

Linguagens de Programação e Compiladores

Cap.2 - Análise Léxica (Pt.3)



Prof. MSc. Renzo P. Mesquita
renzo@inatel.br

Objetivos

- Verificar o que são Autômatos Finitos Não-Deterministas (NFAs) e compreender suas principais diferenças em relação aos Autômatos Finitos Deterministas (DFAs);
- Verificar o que é a Construção de Thompson e como ela auxilia na construção de um NFA capaz de representar uma dada Expressão Regular (REGEX);
- Compreendermos como é possível transformar um AFN em um AFD.



2. Análise Léxica

PARTE 3

2.9. Introdução

2.10. Autômatos Finitos Não-Deterministas;

2.11. Construção de Thompson;

2.12. Transformação AFN para AFD.

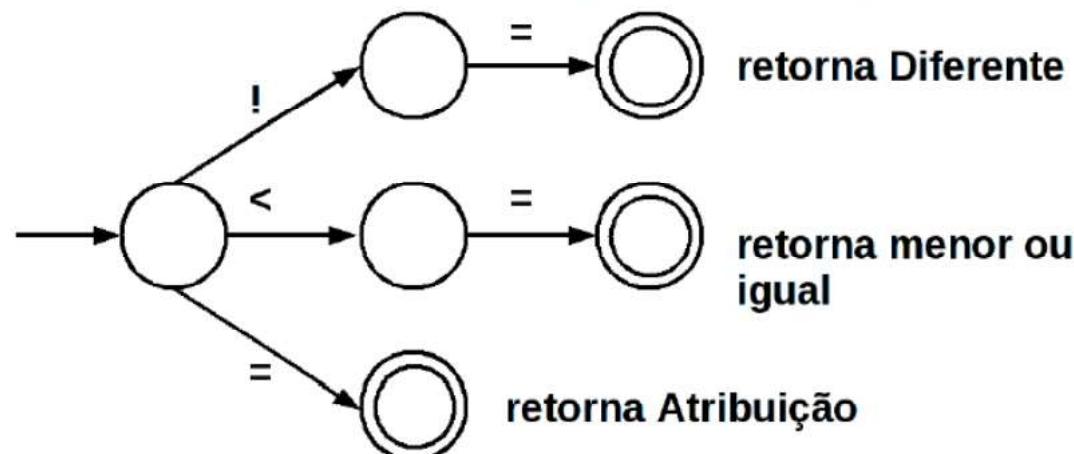


2.9. Introdução

Em uma Linguagem de Programação cada Token pode ser reconhecido por seu próprio Autômato Finito Determinístico;

Por exemplo, cada um dos Tokens !=, <=, = começa com um caracter diferente podendo identificar seus estados iniciais no Autômato.

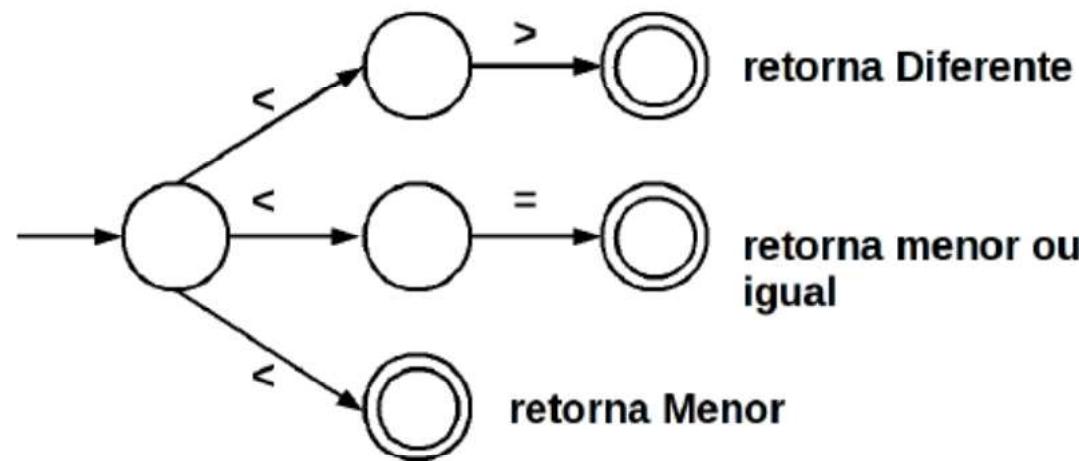
Ex:



2.9. Introdução

Suponha que agora tenhamos vários Tokens que começam com o mesmo caracter <, <= e >.

Ex:



Para este caso, infelizmente ele não se encaixa em um Autômato Finito Determinista, pois em um AFD, dado uma entrada, deve existir sempre uma única transição para um novo estado.

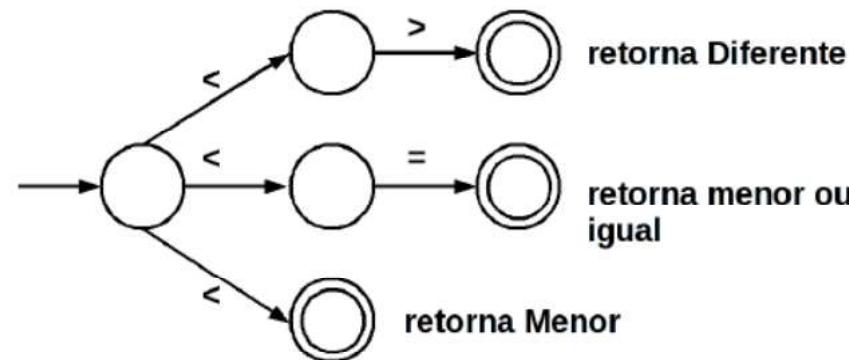
Que solução poderíamos adotar para que esta situação se encaixe em um AFD?



2.9. Introdução

EXEMPLO 1

Que solução poderíamos adotar para que esta situação se encaixe em um AFD?



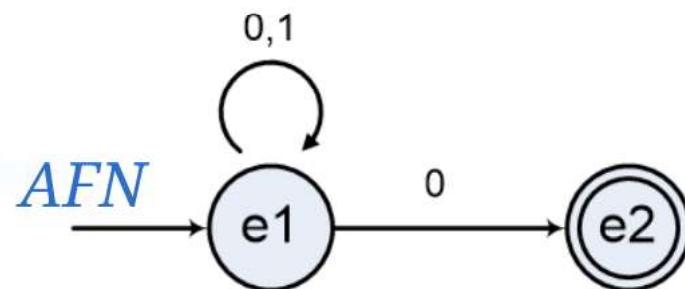
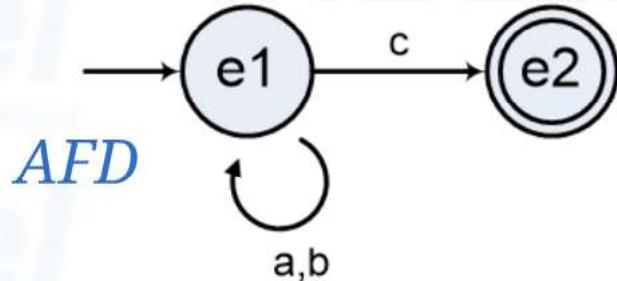
2.9. Introdução

O "Não-Determinismo" é um conceito útil que tem tido grande impacto sobre a teoria da computação.

A Computação é dita DETERMINISTA, quando uma máquina está em um estado e quando lê o próximo símbolo de entrada, sabe-se exatamente qual será o próximo estado, ou seja, o próximo estado está DETERMINADO;

A Computação é dita NÃO-DETERMINISTA, quando uma máquina está em um estado e quando lê o próximo símbolo de entrada, não sabe-se exatamente qual será o próximo estado, ou seja, pode-se assumir vários estados diferentes;

Ex:



2.9. Introdução

As 5 Principais Diferenças entre um DFA e um NFA são:

DFA

- A partir de uma nova entrada, a saída já é previsível;
- No geral gasta menor tempo para reconhecer uma String;
- Não pode utilizar transições com vazio;
- É mais difícil de construir;
- Todo DFA é um NFA;

NFA

- A partir de uma nova entrada, pode-se ter várias saídas;
- No geral gasta maior tempo para reconhecer uma String;
- Pode utilizar transições com vazio;
- É mais fácil de construir;
- Nem todo NFA é um DFA;

2.10. Autômatos Finitos Não-Deterministas

A definição formal de um AFN é bastante similar a dos AFDs. Ambos tem:

A) Estados (Q);

B) Um Alfabeto de entrada (Σ);

C) Uma Função de Transição (δ);

D) Um Estado Inicial (q_0);

E) Uma Coleção de Estados de Aceitação (F);

$$A = (Q, \Sigma, \delta, q_0, F)$$

A diferença entre eles está no tipo de Função de Transição:

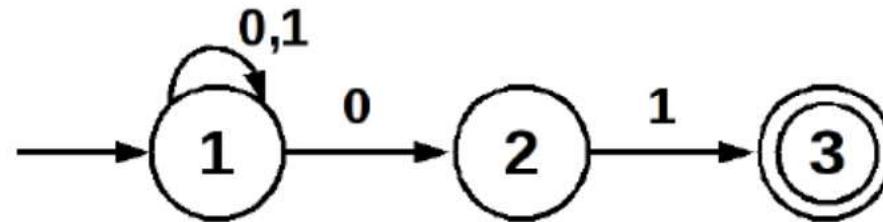
Autômato Finito Determinista: a função de transição recebe um estado, um símbolo de entrada e retorna o próximo estado;

Autômato Finito Não-Determinista: A função de transição recebe um estado, um símbolo de entrada e retorna um conjunto de um ou mais estados;

2.10. Autômatos Finitos Não-Deterministas

EXEMPLO 2

Para o AFN abaixo, represente sua Quíntupla $A = \{Q, \Sigma, \delta, q_0, F\}$.

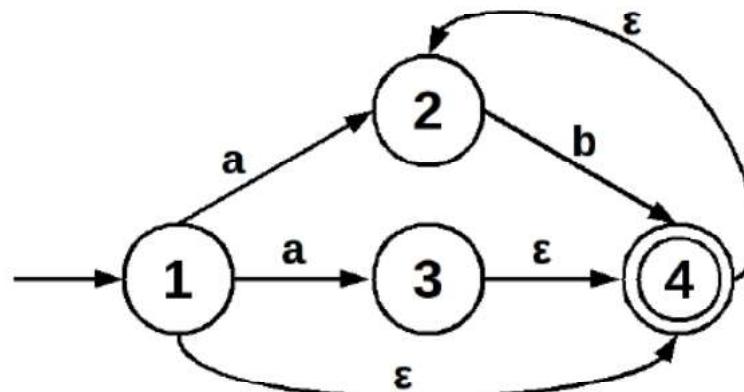


2.10. Autômatos Finitos Não-Deterministas

Uma extensão dos Autômatos Finitos é a possibilidade de **transições sobre ϵ** (String Vazia);

- Um AFN tem a permissão para fazer uma transição espontâneamente, sem receber nenhum símbolo de entrada;
- Cada ϵ encontrado ao longo do caminho é invisível, isto é, ele não contribui com nada para a String formada ao longo do caminho;

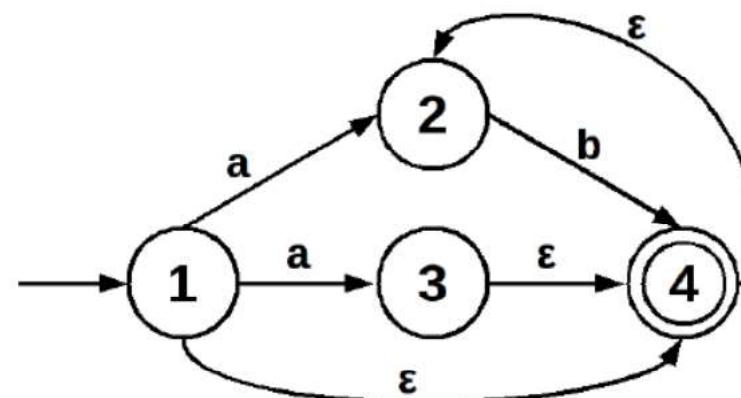
Ex:



2.10. Autômatos Finitos Não-Deterministas

EXEMPLO 3

A partir do AFN abaixo, que caminho mínimo podemos tomar para representar as seguintes Expressões Regulares?



- A) abb
- B) ab^+
- C) ab^*
- D) b^*



2.10. Autômatos Finitos Não-Deterministas

EXEMPLO 4

Construa AFNs que reconheçam as seguintes linguagens sobre o alfabeto $S = \{0,1\}$:

- A) $L1 = \{w \mid w \in \Sigma^* \text{ e } w \text{ começa com } 1 \text{ e termina com } 0\};$
- B) $L2 = \{w \mid w \in \Sigma^* \text{ e } w \text{ termina em } 00\};$
- C) $L3 = \{w \mid w \in \Sigma^* \text{ e } w \text{ possua a substring } 01\};$

Dica: podemos resolvê-los apenas com 3 estados ;)

2.11. Construção de Thompson

Como podemos criar um NFA- ϵ a partir de uma Expressão Regular?

A *Construção de Thompson* permite criar um Autômato Finito Não-Determinístico a partir de uma Expressão Regular, e para isso, alguns passos devem ser obedecidos:

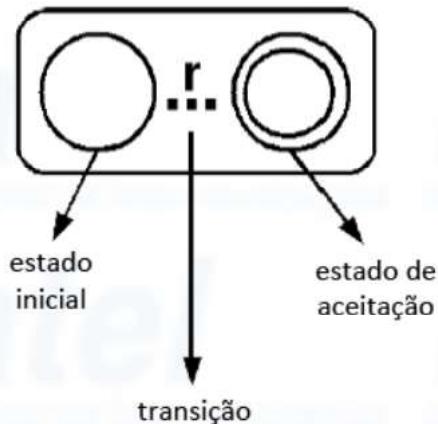
- 1) *Primeiro construímos um NFA para cada subexpressão;*
- 2) *Utilizamos ϵ -transições para juntar cada pedaço de uma Expressão Regular e formar uma máquina correspondente à expressão toda;*
- 3) *Cada operação de um REGEX pode ser obtida pela conexão de NFAs das subexpressões;*

Vamos ver como isso funciona?

2.11. Construção de Thompson

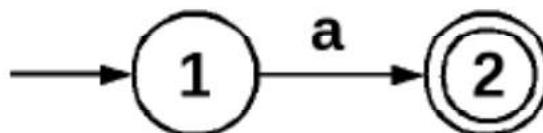
2.11.1. Expressões Regulares Básicas

Uma Expressão Regular básica tem a forma r , ϵ , ou \emptyset , onde:



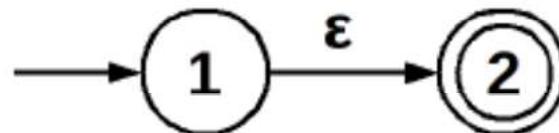
- r representa o casamento de um único caracter do Alfabeto;

Ex:



- ϵ representa um casamento com a cadeia vazia;

Ex:



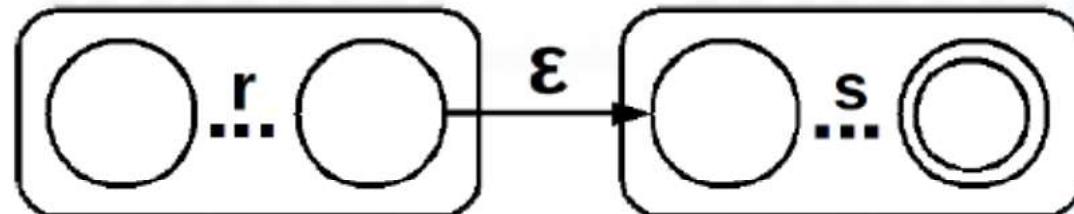
2.11. Construção de Thompson

2.11.2. Concatenação



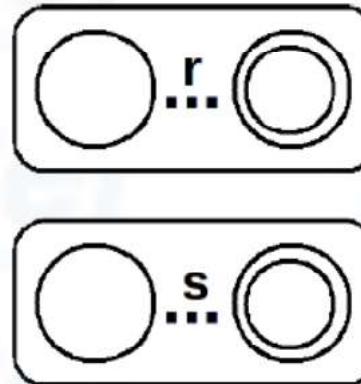
A Expressão Regular tem o estado inicial de r e o estado de aceitação de s . Logo, $L(rs) = L(r)L(s)$, correspondente à Expressão Regular rs .

Ex:

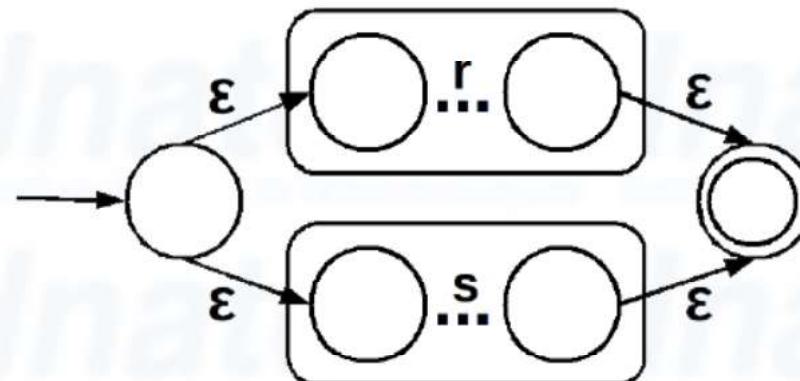


2.11. Construção de Thompson

2.11.3. União

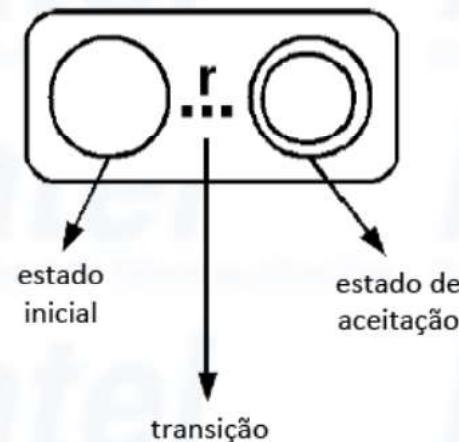


É adicionado um novo estado inicial e um novo estado de aceitação que são conectados por ϵ -transições, formando assim a Expressão Regular $r \mid s$;

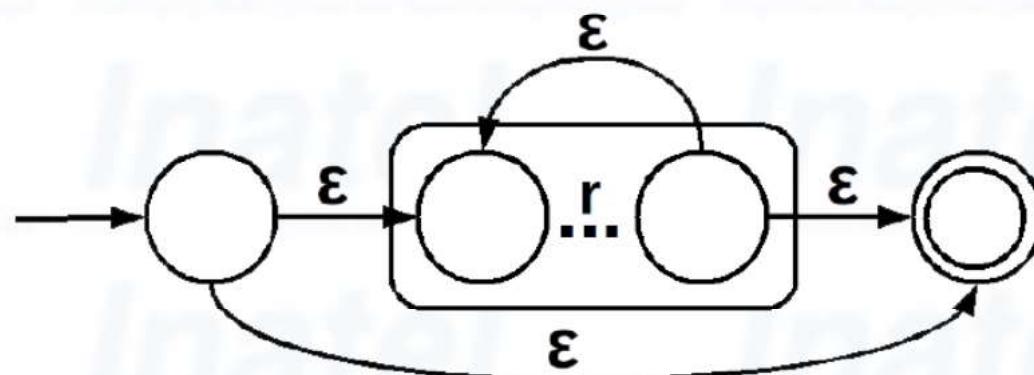


2.11. Construção de Thompson

2.11.4. Repetição (Fechamento *)

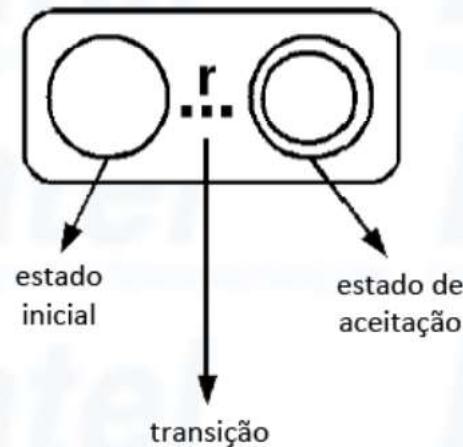


Para que cadeias vazias sejam aceitas, inserimos uma ϵ -transição do novo estado inicial para o estado de aceitação. Formando assim a Expressão Regular r^* :

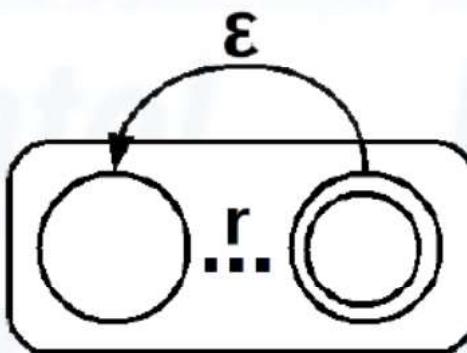


2.11. Construção de Thompson

2.11.4. Repetição (+)



A repetição aparece na nova ϵ -transição **do estado de aceitação de r para seu estado inicial**, permitindo que a máquina de r seja percorrida uma ou mais vezes.



2.11. Construção de Thompson

EXEMPLO 5

Traduza a Expressão Regular $ab|a$ em um AFN utilizando a Construção de Thompson.

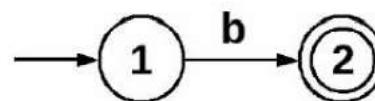
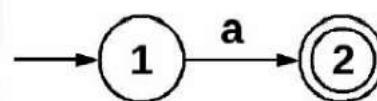


2.11. Construção de Thompson

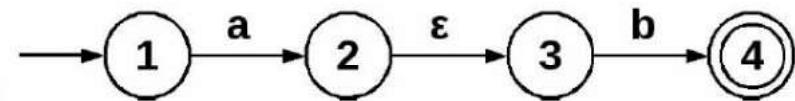
EXEMPLO 5

Traduza a Expressão Regular $ab|a$ em um AFN utilizando a Construção de Thompson.

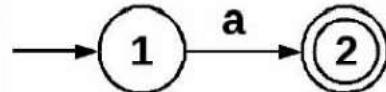
1) Expressões Básicas ('a' e 'b')



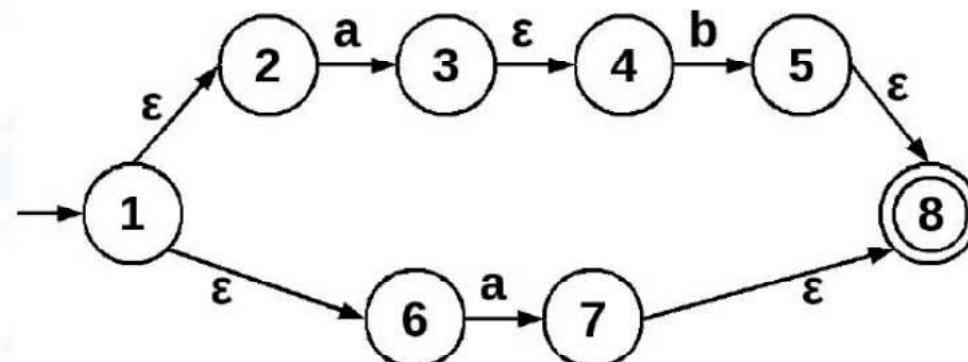
2) Concatenação (ab)



3) Cópia da ER de 'a'



4) União de 2 e 3



2.11. Construção de Thompson

EXEMPLO 6

Traduza a Expressão Regular letra(letra | digito) em um AFN utilizando a Construção de Thompson.*

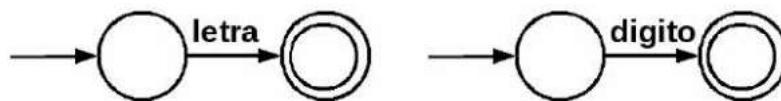


2.11. Construção de Thompson

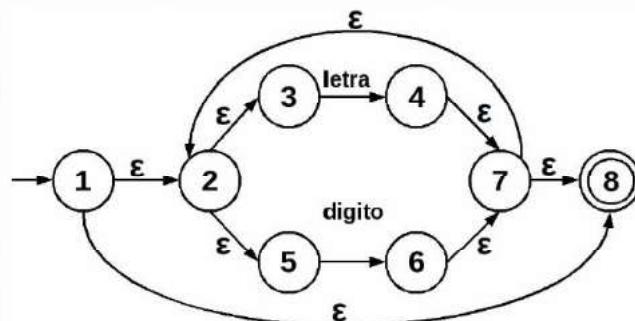
EXEMPLO 6

Traduza a Expressão Regular $\text{letra}(\text{letra} \mid \text{digito})^*$ em um AFN utilizando a Construção de Thompson.

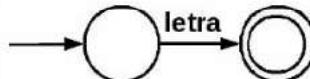
1) Expressões Básicas ('letra' e 'digito')



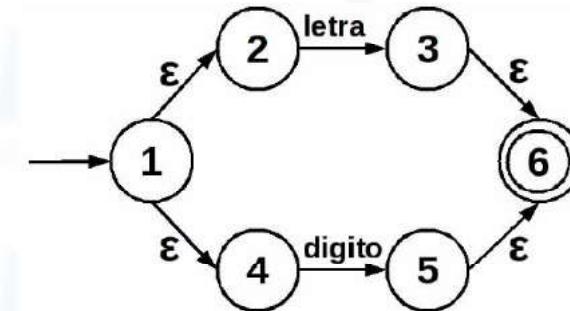
3) Fechamento de 2



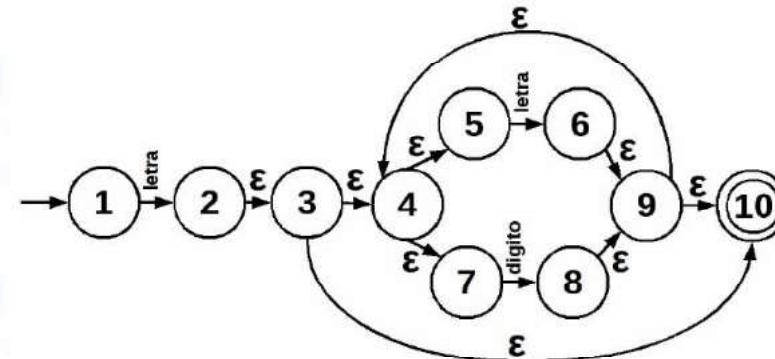
4) Cópia da ER 'letra'



2) União (letra | digito)



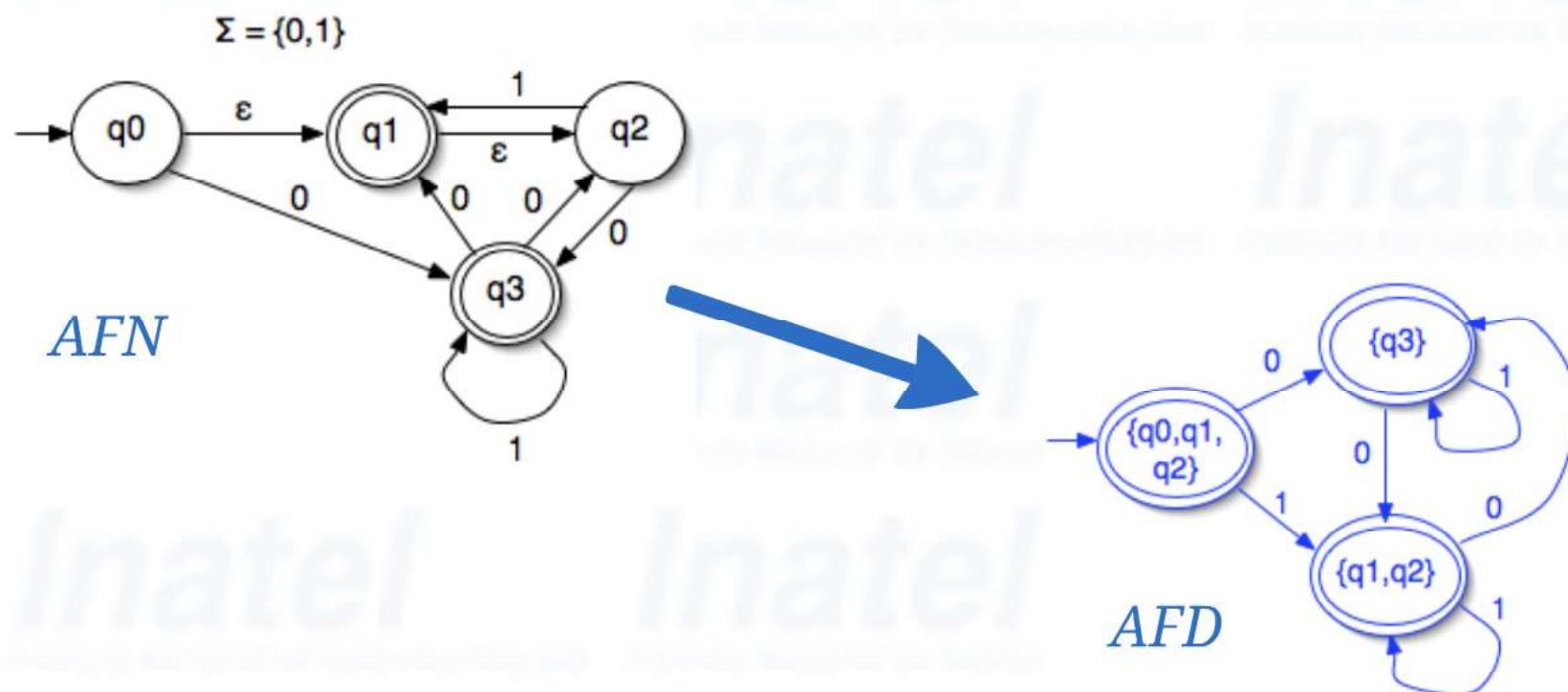
5) Concatenação de 4 e 2



2.12. Transformação AFN para AFD

Um AFN pode se transformar em um AFD equivalente.
Dois Autômatos são considerados equivalentes se aceitam
a mesma linguagem.

Ex:



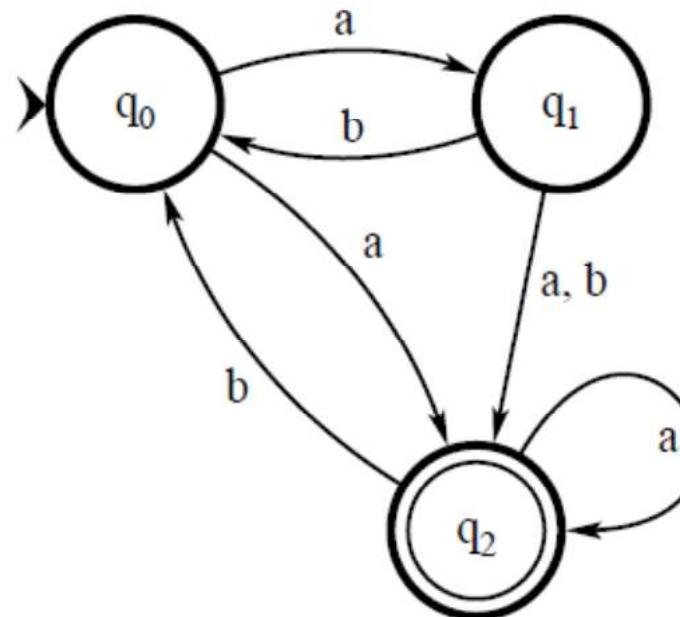
2.12. Transformação AFN para AFD

2.12.1 AFN (sem ϵ -transição) para AFD

Para construir um AFD a partir de um AFN qualquer, devemos passar por 8 passos.

A fim de facilitar o entendimento, vamos demonstrar a transformação por meio de um exemplo.

Considere o seguinte AFN:



2.12. Transformação AFN para AFD

2.12.1 AFN (sem ε -transição) para AFD

PASSO 1) Construir a tabela de Transições do AFN;

	a	b
$\rhd q_0$	$\{q_1, q_2\}$	-
q_1	$\{q_2\}$	$\{q_0, q_2\}$
$* q_2$	$\{q_2\}$	$\{q_0\}$

PASSO 2) Construir a tabela de Transições do AFD através do produto cartesiano dos estados do AFN, incluindo como último conjunto o vazio;

	a	b
$S_0 = \{q_0\}$		
$S_1 = \{q_1\}$		
$S_2 = \{q_2\}$		
$S_3 = \{q_0, q_1\}$		
$S_4 = \{q_0, q_2\}$		
$S_5 = \{q_1, q_2\}$		
$S_6 = \{q_0, q_1, q_2\}$		
$S_7 = \{\}$		

Obs.: Sempre existirão 2^k combinações, onde k é o número de estados do AFN.

2.12. Transformação AFN para AFD

2.12.1 AFN (sem ε -transição) para AFD

PASSO 3) Indicar todos os conjuntos que contém como elemento Estados de Aceitação. Estes conjuntos serão candidatos a Estados de Aceitação do AFD.

	a	b
$S_0 = \{q_0\}$		
$S_1 = \{q_1\}$		
$*S_2 = \{q_2\}$		
$S_3 = \{q_0, q_1\}$		
$*S_4 = \{q_0, q_2\}$		
$*S_5 = \{q_1, q_2\}$		
$*S_6 = \{q_0, q_1, q_2\}$		
$S_7 = \{ \}$		

2.12. Transformação AFN para AFD

2.12.1 AFN (sem ε -transição) para AFD

PASSO 4) Verificar a ocorrência de cada conjunto do AFD em relação a um símbolo e colocar como resultado o conjunto correspondente que pertence ao AFD. Quando existir mais de um elemento no conjunto, a ocorrência passa a ser a união das ocorrências de todas as transações.

	a	b
$S_0 = \{q_0\}$	S_5	S_7
$S_1 = \{q_1\}$	S_2	S_4
$*S_2 = \{q_2\}$	S_2	S_0
$S_3 = \{q_0, q_1\}$	S_5	S_4
$*S_4 = \{q_0, q_2\}$	S_5	S_0
$*S_5 = \{q_1, q_2\}$	S_2	S_4
$*S_6 = \{q_0, q_1, q_2\}$	S_5	S_4
$S_7 = \{ \}$	S_7	S_7

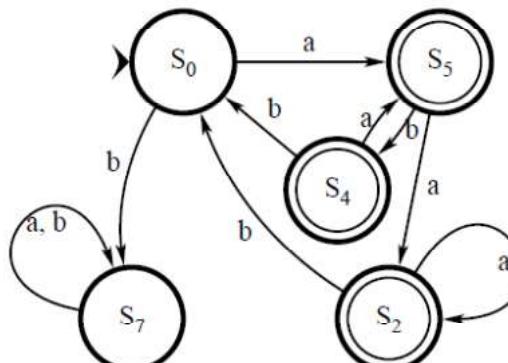
2.12. Transformação AFN para AFD

2.12.1 AFN (sem ϵ -transição) para AFD

PASSO 5) Eliminar as linhas que possuem transições somente com saídas, ou seja, não existe nenhuma transição que chega até ela (estado inacessível);

	a	b
$\rightarrow S_0 = \{q_0\}$	S_5	S_7
$*S_2 = \{q_2\}$	S_2	S_0
$*S_4 = \{q_0, q_2\}$	S_5	S_0
$*S_5 = \{q_1, q_2\}$	S_2	S_4
$S_7 = \{ \}$	S_7	S_7

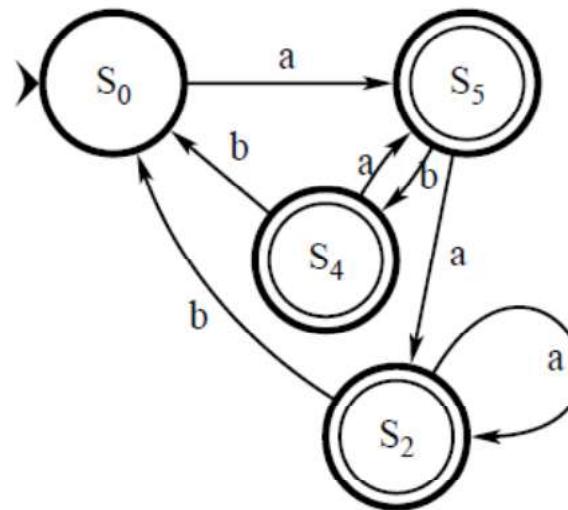
PASSO 6) Montar o AFD a partir da Tabela de Transições do passo 5;



2.12. Transformação AFN para AFD

2.12.1 AFN (sem ε -transição) para AFD

PASSO 7) Eliminar os estados que não possuem saída para outro estado e não são finais;

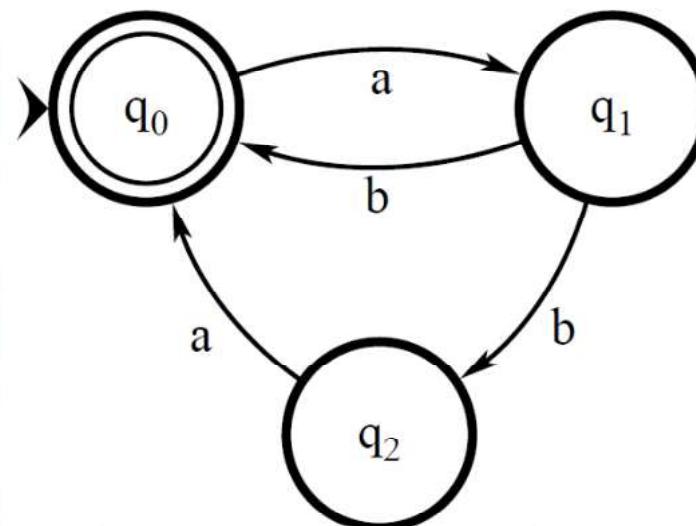


PASSO 8) Neste momento a transformação já está pronta! Basta apenas testar se as entradas do AFN também são válidas no AFD resultante.

2.12. Transformação AFN para AFD

EXEMPLO 7

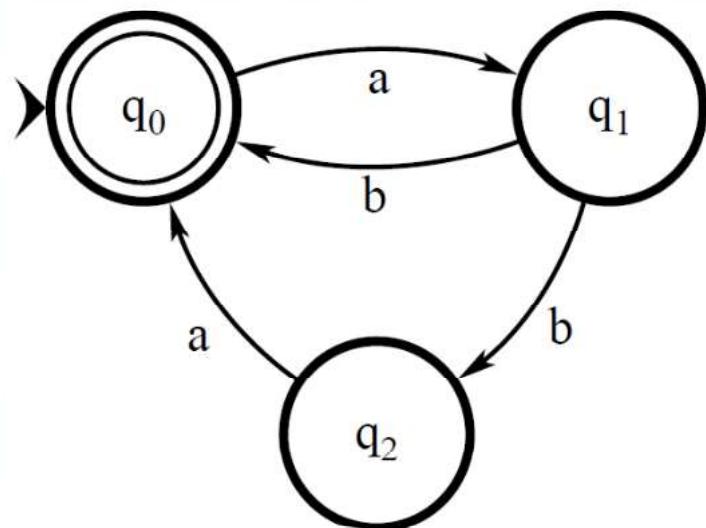
Como ficaria a transformação do AFN abaixo em um AFD?



2.12. Transformação AFN para AFD

EXEMPLO 7

Como ficaria a transformação do AFN abaixo em um AFD?

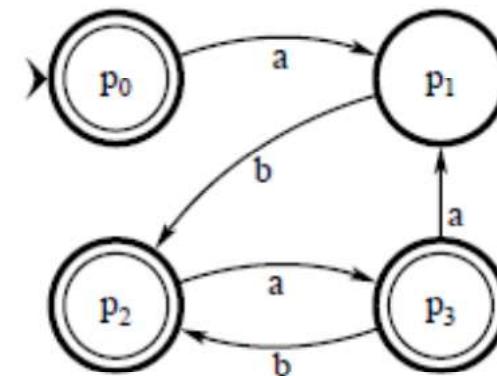


AFN



AFD

$$\begin{aligned} p_0 &= \{q_0\} \\ p_1 &= \{q_1\} \\ p_2 &= \{q_0, q_2\} \\ p_3 &= \{q_0, q_1\} \end{aligned}$$



2.12. Transformação AFN para AFD

2.12.2 AFN (com ϵ -transição) para AFD

Para descrever um Algoritmo que, dado um AFN arbitrário (com transições vazias), construa um AFD equivalente, é necessário:

- Um **método que elimine as ϵ -transições e as múltiplas transições** a partir de um estado para um caracter de entrada único;
- A eliminação das ϵ -transições requer a construção de **ϵ -fechos**;

Mas afinal, o que é um ϵ -fecho?

ϵ -fecho é o conjunto de todos os estados atingíveis por ϵ -transições a partir de um estado.

Tanto o processo de eliminação de ϵ -transições e múltiplas transições a partir de um estado para um caracter consideram conjuntos de estados em vez de estados únicos;



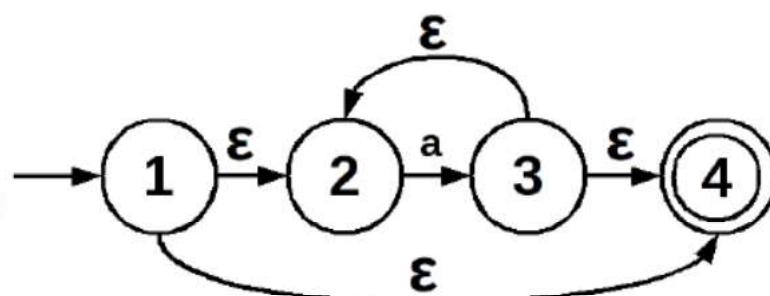
2.12. Transformação AFN para AFD

2.12.2 AFN (com ϵ -transição) para AFD

- O ϵ -fecho de um único estado s é definido como sendo o conjunto de estados atingíveis por uma série com zero ou mais ϵ -transições;
- O conjunto é denotado por \bar{s} ;
- O ϵ -fecho de um estado sempre contém o próprio estado;

EXEMPLO 8

Para o AFN abaixo correspondente à Expressão Regular a^* , encontre o ϵ -fecho de cada um dos estados.



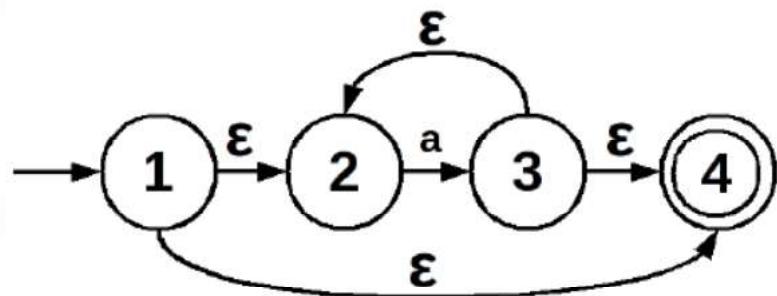
2.12. Transformação AFN para AFD

2.12.2 AFN (com ε -transição) para AFD

- O ε -fecho de um único estado s é definido como sendo o conjunto de estados atingíveis por uma série com zero ou mais ε -transições;
- O conjunto é denotado por \bar{s} ;
- O ε -fecho de um estado sempre contém o próprio estado;

EXEMPLO 8

Para o AFN abaixo correspondente a Expressão Regular a^* , encontre o ε -fecho de cada um dos estados.



$$\bar{1} = \{1,2,4\}$$

$$\bar{2} = \{2\}$$

$$\bar{3} = \{2,3,4\}$$

$$\bar{4} = \{4\}$$

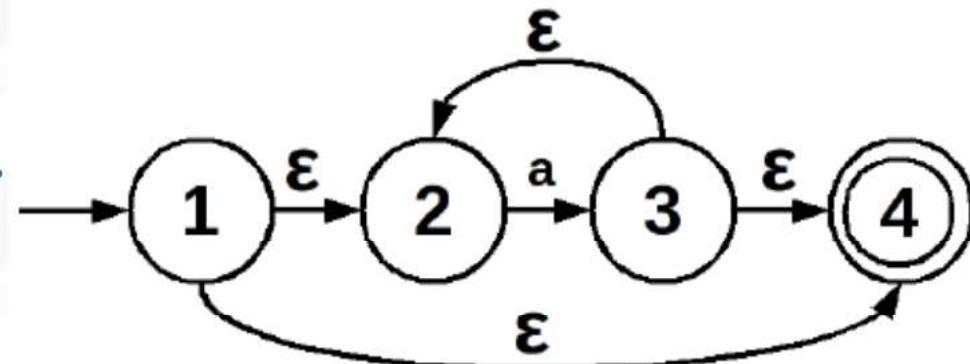
2.12. Transformação AFN para AFD

2.12.2 AFN (com ε -transição) para AFD

Para construir um AFD a partir de um AFN- ε qualquer, devemos passar por **6 passos**.

A fim de facilitar o entendimento, vamos demonstrar a transformação por meio de um exemplo.

Considere o seguinte AFN- ε :



O AFD construído a partir de um AFN M, denomina-se \overline{M} ;

2.12. Transformação AFN para AFD

2.12.2 AFN (com ε -transição) para AFD

PASSO 1) Computa-se o ε -fecho do estado inicial de M , que passa a ser o estado inicial de \bar{M} ;

$$\overline{1} = \{1,2,4\}$$

PASSO 2) Criamos uma tabela que descreve os fechamentos encontrados em M , descreve um novo nome para estes fechamentos a fim de representar um novo estado em \bar{M} e mostra quais as saídas para todas as possíveis entradas;

	Estado do AFN	Estado do AFD	a
›	{1,2,4}	A	

:

2.12. Transformação AFN para AFD

2.12.2 AFN (com ε -transição) para AFD

PASSO 3) Para o conjunto computa-se transições de caracteres "a" (a representa qualquer caractere do alfabeto) da seguinte maneira:

Dado um conjunto S de estados e um caracter "a" do alfabeto:

- *Computa-se o conjunto S_a , onde $S_a = \{t \mid \text{para algum } s \text{ em } S, \text{ existe uma transição de } s \text{ para } t \text{ (próximo estado) em } a\};$*
- *Computa-se o ε -fecho $\bar{S_a}$;*

Ex:

Existe uma transição do estado 2 para o estado 3 em a, e nenhuma transição dos estados 1 ou 4 em a de {1,2,4};

Logo $\{1,2,4\}_a = \{\bar{3}\} = \{2,3,4\}$;

Como não existe nenhuma outra transição de qualquer um dos estados 1, 2 e 4, passa-se para o novo estado {2,3,4}.

Estado do AFN	Estado do AFD	a
{1,2,4}	A	B
{2,3,4}	B	

2.12. Transformação AFN para AFD

2.12.2 AFN (com ϵ -transição) para AFD

PASSO 4) O processo anterior é repetido até que novos estados e transições não sejam mais criados;

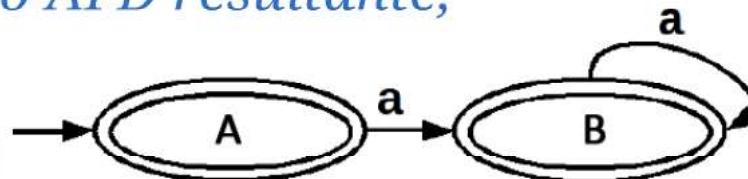
$$\overline{\{2,3,4\}}_a = \overline{\{3\}} = \{2,3,4\}$$

	Estado do AFN	Estado do AFD	a
→	{1,2,4}	A	B
	{2,3,4}	B	B

PASSO 5) Marca como aceitáveis os estados construídos desta maneira que contenham um estado de aceitação de M;

	Estado do AFN	Estado do AFD	a
* →	{1,2,4}	A	B
*	{2,3,4}	B	B

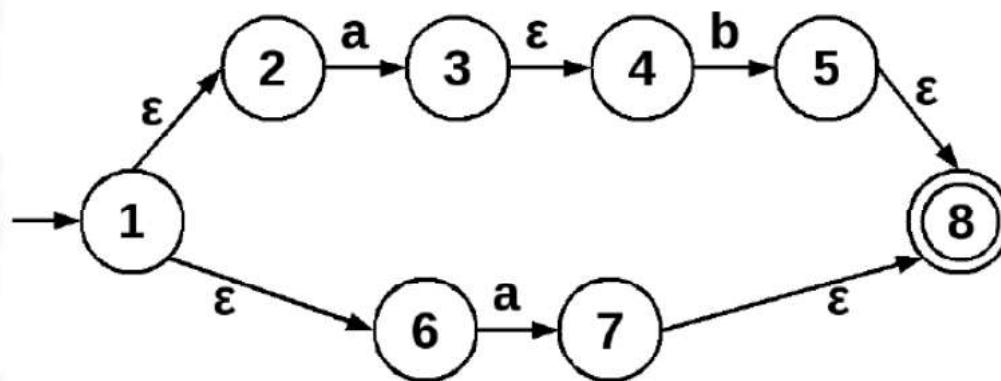
PASSO 6) Desenhe o AFD resultante;



2.12. Transformação AFN para AFD

EXEMPLO 9

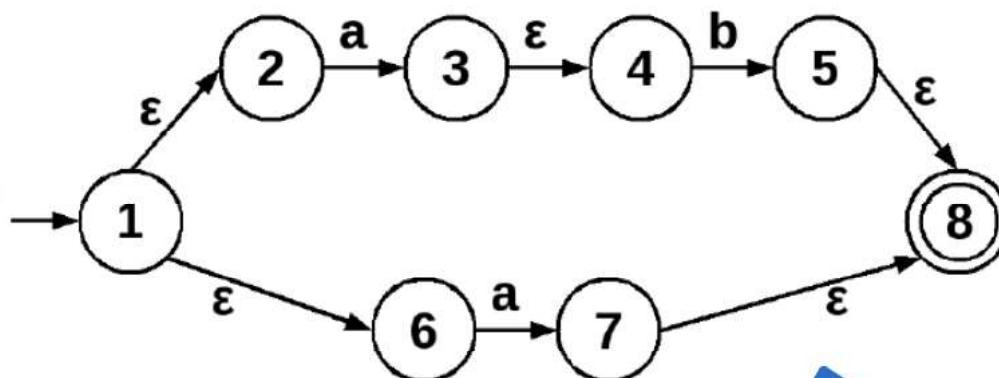
Como ficaria a transformação do AFN- ϵ abaixo em um AFD?



2.12. Transformação AFN para AFD

EXEMPLO 9

Como ficaria a transformação do AFN- ε abaixo em um AFD?



Capítulo 2

Parte 3



EXERCÍCIOS