UNIDADE II

Linguagens Formais e Autômatos

Prof. Me. Roberto Leminski

Autômatos Finitos Não Determinísticos

- Existem Linguagens Regulares cujos Autômatos Finitos acabam sendo complexos demais, com muitos estados e muitas transições entre eles.

- Uma alternativa para simplificar é a construção de um Autômato Finito Não Determinístico.

- Diferente do autômato “comum”, nestes autômatos um elemento da cadeia analisada pode provocar uma mudança para vários estados.

    - Assim, a função de transição não será mais no formato ({estado atual, símbolo na cadeia}, novo estado), passando a ser ({estado atual, símbolo na cadeia}, {conjunto de novos estados}).

Autômatos Finitos Não Determinísticos

- A aplicação de um Autômato Não Determinístico é reduzir o número de estados e transições presentes em um Autômato Determinístico.

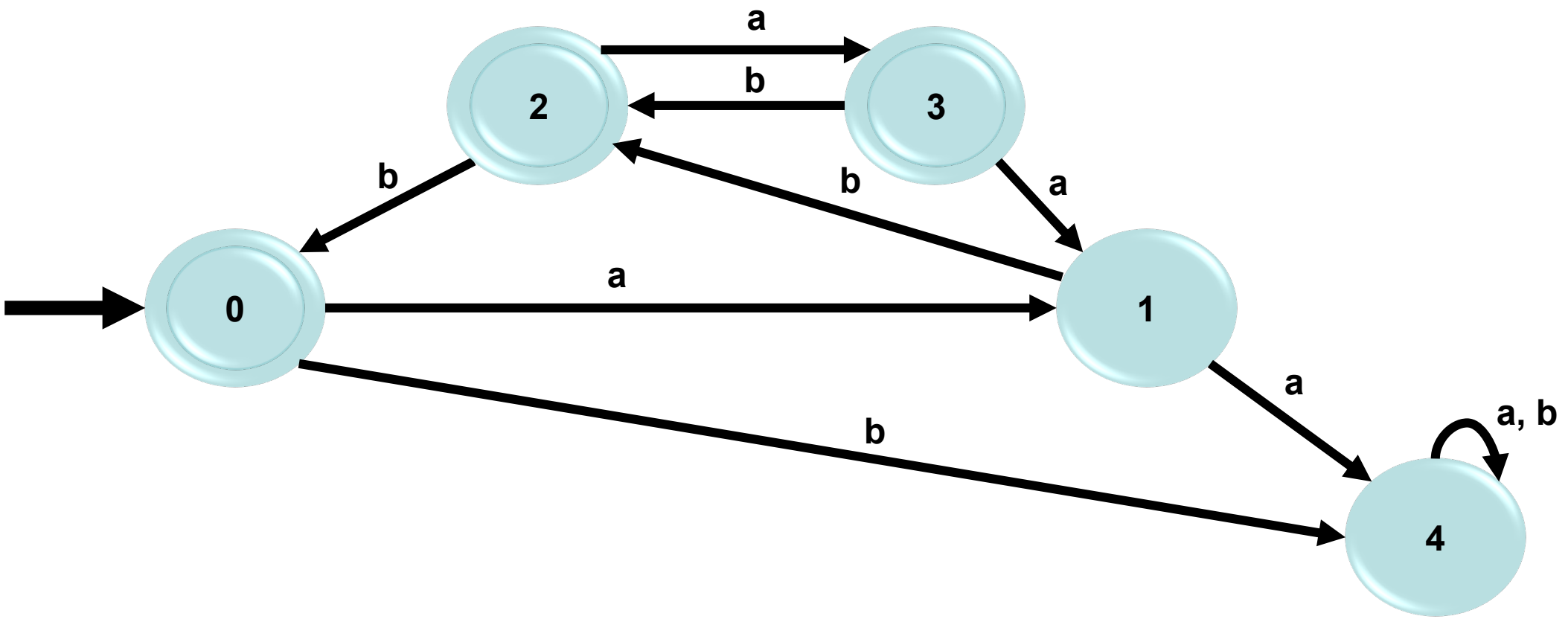
- Assim, apesar de, a cada símbolo lido na cadeia, estarmos percorrendo mais de um caminho, a complexidade da execução do autômato como um todo diminui.

- Se, ao término da cadeia, qualquer uma das rotas traçadas neste autômato tiver terminado em um estado final, a cadeia é aceita.

    - Um exemplo: comparar as versões determinísticas e não determinísticas de autômato para a gramática (ab + aba + abb)*.

Autômatos Finitos Não Determinísticos

O autômato determinístico de (ab + aba + abb)* é o seguinte:

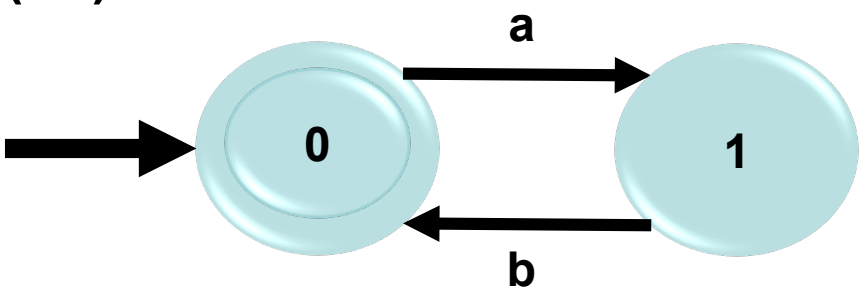


Fonte: Autoria própria.

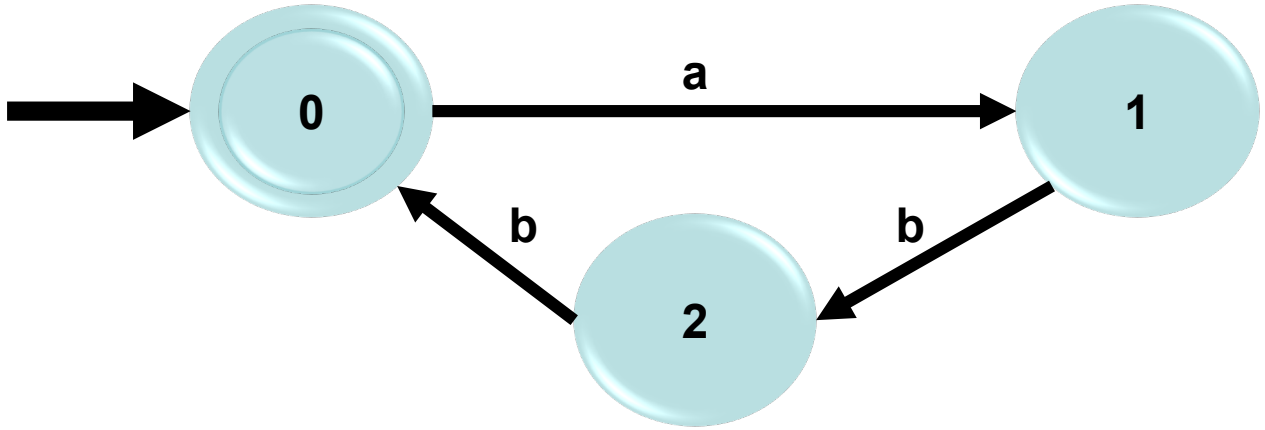
Autômatos Finitos Não Determinísticos

Fazendo um autômato para cada parte da alternância (desconsiderando o estado “sem saída”, para tornar mais claro):

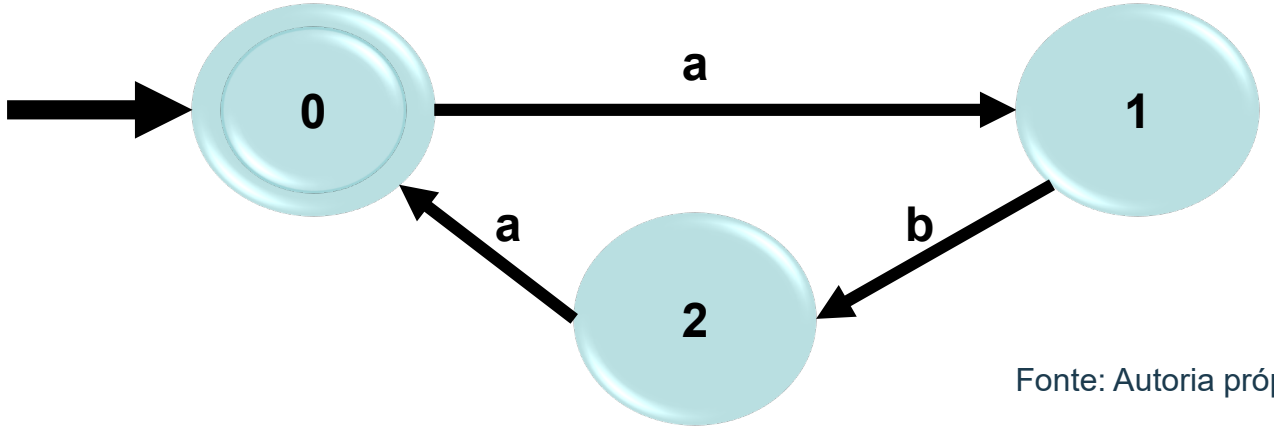
(ab)*



(abb)*



(aba)*

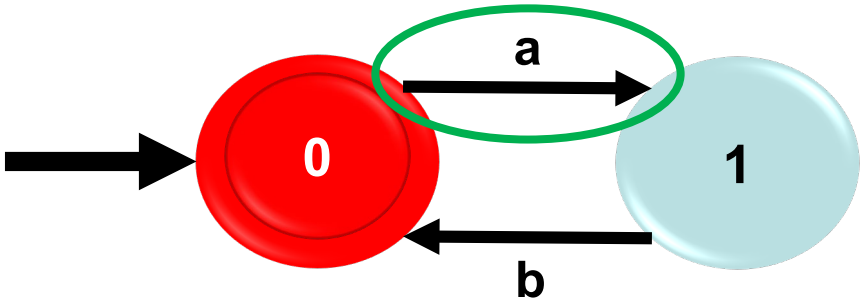


Fonte: Autoria própria.

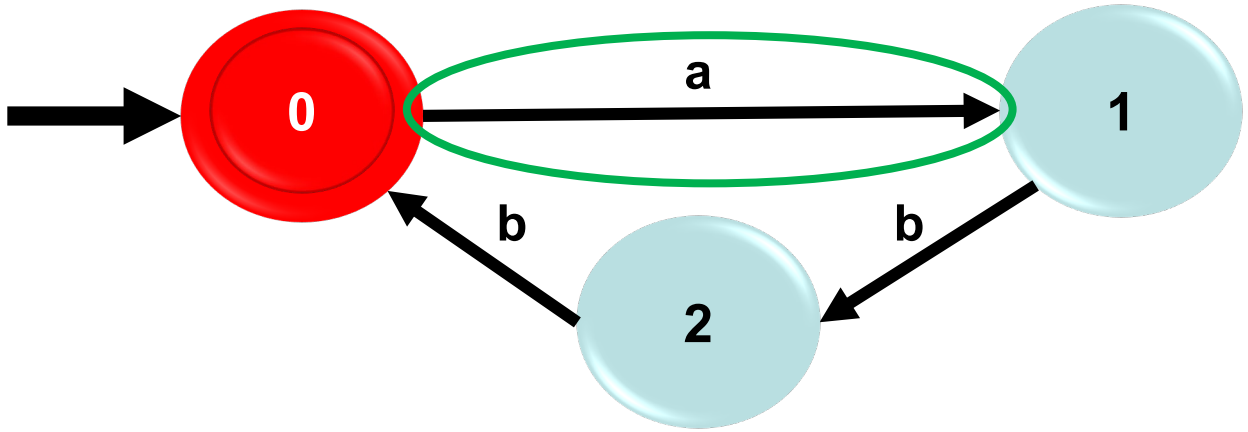
Autômatos Finitos Não Determinísticos

Os três possuem estado final igual ao estado inicial:

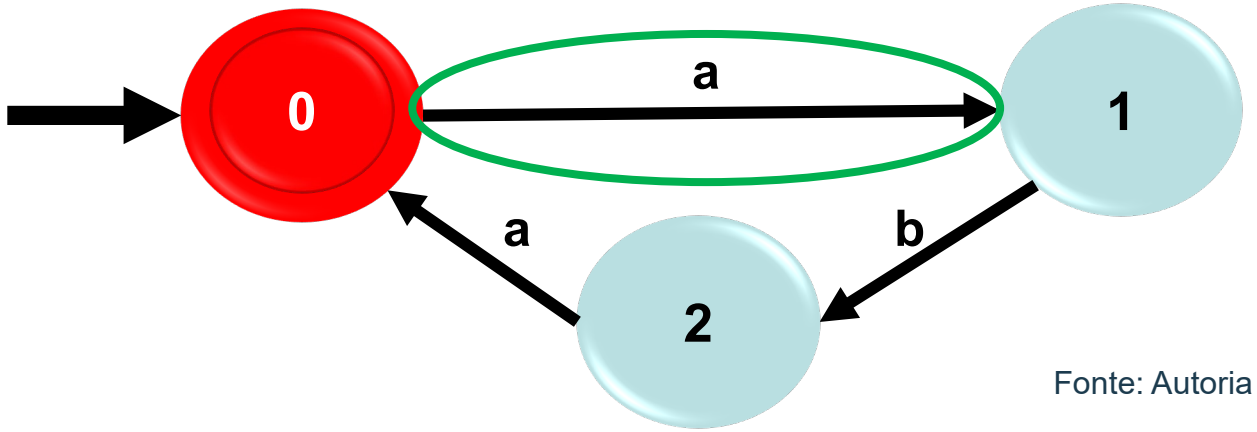
(ab)*



(abb)*



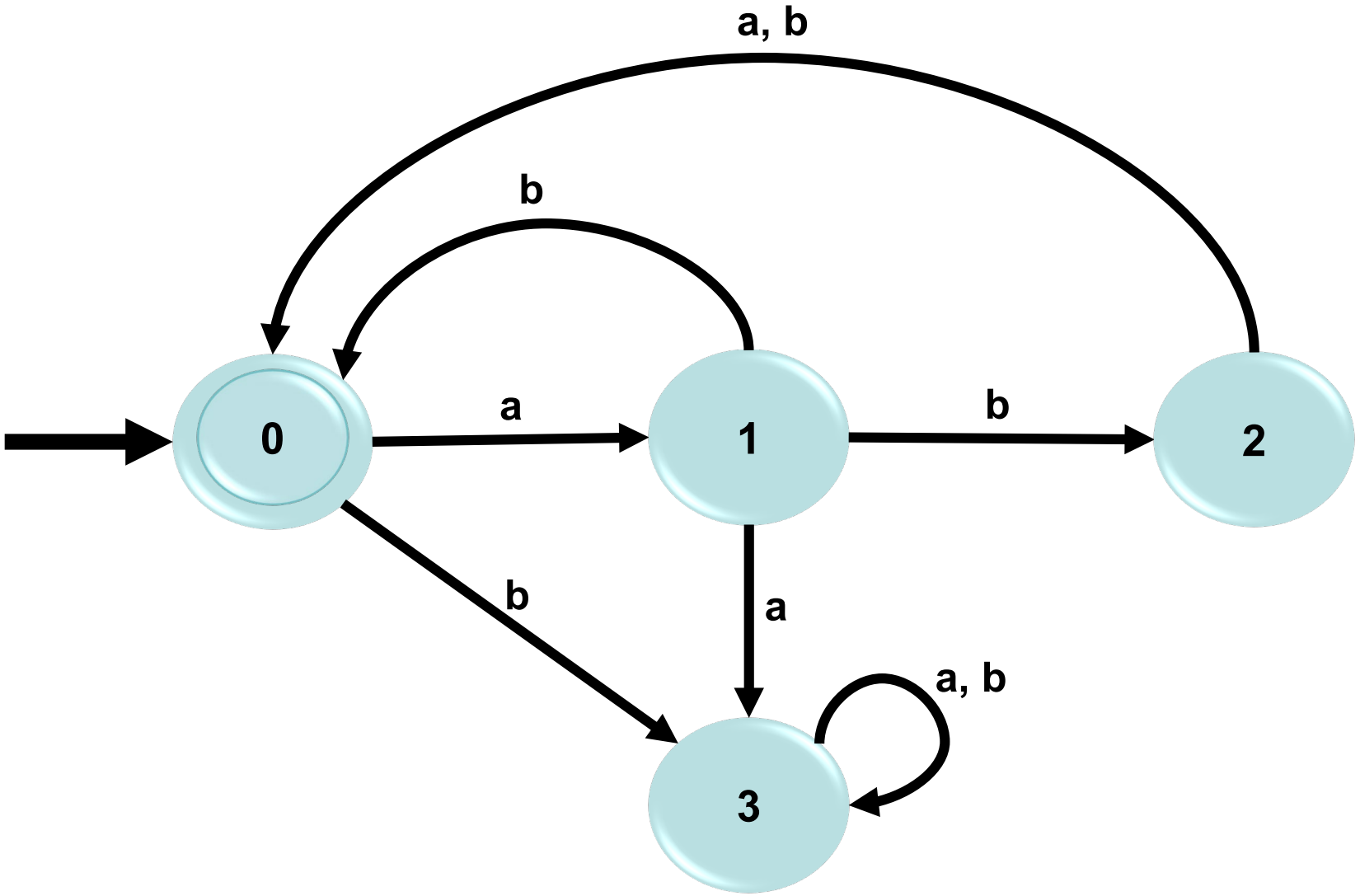
(aba)*



Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Combinando os três autômatos parciais:

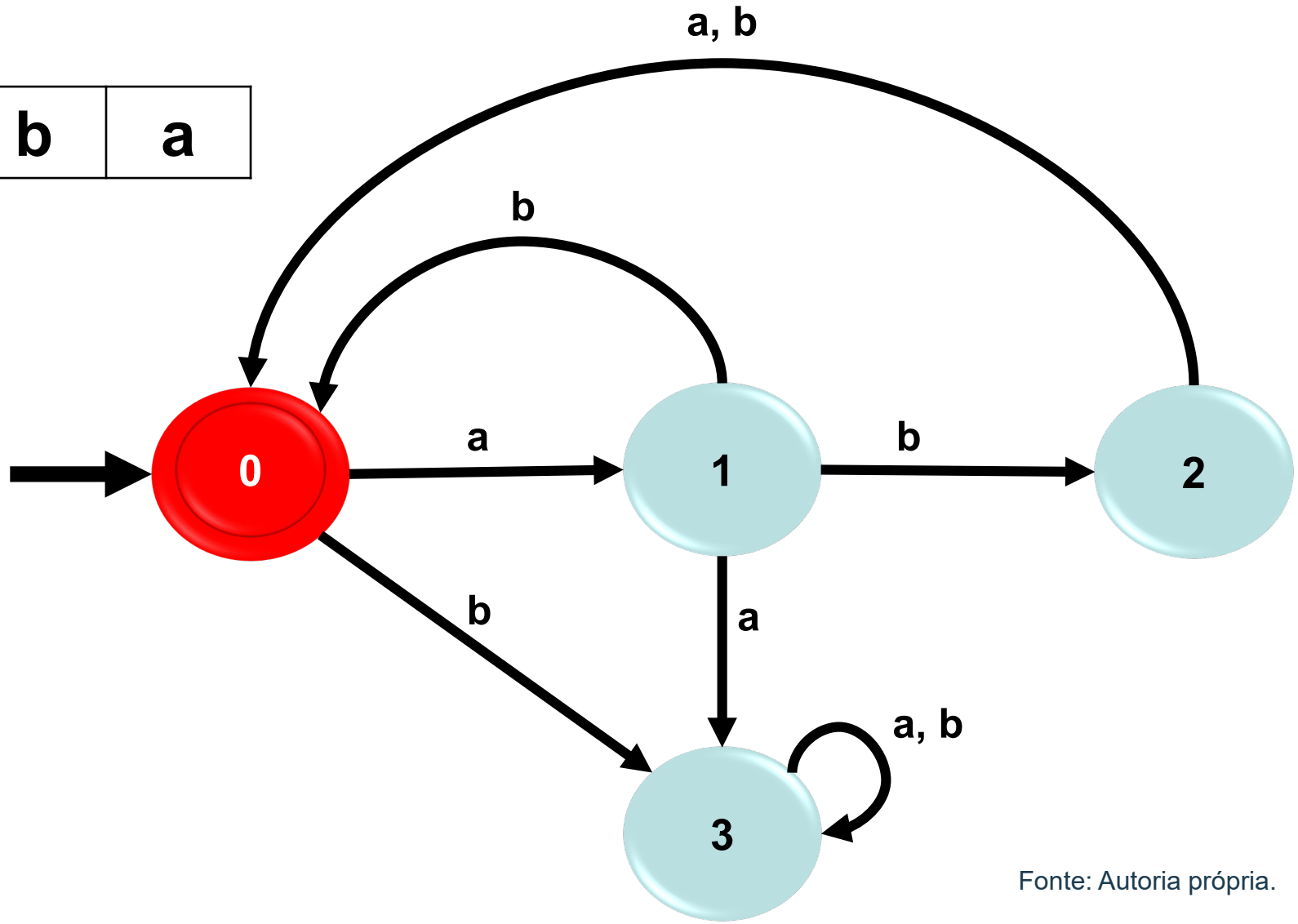


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

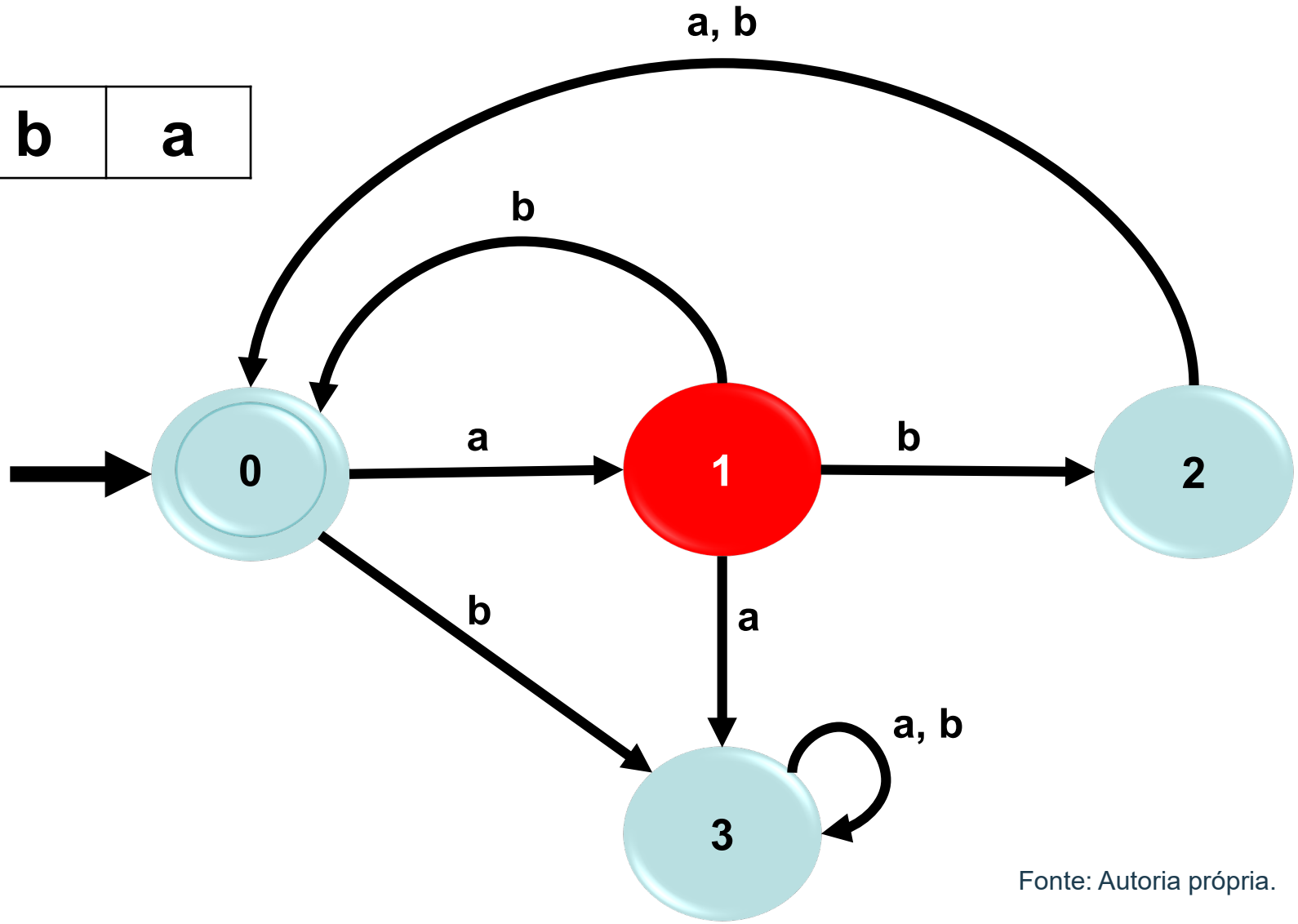


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

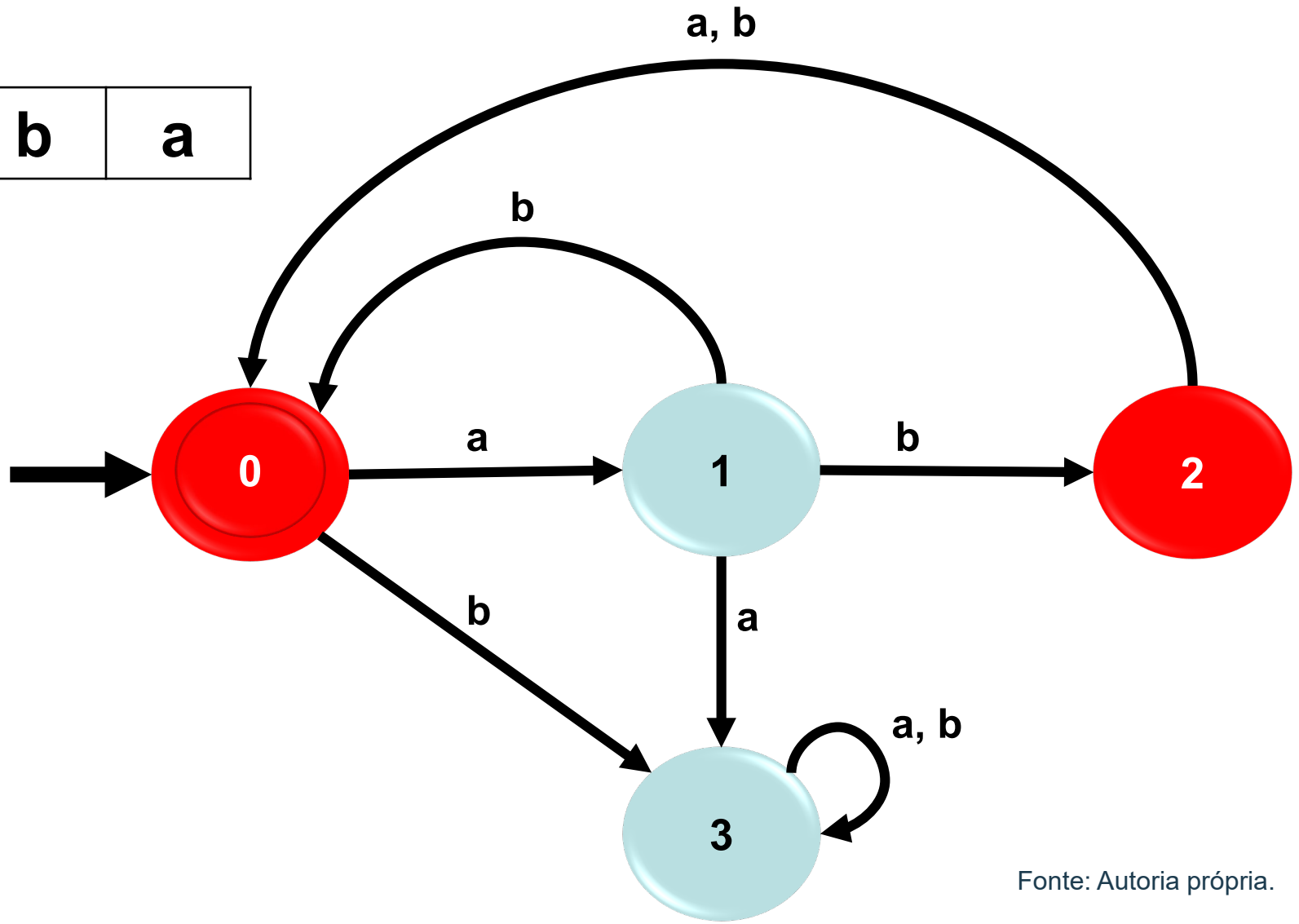


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

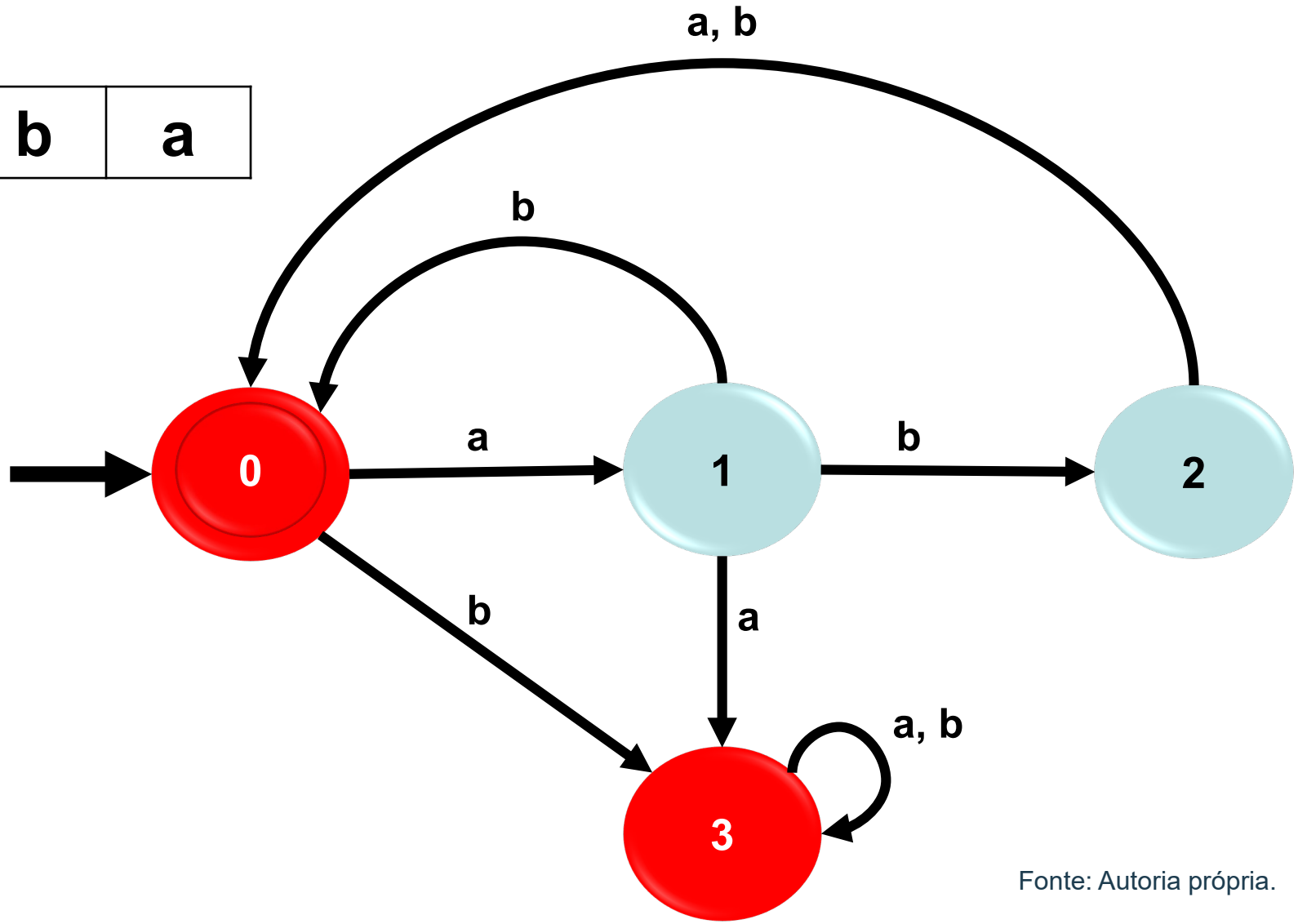


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

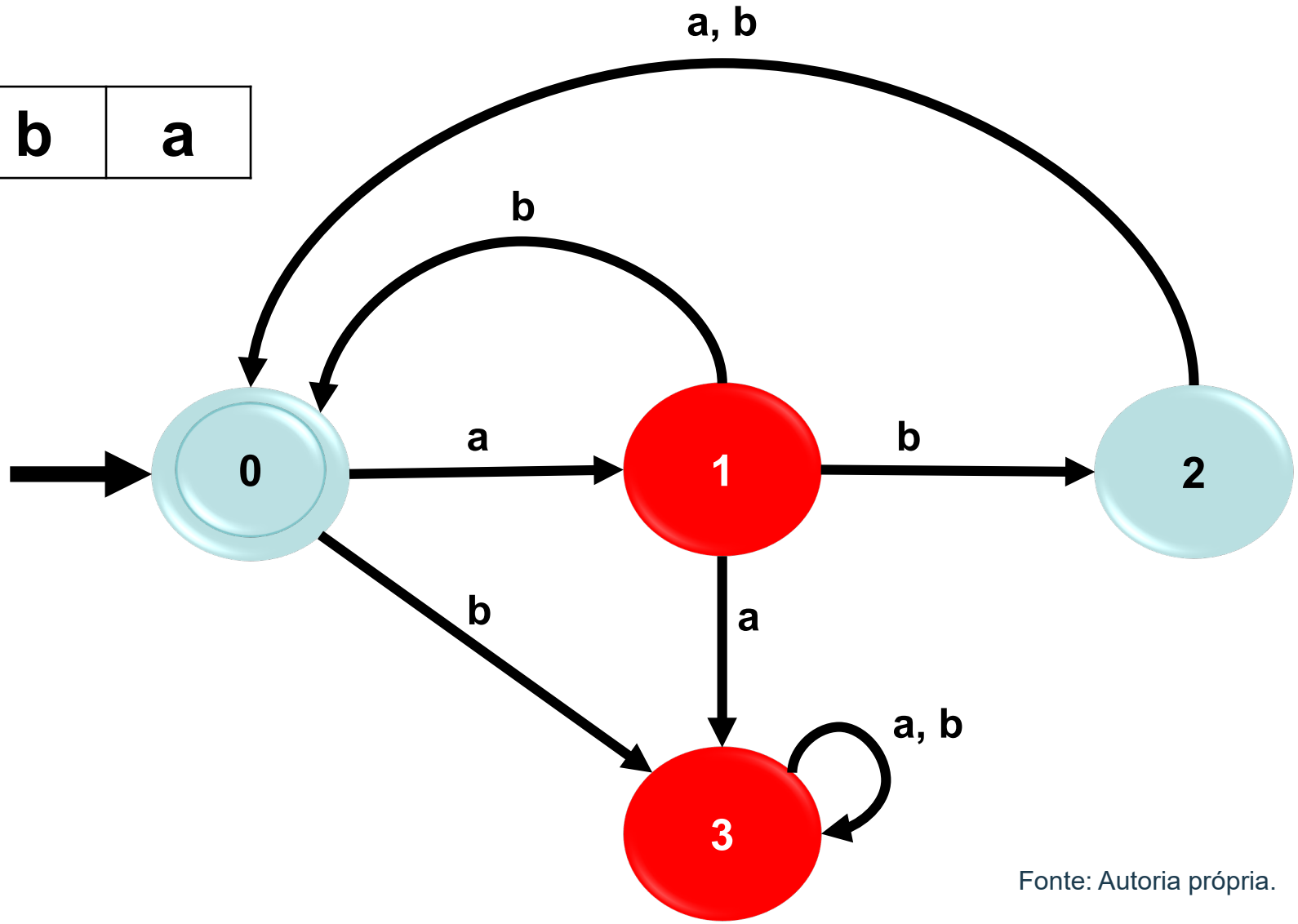


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

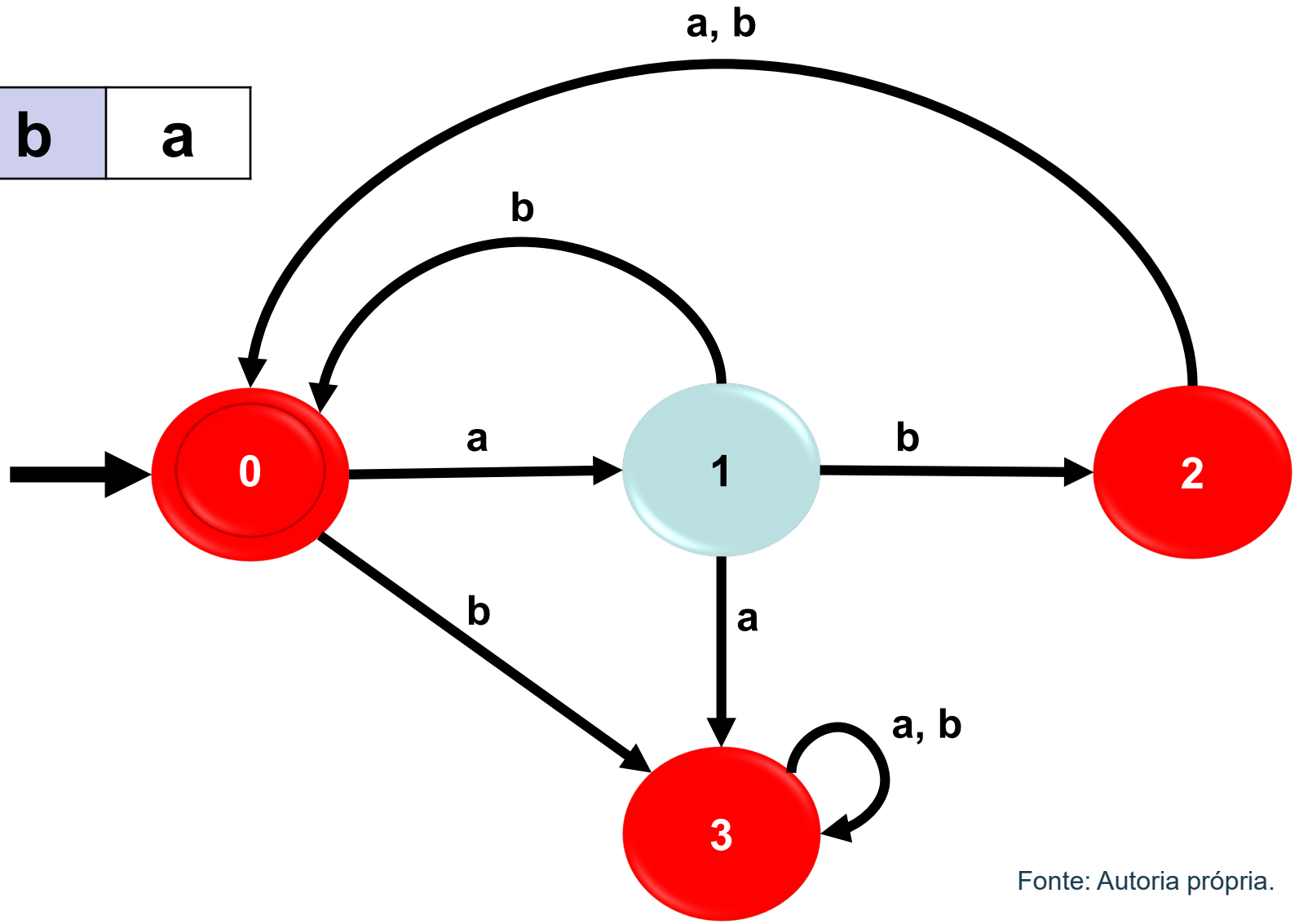


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a

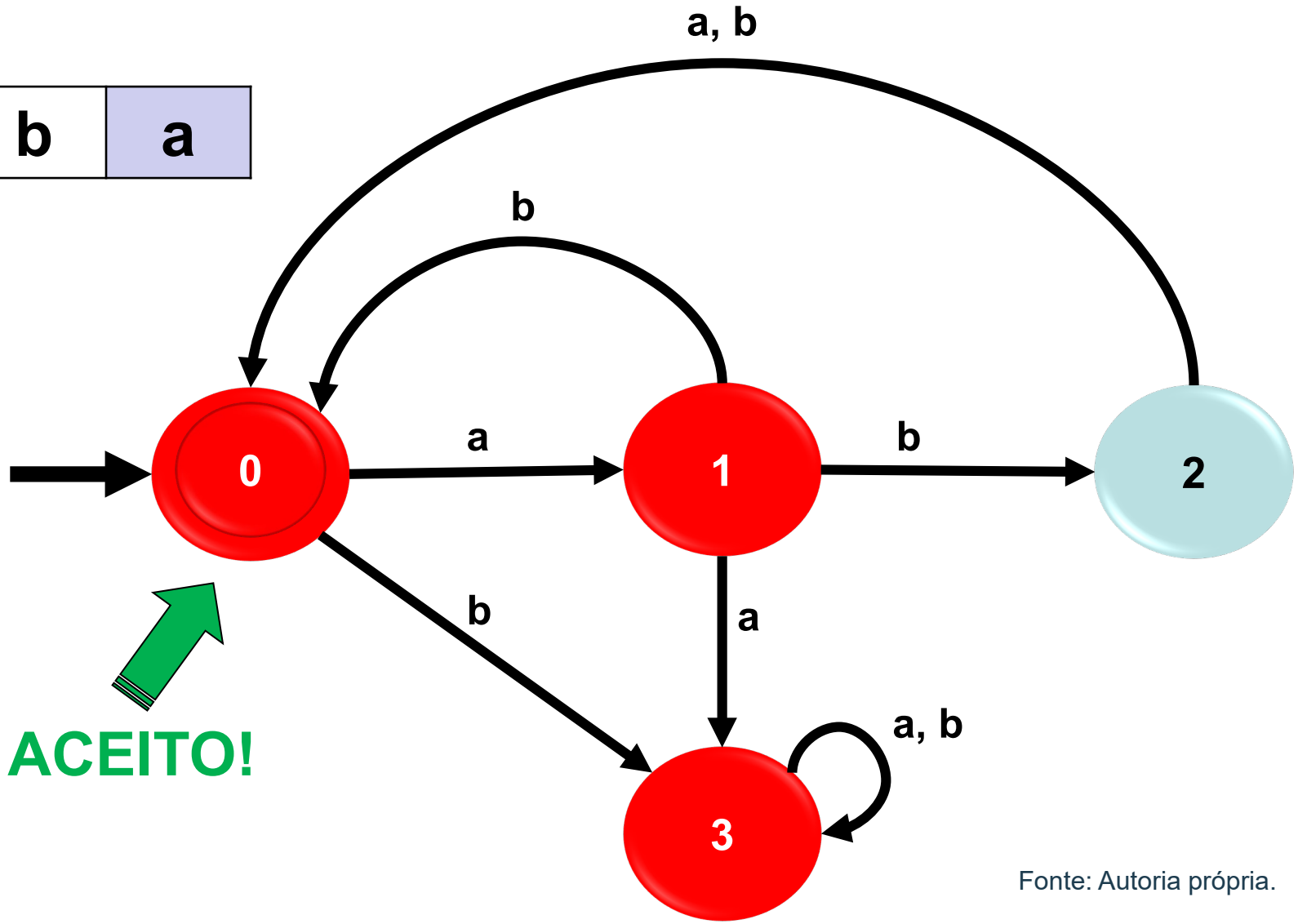


Fonte: Autoria própria.

Autômatos Finitos Não Determinísticos

Testando para a cadeia abbaba:

a	b	b	a	b	a



Fonte: Autoria própria.

Interatividade

Autômatos Finitos Não Determinísticos são utilizados como alternativa para Autômatos Finitos Determinísticos, com a finalidade de simplificar e reduzir o autômato para uma dada gramática regular. Qual é a diferença entre Autômatos Não Determinísticos e Autômatos Determinísticos?

a) Autômatos Não Determinísticos possuem sempre um único estado final.
b) Autômatos Não Determinísticos podem fazer mais de uma transição de estado para uma mesmo símbolo lido na cadeia.
c) Autômatos Não Determinísticos sempre possuirão um número menor de estados.
d) Autômatos Não Determinísticos sempre possuirão um número menor de transições de estados.
e) Autômatos Não Determinísticos não podem sempre ser utilizados para qualquer gramática.

Resposta

Autômatos Finitos Não Determinísticos são utilizados como alternativa para Autômatos Finitos Determinísticos, com a finalidade de simplificar e reduzir o autômato para uma dada gramática regular. Qual é a diferença entre Autômatos Não Determinísticos e Autômatos Determinísticos?

a) Autômatos Não Determinísticos possuem sempre um único estado final.
b) Autômatos Não Determinísticos podem fazer mais de uma transição de estado para uma mesmo símbolo lido na cadeia.
c) Autômatos Não Determinísticos sempre possuirão um número menor de estados.
d) Autômatos Não Determinísticos sempre possuirão um número menor de transições de estados.
e) Autômatos Não Determinísticos não podem sempre ser utilizados para qualquer gramática.

Linguagens Livres de Contexto

- As Linguagens Livres de Contexto (Linguagens de Tipo 2 na Hierarquia de Chomsky) são geradas por Gramáticas Livres de Contexto.

- Trata-se de um dispositivo mais poderoso que aquele das Linguagens Regulares, visto que apresenta regras que permitem expressar como os elementos de uma forma sentencial se organizam em grupos hierárquicos complexos.

  - As Gramáticas Livres de Contexto representam uma importante forma de representação para o processamento das linguagens naturais, bem como das artificiais, em particular das linguagens de programação.

  - O estudo das Linguagens Livres de Contexto é aplicado no projeto e na implementação do analisador sintático, um dos módulos funcionais de um compilador.

Linguagens Livres de Contexto

Seja G = (V, Σ, P, S). Diz-se que G é livre de contexto se as produções apresentarem o seguinte padrão:

$$A \rightarrow b, \text{ onde: } A \in V \text{ e } b \in (V \cup \Sigma)^*$$

- A expressão acima indica que no lado esquerdo da substituição deve existir um e, apenas um, símbolo não terminal e, no lado direito da substituição, podem figurar quaisquer cadeias de símbolos, sejam terminais, não terminais e até mesmo isoladamente a cadeia vazia.

    - De forma mais objetiva, podemos dizer que em uma Linguagem Livre de Contexto, posso fazer um substituição contendo um aninhamento.

    - Um aninhamento é inserir/substituir um símbolo não terminal que esteja entre dois símbolos terminais.

Linguagens Livres de Contexto

- Assim, em uma Linguagem Livre de Contexto podemos fazer estruturas de frases, onde os símbolos seriam palavras.

Um exemplo: consideremos a Linguagem Livre de Contexto com as regras de substituição a seguir (<frase> é a raiz da gramática):

- <frase> → <sujeito><verbo><complemento>
- <sujeito> → O aluno
- <sujeito> → A aluna
- <verbo> → fez a prova
- <verbo> → entregou o exercício
- <complemento> → e teve <adjetivo> nota
- <adjetivo> → uma ótima
- <adjetivo> → uma péssima
- <adjetivo> → ε

Linguagens Livres de Contexto

- <frase> → <sujeito><verbo><complemento>
- <sujeito> → O aluno
- <sujeito> → A aluna
- <verbo> → fez a prova
- <verbo> → entregou o exercício
- <complemento> → e teve <adjetivo> nota.
- <adjetivo> → uma ótima
- <adjetivo> → uma péssima
- <adjetivo> → ε

<frase>

<sujeito><verbo><complemento>

A aluna <verbo><complemento>

A aluna fez a prova <complemento>

A aluna fez a prova e teve <adjetivo> **nota.**

A aluna fez a prova e teve nota.

Linguagens Livres de Contexto

Um outro exemplo: consideremos a Gramática abaixo com V = {+, –, (,), x}, Σ = {S, A, B}, S, sendo a raiz e os conjuntos P de substituições:

- S → A + S
- S → A
- A → B – A
- A → B
- B → (S)
- B → x

- Perceba que o aninhamento na quinta regra irá garantir que cada parêntese aberto será fechado.

Linguagens Livres de Contexto

- $S \rightarrow A + S$
- $S \rightarrow A$
- $A \rightarrow B - A$
- $A \rightarrow B$
- $B \rightarrow (S)$
- $B \rightarrow x$

- S
- $A + S$
- $B + S$
- $(S) + S$
- $(A) + S$
- $(B) + S$
- $(x) + S$
- $(x) + B - A$
- $(x) + x - A$
- $(x) + x - B$
- $(x) + x - x$

Linguagens Livres de Contexto – BNF

- Linguagens Livres de Contexto que contenham aninhamentos não podem ser representadas adequadamente por meio de expressões regulares e, consequentemente, por Autômatos Finitos.

- Uma forma utilizada para representar uma Linguagem Livre de Contexto é o Formalismo de *Backus-Naur* (ou BNF, de "*Backus-Naur Form*").

- Foi originalmente criado por John Backus e Peter Naur no fim dos anos 1950, para descrever resumidamente a linguagem ALGOL.

    - O BNF não possui a força matemática das expressões regulares, tendo como objetivo resumir a gramática.

Linguagens Livres de Contexto – BNF

A notação do BNF é a seguinte:

- Os símbolos não terminais são representados por textos delimitados pelos meta-símbolos “<” e “>”.
- O “→” é substituído por “::=”.
- Todas as alternativas de substituição para um mesmo não terminal são agrupadas, separando-se umas das outras com “|”.
- Os símbolos terminais são denotados sem delimitadores.

Linguagens Livres de Contexto – BNF

Para a gramática do exemplo anterior:

- S → A + S
- S → A

<S> :: = <A> + <S> | <A>

- A → B – A
- A → B

<A> :: = <B> – <A> | <B>

- B → (S)
- B → x

<B> :: = (<S>) | x

Interatividade

Linguagens Livres de Contexto são mais poderosas na representação de ideias do que as Linguagens Regulares. Assinale a alternativa **incorreta** acerca destas Linguagens:

a) Podem possuir regras de substituição nas quais haja aninhamento de símbolos não terminais entre símbolos terminais.

b) Nem todas as Linguagens Livres de Contexto podem ser representadas por meio de expressões regulares.

c) Podem ser representadas de forma resumida pelo Formalismo de *Backus-Naur*.

d) Podem ser sempre representadas na forma de um autômato finito não determinístico.

e) Permitem a construção de sentenças com estruturas de frase, próximas às das Linguagens Naturais.

Resposta

Linguagens Livres de Contexto são mais poderosas na representação de ideias do que as Linguagens Regulares. Assinale a alternativa **incorreta** acerca destas Linguagens:

a) Podem possuir regras de substituição nas quais haja aninhamento de símbolos não terminais entre símbolos terminais.

b) Nem todas as Linguagens Livres de Contexto podem ser representadas por meio de expressões regulares.

c) Podem ser representadas de forma resumida pelo Formalismo de *Backus-Naur*.

d) Podem ser sempre representadas na forma de um autômato finito não determinístico.

e) Permitem a construção de sentenças com estruturas de frase, próximas às das Linguagens Naturais.

Autômatos de Pilha

- Para analisar adequadamente uma Linguagem Livre de Contexto precisamos de um autômato mais poderoso que o Autômato Finito (determinístico ou não).

- Autômatos de Pilha, por serem mecanismos de validação das Linguagens Livres de Contexto (Tipo 2), também podem ser utilizados para a validação de Linguagens Regulares (Tipo 3).

- Para tanto, usaremos um autômato com uma memória auxiliar que se apresenta na forma de uma pilha de dados.

    - Os estados deste autômato serão os possíveis elementos na posição de saída da pilha.

    - Assim, vamos rapidamente rever o conceito de pilha de dados.

Autômatos de Pilha

- São estruturas de dados do tipo LIFO (*Last-In, First-Out*), onde o último elemento a ser inserido, será o primeiro a ser retirado.

- Assim, uma pilha permite acesso a apenas um item dos dados: o último que foi inserido na estrutura.

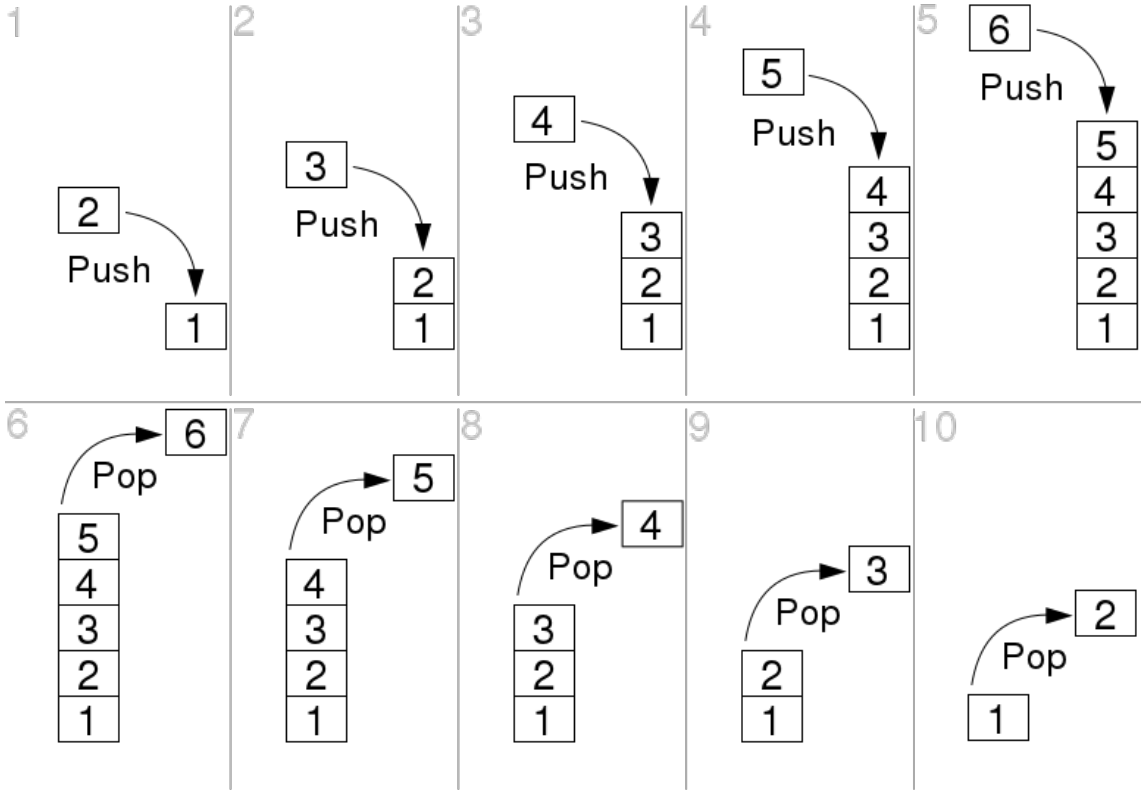
- Para processar o penúltimo item inserido, deve-se remover o último.

    - Muitas Linguagens de Programação, como o Python e o Java, oferecem comandos e métodos próprios para trabalhar com pilhas (e outras estruturas de dados, como filas e listas).

Autômatos de Pilha

- Usualmente se utilizam os termos em inglês *push* (empilhar) para indicar a inserção de um elemento na pilha, e *pop* (desempilhar) para indicar a remoção de um elemento da pilha.

- Um Autômato de Pilha irá construir uma pilha, empilhando ou removendo um elemento de acordo com o símbolo que está sendo analisado no momento da cadeia e do último elemento (na posição de saída da pilha).



Fonte: https://commons.wikimedia.org/wiki/File:Lifo_stack.svg

Autômatos de Pilha

Um autômato de pilha apresenta os seguintes componentes:

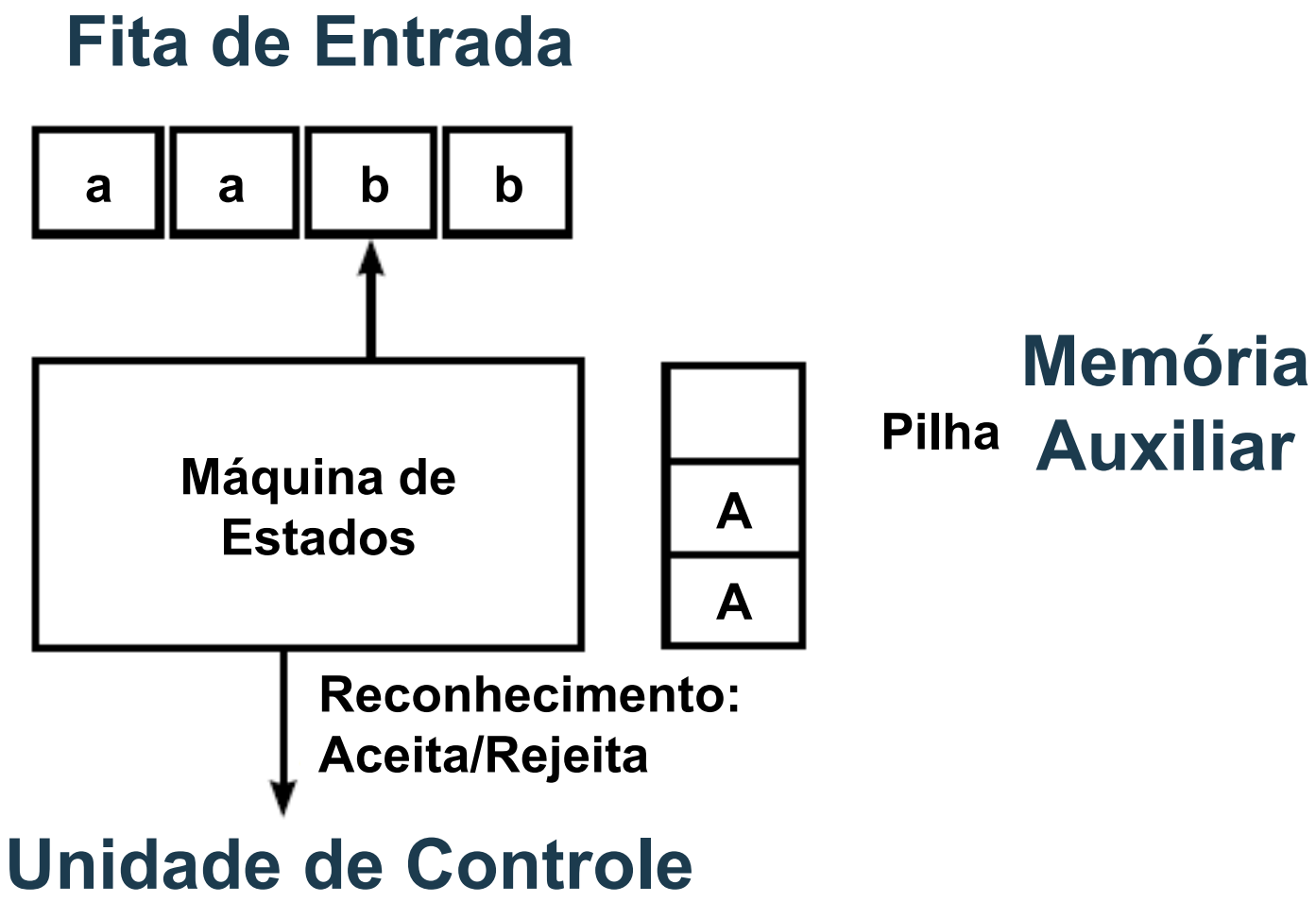
- Fita de entrada: idêntica àquela do autômato finito, sua função é armazenar a cadeia a ser analisada. A fita de entrada é também dividida em células, cujo número é igual ao número de símbolos da cadeia de entrada. Esta fita dispõe de um cursor, que se movimenta apenas da esquerda para a direita.

- Unidade de controle: possui um número finito e predefinido de estados. As transições de estados do autômato de pilha se fazem mediante as especificações previstas por funções de transição, o que leva em conta o símbolo lido e o elemento na posição de saída da pilha.

    - Memória auxiliar organizada em pilha: os símbolos a serem consultados, removidos e armazenados na pilha, em geral, não coincidem com os símbolos do alfabeto de entrada. Os símbolos da pilha apresentam um alfabeto próprio.

Autômatos de Pilha

Um autômato de pilha apresenta os seguintes componentes:



Fonte: Adaptado de: Livro-texto

Autômatos de Pilha

Formalmente, um autômato de pilha pode ser definido como uma sêxtupla M:

$$M = (Q, \Sigma, \Gamma, g, q0, F)$$

- Q é um conjunto finito de estados;
- Σ é um alfabeto (finito e não vazio) de entrada;
- Γ é um alfabeto (finito e não vazio) de pilha;
- g é um conjunto de relações de transição em função de Q, Σ e Γ;
- q0 é o estado inicial;
- $F \subseteq Q$ é o conjunto de estados finais.

Autômatos de Pilha

- Um exemplo: uma autômato de pilha para identificar a correta abertura/fechamento de parênteses e colchetes em uma expressão (pode ser uma expressão matemática ou uma linha de código em um programa de computador).

Utilizaremos as seguintes condições:

- Alfabeto do autômato: 0, x, (
- Estado de aceitação: F = q0 = 0 (pilha começará com o valor 0)

Funções de transição:

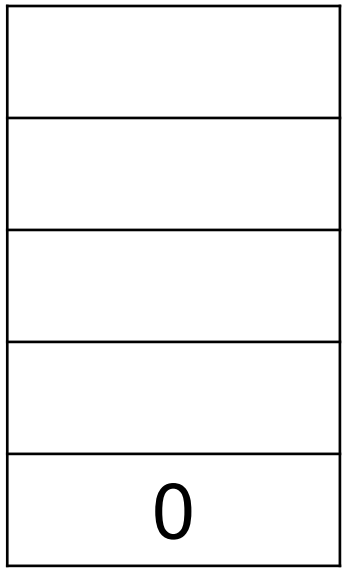
- Ler da fita algo que não seja (ou): não alterar a pilha.
- Ler da fita (e ler da pilha 0: acrescentar (na pilha.
- Ler da fita (e ler da pilha (: acrescentar (na pilha.
- Ler da fita) e ler da pilha (: remover da pilha.
- Ler da fita) e ler na pilha 0: acrescentar x na pilha.

(Para os colchetes, usaremos a mesma estrutura, com a respectiva substituição dos símbolos).

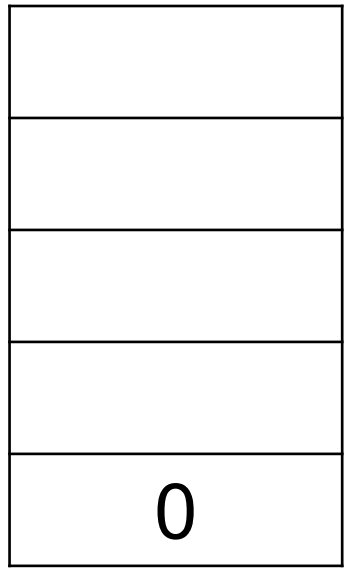
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

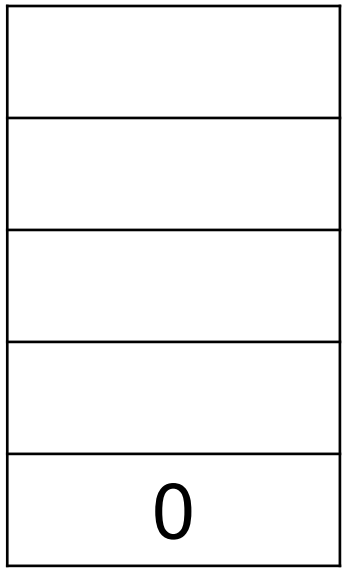


Pilha Colchetes

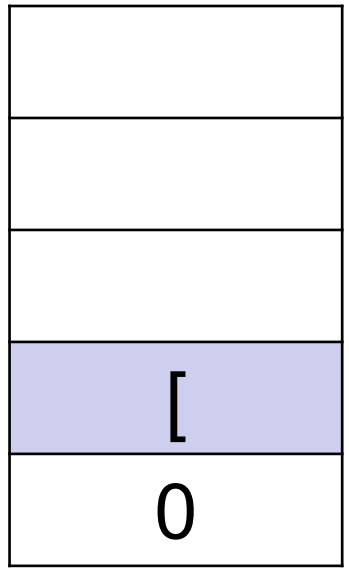
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

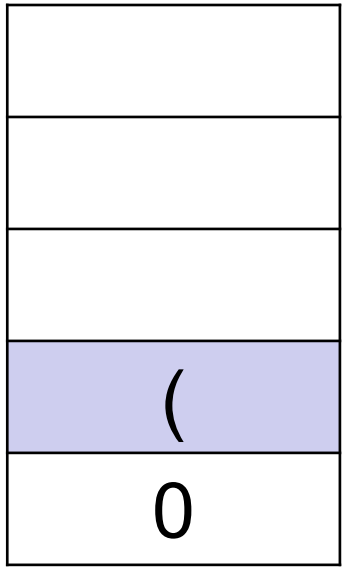


Pilha Colchetes

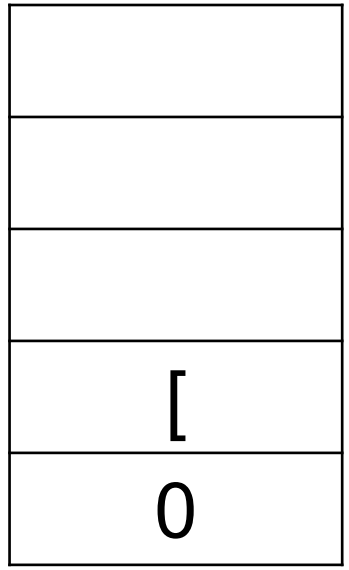
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

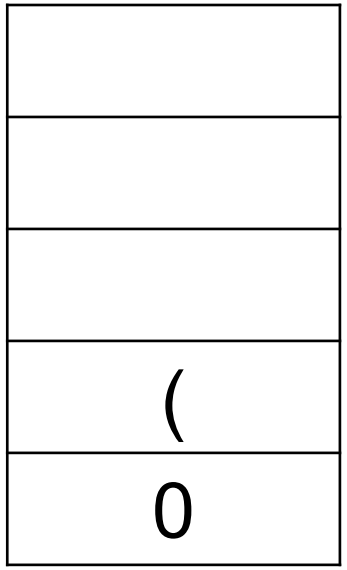


Pilha Colchetes

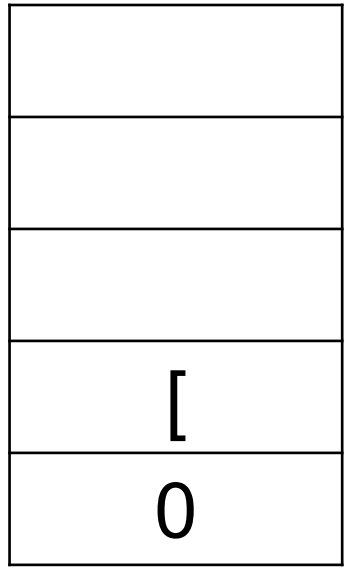
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

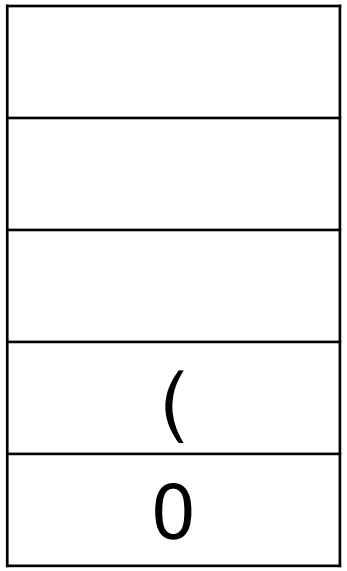


Pilha Colchetes

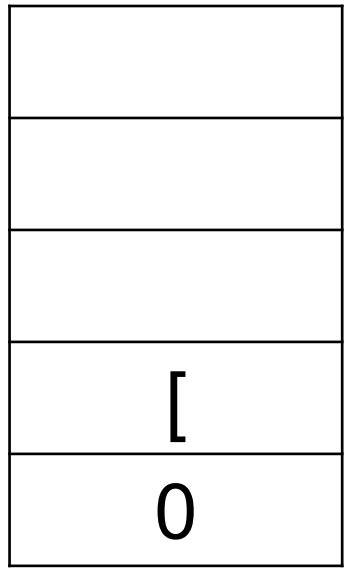
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

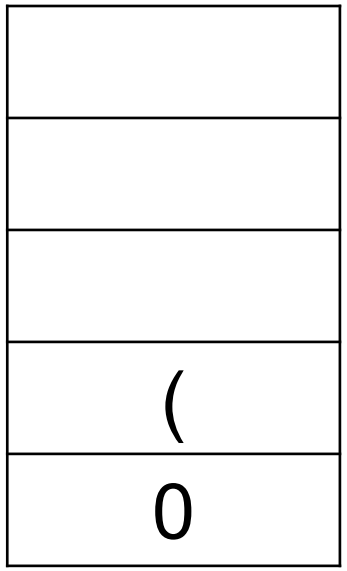


Pilha Colchetes

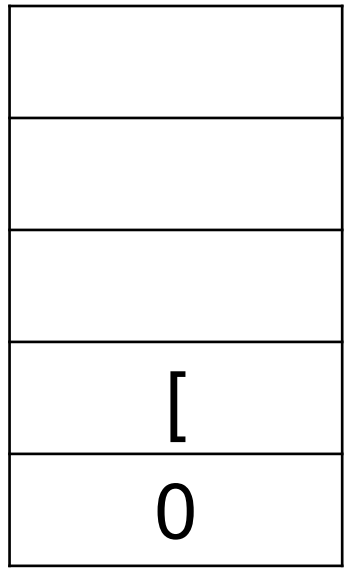
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

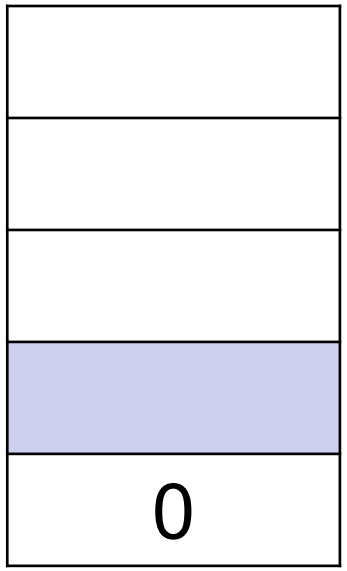


Pilha Colchetes

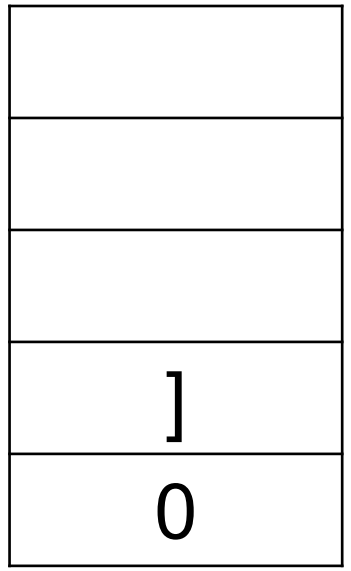
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

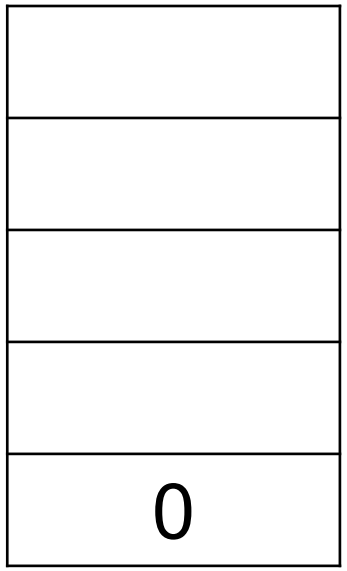


Pilha Colchetes

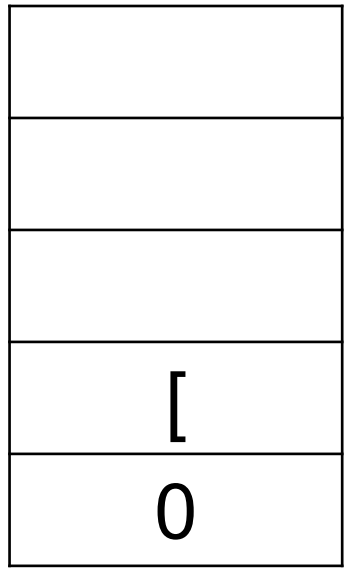
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

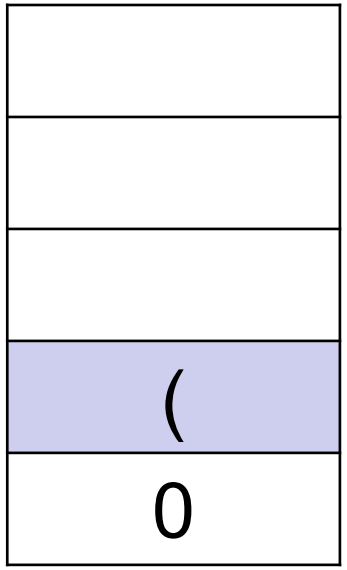


Pilha Colchetes

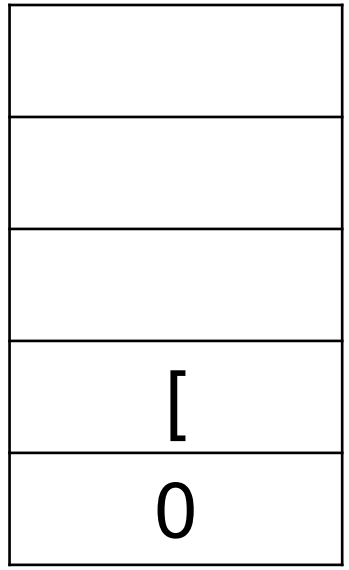
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

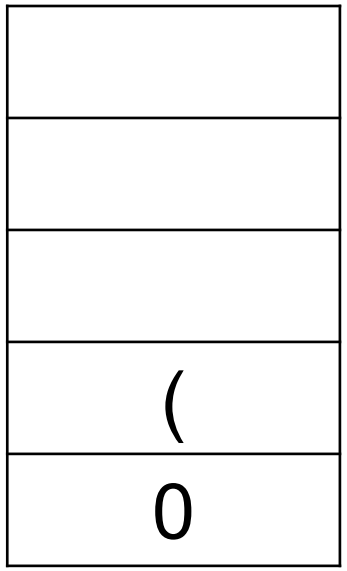


Pilha Colchetes

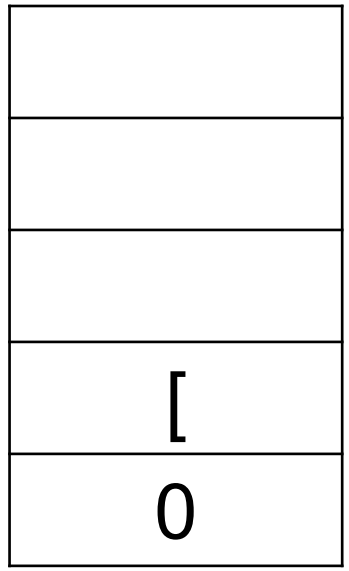
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

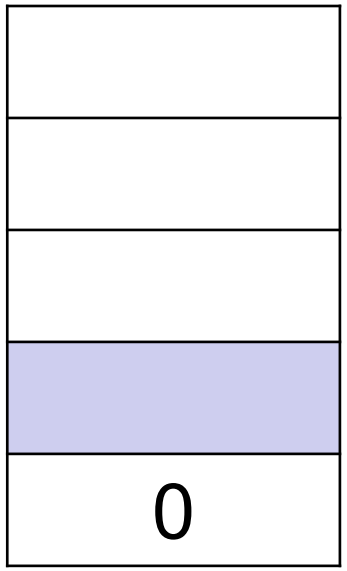


Pilha Colchetes

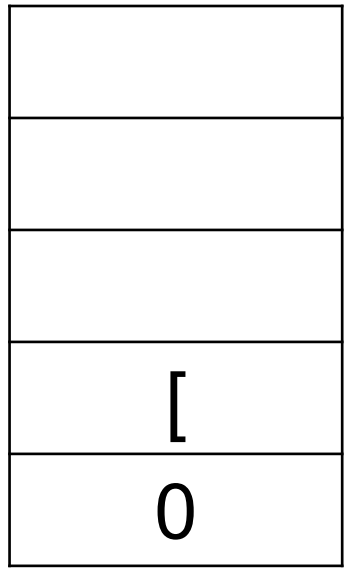
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses

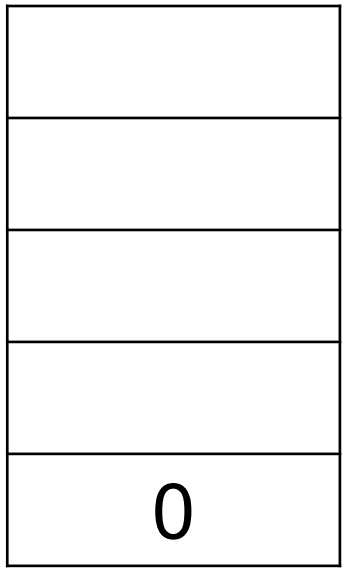


Pilha Colchetes

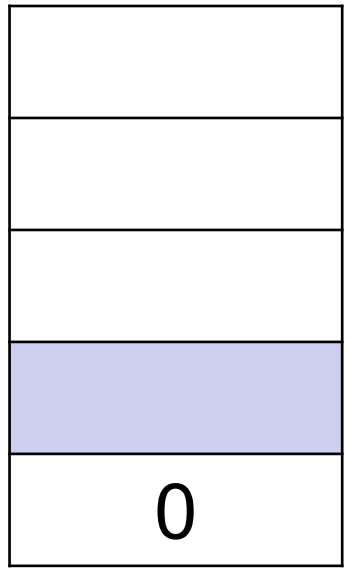
Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

[	(	2	+	3	)	–	(	x	)	]	]



Pilha Parênteses



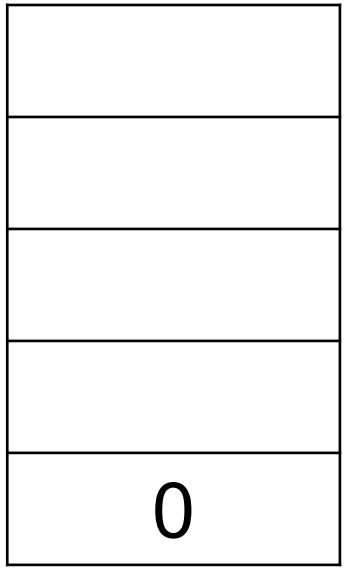
Pilha Colchetes

Autômatos de Pilha

Testando para a cadeia [(2+3)-(x)]]:

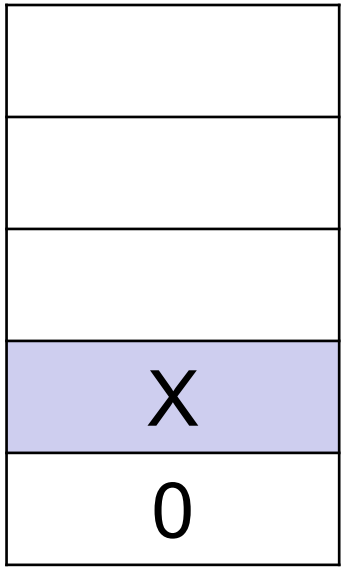
[	(	2	+	3	)	–	(	x	)	]	]

ACEITO!



Pilha Parênteses

REJEITADO!



Pilha Colchetes

Interatividade

Em relação aos Autômatos de Pilha, é correto afirmar que:

a) Não são tão efetivos quanto os Autômatos Finitos Determinísticos.
b) Só podem ser utilizados na aceitação de Gramáticas Livres de Contexto.
c) Utilizam-se de uma estrutura FIFO (*First-In, First-Out*).
d) Não permitem a identificação de sentenças com estruturas de frase.
e) Possuem uma memória auxiliar na forma de uma pilha.

Resposta

Em relação aos Autômatos de Pilha, é correto afirmar que:

a) Não são tão efetivos quanto os Autômatos Finitos Determinísticos.
b) Só podem ser utilizados na aceitação de Gramáticas Livres de Contexto.
c) Utilizam-se de uma estrutura FIFO (*First-In, First-Out*).
d) Não permitem a identificação de sentenças com estruturas de frase.
e) Possuem uma memória auxiliar na forma de uma pilha.

Linguagens Sensíveis ao Contexto

- Tanto as Gramáticas Regulares quanto as Gramáticas Livres de Contexto assumem em suas regras a restrição de que um único símbolo não terminal será substituído por um conjunto de outros símbolos (terminais e/ou não terminais).

- Gramáticas Sensíveis ao Contexto eliminam essa restrição de que o lado esquerdo das regras seja formado por um único símbolo, e de que este seja necessariamente um símbolo não terminal.

    - Estas gramáticas admitem qualquer quantidade de símbolos do lado esquerdo da regra, sejam eles terminais ou não terminais.

Linguagens Sensíveis ao Contexto

- As únicas exigências são que do lado esquerdo exista pelo menos um símbolo não terminal e, também, que o lado direito possua uma quantidade de símbolos não inferior àquela encontrada no lado esquerdo da mesma regra (a cadeia ou sentença não pode diminuir com uma nova transição).

Um exemplo de Gramática Sensível ao Contexto:

- G = ({x, y, y}, {S, B, C}, P, S)
- P = $S \rightarrow xSBC$,
  $S \rightarrow xBC$,
  $CB \rightarrow BC$,
  $xB \rightarrow xy$,
  $yB \rightarrow yy$,
  $yC \rightarrow yz$,
  $zC \rightarrow zz$

Linguagens Sensíveis ao Contexto

- Gramáticas Sensíveis ao Contexto e as linguagens definidas por elas não podem ser analisadas por autômatos, sejam finitos ou de pilha, determinísticos ou não.

- Assim, para analisar estas Linguagens, utilizamos a chamada Máquina de Turing.

- As máquinas de Turing são similares aos autômatos finitos e de pilha e apresentam uma fita de entrada e unidade de controle finito.

- Porém, aqui a fita de entrada não é finita, e sim ilimitada à direita.

    - Por outro lado, o seu início (onde começa o processamento) é marcado pelo símbolo especial: •

Máquina de Turing

Fonte: Adaptado de: Livro-texto

- Sobre a fita de entrada podem ser executadas as operações de leitura e escrita. Ainda, o cursor pode movimentar-se à direita e à esquerda, diferentemente dos autômatos vistos.
- O símbolo gravado e o sentido do movimento são definidos pelo programa da unidade de controle.
- Como a fita é ilimitada à direita, inicialmente, a palavra a ser processada ocupa as células mais à esquerda, e as demais são aqui denotadas por β, ou seja, por células em branco.

Máquina de Turing

- O modelo da máquina de Turing foi proposto por Alan Turing em 1936.

- Neste mesmo ano, Alonzo Church apresentou a Hipótese de Church, que pode ser assim definida: “As máquinas de Turing são versões formais de algoritmos e nenhum procedimento computacional é considerado um algoritmo a não ser que possa ser apresentado na forma de uma máquina de Turing” (NETO, 2003).

Máquina de Turing

- Alguns autores dizem que as máquinas de Turing são tão poderosas quanto as máquinas reais, e são capazes de executar qualquer operação que um programa real executa.

- Máquinas de Turing poderiam de fato ser equivalentes a uma máquina que tenha uma quantidade ilimitada de espaço de armazenamento.

- Porém, memória infinita exigiria, na prática, tempo infinito para ser manipulada, o que não é uma situação real.

Linguagens Irrestritas

- Gramáticas Irrestritas, ou Gramáticas com Estrutura de Frase, são aquelas às quais nenhuma limitação é imposta.
- Estas gramáticas definem as Linguagens Irrestritas.
- Este é o nível 0 na Hierarquia de Chomsky, onde estão as Linguagens Naturais.
- São capazes de gerar linguagens recursivamente enumeráveis.
- Uma Linguagem Irrestrita é um subconjunto recursivamente enumerável no conjunto de todas as cadeias possíveis sob o alfabeto da linguagem.
    - Isso significa que um determinado conjunto é definido por uma função recursiva que produz apenas elementos dele.
    - Linguagens Irrestritas só podem ser analisadas por máquinas de Turing com uma fita infinita em ambas as direções.

Linguagens Recursivamente Enumeráveis

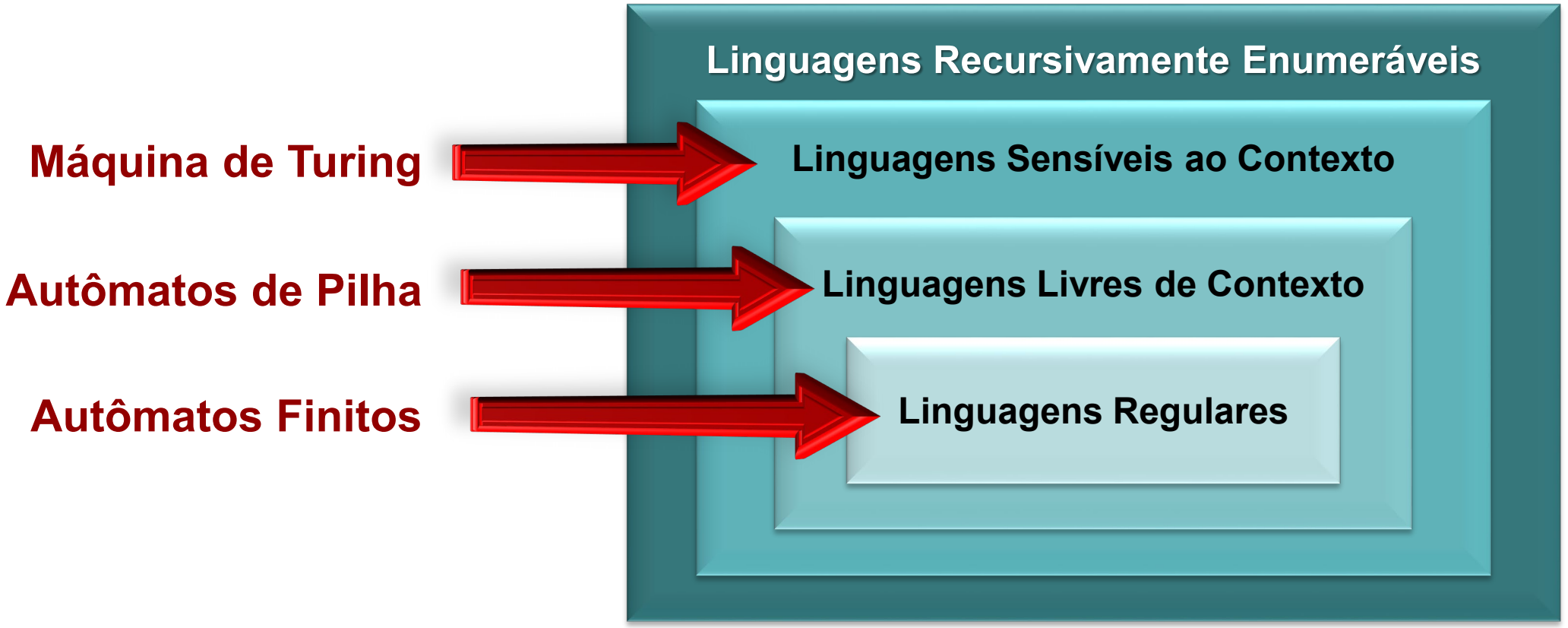
- Um subtipo das Linguagem Irrestrita são as Linguagens Recursivamente Enumeráveis.

- Uma Linguagem Recursivamente Enumerável é uma linguagem formal para a qual existe uma máquina de Turing que irá enumerar todas as cadeias válidas da linguagem.

- Se a linguagem for infinita, o algoritmo de enumeração fornecido pode ser escolhido de forma que evite repetições.

Linguagens Formais e Reconhecedores

- Hierarquia de Chomsky



Fonte: Adaptado de: Livro-texto

Interatividade

Em relação à máquina de Turing, é **incorreto** afirmar que:

a) Permite gravações na fita.
b) O cursor que lê a fita pode se mover em ambas as direções.
c) A fita pode ser infinita em uma ou em ambas as direções.
d) Não permite a identificação de sentenças com estruturas de frase.
e) É um modelo de representar um algoritmo.

Resposta

Em relação à máquina de Turing, é **incorreto** afirmar que:

a) Permite gravações na fita.
b) O cursor que lê a fita pode se mover em ambas as direções.
c) A fita pode ser infinita em uma ou em ambas as direções.
d) Não permite a identificação de sentenças com estruturas de frase.
e) É um modelo de representar um algoritmo.

Referências

- LEWIS, H. R.; PAPADIMITRION, C. H. *Elementos de teoria da computação*. 2. ed. Porto Alegre: Bookman, 2000.
- MENEZES, P. B. *Linguagens formais e autômatos*. 2. ed. Porto Alegre: Sagra Luzzatto, 1998.
- RAMOS, M. V. M.; NETO, J. J.; VEGA, I. S. *Linguagens formais*. Porto Alegre: Bookman, 2009.
- NETO, J. J. *Tópicos sobre os fundamentos da Engenharia da Computação*. PCS-5701, São Paulo, USP, 2003.

ATÉ A PRÓXIMA!