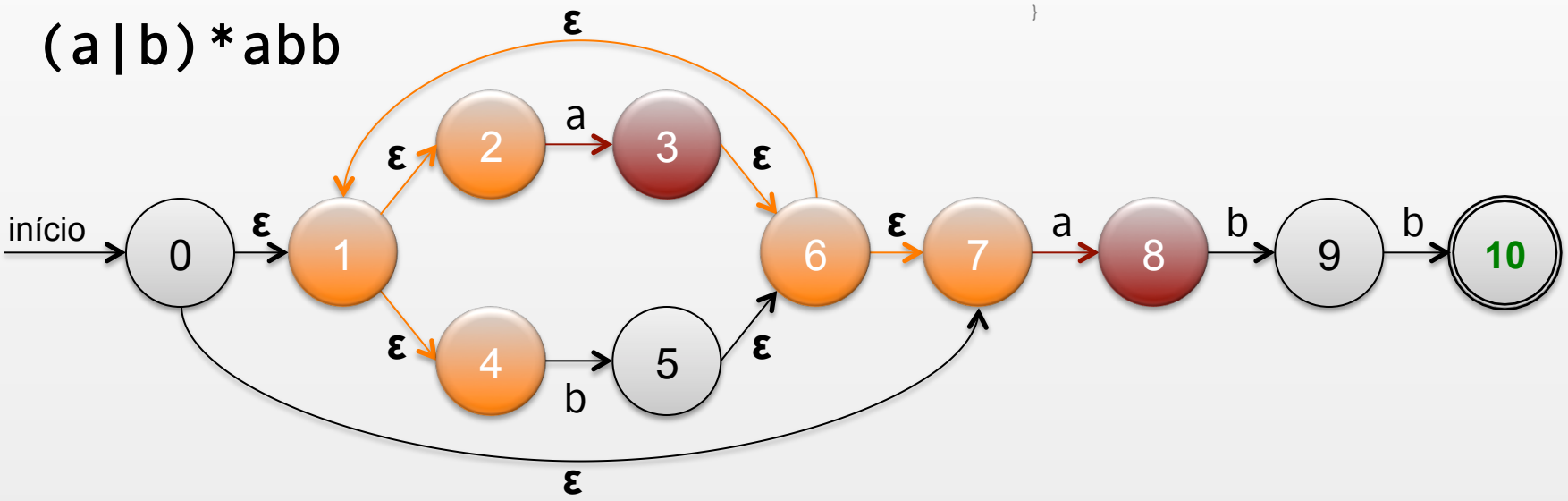
U = **fecho-ε(move(E,b))**.
Adicionado **U** a **TabEstados**.
Adicionado **U** a **TabTrans[E,b]**.

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



E = {1,2,4,5,6,7}.
Marcou-se o estado **E**.
U = **fecho-ε(move(E,a))**.
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,a]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
	{1,2,4,5,6,7,9}

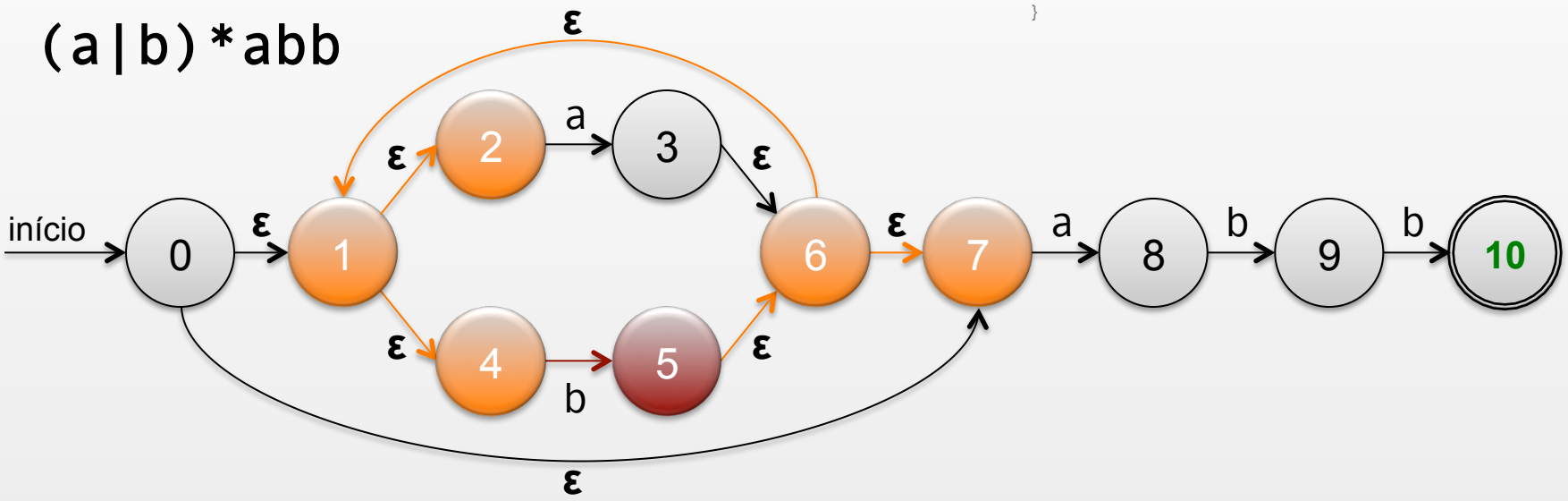
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	
D	{1,2,4,5,6,7,9}		

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)*abb$



TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
	{1,2,4,5,6,7,9}

DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}		

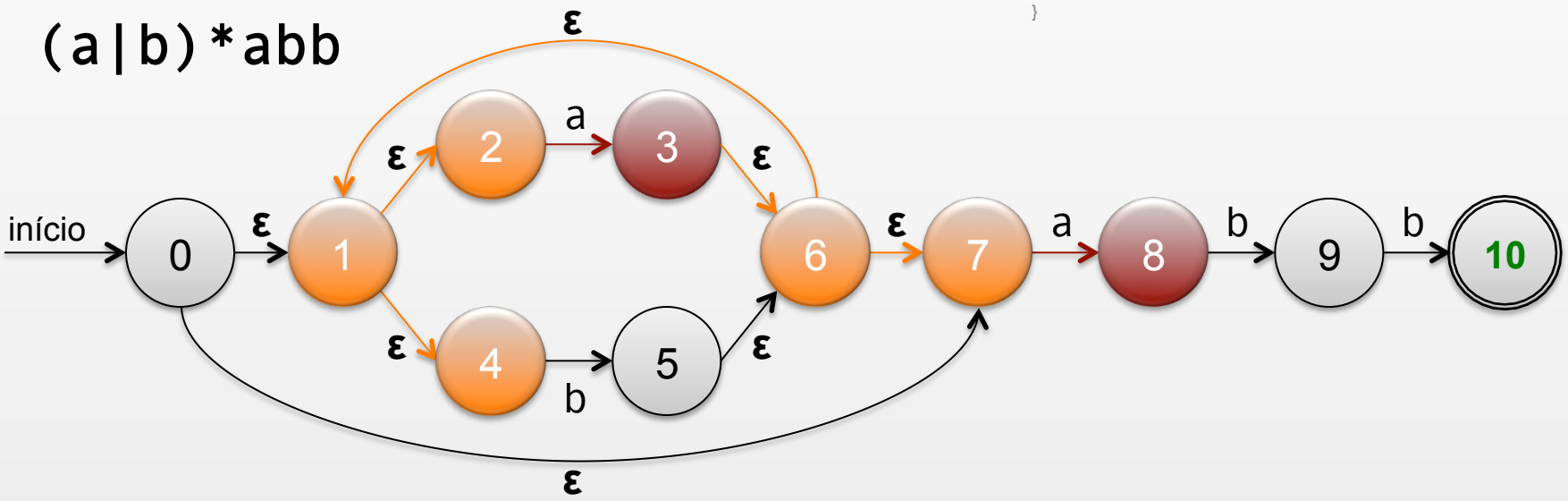
U = **fecho-ε(move(E,b))**.
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,b]**.

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



E = {1,2,4,5,6,7,9}.
Marcou-se o estado **E**.
U = **fecho-ε(move(E,a))**.
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,a]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
✔	{1,2,4,5,6,7,9}

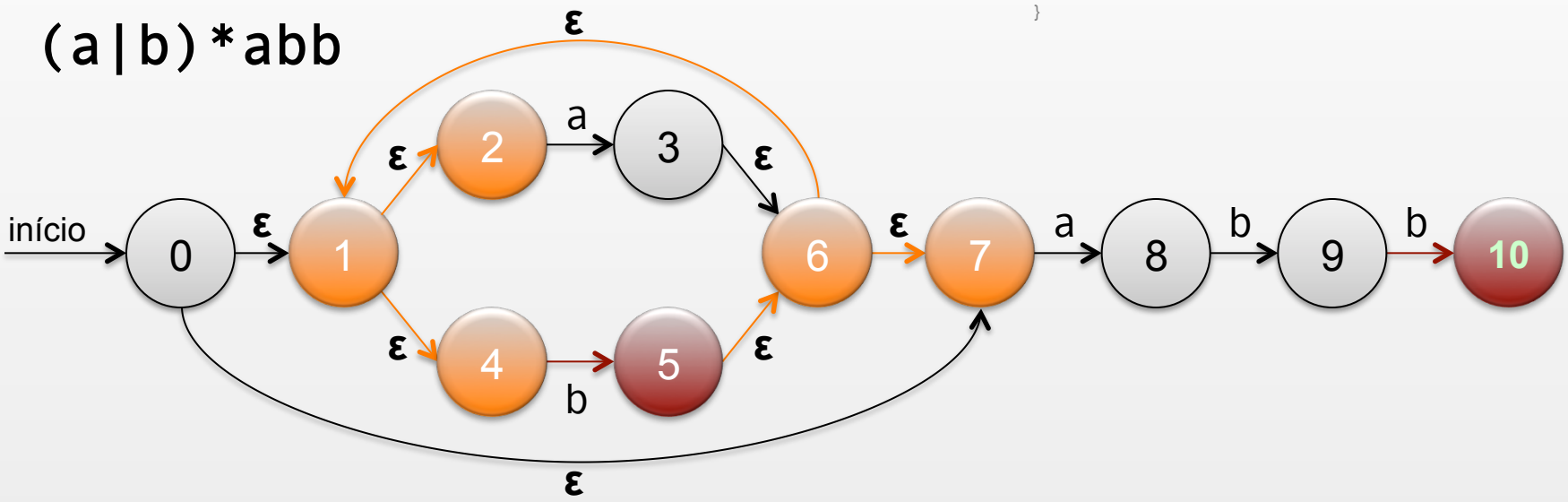
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}	B	

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
✔	{1,2,4,5,6,7,9}
	{1,2,4,5,6,7,10}

DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}	B	E
E	{1,2,4,5,6,7,10}		

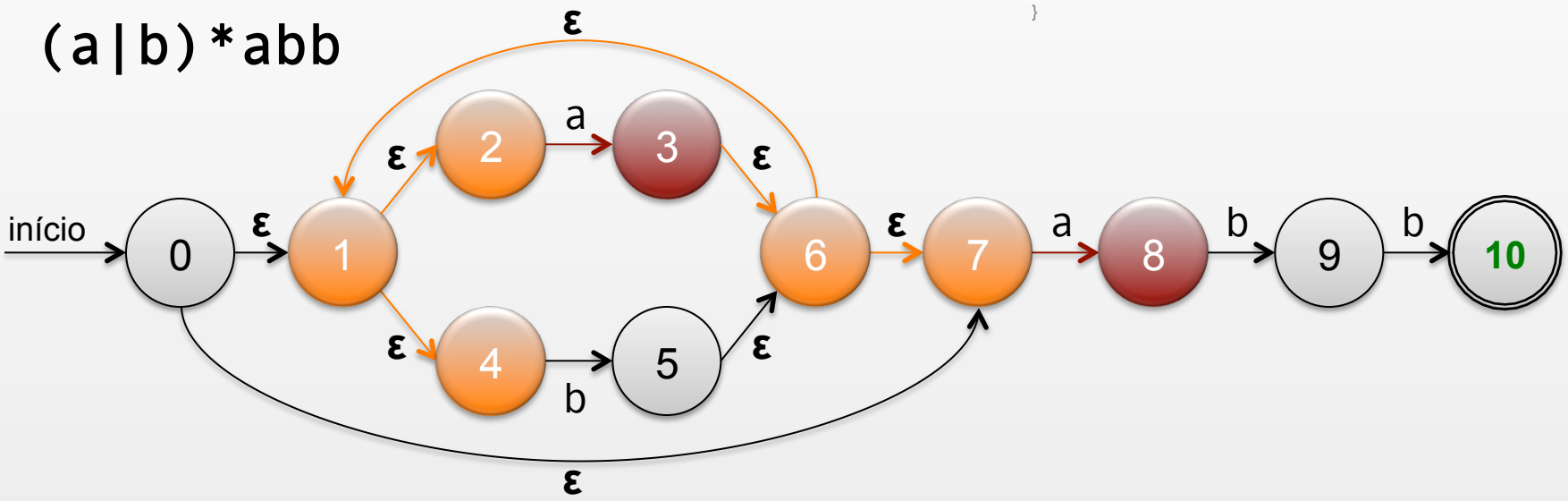
U = fecho-ε(**move(E,b)**).
Adicionado **U** a **TabEstados**.
Adicionado **U** a **TabTrans[E,b]**.

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



E = {1,2,4,5,6,7,10}.
Marcou-se o estado **E**.
U = **fecho-ε(move(E,a))**.
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,a]**.

TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
✔	{1,2,4,5,6,7,9}
✔	{1,2,4,5,6,7,**10**}

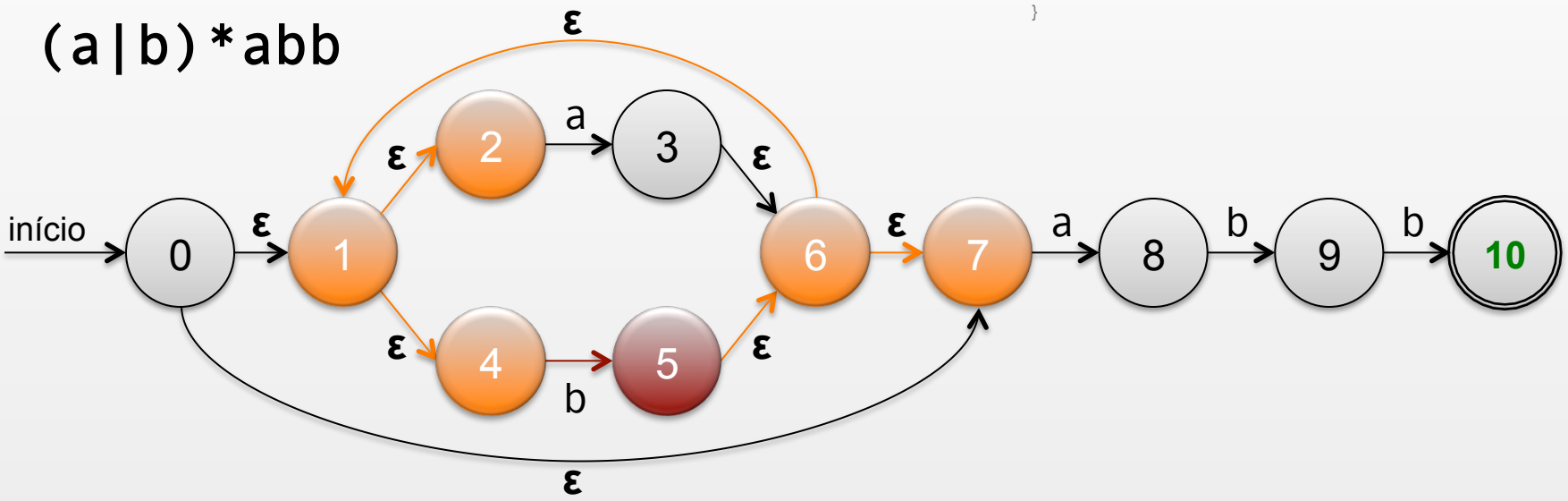
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}	B	**E**
E	{1,2,4,5,6,7,**10**}	B	

```
adicionar fecho-ε(0) como um estado desmarcado a TabEstados;
while( existe um estado desmarcado (E) em TabEstados )
    {
    marcar E;
    for( cada símbolo de entrada s )
        {
        U = fecho-ε( move(E,s) );
        if( U não está em TabEstados )
            adicionar U como um estado desmarcado a TabEstados;
        TabTrans[E,s] = U;
        }
    }
```

Exemplo

Com o NFA para a RE:

$(a|b)*abb$



TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
✔	{1,2,4,5,6,7,9}
✔	{1,2,4,5,6,7,10}

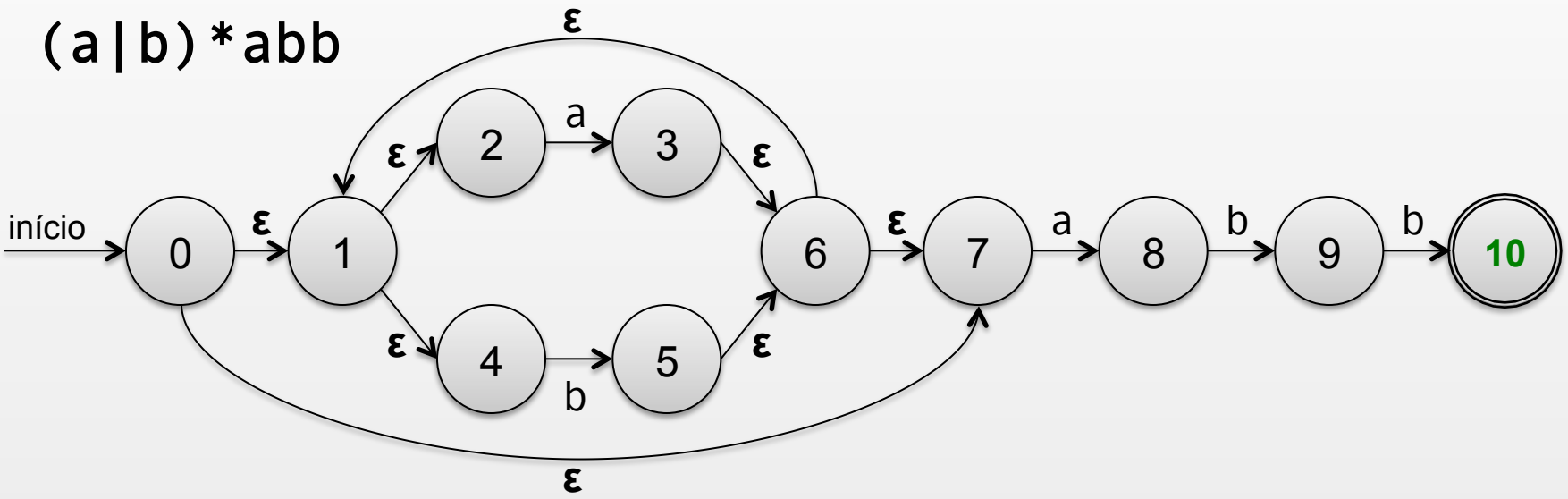
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}	B	E
E	{1,2,4,5,6,7,10}	B	C

U = fecho-ε(**move(E,b)**).
U já existe em **TabEstados**.
Adicionado **U** a **TabTrans[E,b]**.

Exemplo

Com o NFA para a RE:

$(a|b)^*abb$



TabEstados e **TabTrans**:

✔	{0,1,2,4,7}
✔	{1,2,3,4,6,7,8}
✔	{1,2,4,5,6,7}
✔	{1,2,4,5,6,7,9}
✔	{1,2,4,5,6,7,**10**}

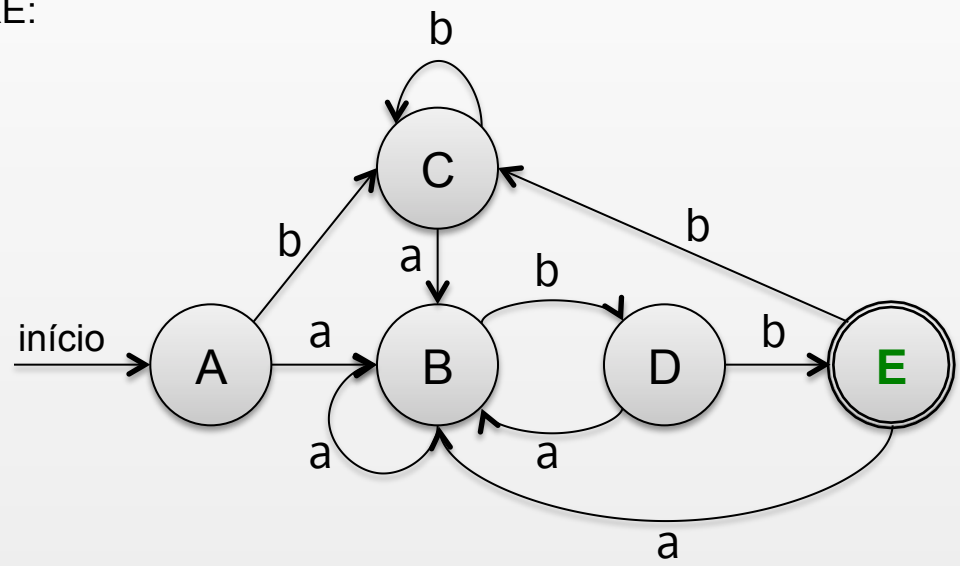
DFA	NFA (E)	a	b
A	{0,1,2,4,7}	B	C
B	{1,2,3,4,6,7,8}	B	D
C	{1,2,4,5,6,7}	B	C
D	{1,2,4,5,6,7,9}	B	**E**
E	{1,2,4,5,6,7,**10**}	B	C

Já não existem mais estados desmarcados em **TabEstados**. Fim.

Exemplo

Com o NFA para a RE:

`(a|b)*abb`



DFA final.

Podemos alterar as letras de A a E por números de 0 a 4 para o fazer parecer mais com os DFAs que temos visto, mas isso não é obrigatório.

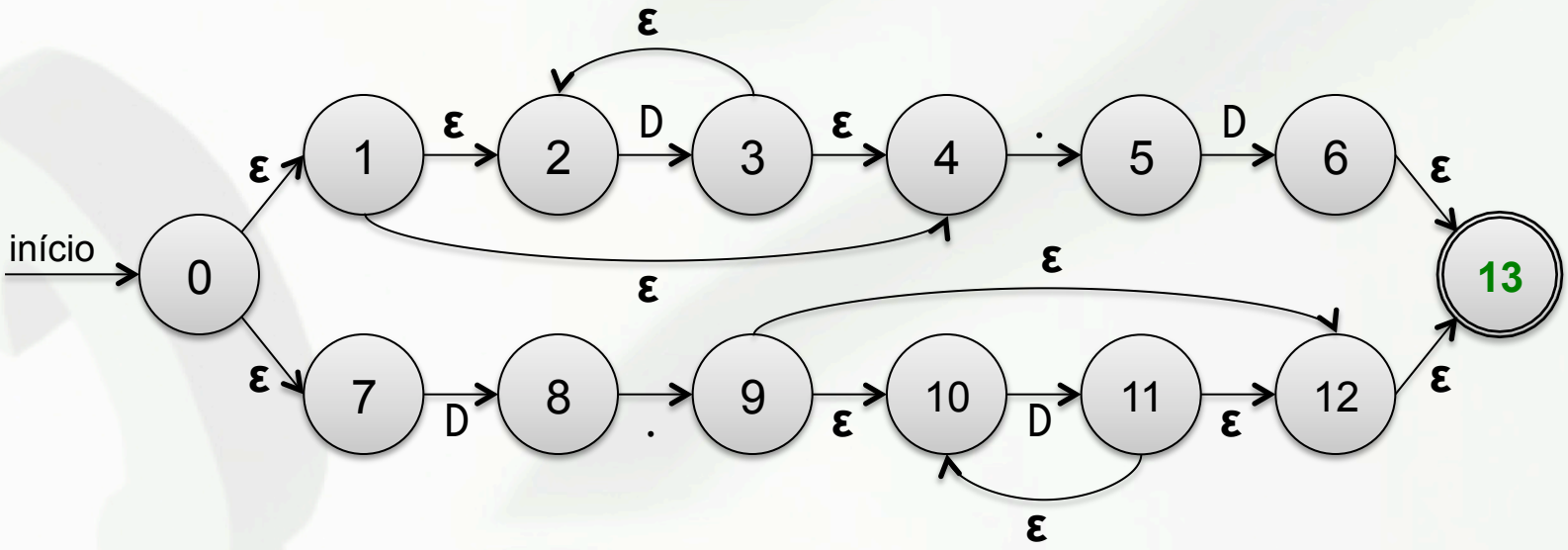
Este DFA não está optimizado: os estados A e C são equivalentes.

	a	b
A	B	C
B	B	D
C	B	C
D	B	**E**
E	B	C

Exercício

Converta para DFA o NFA de:

```
D*.D|D.D*
```



Exercício

Converte a expressão regular indicada abaixo num NFA e depois converte-o para DFA.

a|aa|ba*

MINIMIZAÇÃO DE ESTADOS

Algoritmo

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

No passo 2, estados sem transições a sair deles são estados onde ambas as transições na tabela são para ∅.

No passo 3, transições para ∅ são sempre consideradas como pertencendo ao mesmo conjunto.

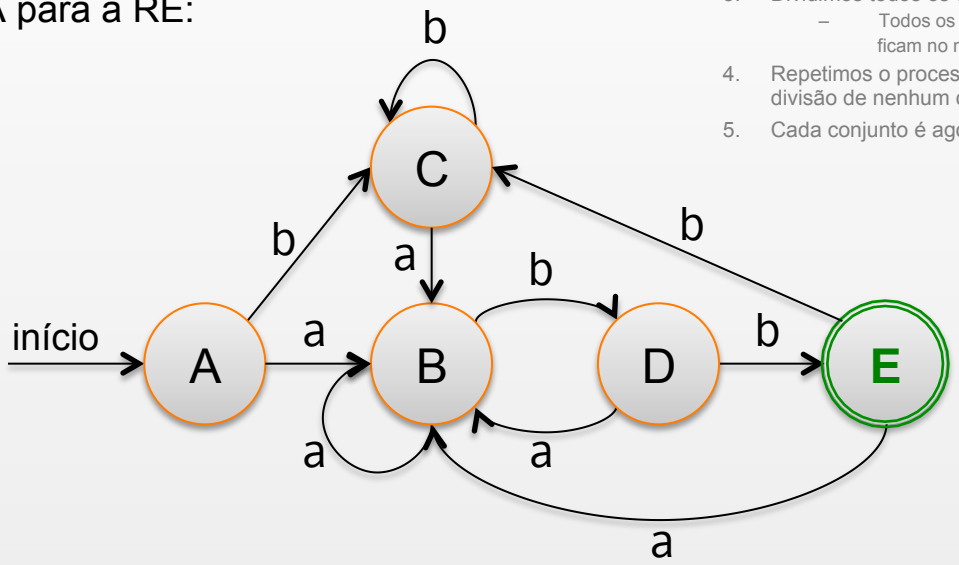
É possível que o processo de minimização gere um "estado morto" de onde não se sai com nenhum símbolo de entrada. É seguro remover esse estado embora o resultado final não seja formalmente um DFA.

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

Exemplo

Com a tabela do DFA para a RE:

$(a|b)^*abb$



	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Divisão inicial: {A,B,C,D} {E}

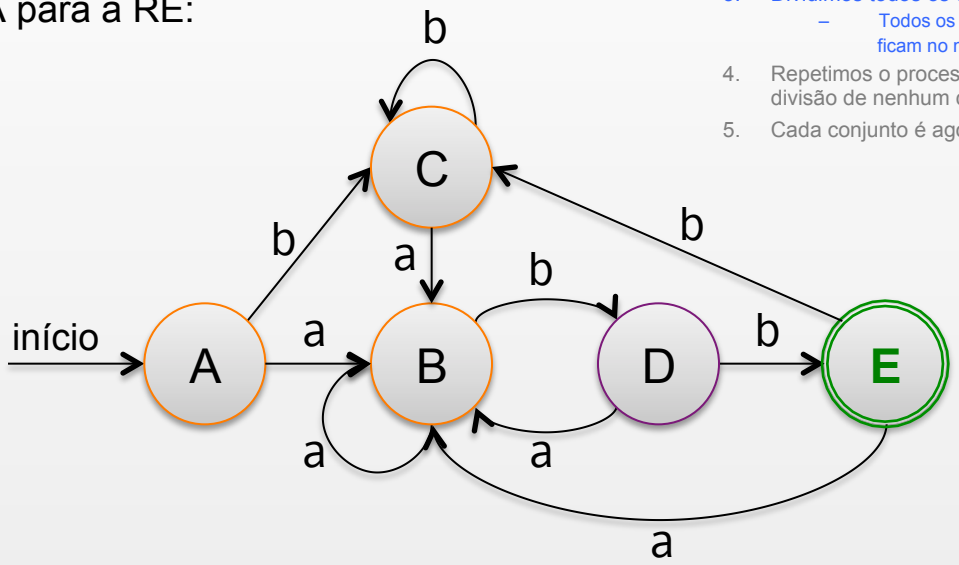
Não existem estados sem transições a saírem deles.

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

Exemplo

Com a tabela do DFA para a RE:

$(a|b)*abb$



	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Divisão atual: {A,B,C,D} {E}

O conjunto {E} não pode ser dividido.

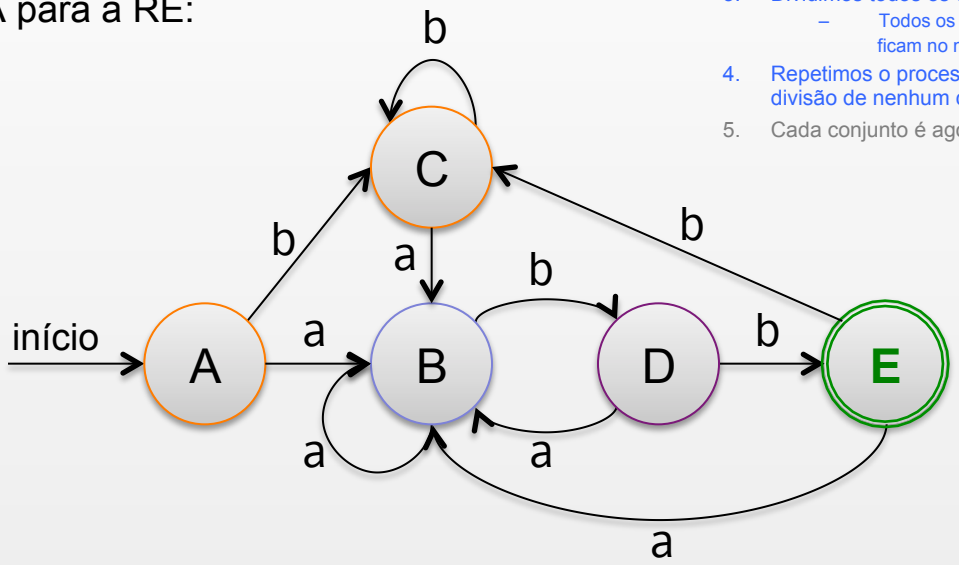
Do conjunto {A,B,C,D} apenas D tem transições para fora deste conjunto, logo dividimos este conjunto em dois: {A,B,C} {D}

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

Exemplo

Com a tabela do DFA para a RE:

$(a|b)*abb$



	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Divisão atual: {A,B,C} {D} {E}

Os conjuntos {D} e {E} não podem ser divididos.

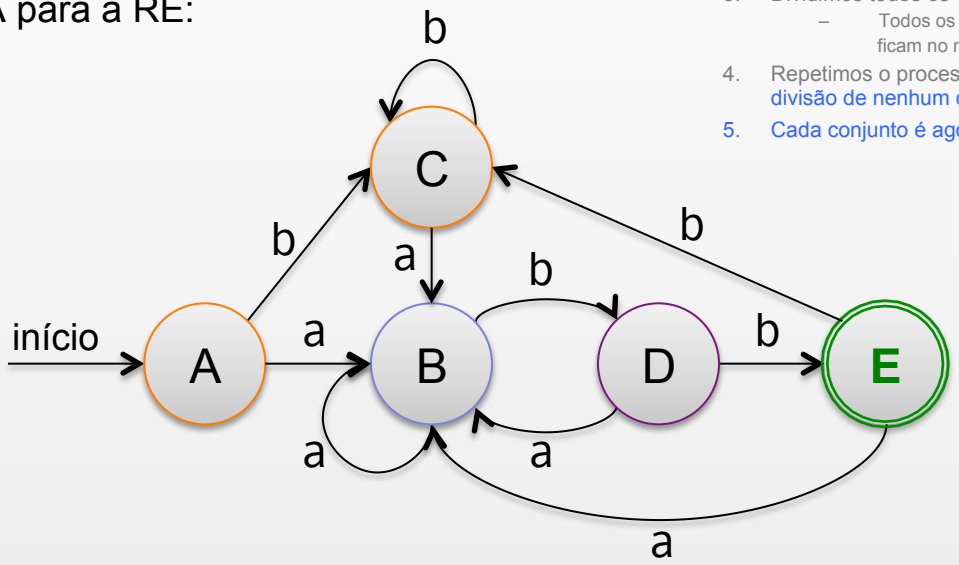
Do conjunto {A,B,C} apenas B tem transições para fora deste conjunto, logo dividimos este conjunto em dois: {A,C} {B}

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

Exemplo

Com a tabela do DFA para a RE:

$(a|b)^*abb$



	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Divisão atual: {A,C} {B} {D} {E}

Nenhum dos conjuntos pode ser dividido.

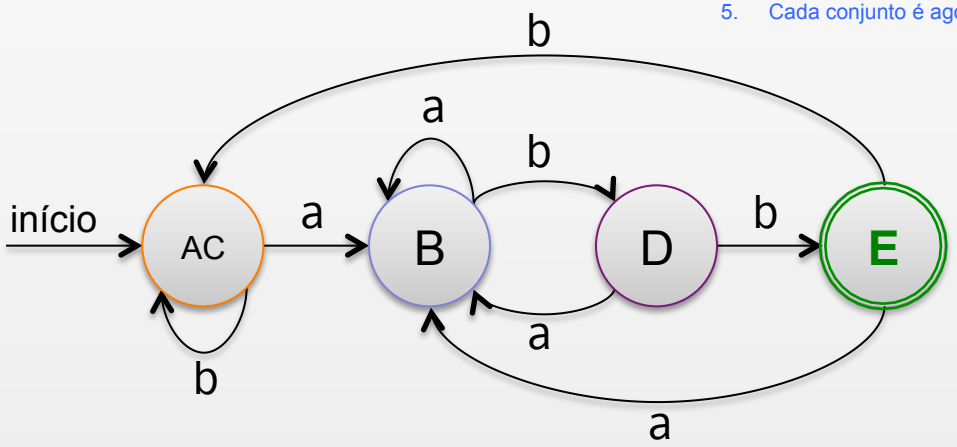
Cada um destes conjuntos é então um estado do DFA minimizado.

1. Dividimos os estados em dois conjuntos
   - Um com os estados finais
   - Outro com os restantes
2. Dividimos cada um dos dois conjuntos em dois subconjuntos
   - Um com os estados que não têm transições a sair deles
   - Outro com os restantes
3. Dividimos todos os conjuntos em vários subconjuntos
   - Todos os estados que transitam para um mesmo grupo ficam no mesmo subconjunto
4. Repetimos o processo a partir de 3 até não ser possível a divisão de nenhum conjunto.
5. Cada conjunto é agora um estado do DFA minimizado.

Exemplo

Com a tabela do DFA para a RE:

$(a|b)^*abb$



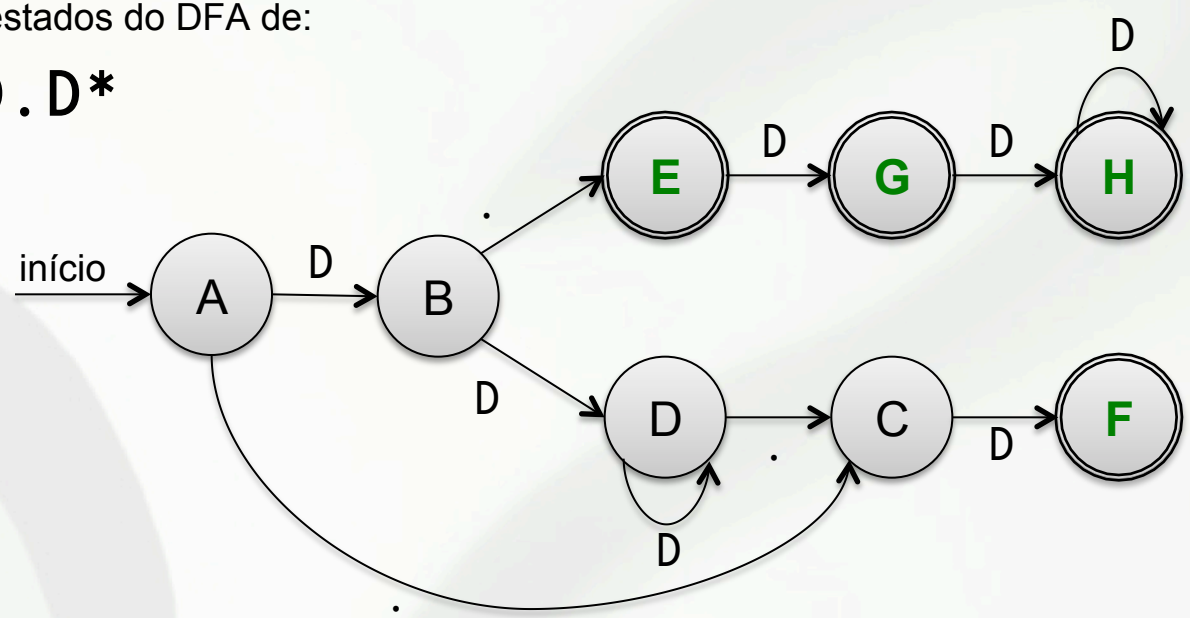
	a	b
AC	B	AC
B	B	D
D	B	**E**
E	B	AC

Damos o nome AC a qualquer um dos estados A e C.
Ao substituir na tabela, duas linhas ficam idênticas: eliminamos uma.
A tabela do DFA tem assim 4 estados.

Exercício

Minimize os estados do DFA de:

D*.D|D.D*



DFA	NFA (E)	D	.
A	{0,1,2,4,7}	B	C
B	{2,3,4,8}	D	E
C	{5}	F	∅
D	{2,3,4}	D	C

DFA	NFA (E)	D	.
E	{5,9,10,12,13}	G	∅
F	{6,13}	∅	∅
G	{6,10,11,12,13}	H	∅
H	{10,11,12,13}	H	∅

Questões?