TEORIA DA COMPUTAÇÃO

Aula 02 – Linguagens Regulares

DIEGO KASUO NAKATA DA SILVA

Visão Geral

Tópicos do capítulo

- Introdução
- Autômatos Finitos
- Não-Determinismo
- Expressões Regulares
- Linguagens Não-Regulares.

INTRODUÇÃO

Introdução

“O que é um computador?”

- É um objeto bastante complicado para permitir que estabeleçamos teorias manipuláveis sobre ele.
- Usaremos um computador mais simples: **modelo computacional**, um computador idealizado.
- Usaremos vários modelos computacionais.
- **Máquina de estados finitos** ou **autômatos finitos** são modelos bem simples.

AUTÔMATOS FINITOS

Autômatos Finitos

- **Autômatos Finitos** são bons modelos para computadores com uma quantidade extremamente limitada de memória.
- Possuem completa falta de memória fora do seu processador central fixo
- Está presente em dispositivos eletromecânicos.
- Recebem como entrada uma *string*, não entrega uma saída e sim, uma indicação de se a entrada foi considerada aceitável ou não.
- É um dispositivo de reconhecimento da linguagem.

Autômatos Finitos

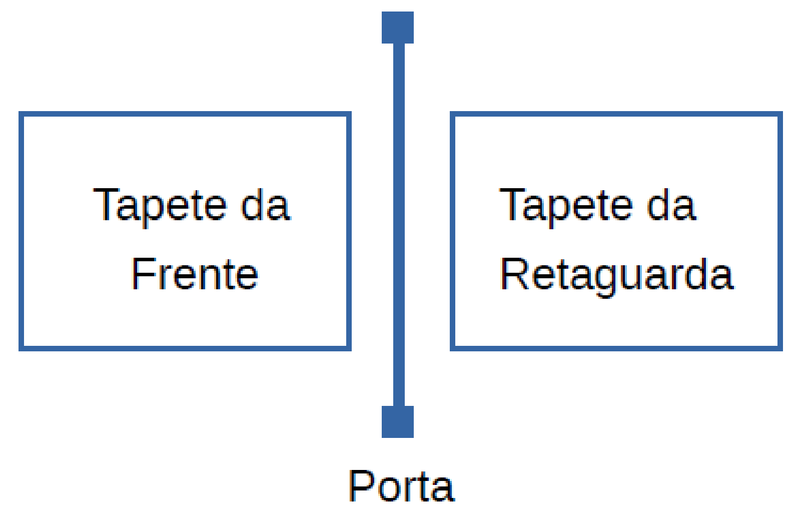
- Como funcionam as portas que abrem e fecham automaticamente?
- E as lavadoras de louça/roupa?
- E os termômetros eletrônicos, relógios digitais, calculadoras e máquinas de venda automática?

Todos esses dispositivos eletromecânicos têm uma controladora que nada mais é do que um autômato finito.

Autômatos Finitos

- <u>Exemplo</u>: Porta Automática

Figura: Visão superior de uma porta automática

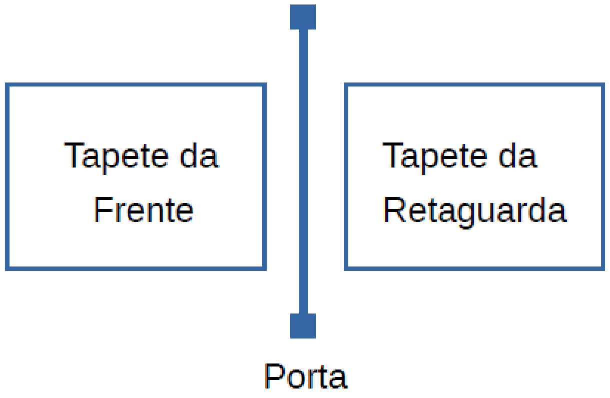


Há um tapete na frente da porta e outro atrás. Ambos detectam a presença de pessoas prestes a atravessar a passagem, abrindo a porta quando alguém se aproxima e mantendo-a aberta até que a pessoa se afaste.

Autômatos Finitos

- Exemplo: Porta Automática

Figura: Visão superior de uma porta automática

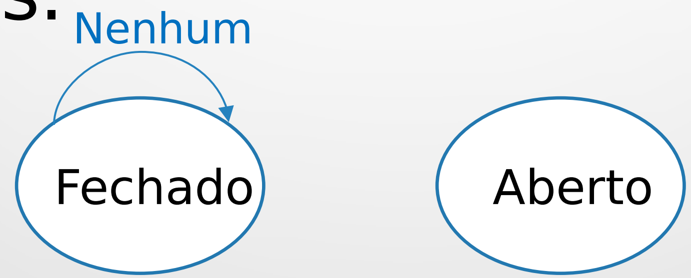


1. O controlador pode estar em um de dois possíveis estados (aberto ou fechado).
2. Há 4 condições de entrada possíveis: frente (uma pessoa está no tapete da frente), atrás (uma pessoa está no tapete de dentro), ambos (há pessoas sobre os dois tapetes) e nenhum (não há ninguém sobre os tapetes).
3. A transição de estado do controlador depende do estímulo (entrada) que recebe.

Autômatos Finitos

- **Diagrama de Estados**

Figura: Estados são representados por círculos e entradas por arcos.



- Estando no estado **fechado** e recebendo como entrada nenhum, o controlador permanece como fechado.

Autômatos Finitos

- **Diagrama de Estados**

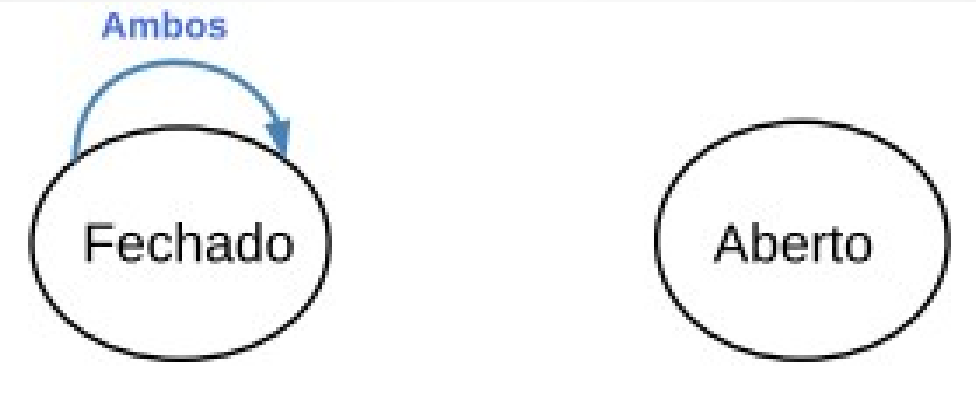
Figura: Estados são representados por círculos e entradas por arcos.

- Estando no estado **aberto** e recebendo como entrada ambos, o controlador permanece no estado aberto.

Autômatos Finitos

- **Diagrama de Estados**

Figura: Estados são representados por círculos e entradas por arcos.

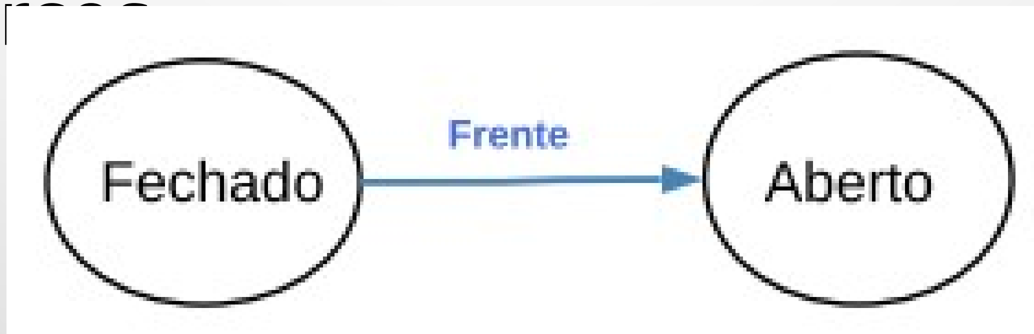


- Estando no estado **fechado** e recebendo como entrada ambos, o controlador permanece como fechado porque se a porta for aberta ela pode bater em alguém sobre o tapete posterior.

Autômatos Finitos

- **Diagrama de Estados**

Figura: Estados são representados por círculos e entradas por ar...

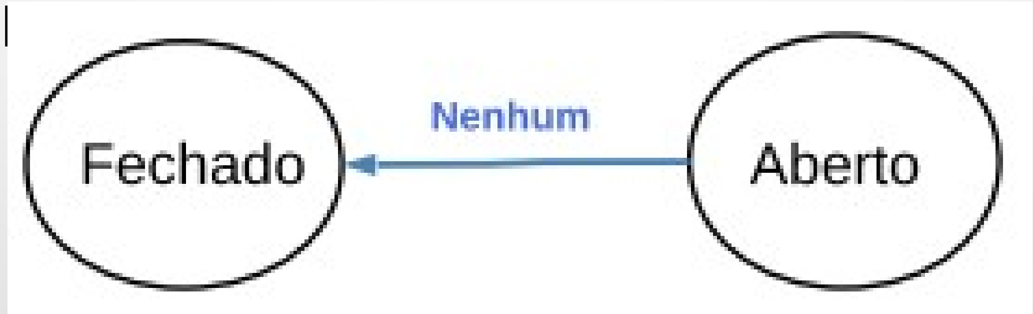


- Estando no estado **fechado** e recebendo como entrada frente, o controlador move para o estado aberto.

Autômatos Finitos

- **Diagrama de Estados**

Figura: Estados são representados por círculos e entradas por a



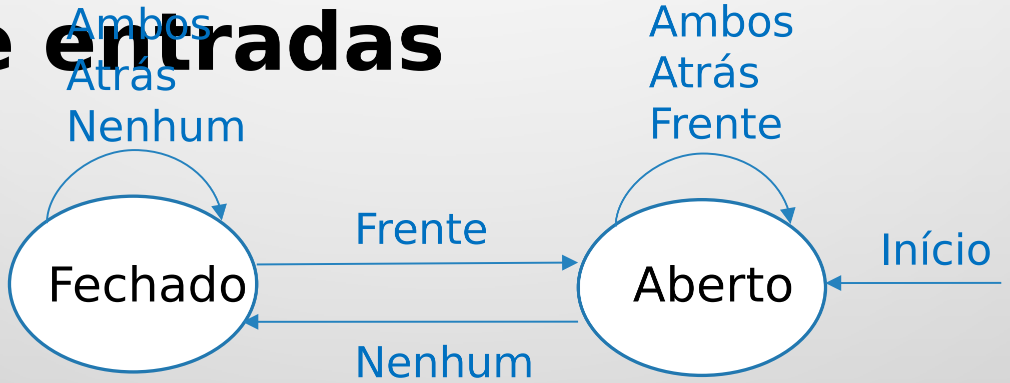
- Estando no estado **aberto** e recebendo como entrada nenhum, o controlador move para o estado fechado.

Autômatos Finitos

- **Tabela de transição de Estado**

	Nenhum	**Frente**	**Atrás**	**Ambos**
Fechado	Fechado	Aberto	Fechado	Fechado
Aberto	Fechado	Aberto	Aberto	Aberto

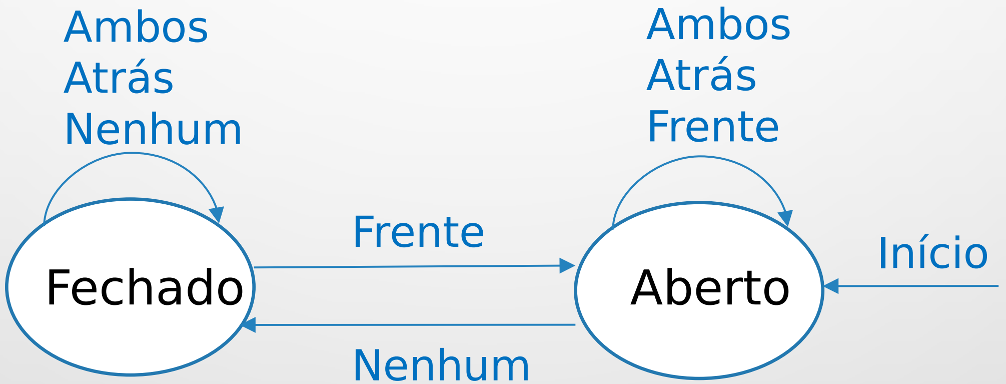
- **Diagrama de estados para uma dada sequência de entradas**



Autômatos Finitos

- **Diagrama de Estados**

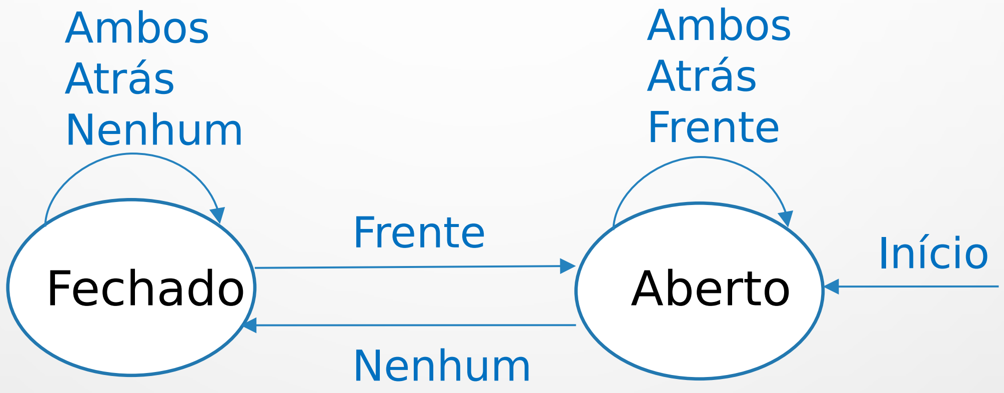
Figura: Diagrama de estados para uma dada sequência de entradas, iniciando-se pelo estado aberto.



- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

Autômatos Finitos

- **Diagrama de Estados**



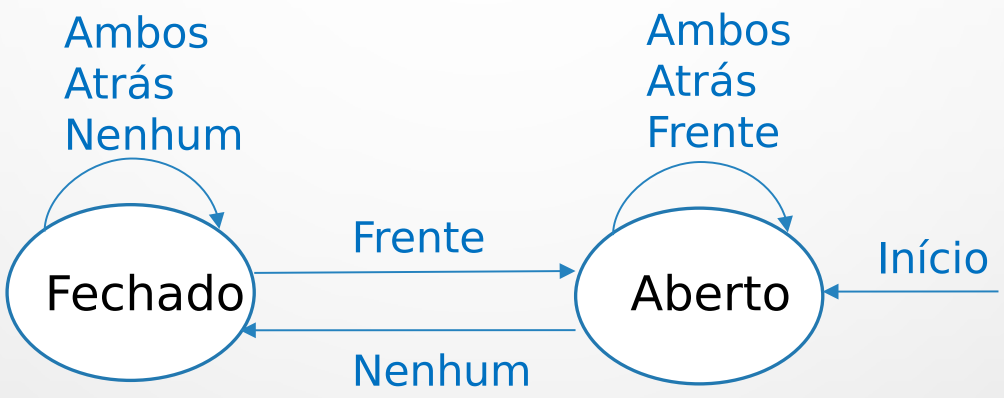
- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

Solução:

aberto (início) → aberto;

Autômatos Finitos

- **Diagrama de Estados**



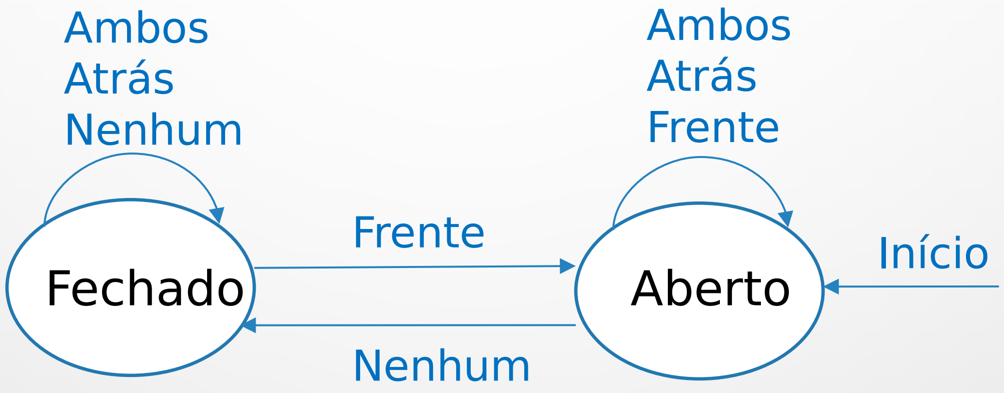
- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

Solução:

aberto (início) → aberto; aberto → aberto;

Autômatos Finitos

- **Diagrama de Estados**



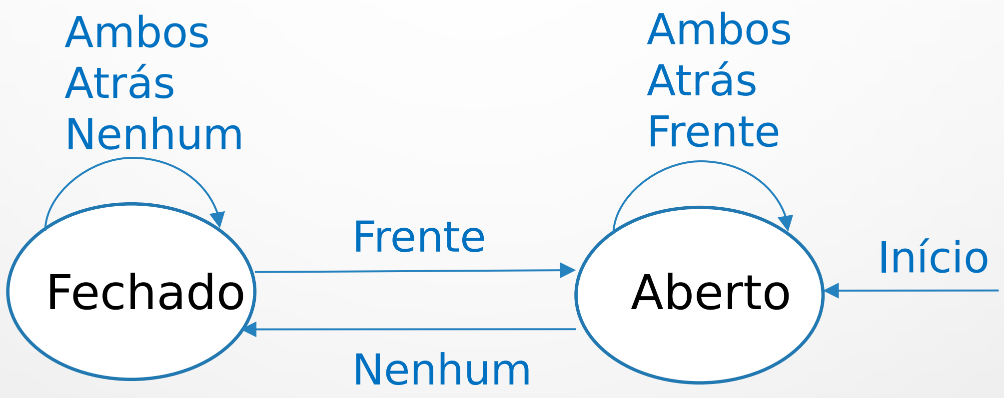
- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

Solução:

aberto (início) → aberto; aberto → aberto; aberto → fechado;

Autômatos Finitos

- **Diagrama de Estados**



- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

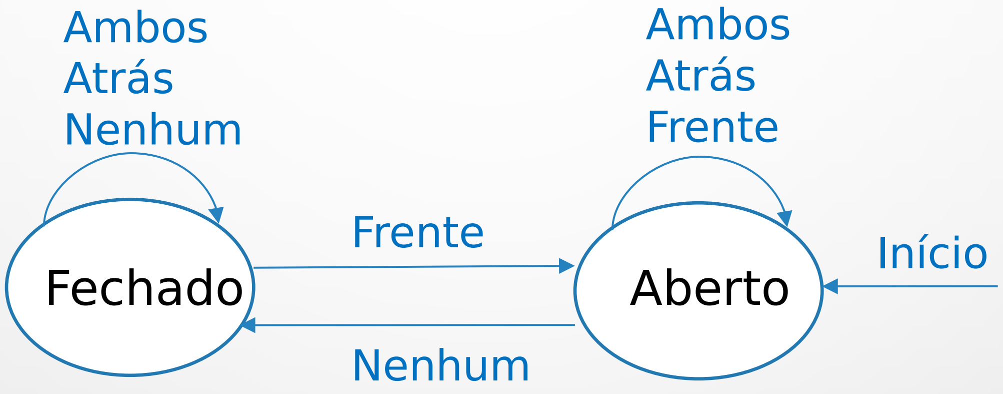
Solução:

aberto (início) → aberto; aberto → aberto; aberto → fechado;

fechado → aberto

Autômatos Finitos

- **Diagrama de Estados**



- Um controlador poderia iniciar no estado **aberto** e receber a série de entradas: frente, atrás, nenhum, frente e ambos. Qual seria o estado final?

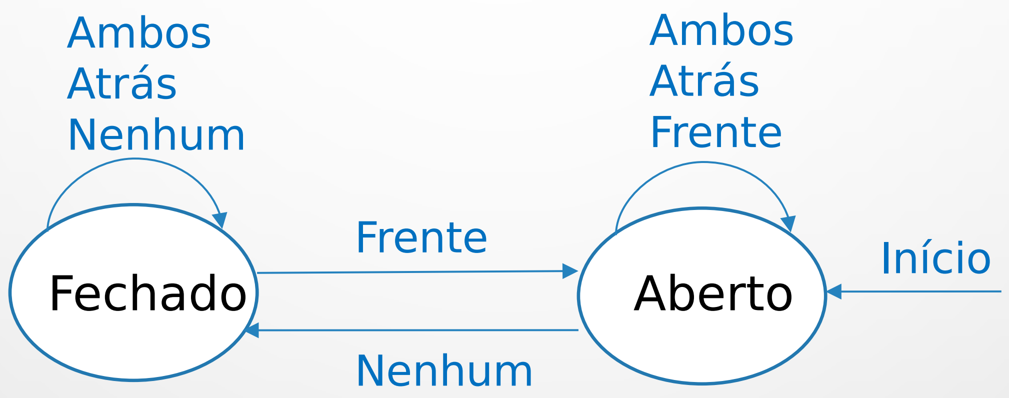
Solução:

aberto (início) → aberto; aberto → aberto; aberto → fechado;

fechado → aberto; aberto → aberto (fim)

Autômatos Finitos

- **Diagrama de Estados**



Um controlador poderia iniciar no estado **fechado** e receber a série de entradas: frente, atrás, nenhum, frente, ambos, nenhum, atrás e nenhum. Qual seria o estado final?

Autômatos Finitos

- **Exemplo: Interruptor de Lâmpada**

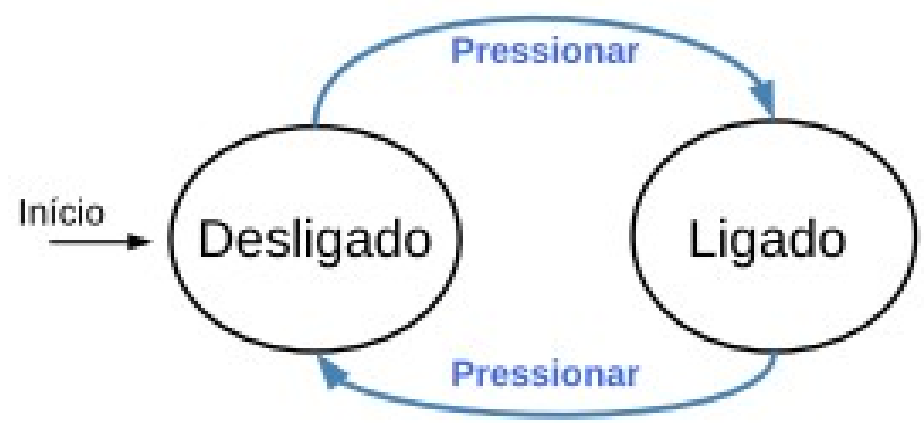
Figura: Lampada e interruptor liga/desliga

O dispositivo memoriza se está no estado ligado ou desligado, e permite que um botão seja pressionado para um efeito diferente, dependendo do estado do interruptor.

Autômatos Finitos

- **Exemplo: Interruptor de Lâmpada**

Figura: Diagrama de estados para o interruptor liga/desliga



Frequentemente o objetivo é designar o estado final ou de aceitação, que é o estado ao qual se chega após uma sequência de entradas.

Como fica a tabela de transição de estados para esse caso?

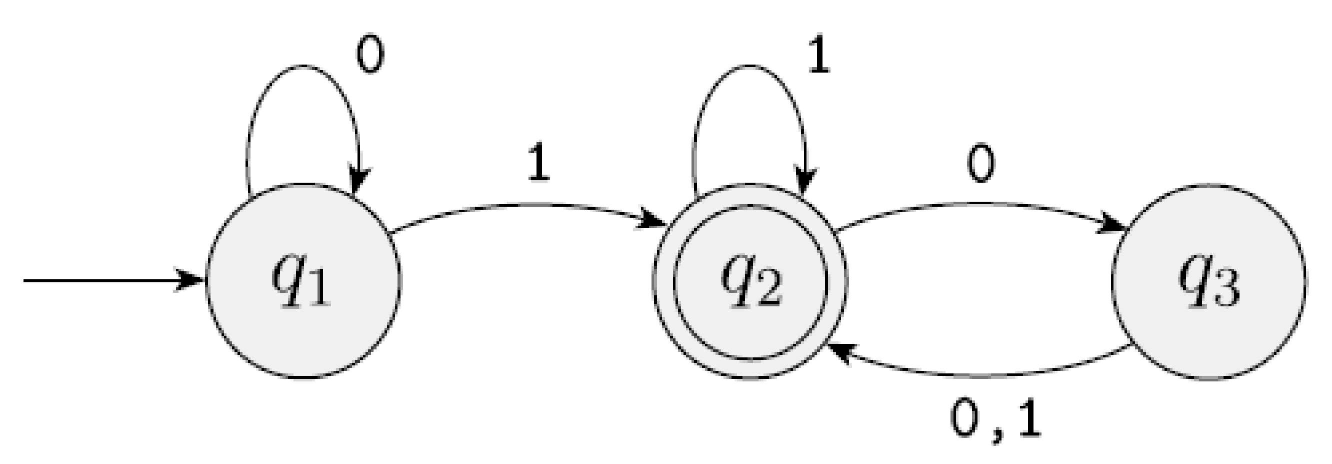
Autômatos Finitos

- Pensar num controlador de porta automática como um autômato finito é útil porque isso sugere formas padrão de representação como nas figuras anteriores. Esse controlador é um computador que tem somente um único bit de memória, capaz de registrar em quais dos dois estados o controlador está.
- Outros dispositivos comuns têm controladores com memórias um pouco maiores. Em um controlador de elevador um estado pode representar o andar no qual o elevador está e as entradas poderiam ser os sinais recebidos dos botões.

Autômatos Finitos

- Autômatos finitos e suas contrapartidas probabilísticas ***cadeias de Markov*** são ferramentas úteis quando estamos tentando reconhecer padrões em dados. Esses dispositivos são utilizados em processamento de voz e em reconhecimento de caracteres óticos. Cadeias de Markov têm sido usadas para modelar e fazer previsões de mudança de preços em mercados financeiros.

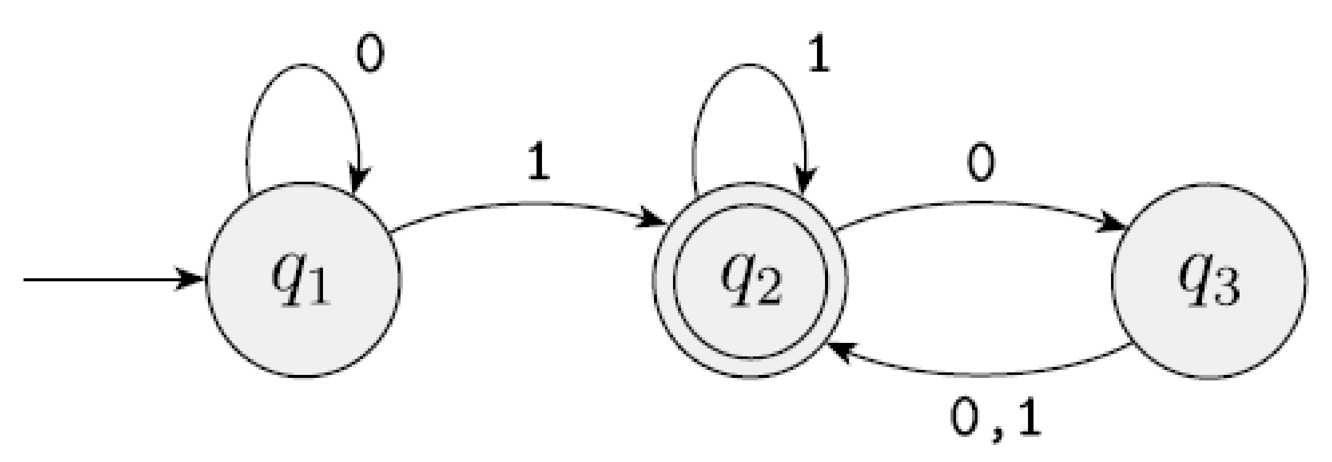
Autômatos Finitos



Autômato finito .

- A figura acima é denominada **diagrama de estado** de .
- O autômatos têm três **estados**: .
- O **estado inicial** é .
- O **estado de aceitação** é .

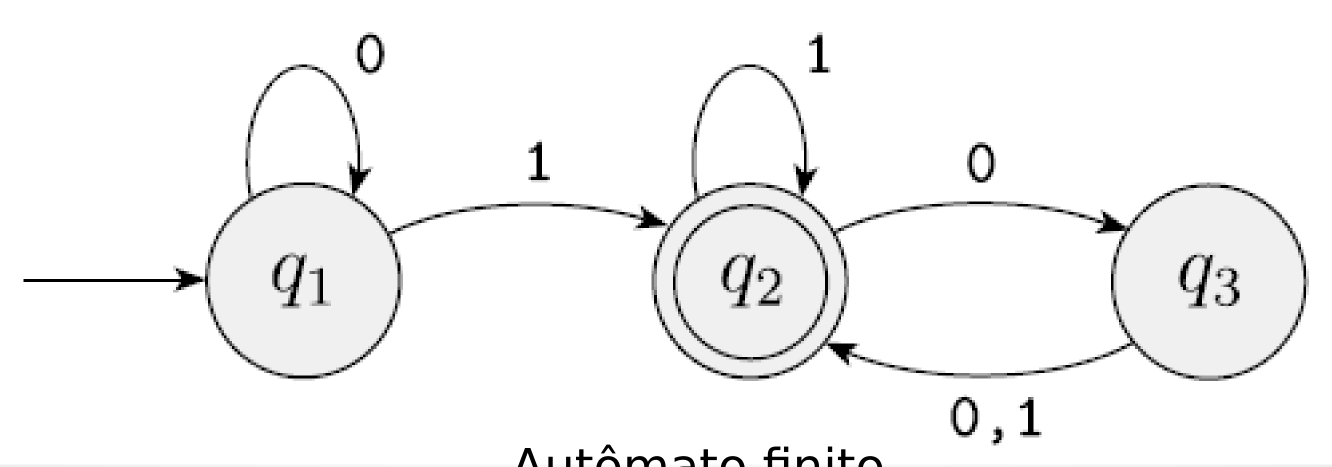
Autômatos Finitos



Autômato finito .

- Se ele recebe 1101, ele processa essa cadeia e produz uma saída.
- A saída é **aceita** ou **rejeita**.
- A saída é **aceita** se está no estado de aceitação e **rejeita** se ele não está.

Autômatos Finitos



Autômato finito .

- Alimentando a cadeia com a entrada 1101, o processamento procede da seguinte forma:

1. Começa no estado no estado .

2. Lê 1, segue a transição de para .

3. Lê 1, segue a transição de para .

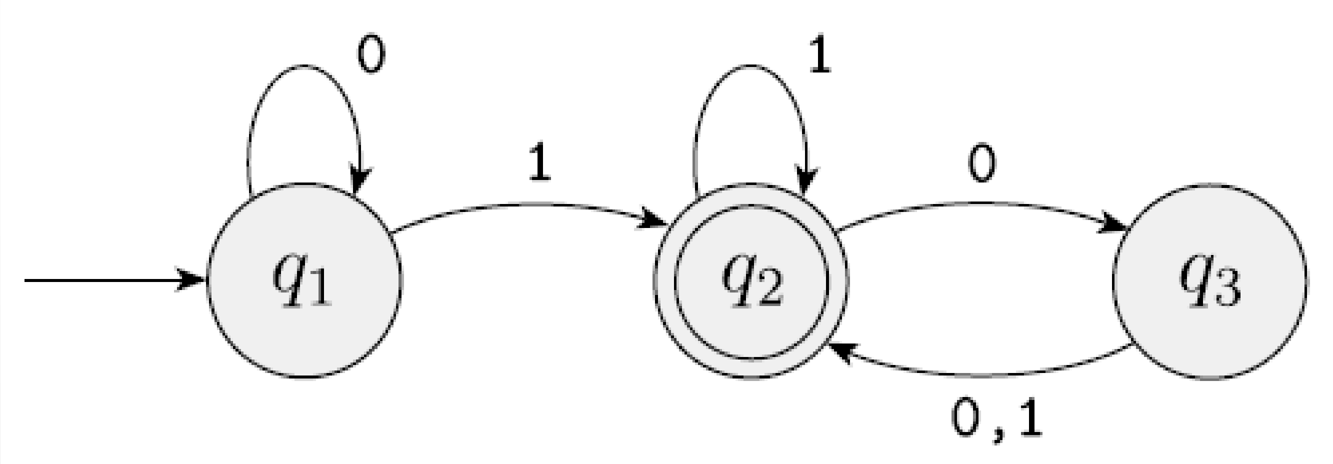
4. Lê 0, segue a transição de para .

5. Lê 1, segue a transição de para .

6. Aceite porque está no estado de transição no final da entrada.

Autômatos Finitos

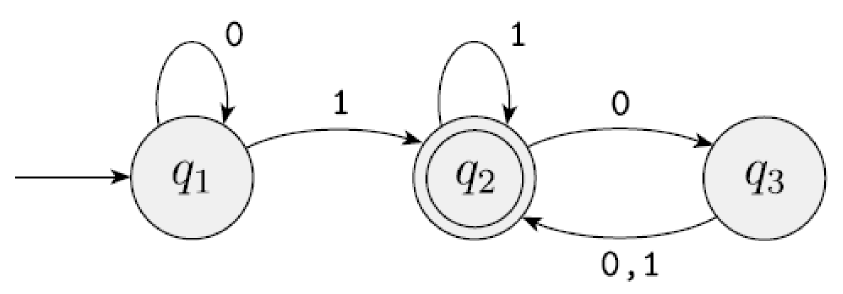
Autômato finito .



- Exemplos: Processar as cadeias abaixo em :

1. 1
2. 01
3. 11
4. 0101010101
5. 100
6. 0100
7. 110000
8. 0101000000

Autômatos Finitos



- Que cadeias aceita? Que cadeias rejeita?
- Os experimentos mostraram que:

➢ aceita qualquer cadeia que termine com o símbolo 1, pois assim vai para o estado de aceitação .

➢ aceita qualquer cadeia que termine com números pares de 0s seguindo o símbolo 1.

➢ rejeita cadeias como 0, 10, 101000.

Obs: Esse tipo de análise ajuda a descrever a linguagem aceita por , ou seja, todas as cadeias de um dado conjunto.

Autômatos Finitos

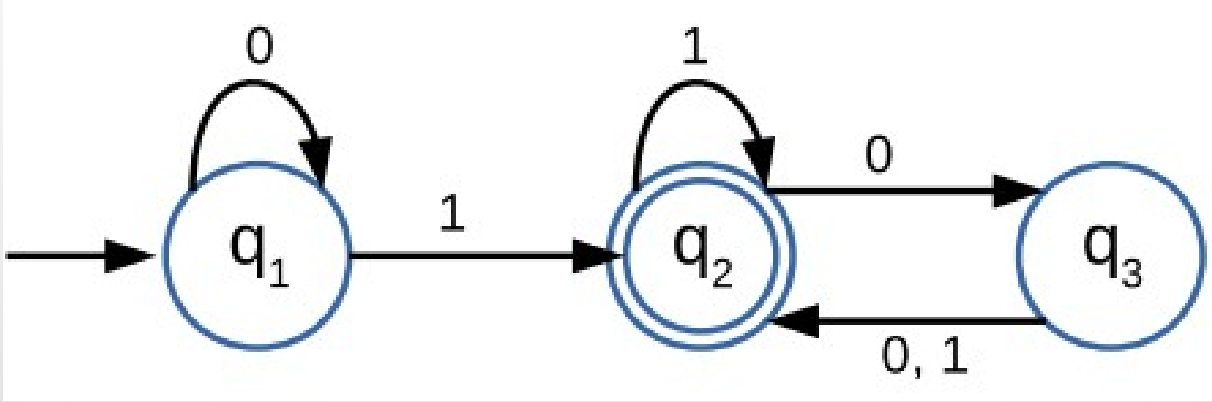
Autômato finito

Um autômato finito é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde:

1. Q é um conjunto finito de **estados**.
2. Σ é um conjunto finito conhecido como **alfabeto**.
3. $\delta : Q \times \Sigma \rightarrow Q$ é a **função de transição.**
4. $q_0 \in Q$ é o **estado inicial**.
5. $F \subseteq Q$ é o conjunto de **estados de aceitação (ou estados finais)**.

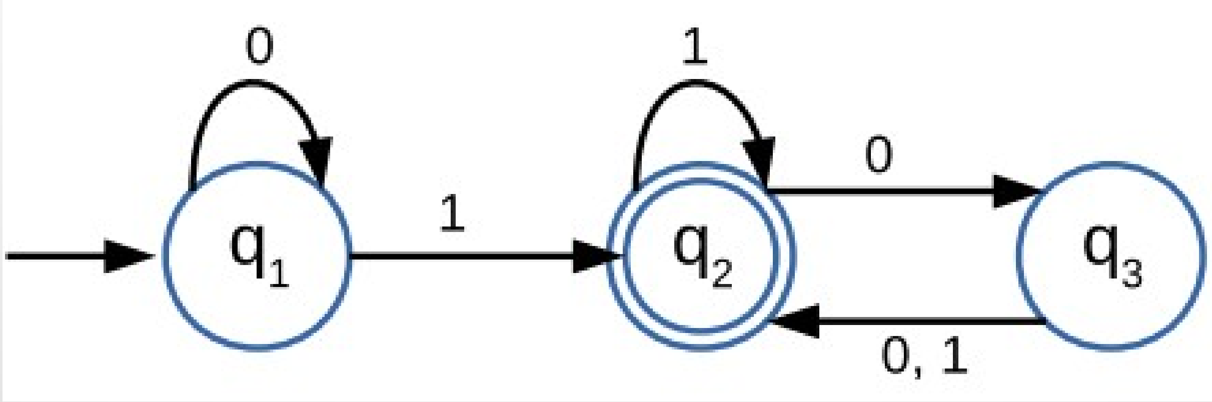
Autômatos Finitos

- Exemplo: O Autômato finito chamado que tem 3 estados.



Autômatos Finitos

- <u>Exemplo</u>: O Autômato finito chamado que tem 3 estados.

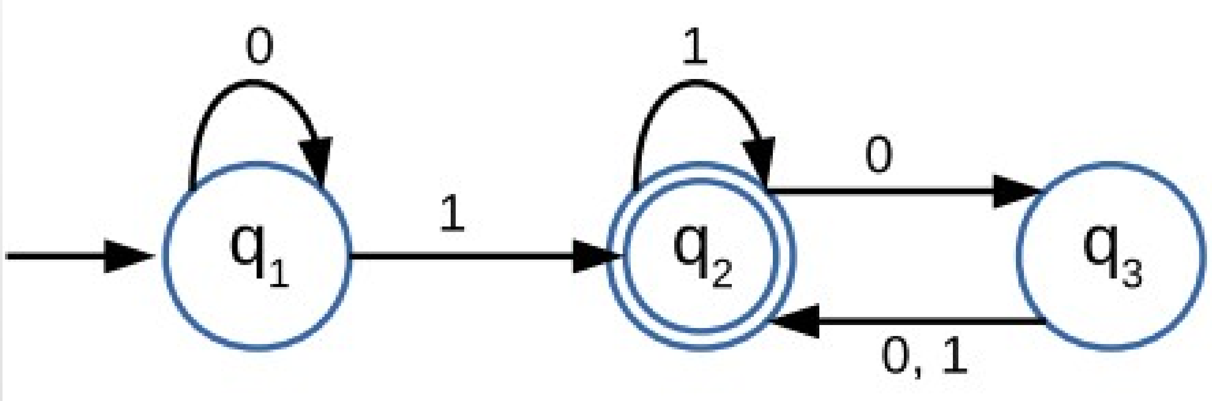


3.

	0	1

Autômatos Finitos

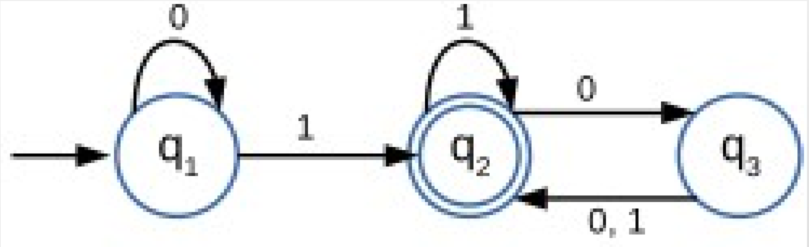
- Exemplo: O Autômato finito chamado que tem 3 estados.



4. é o **estado inicial**

Autômatos Finitos

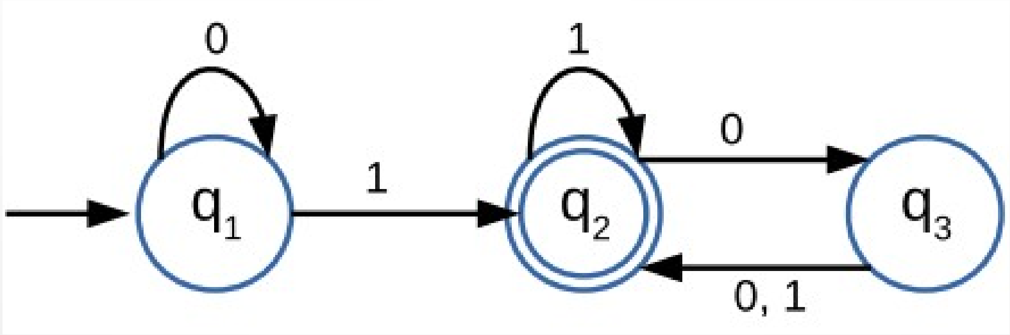
- Se A é o conjunto de todas as cadeias que a máquina M aceita, então A é a linguagem da máquina M, ou seja L(M) = A.
- Dizemos que M reconhece A ou M aceita A.
- Uma máquina pode aceitar várias cadeias (ou nenhuma cadeia), mas sempre reconhece uma única linguagem (linguagem vazia).



A = {w|w contém pelo menos um número 1 e um número par de 0 segue o último 1}

Autômatos Finitos

- Exemplo 1: Diagrama de estados do autômato finito .



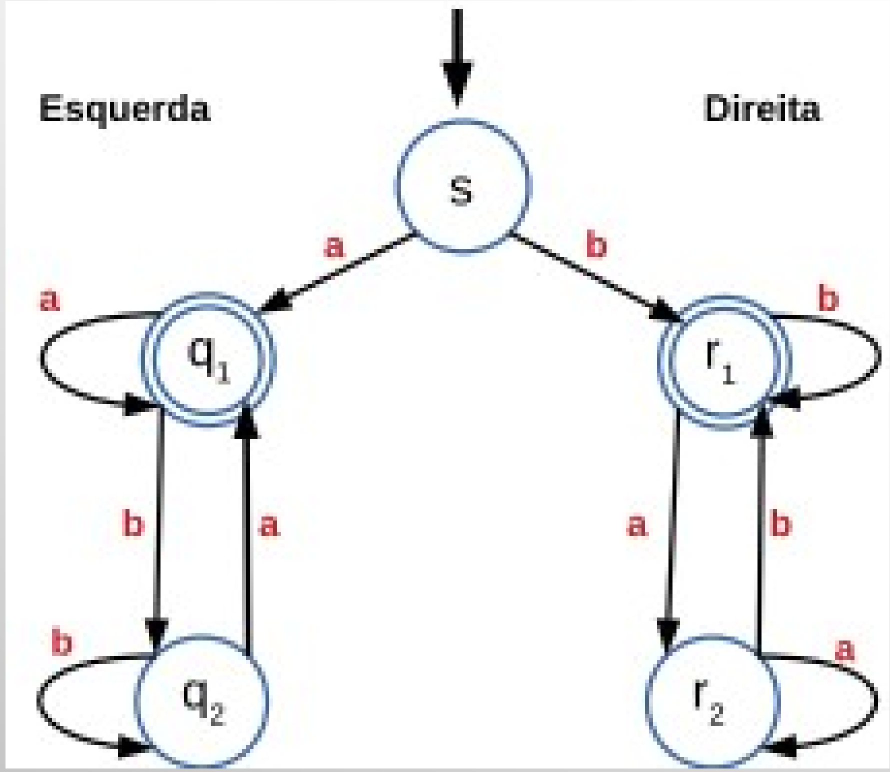
- Na descrição formal
- A função de transição δ é:

	0	1

- aceita a cadeia 1101?

Autômatos Finitos

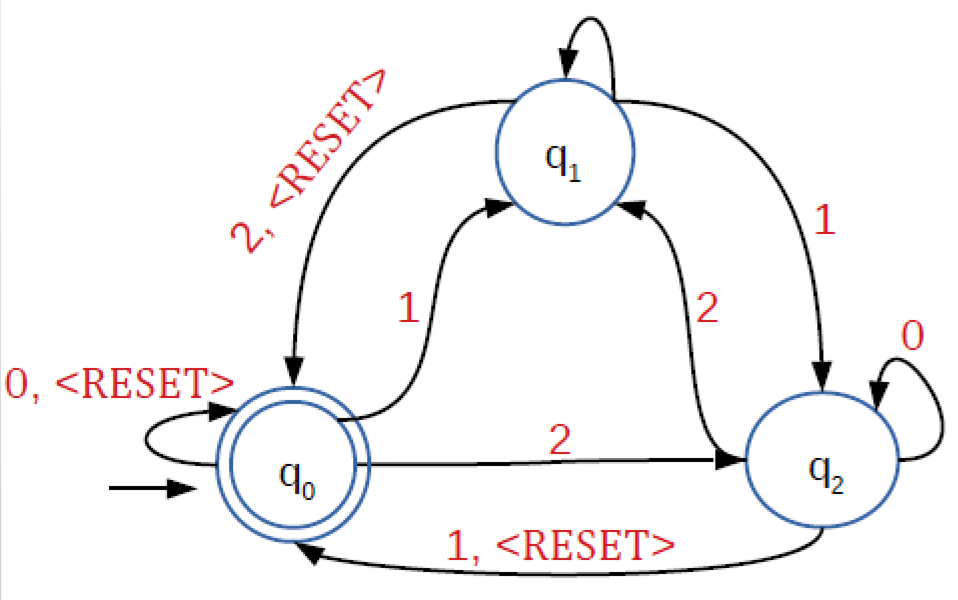
- <u>Exemplo 2</u>: Diagrama de estados do autômato finito .



- tem dois estados de aceitação: e
- opera sobre o alfabeto Σ = {a,b}
- Cadeias aceitas: {a,b,aa,bb,bab}
- Cadeias não aceitas: {ab,ba,bbba}

Autômatos Finitos

- <u>Exemplo 3</u>: Diagrama de estados do autômato finito .



- opera sobre o alfabeto Σ = { <RESET>, 0, 1, 2}.
- A máquina mantém um contador da soma dos símbolos numéricos de entrada que ela lê, módulo 3.
- Quando recebe o símbolo < RESET >, reinicia o contador para 0.
- **aceita** se a soma for 0, módulo 3 (se for múltiplo de 3).

Definição Formal de Computação

Computação de um autômato
Seja um autômato finito e suponha seja uma cadeia onde cada é um membro do alfabeto Σ. Então **aceita** se existe uma sequência de estados em com três condições: 1. (A máquina começa no estado inicial) 2. , para (A máquina muda de estado conforme a função de transição). **3.** (A máquina aceita sua entrada se ela termina em um estado de aceitação)

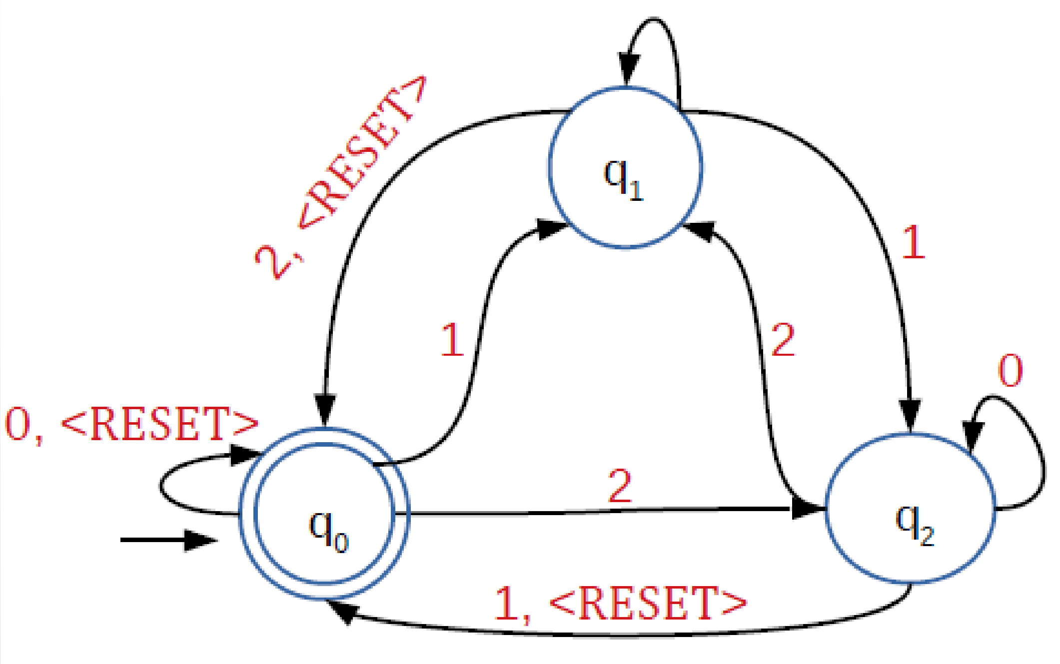
Definição Formal de Computação

Linguagem Regular

- Dizemos que **reconhece a linguagem** se .
- Uma linguagem é dita **linguagem regular** se algum autômato finito a reconhece.

Definição Formal de Computação

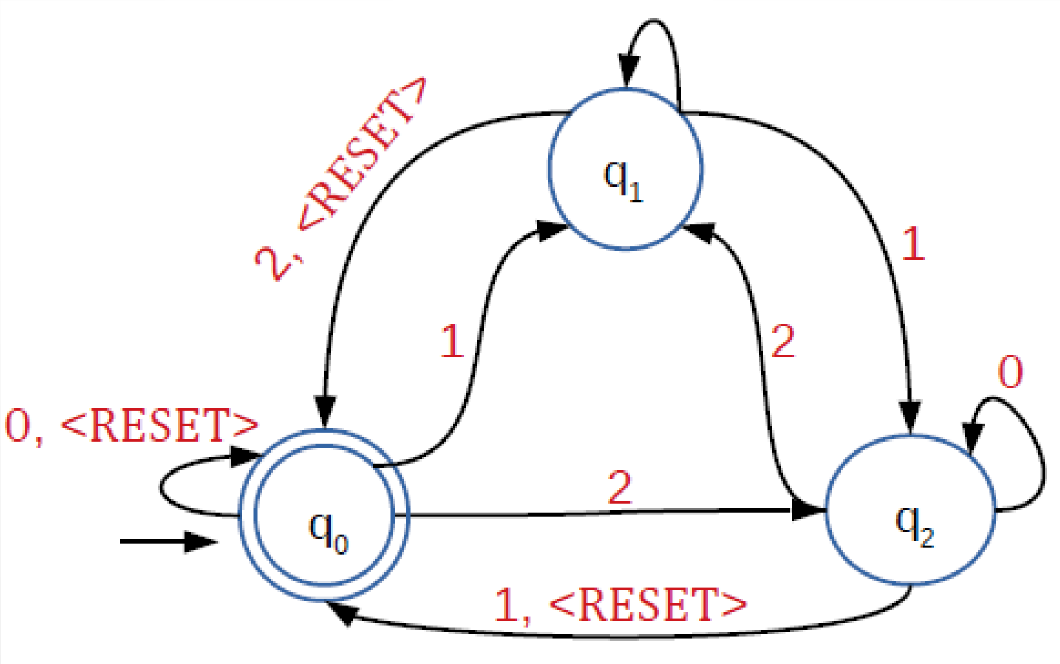
- <u>Exemplo</u>: Diagrama de estados do autômato finito .



- Seja 012
- aceita conforme a definição formal de computação porque a sequência de estados na qual entra quando está computando sobre é

Definição Formal de Computação

- Exemplo: Diagrama de estados do autômato finito .



- A cadeia satisfaz as 3 condições da definição formal de computação
- A linguagem de é

- Como reconhece , ela é uma linguagem regular.

Projetando Autômatos Finitos

- Projetar é um Processo Criativo.
- Ele não pode ser reduzido a uma receita ou fórmula simples.
- Ponha-se a si próprio no lugar da máquina que você está tentando projetar.
- Fazer de conta que você é a máquina é um truque psicológico que ajuda a sua mente inteira no processo de projetar.
- Você está fazendo de conta que é um autômato finito e que esse tipo de máquina tem somente um número finito de estados, o que significa memória finita.

Projetando Autômatos Finitos

- Por exemplo, suponha que o alfabeto seja {0,1} e que a linguagem consista de todas as cadeias com um número ímpar de 1s. Você deseja construir um autômato finito para reconhecer essa linguagem.

Como fazer?

Projetando Autômatos Finitos

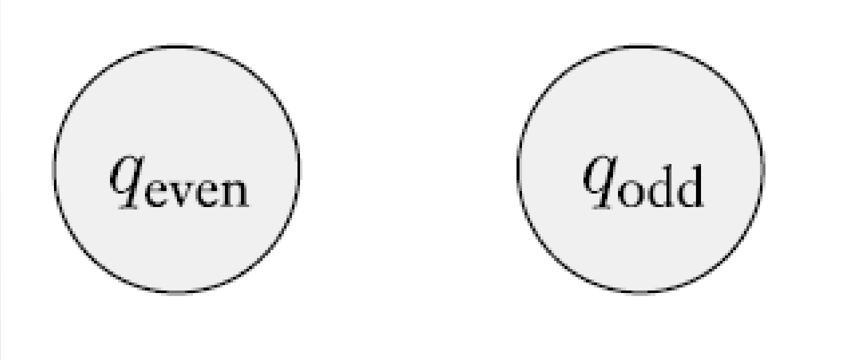
- Fazendo de conta ser o autômato, você começa obtendo uma cadeia de entrada de 0s e 1s símbolo a símbolo.
- Você precisa lembrar a cadeia inteira vista até então para determinar se o número de 1s é ímpar? É claro que não.
- Simplesmente lembrar se o número de 1s visto até então é par ou ímpar e manter essa informação à medida lê novos símbolos.
- Uma vez que você tenha determinado a informação necessária para lembrar sobre a cadeia à medida que ela está sendo lida, você representa essa informação como uma lista finita de possibilidades.

Projetando Autômatos Finitos

- Nessa instância, as possibilidades seriam

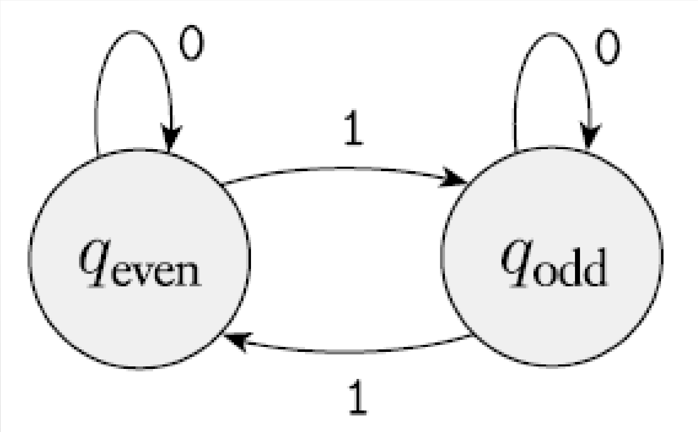
1. par até agora, e
2. ímpar até agora.

- Depois disso, você atribui um estado a cada uma das possibilidades.



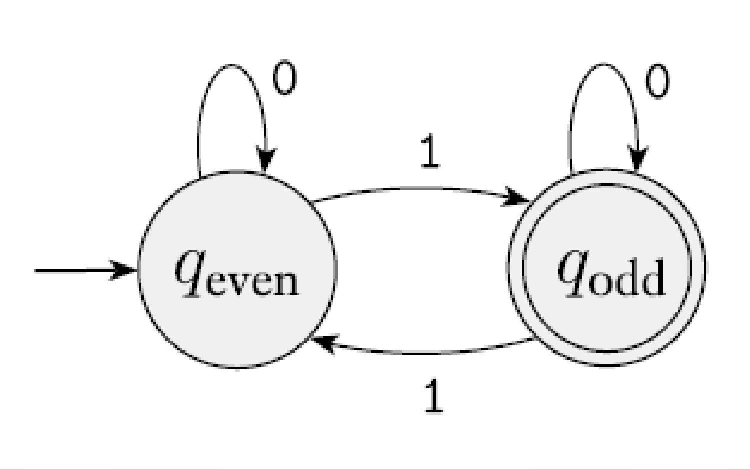
Projetando Autômatos Finitos

- A seguir, você atribui as transições vendo como ir de uma possibilidade para outra ao ler um símbolo.
- Portanto, se o estado representa a possibilidade par e o estado representa a possibilidade ímpar, você faria as transições trocar de estado com um 1 e permanecer como está com um 0, como mostrado aqui.



Projetando Autômatos Finitos

- A seguir, você coloca como estado inicial o estado correspondendo à possibilidade associada com ter visto 0 símbolos até então (a cadeia vazia).
- Depois, por como estados de aceitação aqueles correspondendo a possibilidades nas quais você deseja aceitar a cadeia de entrada.



Projetando Autômatos Finitos

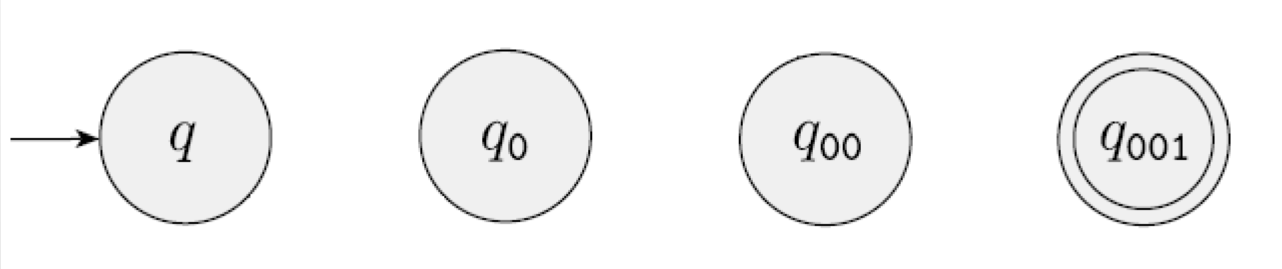
- Exemplo: projetar um autômato finito para reconhecer a linguagem regular de todas as cadeias que contém a cadeia 001 como uma subcadeia. Por exemplo, 0010, 1001, 001 e 11111110011111 estão todas na linguagem, mas 11 e 0000 não estão.

Projetando Autômatos Finitos

- Pensando como autômato, existem quatro possibilidades:

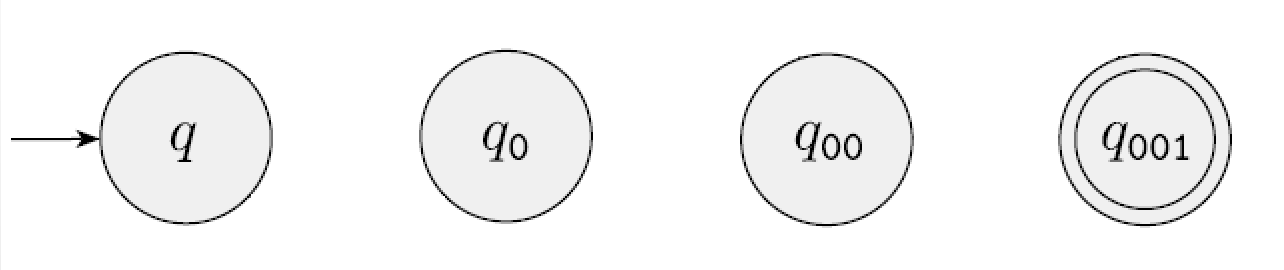
1. não tem visto quaisquer símbolos do padrão,
2. acaba de ver um 0,
3. acaba de ver 00, ou
4. acaba de ver o padrão inteiro 001.

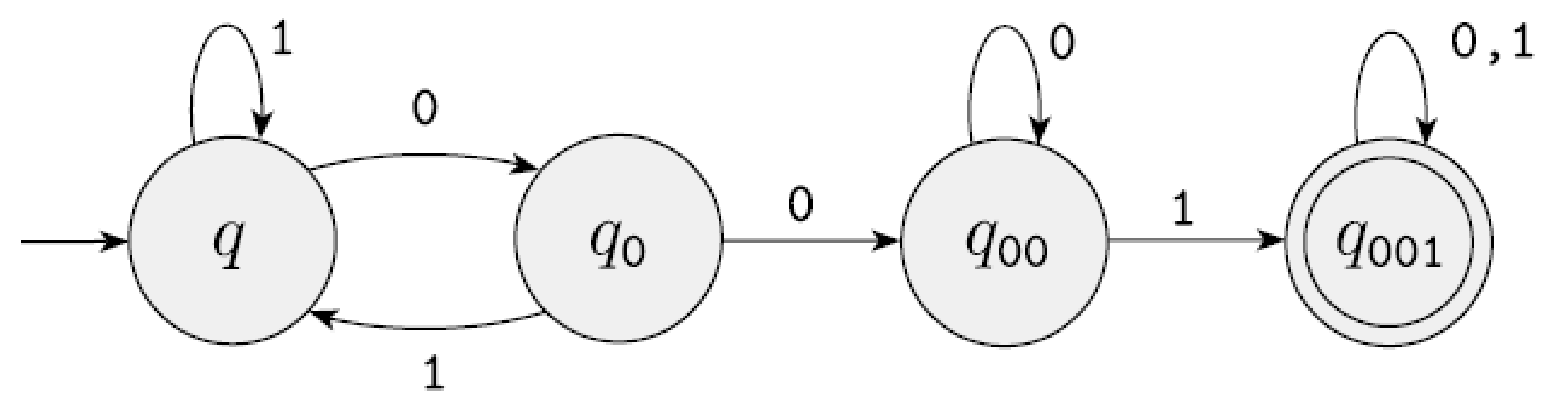
- Atribua os estados a essas possibilidades.



Projetando Autômatos Finitos

- Depois disso analise as transições que podem ocorrer.





Operações Regulares

- Em aritmética, os objetos básicos são números (1, 2, 3, ...) e ferramentas são operações para manipulá-los
- Na teoria da computação, os objetos básicos são linguagens e as ferramentas (operações para manipular linguagens - operações regulares)

Operações Regulares

Operações regulares

Sejam A e B linguagens. Definimos as operações regulares **união**, **concatenação** e **estrela** da seguinte forma.

- **União**:
- **Concatenação**:
- **Estrela**:

Operações Regulares

- Seja o conjunto dos números naturais.
- é fechado sob multiplicação, ou seja,

$$\forall x, y \in \mathbb{N}, x * y \in \mathbb{N}$$

- não é fechado sob divisão, ou seja,

$$\forall x, y \in \mathbb{N}, x \div y \notin \mathbb{N}$$

- Uma coleção de objetos **é fechada** sob alguma operação se, aplicando-se essa operação a membros da coleção, o resultado ainda é um membro dessa coleção.
- A coleção de linguagens regulares é fechada sob todas as três das operações regulares (união, concatenação e estrela).

Operações Regulares

- Exemplo: Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras . Se e , então:

1. **UNIÃO**:

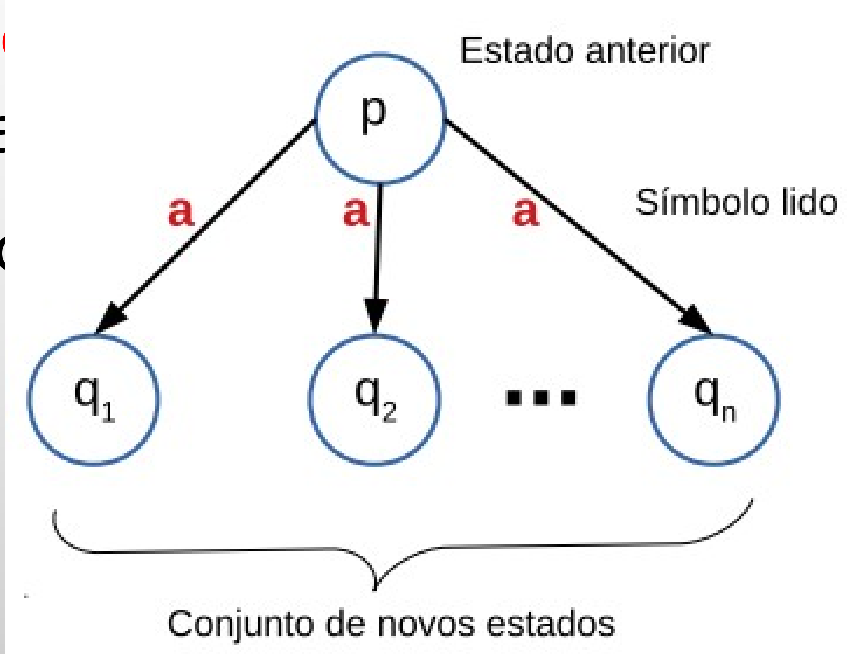
2. **CONCATENAÇÃO**:

3. **ESTRELA**:

Não-Determinismo

- Até agora, todo passo de uma computação segue de uma maneira única do passo precedente (os estados estão determinados e sabemos qual o próximo): computação determinística.
- Em uma máquina não-determinísti m existir para o próximo estado em qua
- Não-determinismo é uma generaliza

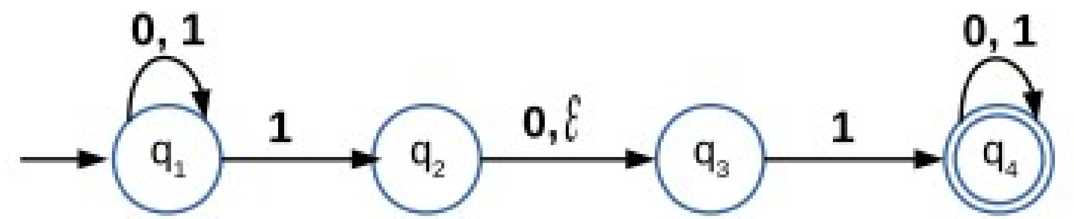
determinismo.



Não-Determinismo: Diferença entre AFD e AFN

- Todo estado de um AFD tem exatamente **uma seta** de transição saindo para cada símbolo do alfabeto.
- AFN pode ter de nenhuma a **muitas setas** saindo para cada símbolo do alfabeto.

Figura: Autômato finito não-determinístico N_1.

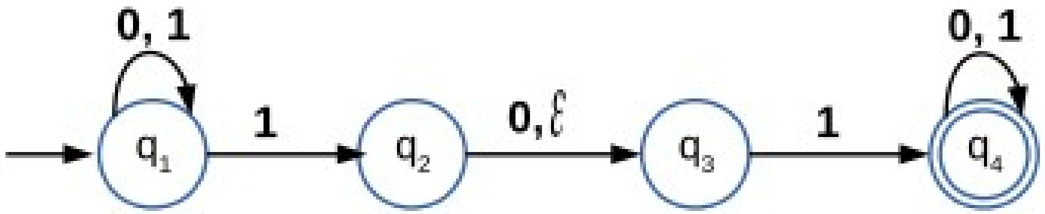


Não-Determinismo: Diferença entre AFD e AFN

- Em um AFD, os rótulos sobre as setas de transição são símbolos do alfabeto.

- Um AFN pode ter setas rotuladas com membros do alfabeto ou com .

Figura: Autômato finito não-determinístico N_1.



Não-Determinismo

- Ao processar uma cadeia de entrada, caso ocorra um direcionamento para um estado com múltiplas maneiras de prosseguir, a máquina divide-se em múltiplas cópias de si mesma e segue todas as possibilidades em paralelo.
- Cada cópia da máquina toma uma das possíveis maneiras de prosseguir e continua sua execução como um AFD, recursivamente

Não-Determinismo

- Se o símbolo de entrada não aparece sobre qualquer seta saindo do estado ocupado por uma cópia da maquina, aquela cópia morre, juntamente com o ramo da computação associado a ela.
- Quando o símbolo ε (ou λ) é encontrado sobre uma seta, sem ler qualquer entrada (antes e depois), a máquina divide-se e as transições são consideradas: transição epsilon ou transição espontânea.
- Não-determinismo pode ser visto como uma espécie de computação paralela na qual múltiplos e independentes “processos” ou “*threads*” podem estar rodando concorrentemente.

Não-Determinismo

1. Dada uma cadeia de entrada:

   AFD: Executa uma única sequência de movimentos

   AFN: Executa várias sequências distintas

2. Aceita a cadeia de entrada:

   AFD: Parada em uma configuração final

   AFN: Parada em uma configuração final

3. Rejeita a cadeia de entrada:

   AFD: Parada em uma configuração não-final

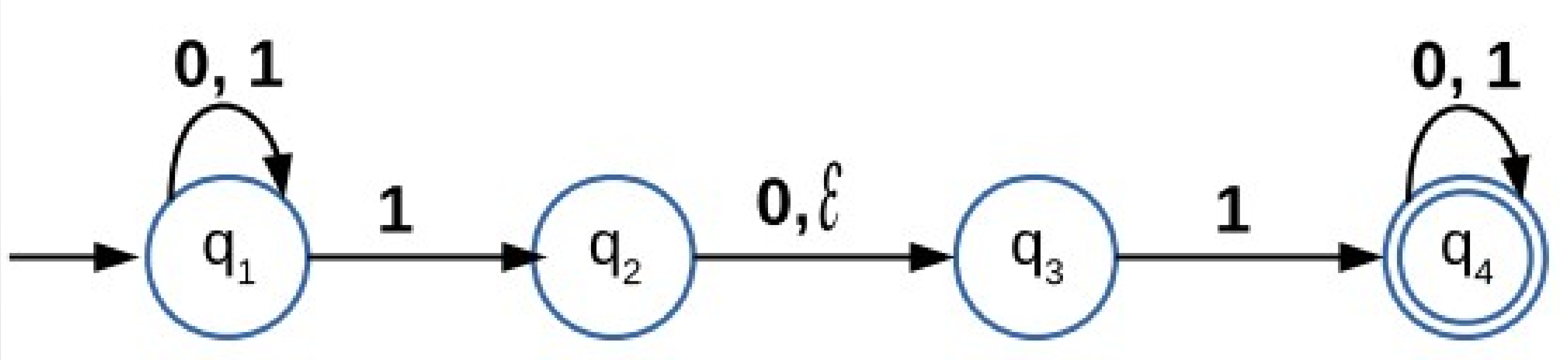
   AFN: Parada mesmo sem atingir nenhuma configuração final

Definição Formal de AFN

Autômato Finito Não-determinístico
Um Autômato Finito Não-determinístico é uma 5-upla , onde: 1. é um conjunto finito de **estados**. **2.** é um **alfabeto** finito. 3. é **a função de transição.** **4.** é o **estado inicial.** **5.** é o conjunto de **estados de aceitação (ou estados finais).**

Definição Formal de AFN

Seja o AFN :

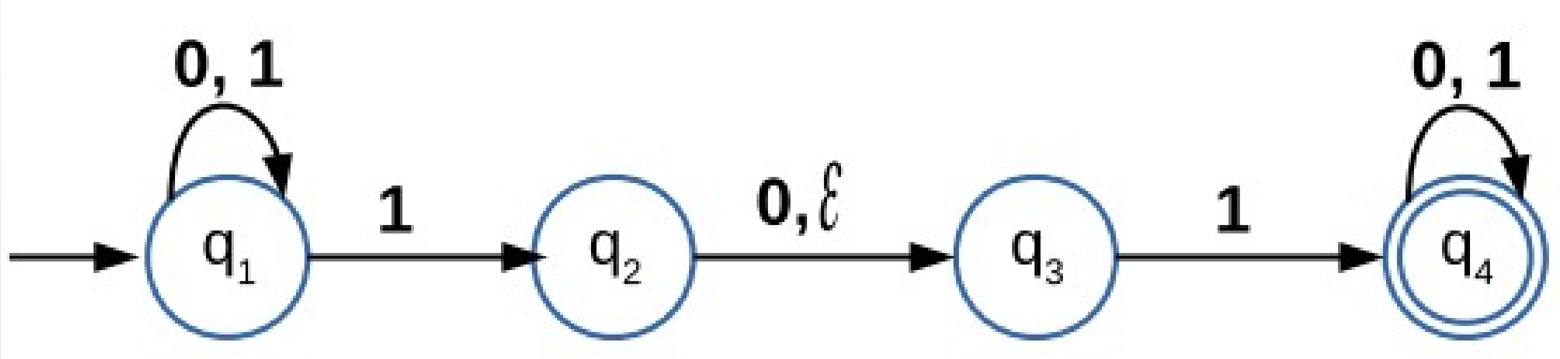


1.

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

Definição Formal de AFN

Seja o AFN :



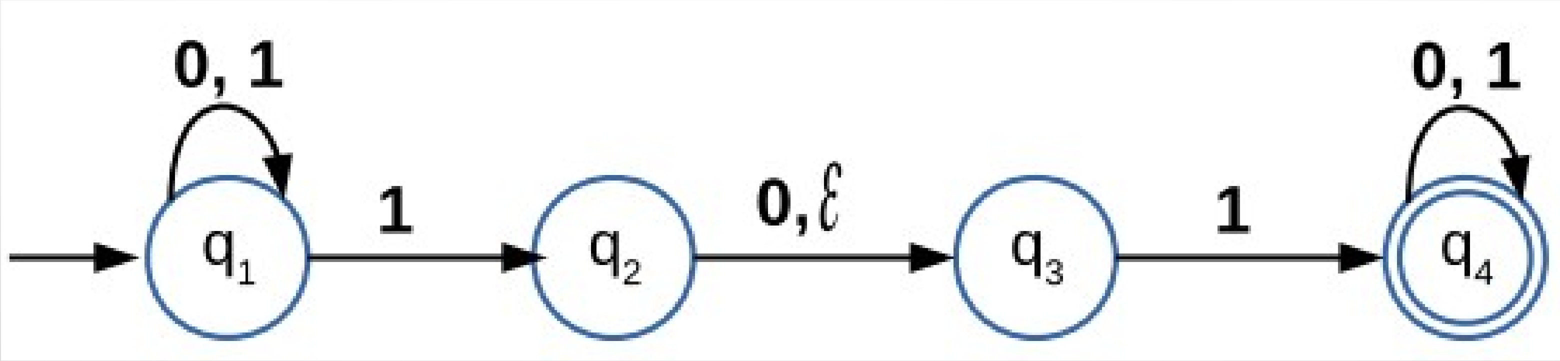
1.

2. é o **estado inicial.**

3. .

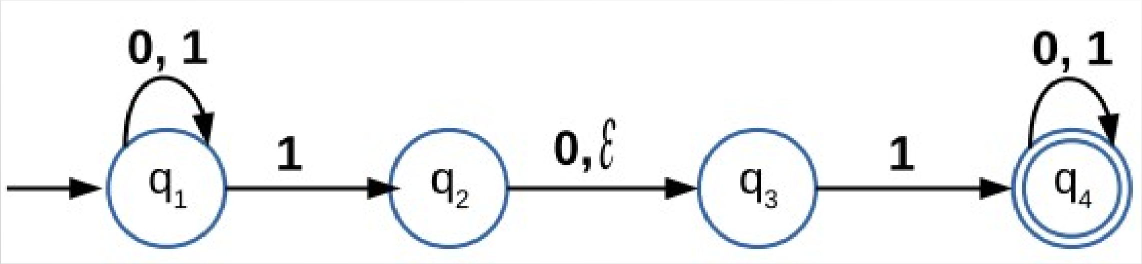
Definição Formal de AFN

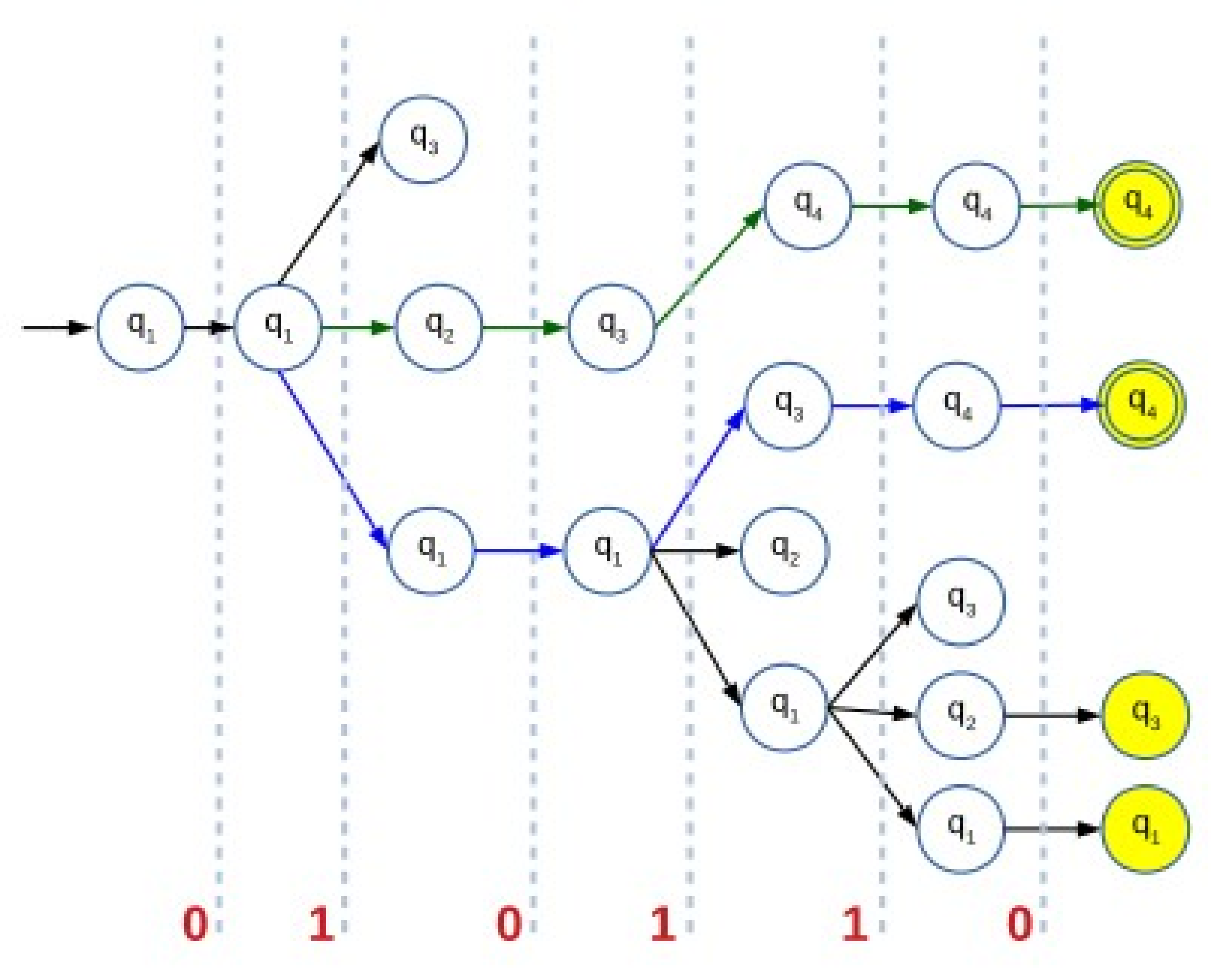
<u>Exemplo 1</u>: Computar **010110** em



Definição Formal de AFN

Exemplo 1: Computação de sobre a entrada **010110**:

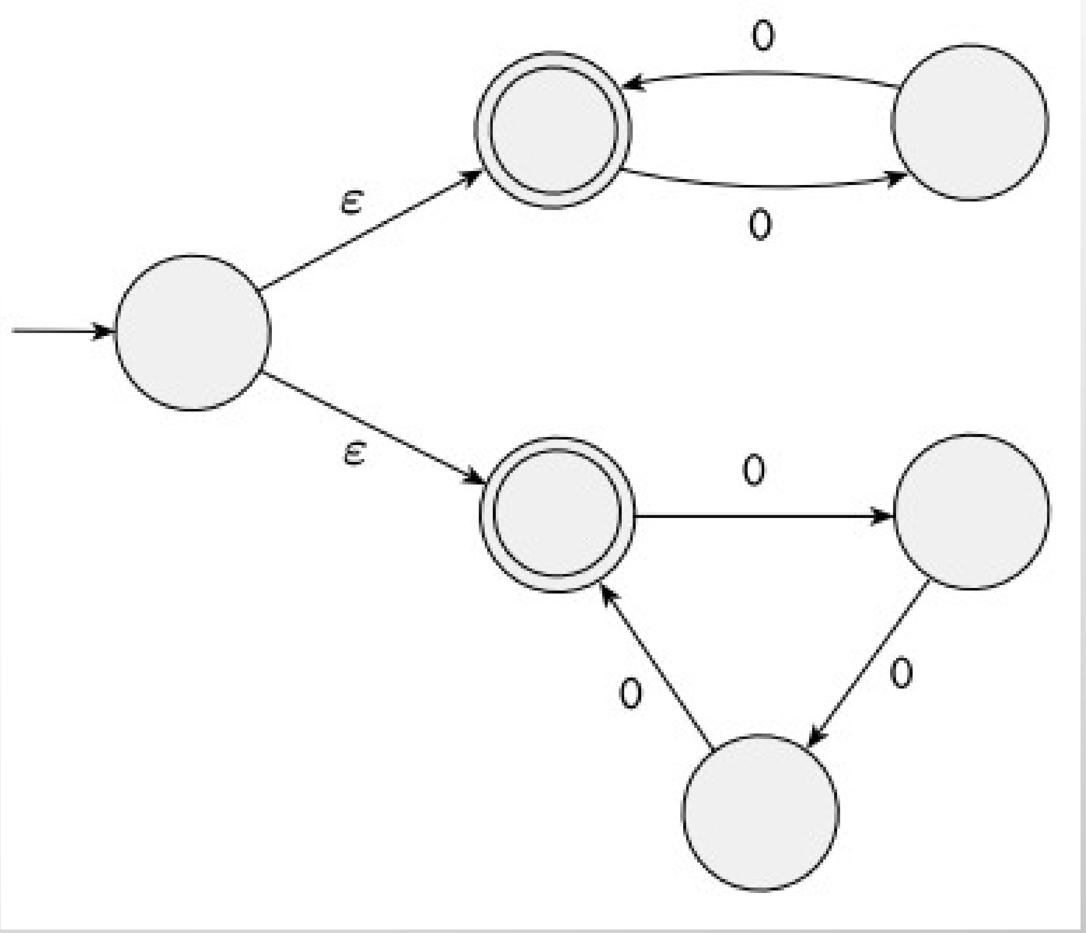




Definição Formal de AFN

<u>Exemplo 2</u>: Considere o AFN

- Essa máquina demonstra a conveniência de se ter setas ε.
- Ele aceita todas as cadeias da forma onde é um múltiplo de 2 ou 3. (Lembre-se de que o expoente denota repetição, e não exponenciação numérica.)
- Por exemplo, aceita as cadeias ε, 00, 000, 0000 e 000000, mas não 0 ou 00000



Definição Formal de Computação para AFN

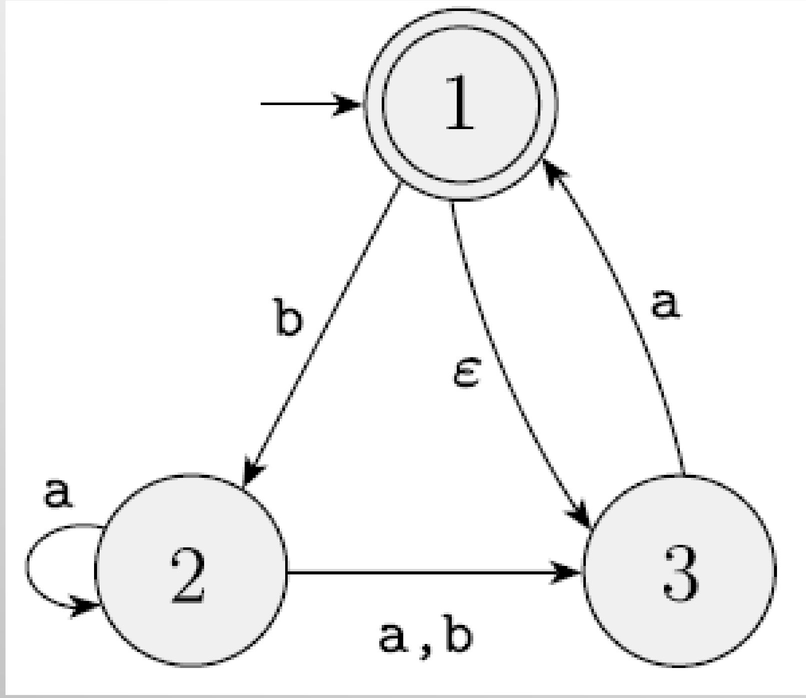
Computação de um AFN
Seja um AFN e uma cadeia sobre o alfabeto Σ. Então dizemos que **aceita** se podemos escrever como onde cada é um membro de e existe uma sequência de estados em com três condições: 1. (A máquina começa no estado inicial) 2. , para (O estado é um dos próximos estados possíveis). **3.** (A máquina aceita sua entrada se o último um estado é um estado de aceitação)

Equivalência de AFDs e AFNDs

1. AFNs e AFDs reconhecem a mesma classe de linguagem
2. Duas máquinas são equivalentes se elas reconhecem a mesma linguagem.
3. Todo AFN tem um AFD equivalente.
4. Uma linguagem é regular se, e somente se, algum AFN a reconhece.

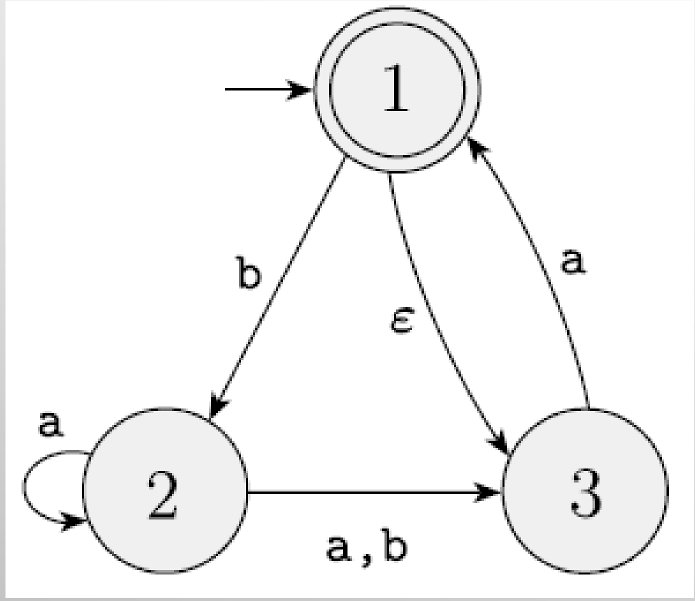
Conversão de AFN para AFD

- Vamos ilustrar o procedimento para converter um AFN para um AFD usando a máquina .



Conversão de AFN para AFD

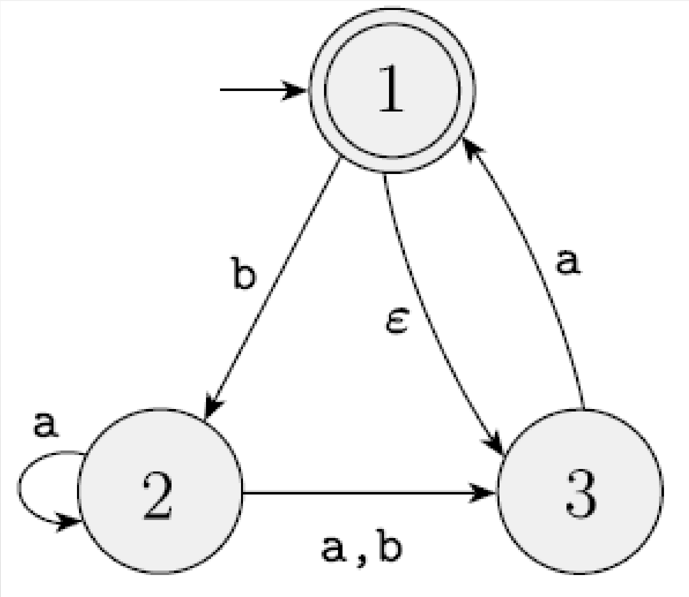
- Para construir um AFD que seja equivalente a , primeiro determinamos os estados de .
- tem três estados, {1, 2, 3}, assim construímos com oito estados, .



- O Conjunto de estados de é:

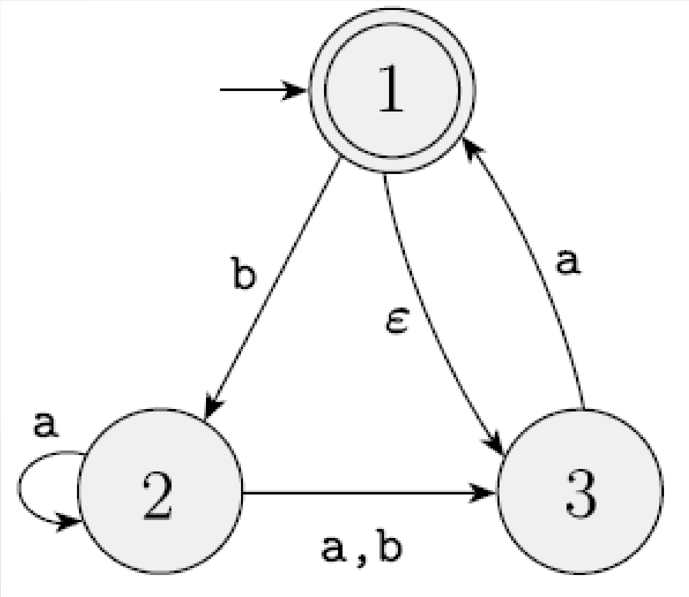
Conversão de AFN para AFD

A seguir, determinamos os estados inicial e de aceitação de D. O estado inicial é $E(\{1\})$, o conjunto de estados que são atingíveis a partir de 1 viajando ao longo de setas ε, mais o próprio 1. Uma seta ε vai de 1 para 3, portanto $E(\{1\}) = \{1, 3\}$. Os novos estados de aceitação são aqueles contendo o estado de aceitação de N_4; assim, $\{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$.



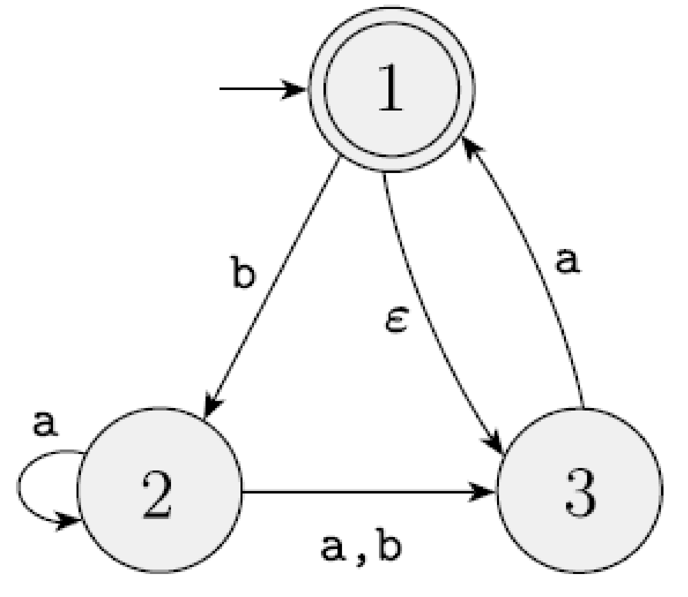
Conversão de AFN para AFD

Em D, o estado $\{2\}$ vai para $\{2,3\}$ na entrada a, porque em N_4, o estado 2 vai para ambos 2 e 3 na entrada a e não podemos ir mais longe a partir de 2 ou 3 ao longo de setas ε. O estado $\{2\}$ vai para o estado $\{3\}$ na entrada b, porque em N_4, o estado 2 vai apenas para o estado 3 na entrada b e não podemos ir mais longe a partir de 3 ao longo de setas ε.



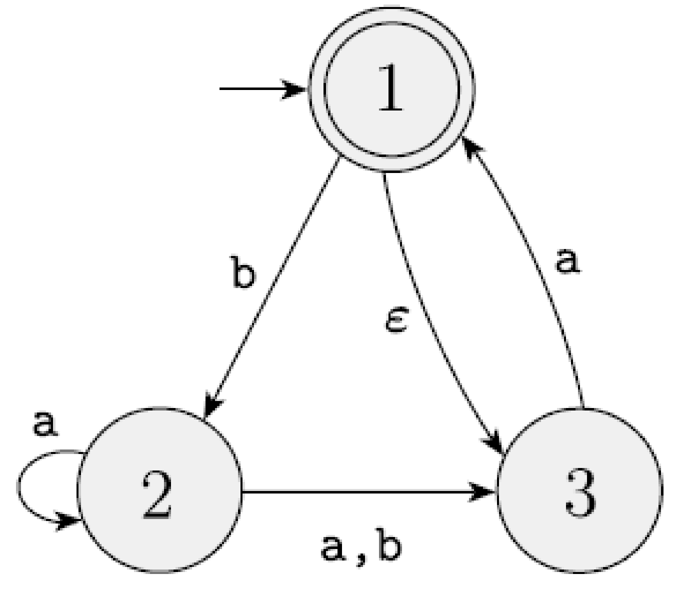
Conversão de AFN para AFD

- O estado vai para na entrada , porque nenhuma seta sai dele. Ele vai para na entrada .



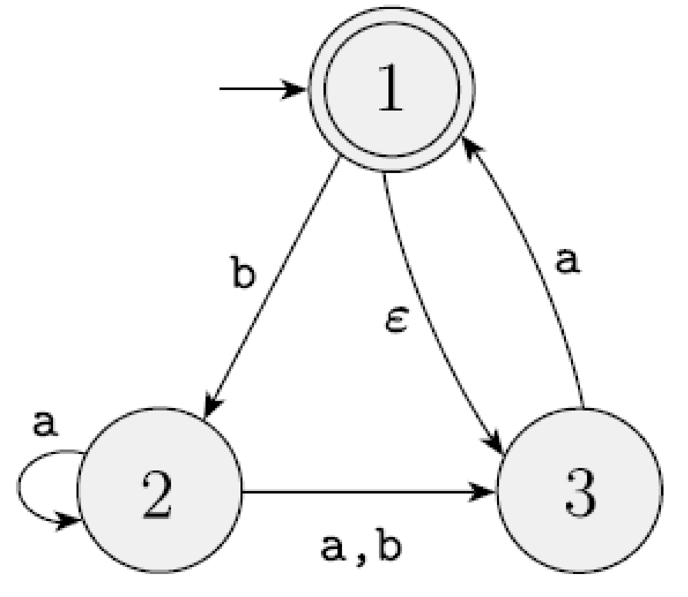
Conversão de AFN para AFD

O estado $\{3\}$ vai para $\{1,3\}$ na entrada a, porque em N_4, o estado 3 vai para 1 na entrada a e 1 por sua vez vai para 3 com uma seta ε. O estado $\{3\}$ na entrada b vai para $\emptyset$.



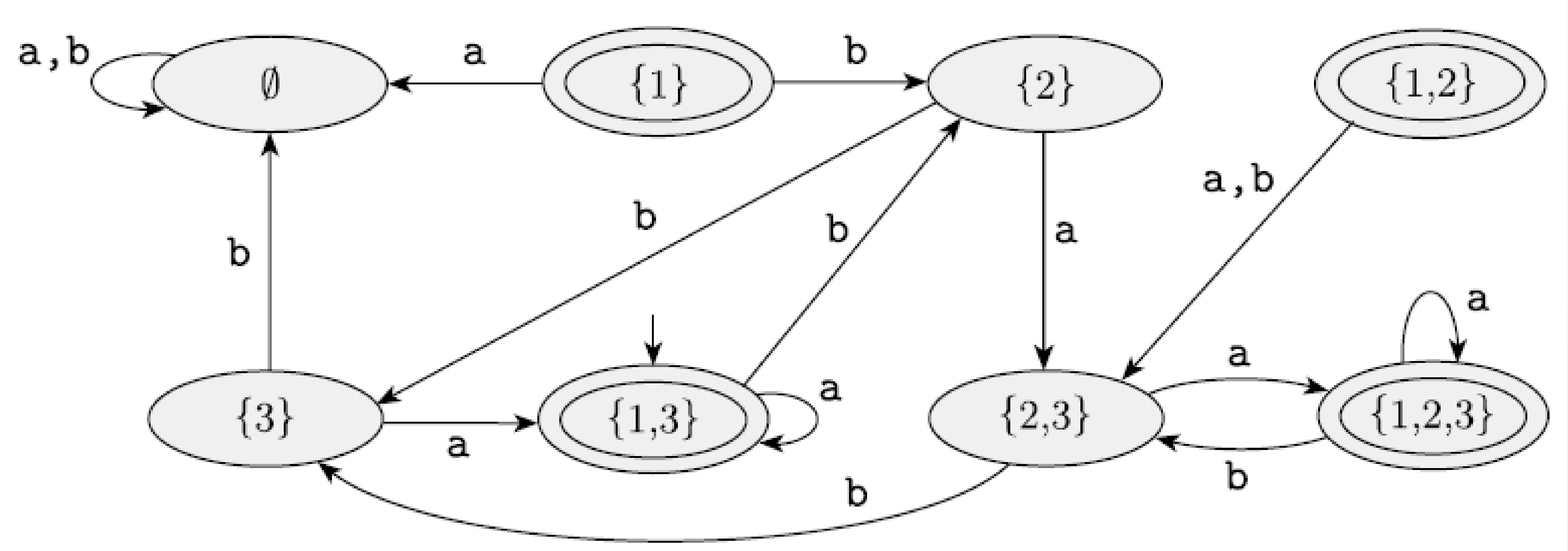
Conversão de AFN para AFD

- O estado na entrada vai para .
- O estado na entrada vai para .
- E assim por diante.



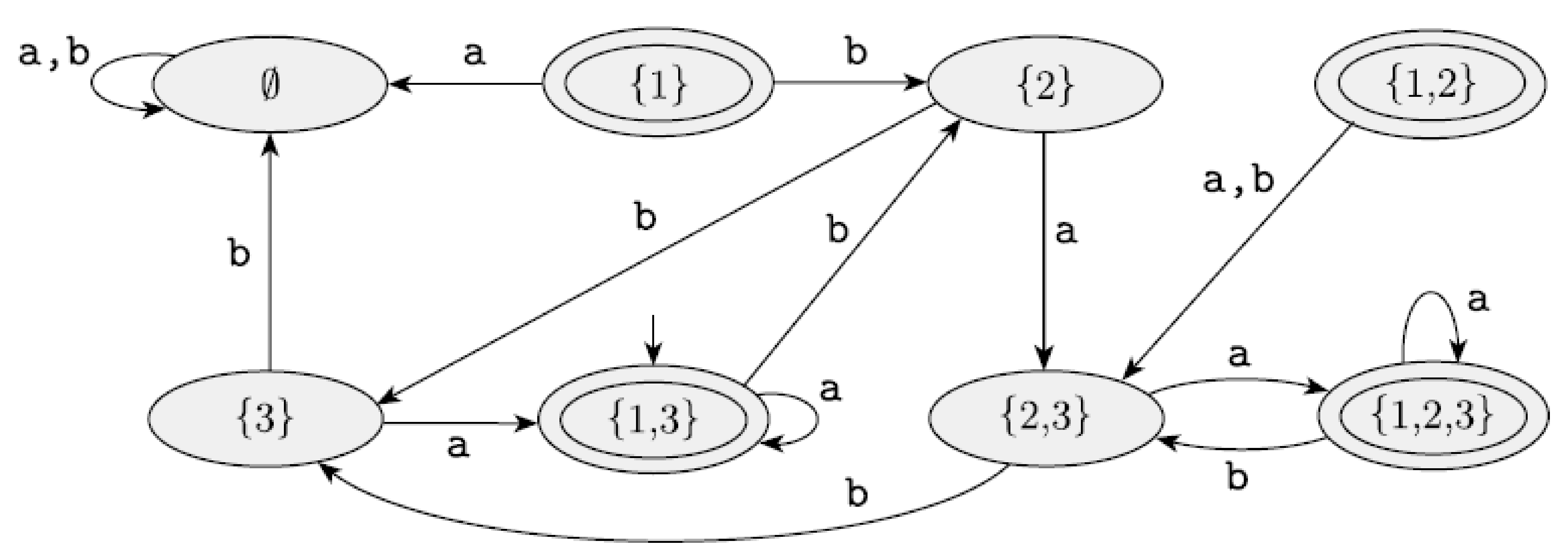
Conversão de AFN para AFD

- Após isso, obtém-se o seguinte diagrama de estados para .



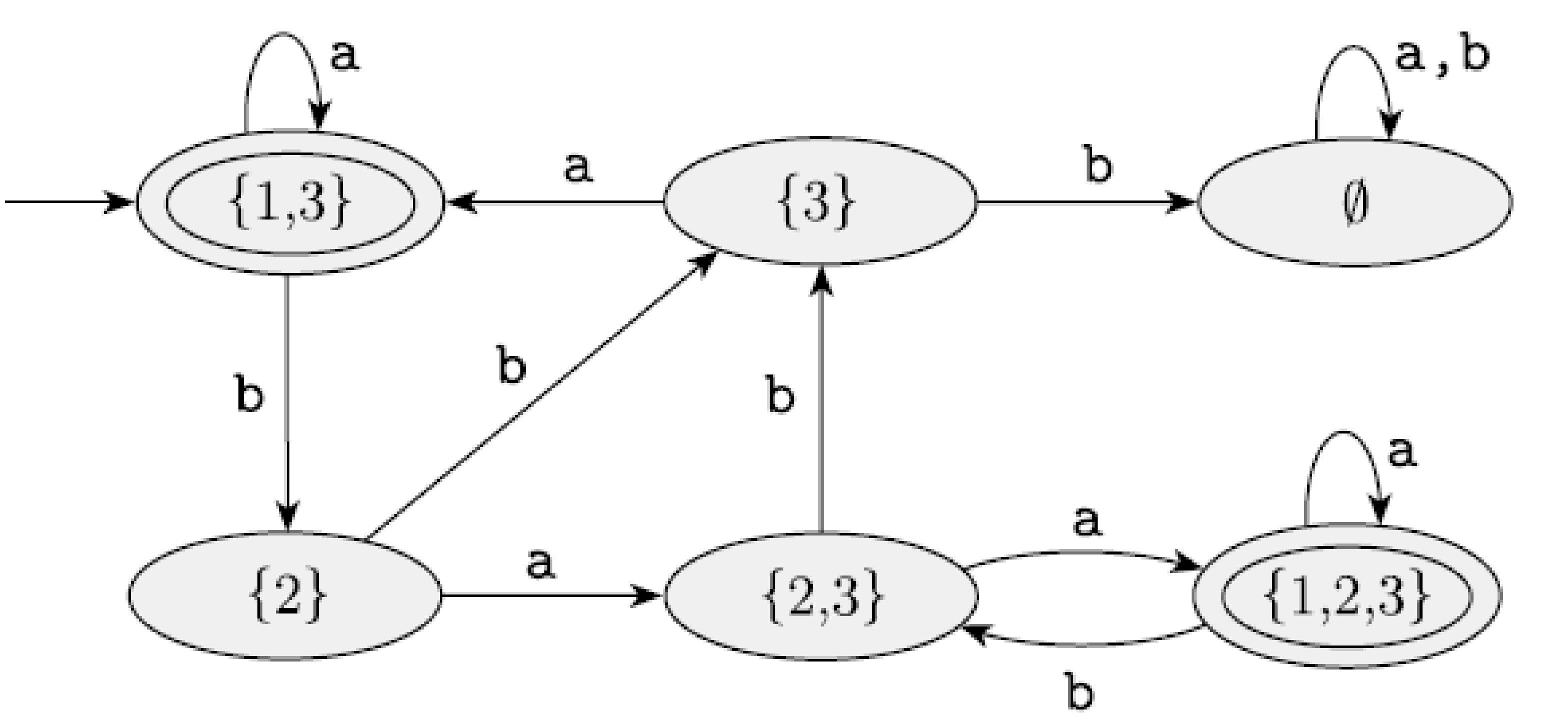
Conversão de AFN para AFD

- Podemos simplificar essa máquina observando que nenhuma seta aponta para os estados e , portanto eles podem ser removidos sem afetar o desempenho da máquina.



Conversão de AFN para AFD

- Simplificando, tem-se:



Expressões Regulares

- Na aritmética, podemos usar operações e para construir expressões como:

- Em Teoria da Computação, podemos usar operações regulares para construir expressões descrevendo linguagens, que são chamadas de **expressões regulares**

Linguagem L

L = {w | w é qualquer cadeia começando com 0 ou 1 seguido por um número qualquer de zeros.}

Expressões Regulares

- Na expressão:

- **Primeira parte**: os símbolos 0 e 1 são abreviações para {0} e {1}, sendo que
- **Segunda parte**: significa , ou seja todas as cadeias contendo qualquer número de 0s
- Como em Álgebra o símbolo de multiplicação, , geralmente está implícito, o símbolo de concatenação também está implícito nas expressões regulares:

Expressões Regulares

- Expressões regulares têm um papel importante em Ciência da Computação:

1. Em aplicações envolvendo textos, usuários podem fazer busca por cadeias que satisfaçam determinados padrões
2. Muitos utilitários e linguagens de programação provêm mecanismos para descrever padrões usando expressões regulares

Expressões Regulares

- Exemplos:

1. Exemplo: O CEP é da forma , onde é um dígito qualquer entre 0 e 9. Essa expressão representa toda a gama de CEPs no nosso país
2. Compiladores: expressões regulares podem representar qualquer trecho do código, e acusar erro de sintaxe, caso o código escrito não case com as expressões regulares

Expressões Regulares

- Na Expressão:

- Começa com a linguagem e aplica a operação *
- Valor da expressão: linguagem constituída de todas as possíveis cadeias de 0s e 1s.
- Se , podemos escrever Σ como abreviação para a expressão regular .

Expressões Regulares

- Se for uma alfabeto qualquer, a expressão regular descreve a linguagem que contém todas as cadeias de comprimento 1 sobre esse alfabeto.
- descreve a linguagem constituída de todas as cadeias sobre Σ.
- 1 é a linguagem que contém todas as cadeias que terminam em 1.
- A linguagem contém todas as cadeias que começam com 0 ou terminam em 1.

Expressões Regulares

- Na aritmética, tem precedência sobre +
- Para fazer com que a adição seja feita primeiro em uma expressão, usamos parênteses.

- Em operações regulares, vale a seguinte ordem:

(1) operação estrela: *, **(2)** concatenação: ∘, **(3)** união:

- Parênteses podem forçar uma ordem diferente.

Definição Formal de uma Expressão Regular

Expressão Regular

Dizemos que R é uma expressão regular se R for indutivamente:

1. para algum , sendo uma linguagem.
2. , sendo uma linguagem,
3. , linguagem vazia,
4. , onde e são expressões regulares,
5. , onde e são expressões regulares, ou
6. , onde é uma expressão regular.

; sempre serão menores do que

Definição Formal de uma Expressão Regular

Observações:

- Não Confundir e :

- : Linguagem contendo uma única cadeia
- : Linguagem que não contém nenhuma cadeia

- Os parênteses em uma expressão podem ser omitidos. Quando isso ocorre, seguimos as regras de precedência.

Definição Formal de uma Expressão Regular

Linguagem de uma Expressão Regular

A linguagem definida por uma expressão regular , denotada por , é definida da seguinte maneira:

1. (Linguagem vazia) e ,
2. para algum ,

3. ,
4. , onde

Dizemos que e são equivalentes, , se elas representam a mesma linguagem, i.e., .

Definição Formal de uma Expressão Regular

Observações:

- Podemos usar como abreviação para

- A linguagem   tem todas as cadeias que são 0 ou mais concatenações de cadeias de
- A linguagem tem todas as cadeias que resultam de 1 ou mais concatenações de cadeias de
-
- é uma abreviação para a concatenação de Rs
- Distinguir expressão regular de linguagem regular .

Definição Formal de uma Expressão Regular

Exemplos (i)

- Assumindo , o valor de cada expressão abaixo é:

1. { contém um único }
2. { contém pelo menos um símbolo }
3. { contém como subcadeia}
4. { todo em é seguido por pelo menos um 1}

Definição Formal de uma Expressão Regular

Exemplos (ii)

- Assumindo , o valor de cada expressão abaixo é:

1. { é uma cadeia de comprimento par}
2. { é uma cadeia cujo comprimento é um múltiplo de }

3.

Definição Formal de uma Expressão Regular

Exemplos (iii)

- Assumindo , o valor de cada expressão abaixo é:

1. , concatenar o conjunto vazio a qualquer conjunto produz o conjunto vazio.
2. , a operação estrela junta qualquer número de cadeias da linguagem para obter uma cadeia no resultado. Se a linguagem for vazia, a operação estrela pode juntar 0 cadeias, dando apenas a cadeia vazia.

Definição Formal de uma Expressão Regular

Identidades sob expressão regular

1. , adicionar a linguagem vazia a qualquer outra linguagem não a modificará.
2. , juntar a cadeia vazia a qualquer outra cadeia não a modificará.

Definição Formal de uma Expressão Regular

Identidades sob expressão regular

1. Se , então ,

   mas .
2. Se , então , mas

   mas .

Aplicação de Expressão Regular

1. Expressões regulares são úteis no **desenho de compiladores** para linguagens de programação.
2. *Tokens* (variáveis e constantes) podem ser descritos com expressões regulares.

Aplicação de Expressão Regular

Exemplos (iv)

A constante numérica

Pode ser descrita como um membro da linguagem:

onde

- Uma vez que a sintaxe dos tokens tenha sido descrita com expressões regulares, sistemas automáticos podem gerar o analisador léxico (parte que processa o programa de entrada).

Equivalência com Autômatos Finitos

- Expressões regulares e autômatos finitos tem o mesmo poder descritivo.
- Qualquer expressão regular pode ser convertida num autômato finito que reconhece a linguagem que ela descreve, e vice-versa.

Equivalência com Autômatos Finitos

- Teorema:

Uma linguagem é regular se e somente se alguma expressão regular a descreve.

Equivalência com Autômatos Finitos

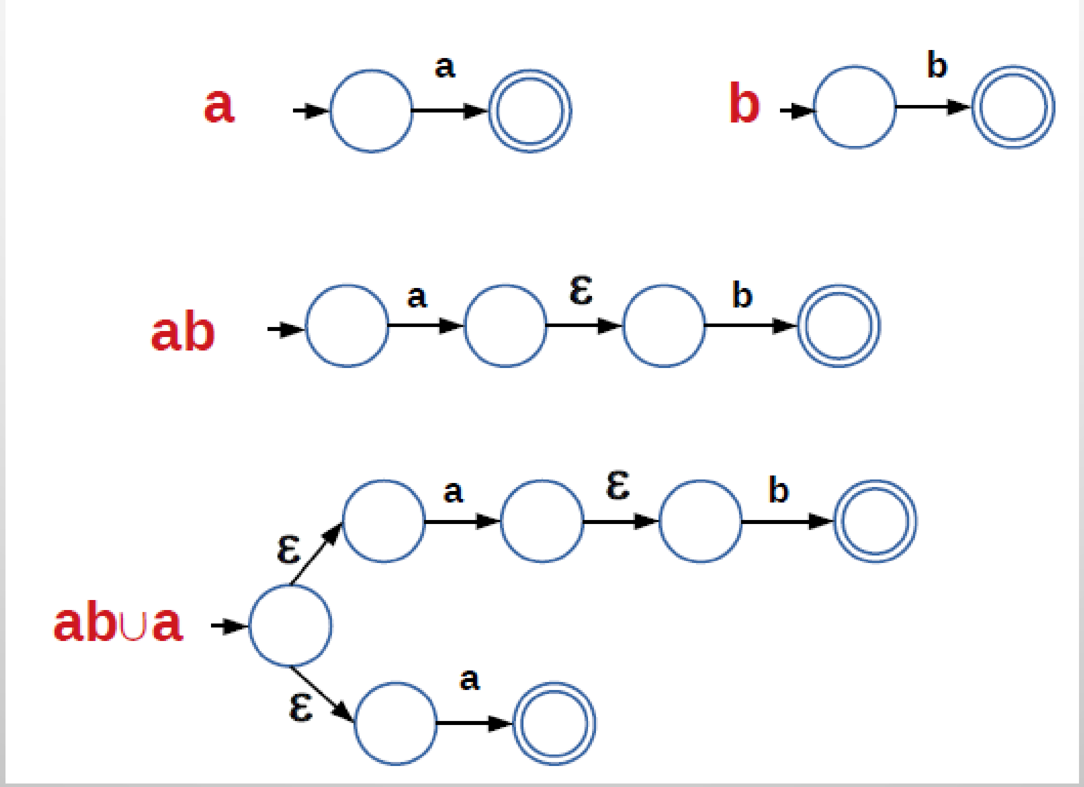
Exemplos (vi)

- Converteremos a expressão regular em um AFN numa sequência de estágios.
- Construiremos a partir das subexpressões menores em direção às maiores até que tenhamos um AFN para a expressão original.

Equivalência com Autômatos Finitos

Exemplos (vi)

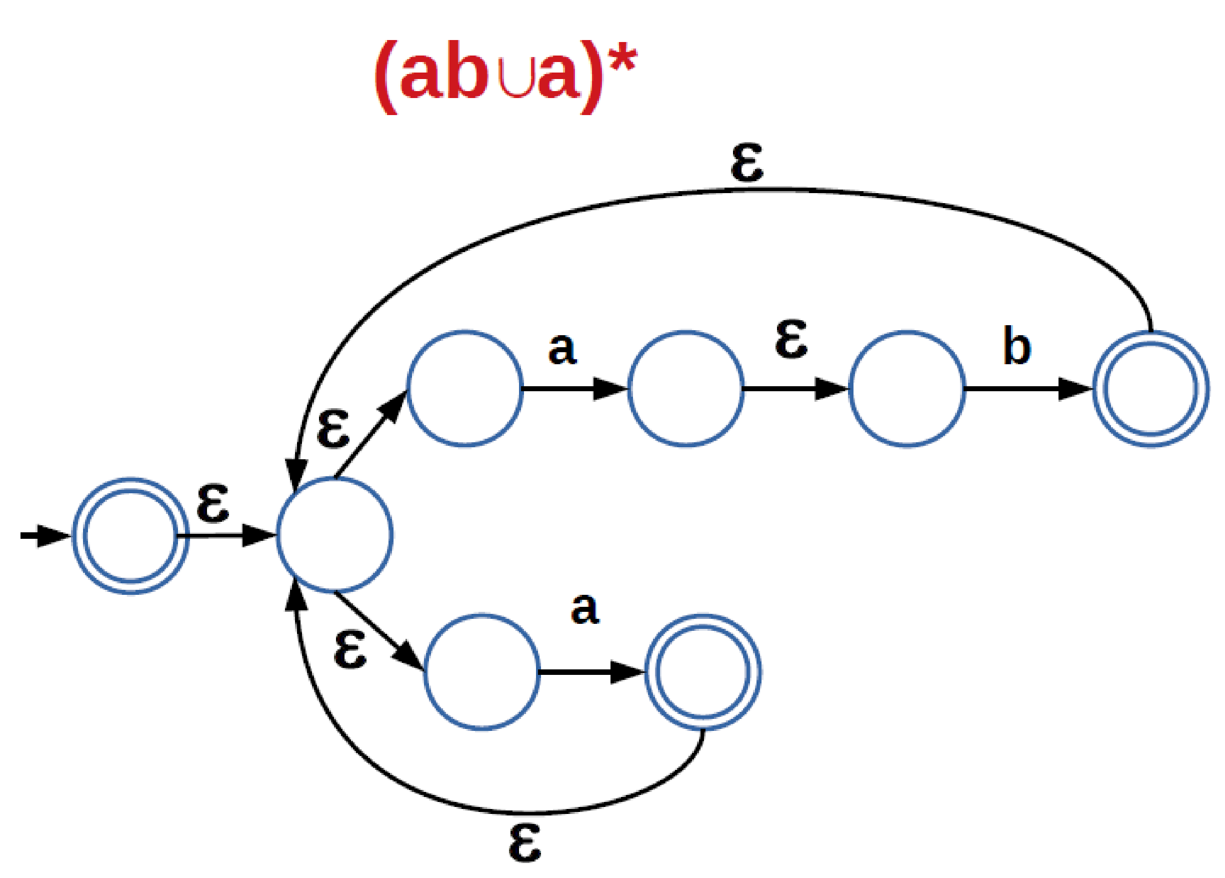
- Construiremos a partir das subexpressões menores em direção às maiores até que tenhamos um AFN para a expressão original.



Equivalência com Autômatos Finitos

Exemplos (vi)

Resultado



Equivalência com Autômatos Finitos

Exemplos (vii)

Exiba a linguagem na notação de conjunto.

Equivalência com Autômatos Finitos

Exemplos (viii)

Para , a expressão é regular e denota a linguagem .

Isso pode ser verificado considerando-se as várias parte de :

- 1ª parte: significa qualquer posição de uma *string* e
- 2ª parte: representa ou um ou um duplo .

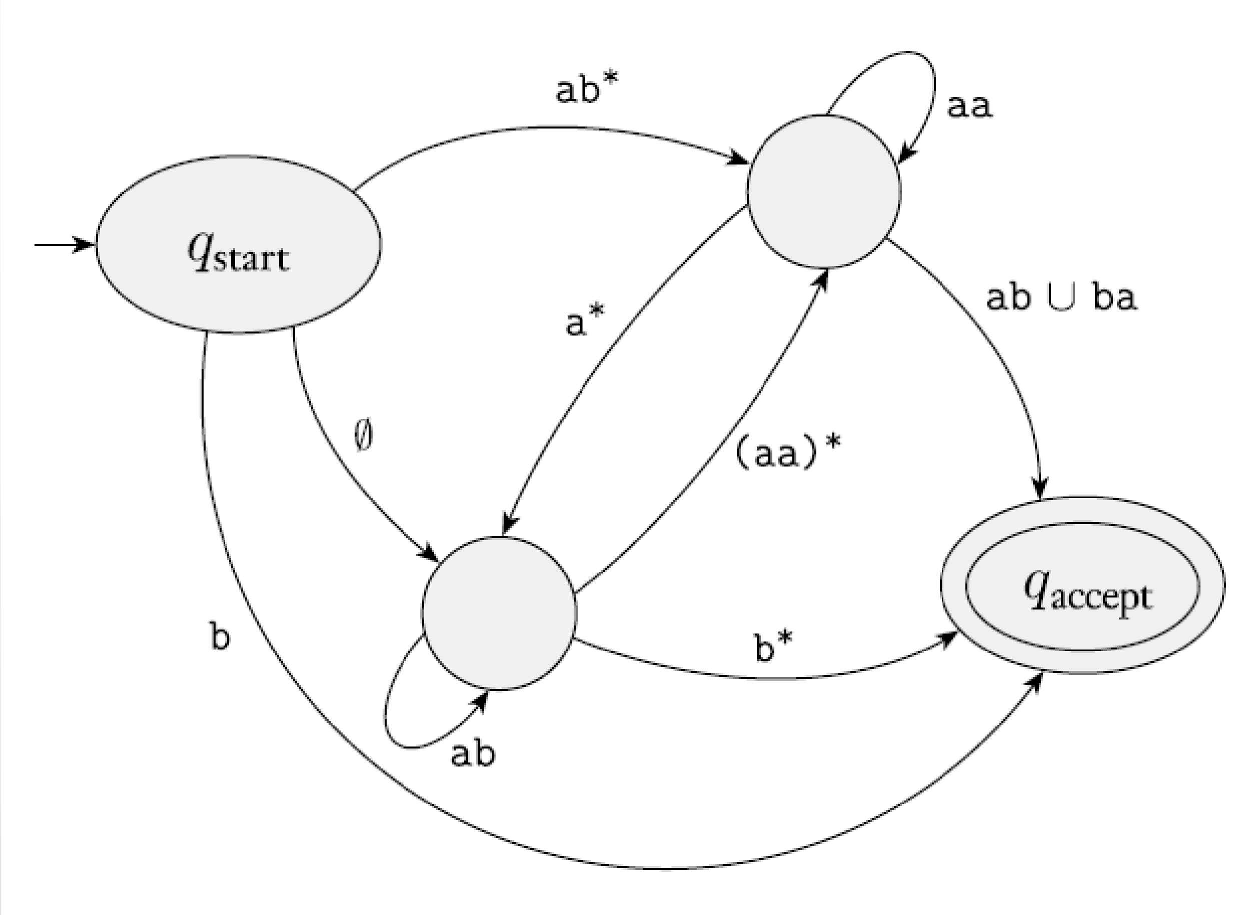
Consequentemente, é o conjunto de todas as *strings* em , terminadas ou por um ou por um .

Autômato Finito Não-Determinístico Generalizado (AFNG)

- AFNs generalizados (AFNG) são AFNs nos quais as setas de transição *podem ter quaisquer expressões regulares como rótulos*, em vez de apenas membros do alfabeto e .
- Por ser não-determinístico, um AFNG pode ter várias maneiras de processar uma mesma cadeia de entrada.

Autômato Finito Não-Determinístico Generalizado (AFNG)

- Exemplo de AFNG:



Autômato Finito Não-Determinístico Generalizado (AFNG)

- Regras para um AFNG:

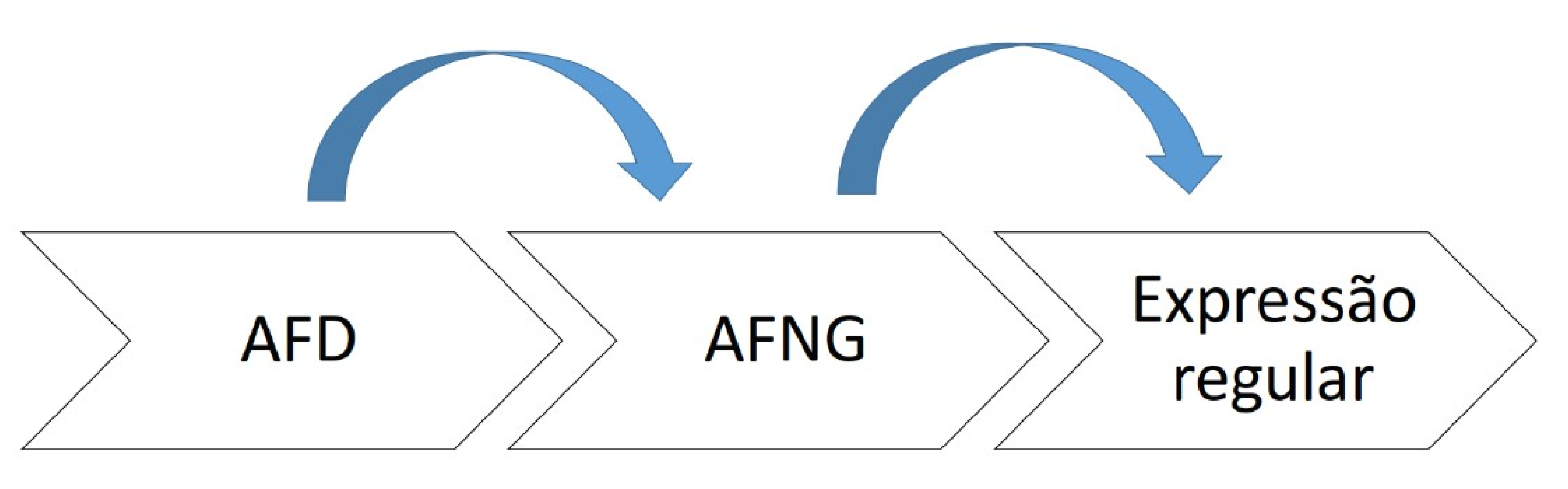
➢ **Estado inicial**: tem setas saindo para todos os outros estados, mas nenhuma seta chegando nele.

➢ **Estado de aceitação**: existe apenas um, e ele tem setas chegando de todos os outros estados, mas nenhuma saindo dele. O estado de aceitação deve ser diferente do estado inicial.

➢ **Demais estados**: cada um possui seta para todos os demais, inclusive para si.

Autômato Finito Não-Determinístico Generalizado (AFNG)

- Conversões:



Autômato Finito Não-Determinístico Generalizado (AFNG)

- Regras para um AFNG:

1. Adicionar NOVO estado inicial com uma seta de transição apontando para o estado inicial antigo.
2. Adicionar NOVO estado de aceitação com uma seta de transição chegando dos estados de aceitação antigo.
3. Se existirem setas com múltiplos rótulos (ou se há múltiplas setas entre dois estados na mesma direção), substitui-se cada uma por uma única seta cujo rótulo é a união dos rótulos anteriores.
4. Adicionar setas com rótulos ; entre estados que não tenham setas.

Autômato Finito Não-Determinístico Generalizado (AFNG)

- Converter AFNG para Expressão Regular:

➢ Suponha um AFNG com estados.

➢ Como um AFNG deve ter um estado inicial diferente do de aceitação, sabemos que .

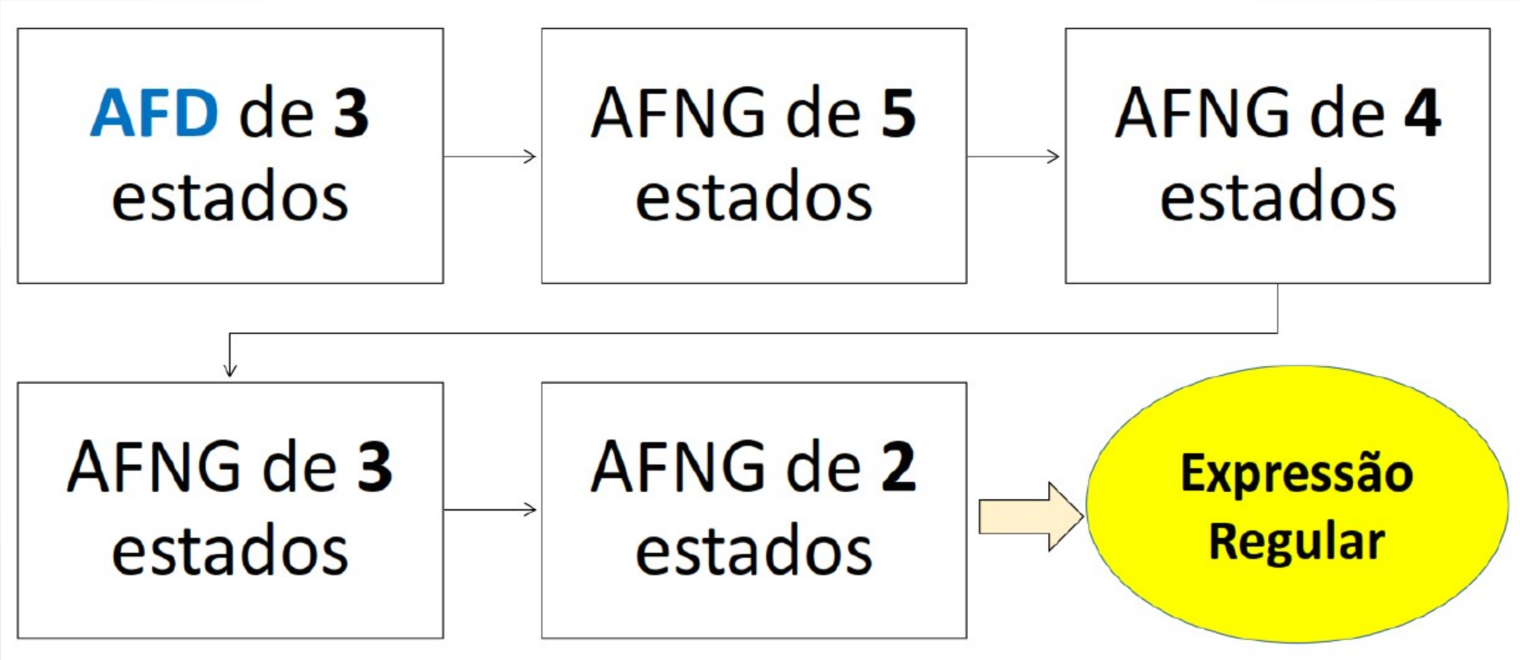
➢ Se , construímos um AFNG equivalente com estados. Esse passo repete-se até que o AFNG esteja reduzido a dois estados.

➢ Se , o AFNG tem uma única seta, que vai do estado inicial para o estado de aceitação.

➢ O rótulo dessa seta é a expressão regular equivalente.

Autômato Finito Não-Determinístico Generalizado (AFNG)

- Estágios na conversão:



Autômato Finito Não-Determinístico Generalizado (AFNG)

- Estágios na conversão:

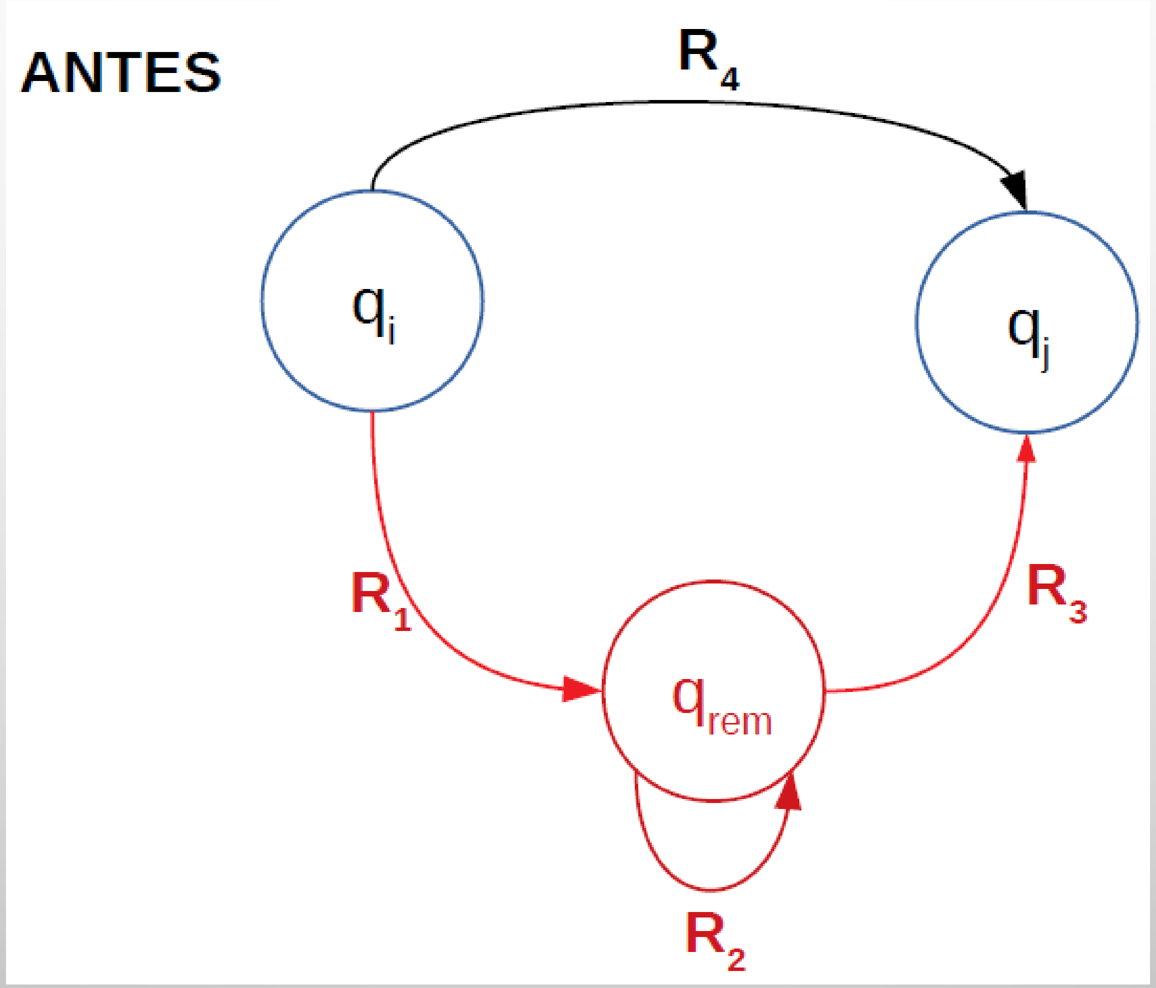
➢ O passo crucial está na construção de um AFNG equivalente com um estado a menos quando .

➢ Fazemos isso selecionando um estado, removendo-o da máquina e reparando o resto de forma que seja reconhecida ainda a mesma linguagem.

➢ **Qualquer estado serve**, desde que não seja o estado inicial ou de aceitação.

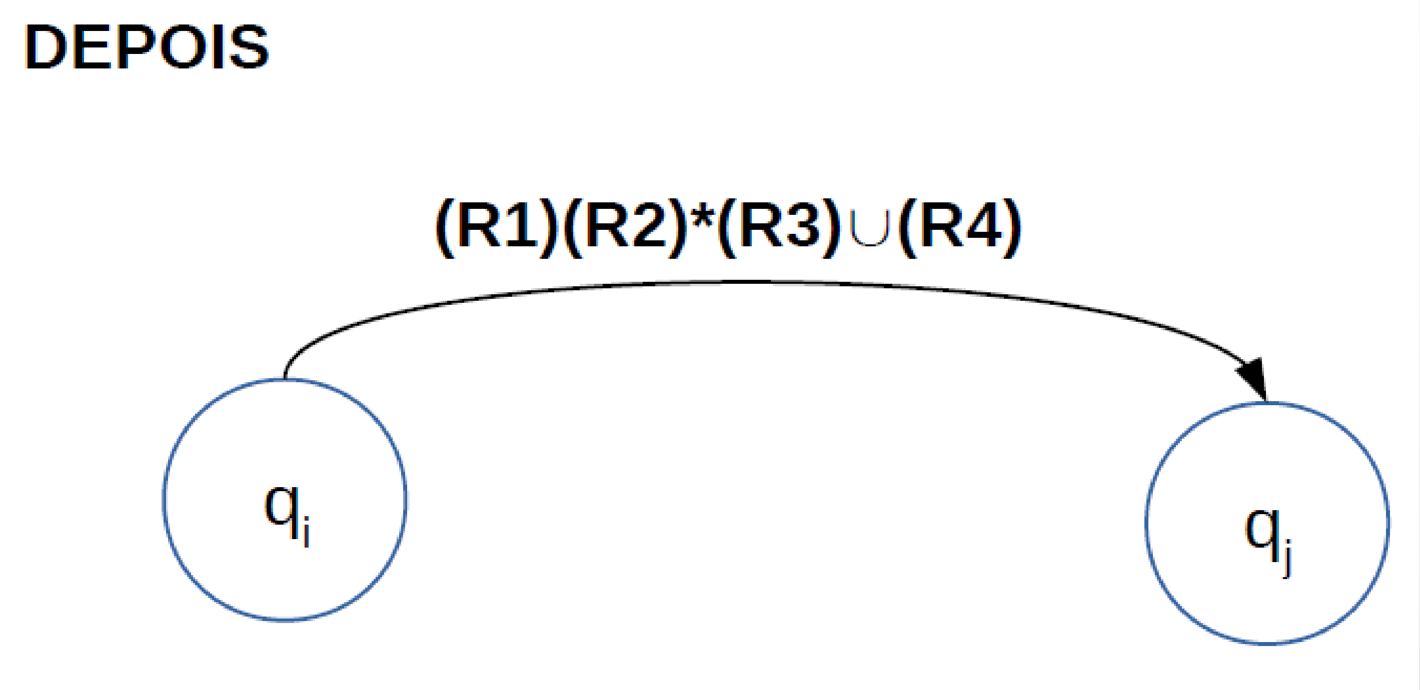
Autômato Finito Não-Determinístico Generalizado (AFNG)

- Construindo um AFNG:



Autômato Finito Não-Determinístico Generalizado (AFNG)

- Construindo um AFNG:



Definição Formal de AFNG

AFNG

Um AFNG é uma 5-upla , onde:

- é o conjunto finito de estados
- Σ é o alfabeto de entrada
- ª é a função de transição
- é o estado inicial
- é o estado de aceitação

ª é a coleção do todas as expressões regulares sobre o alfabeto Σ.

Definição Formal de AFNG

Um AFNG aceita uma cadeia em se , onde cada está em , e existe uma sequência de estados ; tal que

1. é o estado inicial
2. é o estado de aceitação
3. Para cada , temos , onde ; em outras palavras, é a expressão sobre a seta de a .

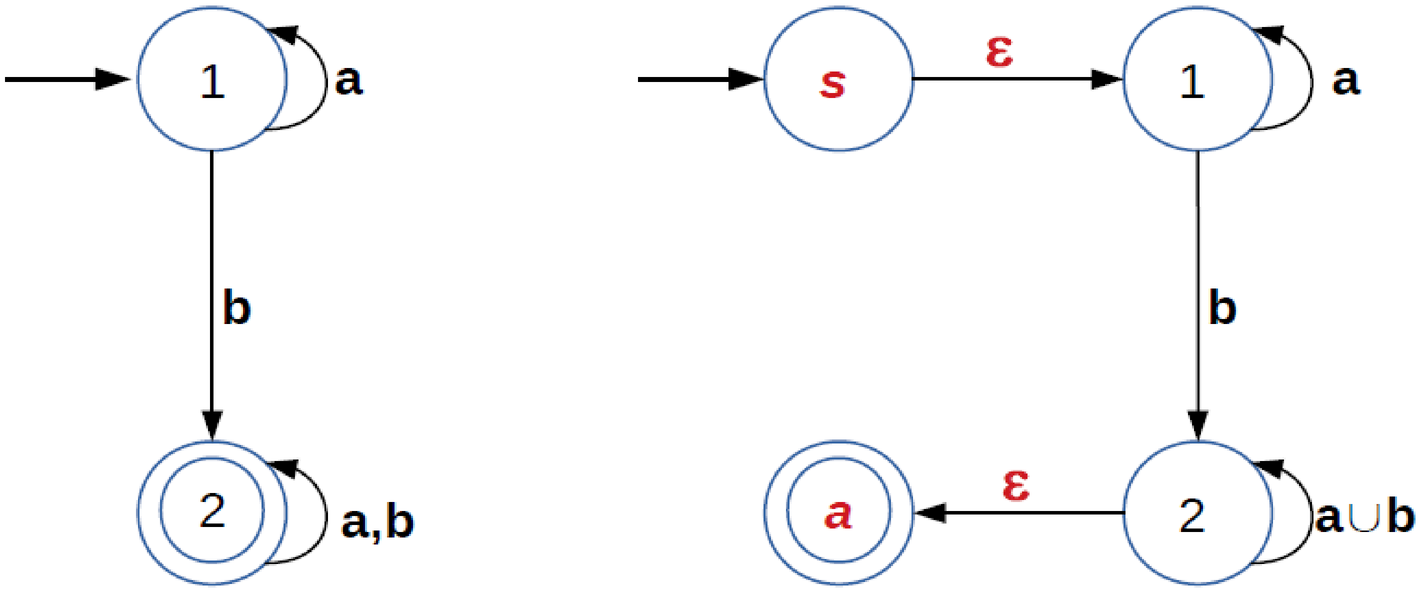
Definição Formal de AFNG

Procedimento CONVERT(G):

1. Seja k o número de estados de G.
2. Se $k = 2$, então G tem um estado inicial e um de aceitação conectados por uma seta que é rotulada com uma expressão regular R. Retorne a expressão R.
3. Se $k > 2$, selecione qualquer $q_{rem} \in Q$ diferente de q_i e q_a e seja G' o AFNG $(Q', \Sigma, \delta', q_i, q_a)$, onde $Q' = Q - q_{rem}$, e para qualquer $q_i \in Q' - q_a$ e qualquer $q_j \in Q' - q_i$ seja $\delta'(q_i, q_j = (R_1)(R_2) * (R_3) \cup (R_4))$, para
   - $R_1 = \delta(q_i, q_{rem})$
   - $R_2 = \delta(q_{rem}, q_{rem})$
   - $R_3 = \delta(q_{rem}, q_j)$
   - $R_4 = \delta(q_i, q_j)$
4. Compute CONVERT(G') e retorne o valor

Exemplo 1: Converter o AFD da figura (a) para uma expressão regular

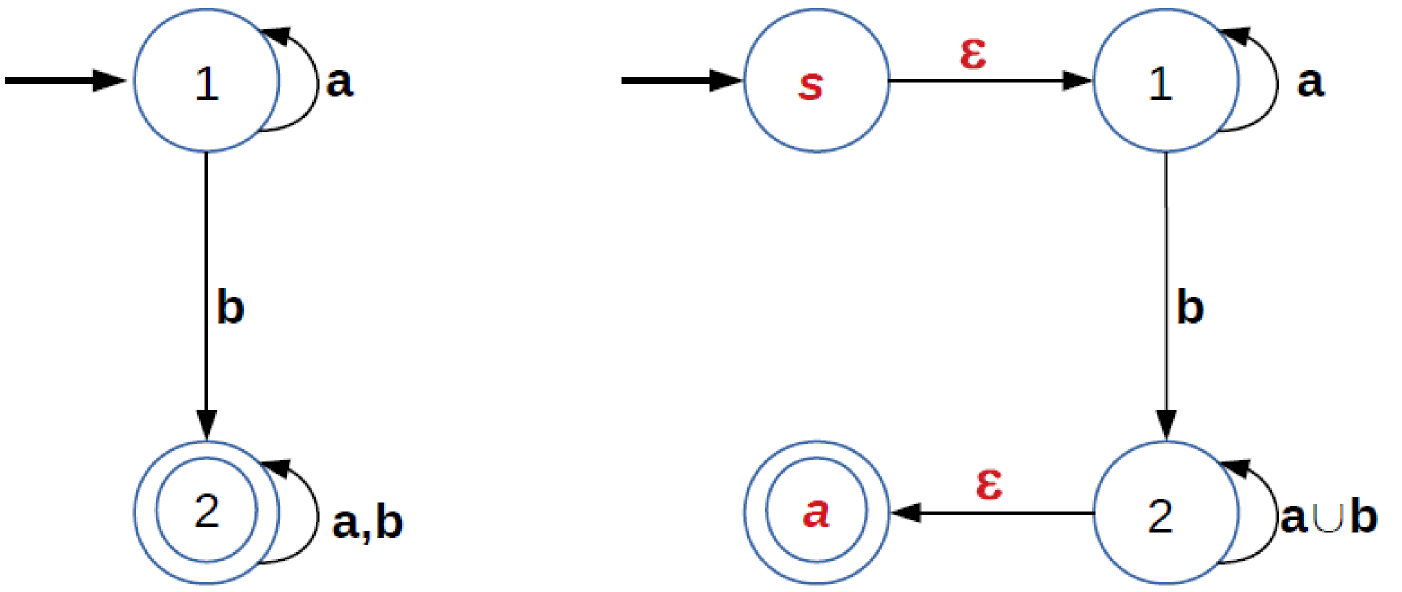
- Começamos um o DFA de 2 estados **(a)**
- Na figura **(b)** construímos um AFNG de 4 estados adicionando um estado inicial e outro de aceitação



(a) (b)

Exemplo 1: Converter o AFD da figura (a) para uma expressão regular

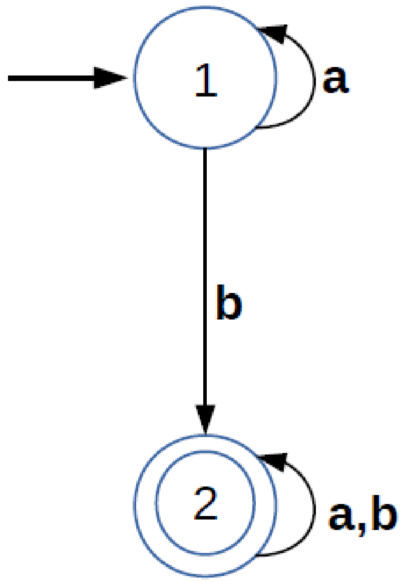
- Para deixar a figura legível, não desenhamos setas com $\emptyset$, mesmo que elas estejam presentes
- Substituimos o rótulo a,b no autoloop do estado 2 por $a \cup b$ no ponto correspondente do AFNG.



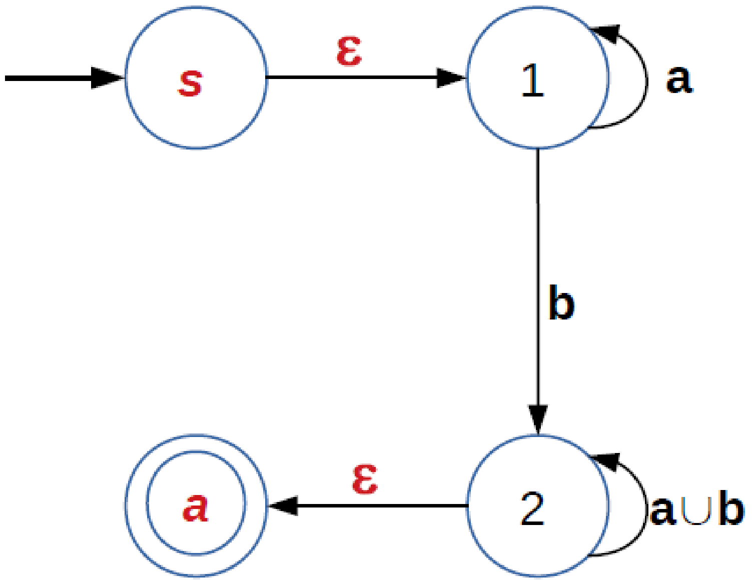
(a) (b)

Exemplo 1: Converter o AFD da figura (a) para uma expressão regular

- Começamos um o DFA de 2 estados **(a)**
- Na figura **(b)** construímos um AFNG de 4 estados adicionando um estado inicial e outro de aceitação



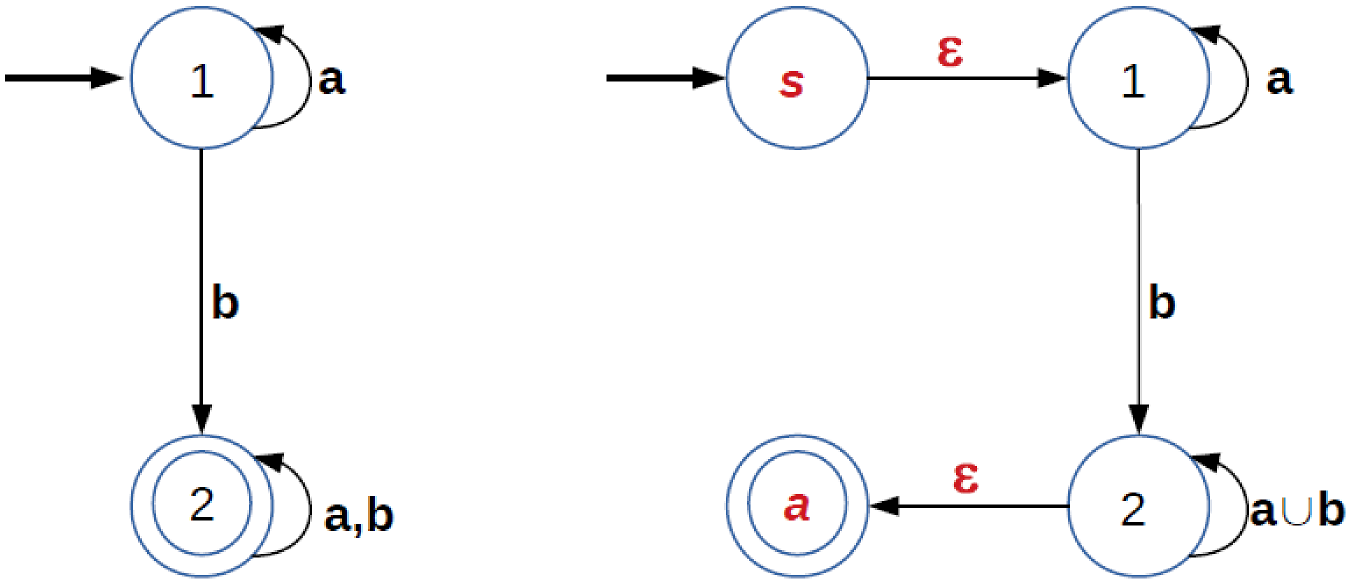
(a)



(b)

Exemplo 1: Converter o AFD da figura (a) para uma expressão regular

- Para deixar a figura legível, não desenhamos setas com $\emptyset$, mesmo que elas estejam presentes
- Substituimos o rótulo a,b no autoloop do estado 2 por $a \cup b$ no ponto correspondente do AFNG.

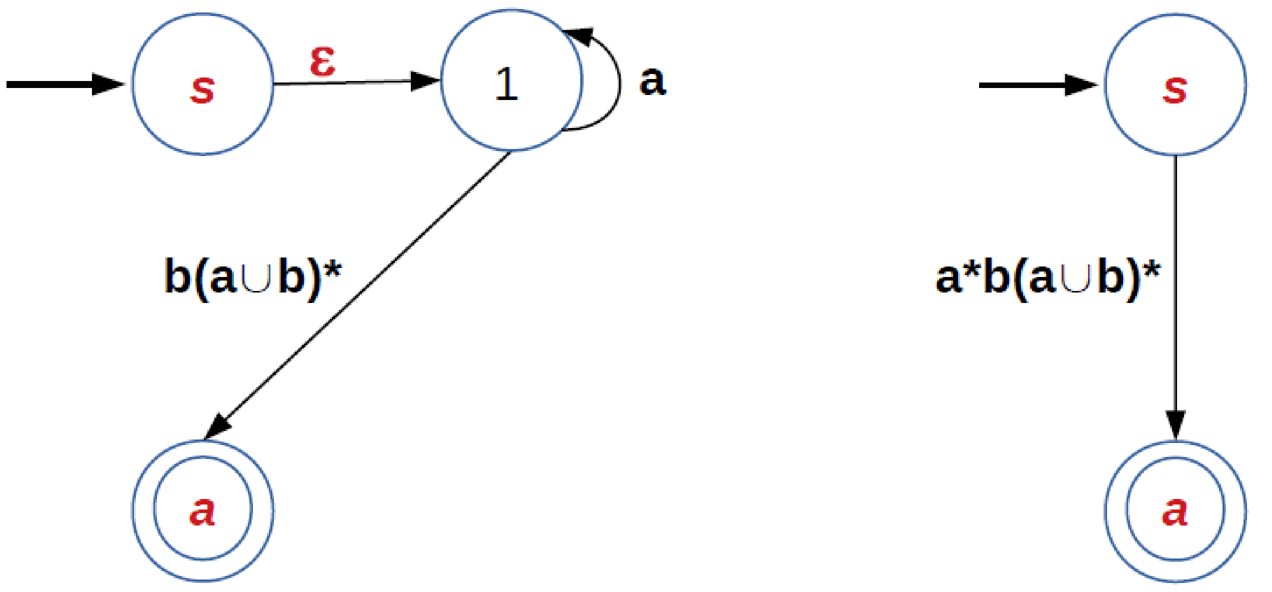


(a)

(b)

Exemplo 1: Converter o AFD da figura (a) para uma expressão regular

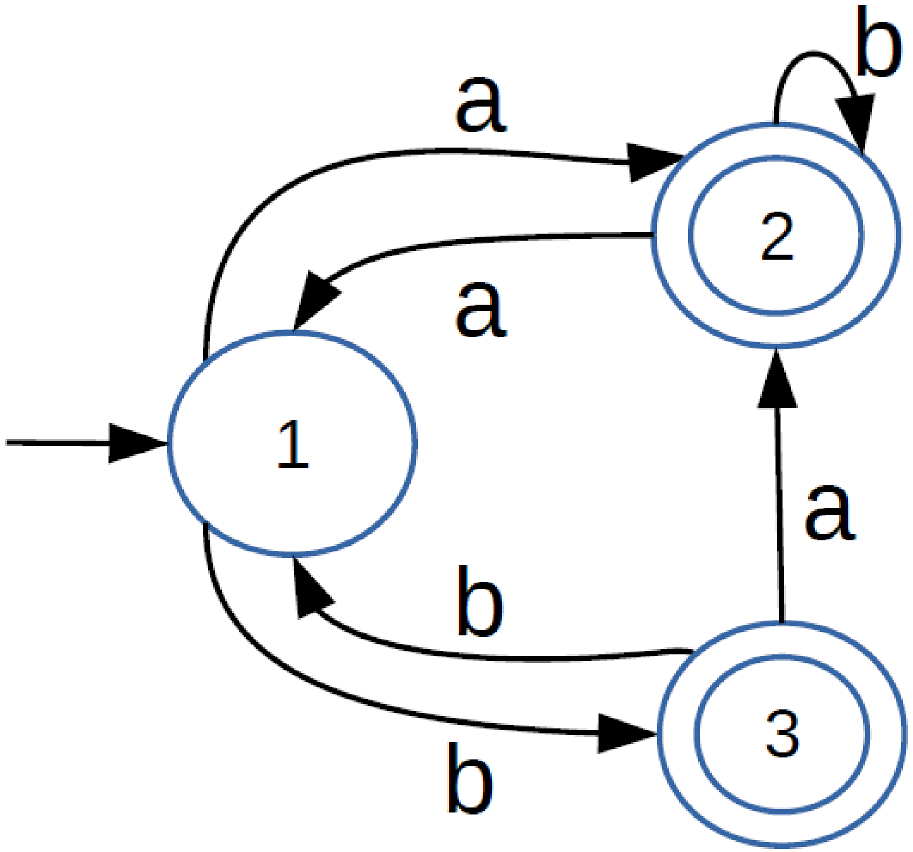
- Em **(c)** removemos o estado 2 e atualizamos os rótulos das setas
- $q_k = q_a$ é o estado de aceitação
- Na figura **(d)** removemos o estado 1 da parte (c) e seguimos o mesmo procedimento



(c) (d)

Exemplo 2: Converter o AFD da figura (a) para uma expressão regular

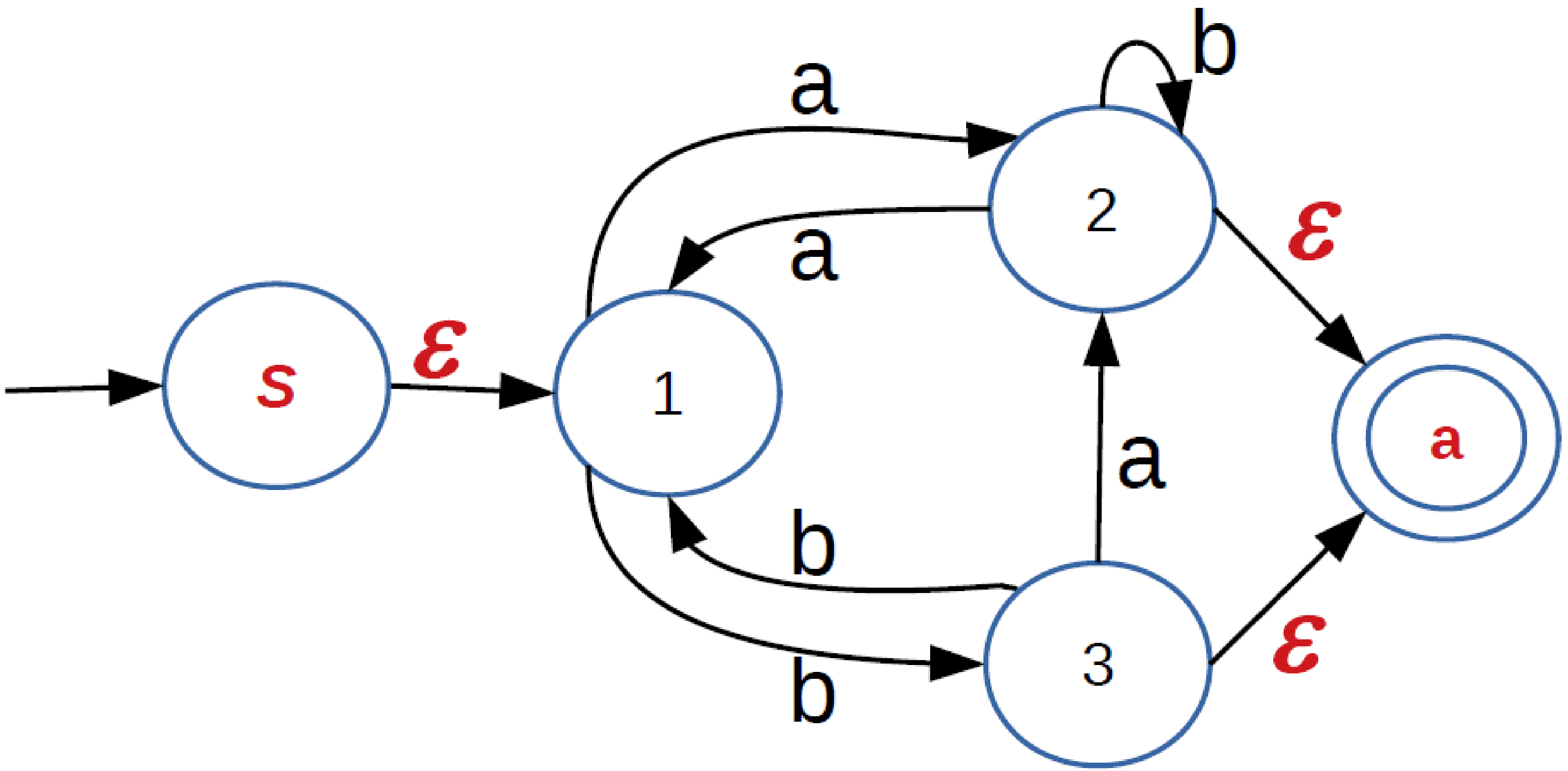
- Começamos um o DFA de 3 estados **(a)**



(a)

Exemplo 2: Converter o AFD da figura (a) para uma expressão regular

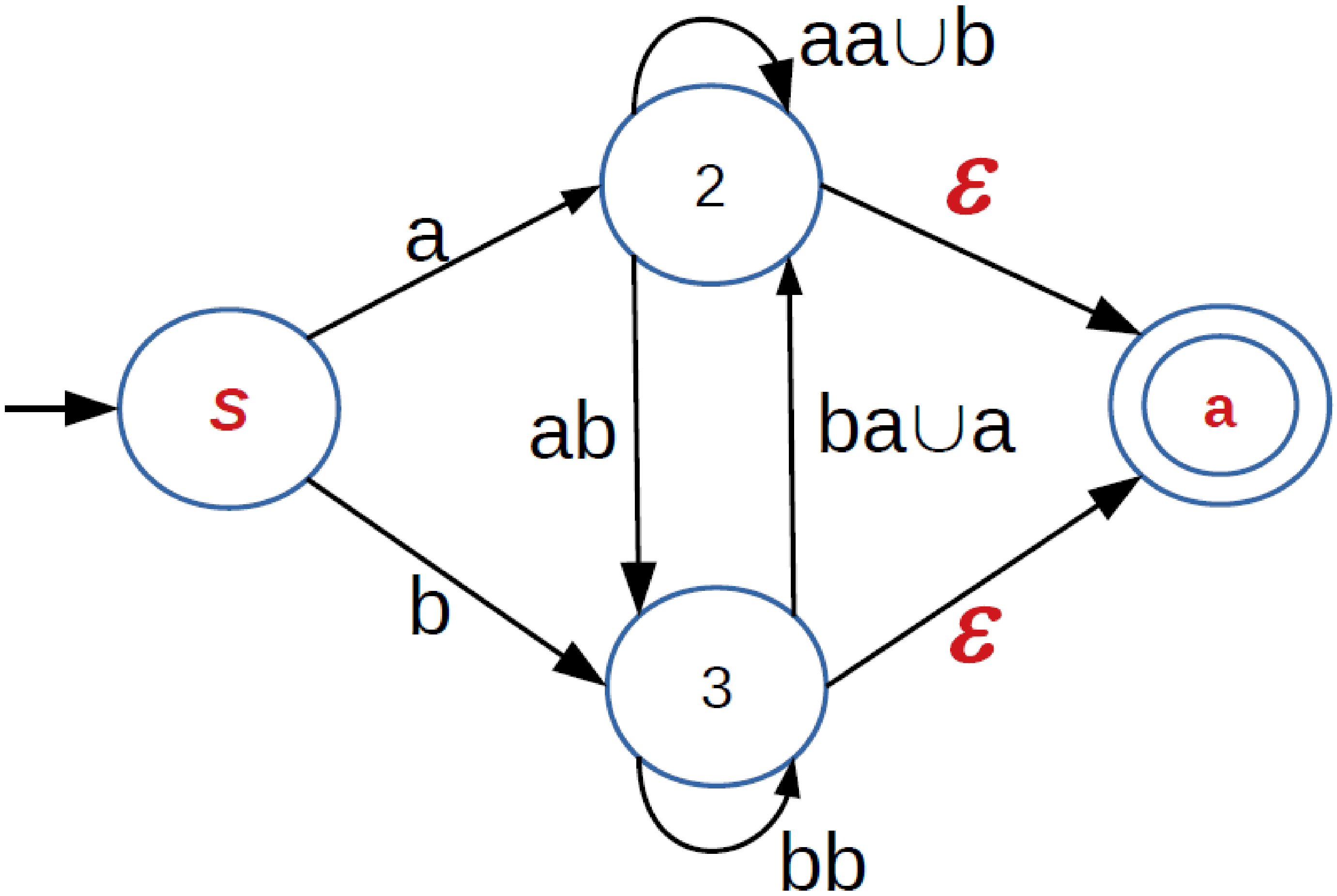
- Na figura **(b)** construímos um AFNG de 5 estados adicionando um **estado inicial** (s) e outro de **aceitação** (a)



(b)

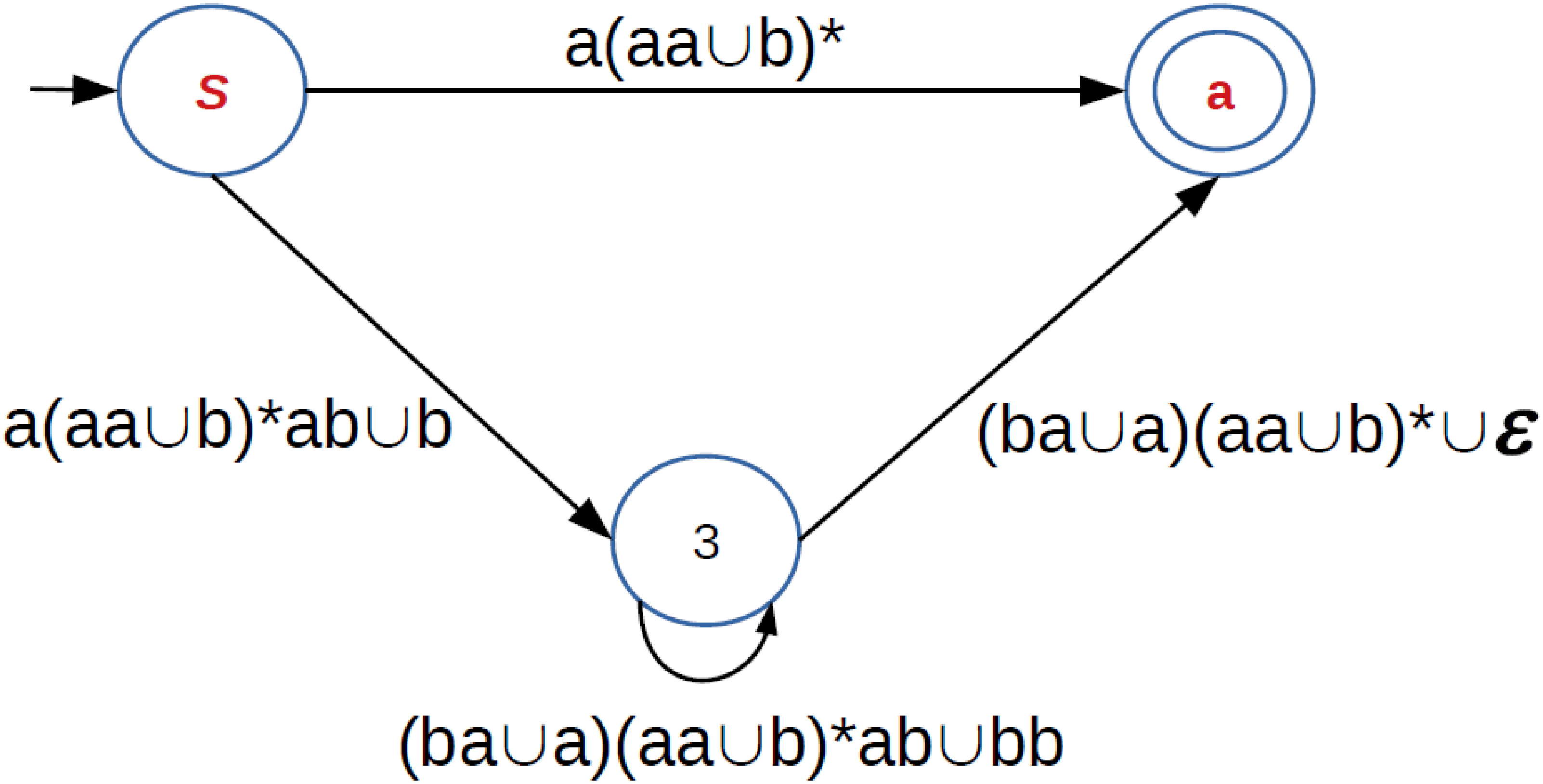
Exemplo 2: Converter o AFD da figura (a) para uma expressão regular

- Na figura **(c)**, removemos o estado (1)



Exemplo 2: Converter o AFD da figura (a) para uma expressão regular

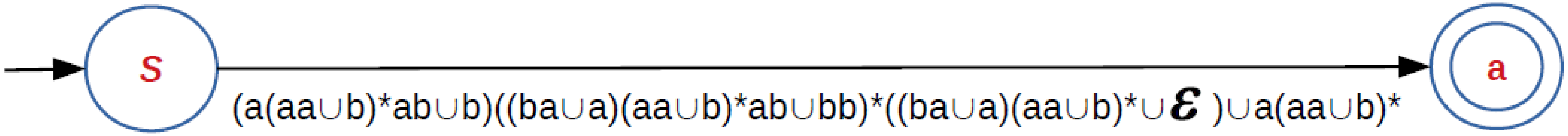
- Na figura **(d)** removemos o estado **2**



(d)

Exemplo 2: Converter o AFD da figura (a) para uma expressão regular

- O resultado obtido:



(e)