EAD
EDUCAÇÃO A DISTÂNCIA

UNIDADE I

Aspectos Teóricos da Computação

Prof. Me. Hugo Insua

Conjuntos

- Um conjunto é uma coleção não ordenada de objetos.

- A ordem dos elementos em um conjunto não tem impacto em sua definição, e o número de vezes que um mesmo elemento aparece em um conjunto não é relevante. Portanto, podemos afirmar que: A = {maçã, banana, laranja} = {banana, laranja, maçã} = {maçã, maçã, banana, banana, laranja, laranja}.

Conjuntos – Relação de Pertinência

- Chamamos de pertinência a relação de um elemento qualquer pertencer ou não pertencer a um determinado conjunto.

Dado o conjunto D = {÷, ≥ , %, +}, dizemos que:

- $+ \in D$.

- $? \notin D$.

Conjuntos – Classificação

Os conjuntos podem ser:

- Finitos: São conjuntos em que se pode determinar a quantidade de elementos (cardinalidade), ou seja, enumerá-las.

- Infinitos: São conjuntos em que não se pode determinar a cardinalidade.

- Vazio: É o conjunto que não possui elementos, ou seja, a cardinalidade é 0 (zero).

    - Representamos o conjunto vazio por { } ou ∅.

Conjuntos – Classificação

Exemplos:

- Seja o conjunto A = {a, e, i, o, u}. A é um conjunto finito, representado pela enumeração de seus elementos a, e, i, o, u.

- O conjunto dos números Naturais (N) é um exemplo de conjunto infinito. N = {0, 1, 2, 3, 4, ...}.

Conjuntos – Representação por Propriedades

Os conjuntos podem ser descritos com base em suas propriedades, como exemplificado a seguir:

- $E = \{y \mid y \in R \ \text{ e } y > 0\}$ representa o conjunto de números reais positivos.

Conjuntos – Relação de Inclusão

- Relação de inclusão entre conjuntos é a maneira de descrever como um conjunto pode estar contido em outro, ou seja, quando um conjunto é subconjunto de outro.

- É representada por dois símbolos principais: $\subseteq$ e $\subset$, que denotam diferentes tipos de inclusão.

- $\subseteq$ é usado quando estamos indicando que um conjunto é subconjunto do outro e que, inclusive, podem ser iguais.

  - Formalmente, se $A \subseteq B$ (A está contido em B).

Conjuntos – Relação de Inclusão

- "⊂" representa uma inclusão própria, o que significa que um conjunto é um subconjunto estrito do outro. Se $A \subset B$ (A está contido propriamente em B ou A é subconjunto próprio de B), isso implica que todos os elementos de A estão em B, mas A não é igual a B. Em outras palavras, A está contido em B, mas B tem pelo menos um elemento que não está em A.

- Exemplo: Sejam os conjuntos $A = \{0, 1, 3, 5, 15, 25\}$ e $B = \{0, 1, 15, 25\}$, é possível estabelecer que: $B \subseteq A$, $B \subset A$, $A \subseteq A$, $B \subseteq B$.

- O conjunto vazio, por definição, é um conjunto que não contém nenhum elemento. Devido a essa característica, o conjunto vazio é considerado um subconjunto de qualquer conjunto.

Conjuntos – Operações

- União: Sejam A e B dois conjuntos. A união de A e B, denotada como $A \cup B$, consiste em todos os elementos que pertencem a A ou a B, ou seja, $A \cup B = \{ x \mid x \in A \text{ ou } x \in B\}$.

- Exemplo: Considerando A = {maçã, banana, laranja} e B = {banana, uva, kiwi}, a união de A e B é {maçã, banana, laranja, uva, kiwi}.

- Interseção: Sejam A e B dois conjuntos. A interseção de A e B, denotada como $A \cap B$, contém todos os elementos que pertencem tanto a A quanto a B, ou seja,
$A \cap B = \{ x \mid x \in A \text{ e } x \in B\}$.

- Exemplo: Considerando A = {maçã, banana, laranja} e B = {banana, uva, kiwi}, a interseção de A e B é $A \cap B$ = {banana}.

Conjuntos – Operações

- Diferença: Sejam A e B dois conjuntos. A diferença de A por B, denotada como A - B, consiste em todos os elementos que pertencem a A, mas não a B, ou seja, $A - B = \{ x \mid x \in A \text{ e } x \notin B\}$.

- Exemplo: Considerando A = {maçã, banana, laranja} e B = {banana, uva, kiwi}, a diferença A - B = {maçã, laranja} e a diferença B - A = {uva, kiwi}.

- Complementação: A complementação de um conjunto A, denotada como A', é o conjunto que contém todos os elementos de um conjunto universo U que não pertencem a A, ou seja, $A' = \{x \mid x \in U \text{ e } x \notin A\}$.

Conjuntos – Operações

- Exemplo: Suponha que A seja o conjunto de números pares e U seja o conjunto de números inteiros. A complementação A’ seria o conjunto de números inteiros que não são pares, ou seja, os números ímpares.

- A = {..., -6, -4, -2, 0, 2, 4, 6, ...} U = {..., -3, -2, -1, 0, 1, 2, 3, ...} A' = {... , -3, -1, 1, 3, ...}

Sequências e Uplas

- Sequências ou listas são coleções ordenadas de objetos representadas entre parênteses. Dessa forma, a lista (a, b, c) é diferente da lista (c, b, a).

- Podemos classificar as sequências como finitas (uplas) ou infinitas. Uma sequência com n elementos é denominada n-upla. Assim, a sequência (a, b, c) é uma 3-upla.

Relação e Função

- Em matemática, uma relação é um conjunto de pares ordenados que relacionam elementos de dois conjuntos. Formalmente, uma relação R entre dois conjuntos A e B é definida como um subconjunto do produto cartesiano A x B, que é o conjunto de todos os pares ordenados (a, b), onde a é um elemento de A e b é um elemento de B.

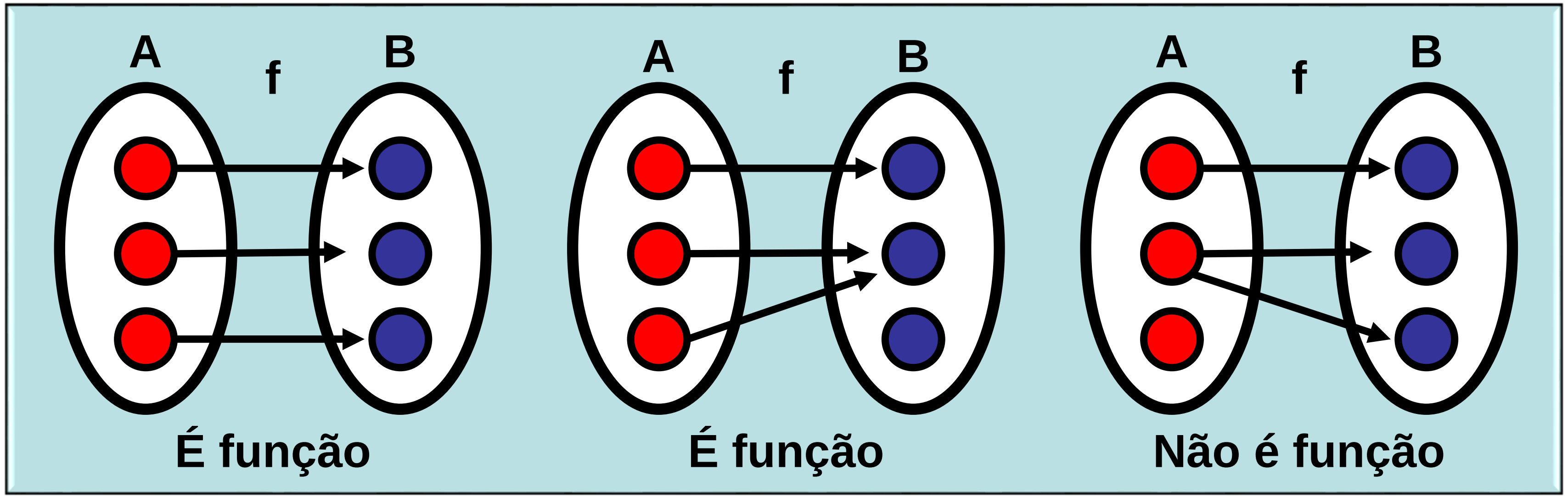
- Por exemplo, considere os conjuntos A = {1, 2, 3} e B = {a, b, c}. A relação R = {(1, a), (2, b), (2, c), (3, a)} é um subconjunto do produto cartesiano A x B, pois todos os seus elementos são pares ordenados cujos primeiros elementos pertencem a A e os segundos elementos pertencem a B. Observe que os elementos (2, b) e (2, c) mostram que o elemento 2 de A está relacionado a ambos os elementos b e c de B.

Relação e Função

- Uma <u>função</u>, em matemática e programação, é uma <u>relação</u> que associa um conjunto de elementos de entrada (domínio) a um conjunto de elementos de saída (contradomínio) de forma que para cada elemento de entrada, há exatamente um elemento de saída correspondente.

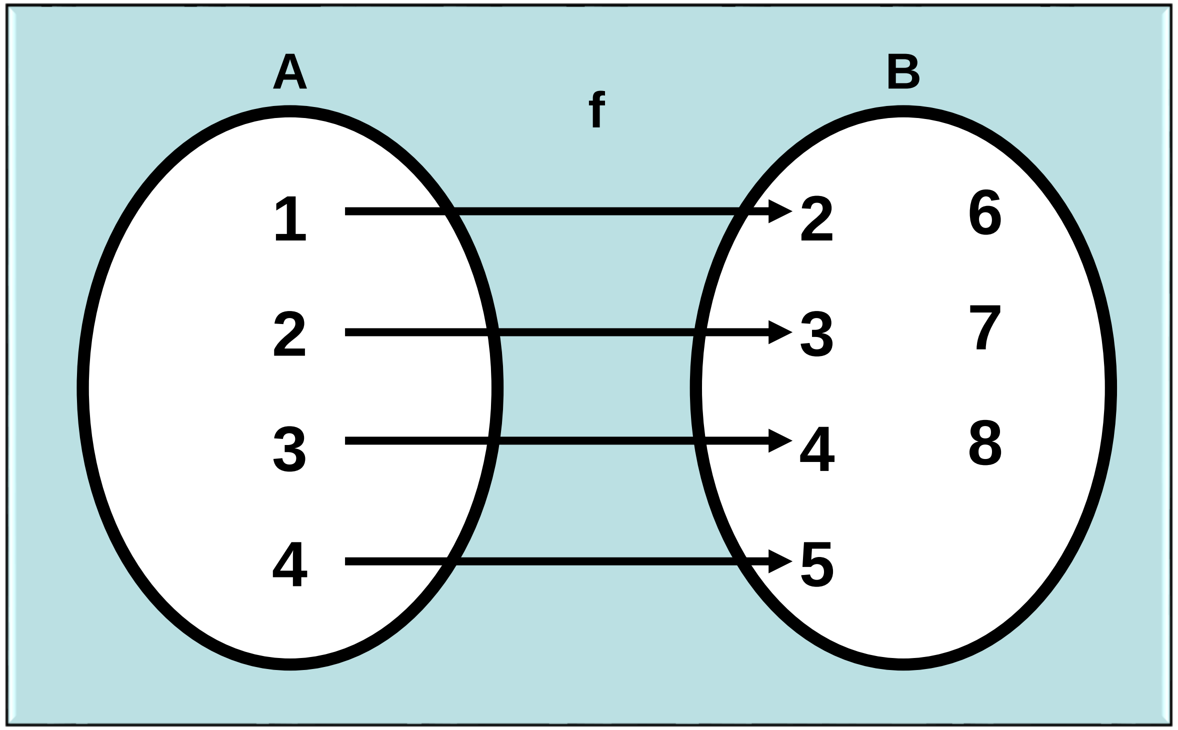
Formalmente, uma função pode ser definida da seguinte maneira: Seja A um conjunto chamado domínio e B um conjunto chamado contradomínio. Uma função f de A para B é uma regra ou correspondência que associa cada elemento x em A a um único elemento y em B. Isso é denotado como:

- f: A → B ou f(x) = y



Fonte: autoria própria.

Relação e Função



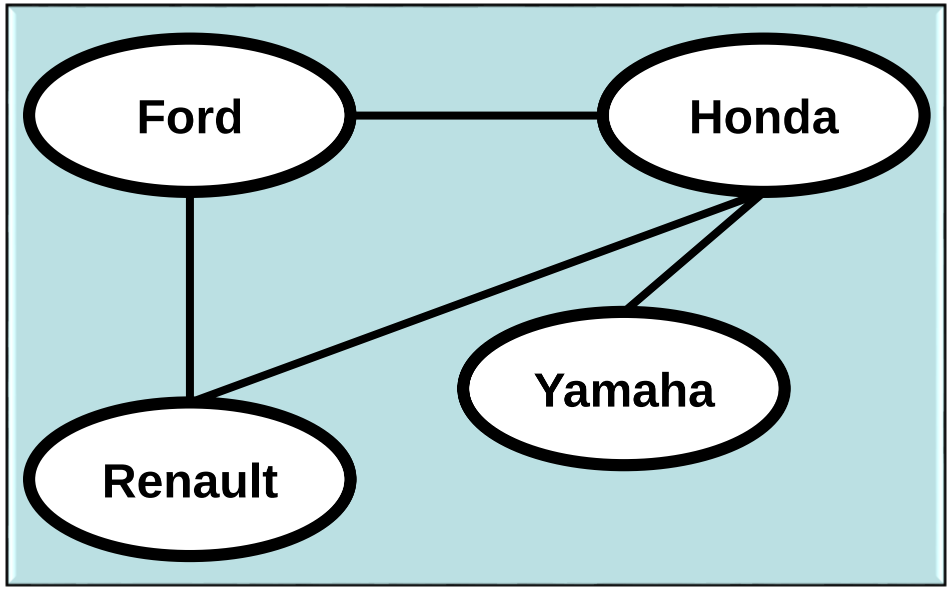
Fonte: autoria própria.

Exemplo:

Dados os conjuntos A = {1, 2, 3, 4} e B = {2, 3, 4, 5, 6, 7, 8}. A função F: A → B que determina a relação entre os elementos de A e B é x → x + 1. Sendo assim, f(x) = x +1, em outras palavras, cada x do conjunto A é transformado em x + 1 no conjunto B. Portanto:

- O conjunto A é o Domínio;
- {2,3,4,5} é o Contradomínio; e
- B é chamado de Imagem.

Grafos



Fonte: autoria própria.

Um grafo G(N,A) é definido pelo par de conjuntos
N e A, onde:

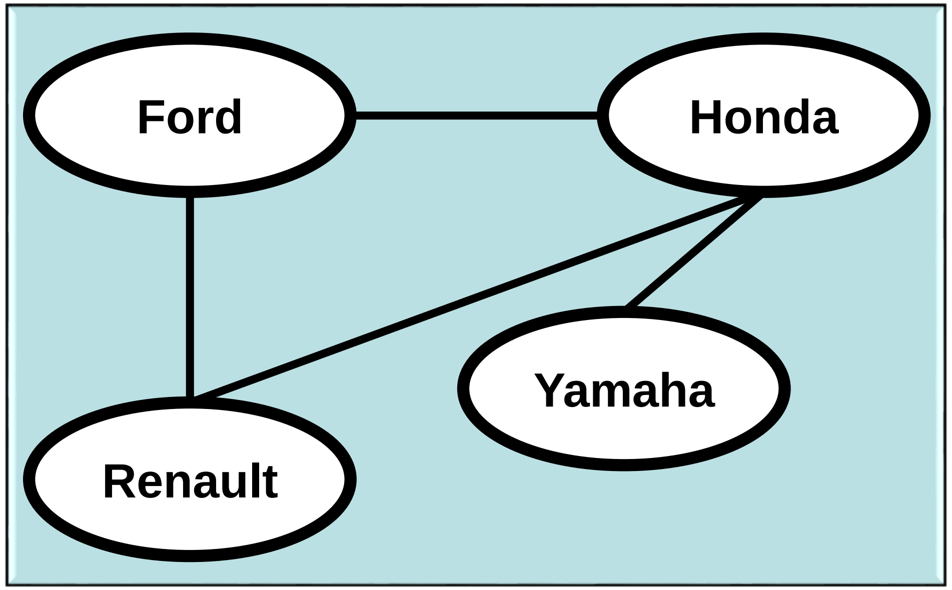
- N – conjunto não vazio: os nodos ou vértices do grafo;
- A – conjunto de pares ordenados a=(n, m), n e m ∈ N: as arestas do grafo.

<u>Exemplo</u>:

Seja o grafo G(N, A) dado por:

- N = {a | a é uma indústria}
- A = {(n, m) | < n é concorrente de m >}

Grafos



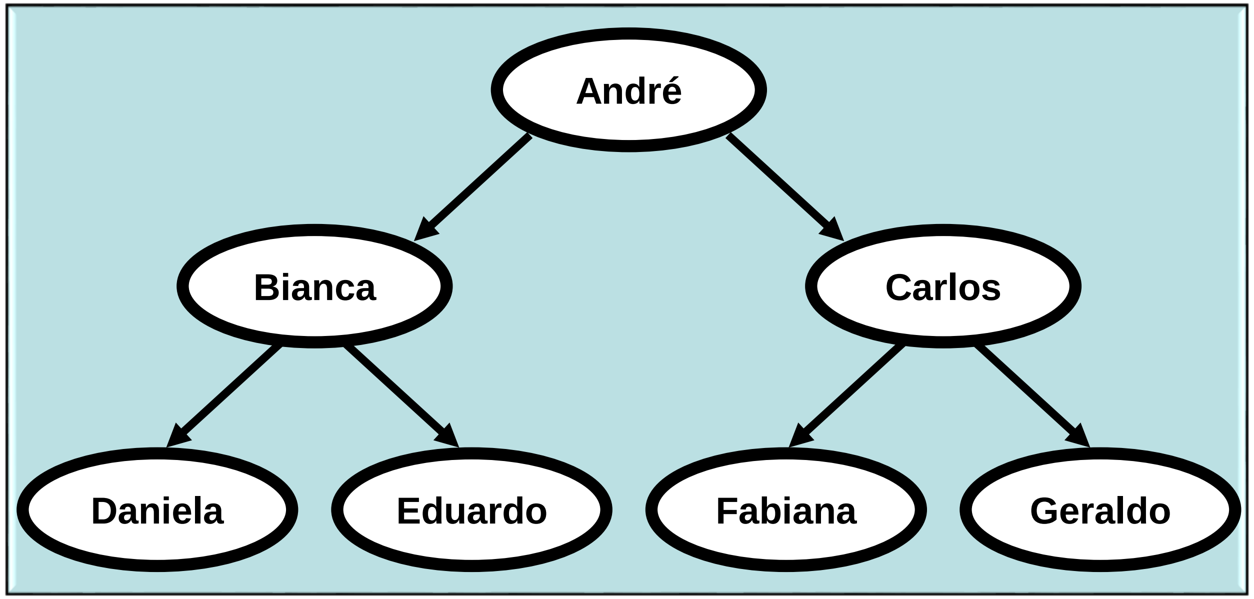
Fonte: autoria própria.

GRAU

- O grau de um vértice em um grafo é o número de arestas que incidem sobre esse vértice, ou seja, é o número de arestas que estão conectadas a ele.

- Por exemplo, os vértices Ford e Renault são grau 2, Honda é grau 3 e Yamaha, grau 1.

Grafos



Fonte: autoria própria.

Dígrafo (Grafo Orientado)

- É o grafo em que as arestas têm uma direção específica, indicando a relação entre dois vértices.

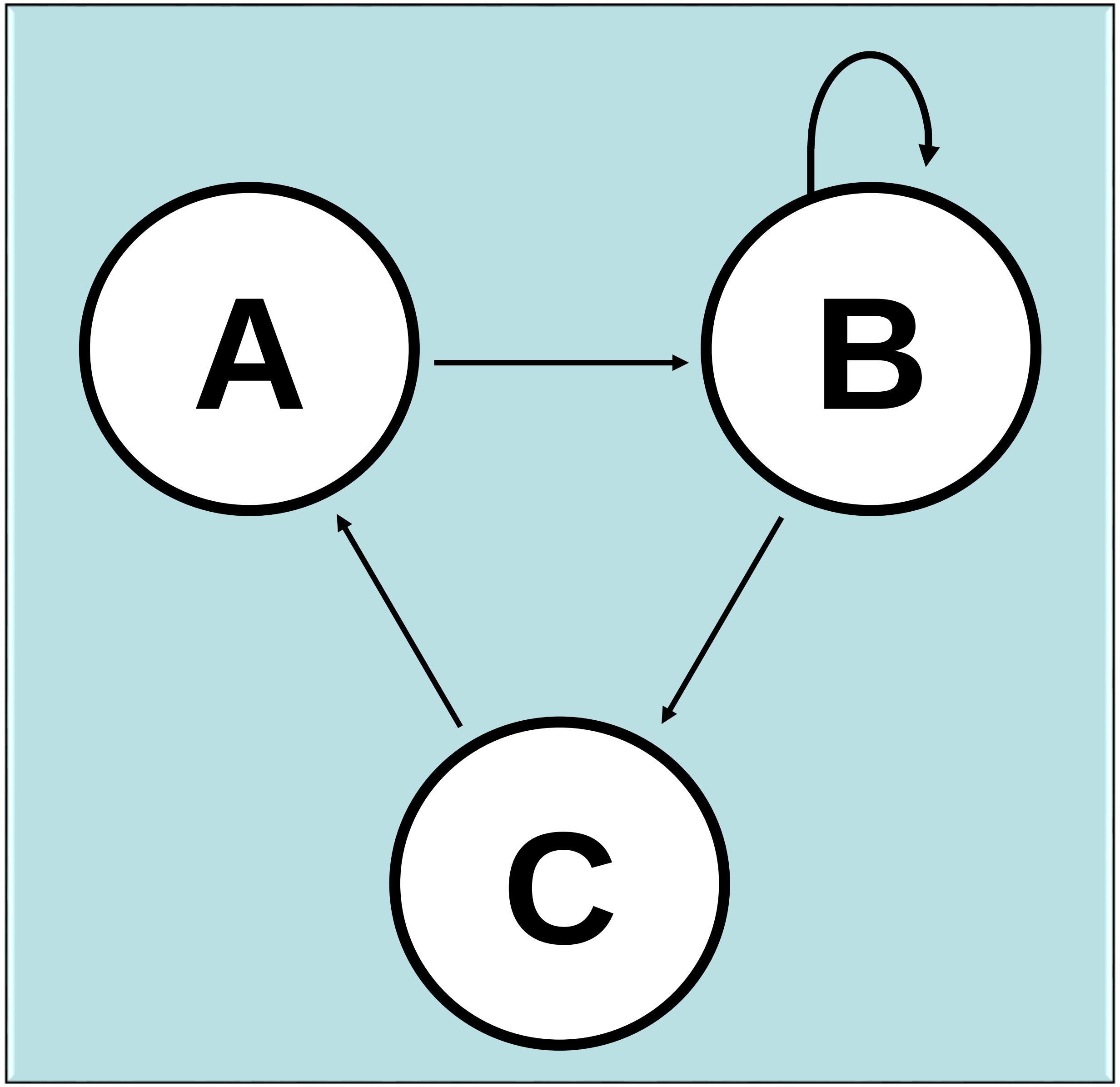
Considere, agora, o grafo G(N, A) definido por:

- V = {r | r é funcionário da empresa E}
- A = {(n, m) | < n é chefe de m >}

Grafos

Laço

- Representa um laço a aresta que conecta um vértice a ele mesmo.



Fonte: autoria própria.

Grafos

Caminho

Caminho é uma rota que percorre, no grafo, as arestas de mesma orientação, passando por diferentes vértices, sem repeti-los. Alguns exemplos comuns incluem:

1. Caminho simples: Um caminho em que nenhum vértice é visitado mais de uma vez. Ou seja, não há repetição de vértices no caminho.
2. Caminho fechado ou ciclo: Um caminho em que o primeiro e o último vértices são os mesmos, formando um ciclo.
3. Caminho mínimo: Um caminho entre dois vértices com o menor comprimento possível, geralmente medido em termos do número de arestas percorridas.

Grafos

4. Caminho Euleriano: Um caminho que passa por todas as arestas de um grafo exatamente uma vez.

5. Caminho Hamiltoniano: Um caminho que passa por todos os vértices de um grafo exatamente uma vez.

Interatividade

Alberto é representante comercial. Ele recebe mensalmente um salário composto de duas partes: uma fixa, no valor de R$ 1400,00, e uma variável, que corresponde a uma comissão de 6% sobre o total de vendas que ele faz durante o mês. Considere “S” o salário mensal e “x” o total das vendas do mês. A função matemática que calcula S em função de x é:

a) $S = 1400x + 6$
b) $S = 6x + 1400$
c) $S = 1400 + 0{,}06x$
d) $S = 1400x + 0{,}06$
e) $S = 1400x – 0{,}6$

Resposta

Alberto é representante comercial. Ele recebe mensalmente um salário composto de duas partes: uma fixa, no valor de R$ 1400,00, e uma variável, que corresponde a uma comissão de 6% sobre o total de vendas que ele faz durante o mês. Considere “S” o salário mensal e “x” o total das vendas do mês. A função matemática que calcula S em função de x é:

a) $S = 1400x + 6$
b) $S = 6x + 1400$
c) $S = 1400 + 0{,}06x$
d) $S = 1400x + 0{,}06$
e) $S = 1400x - 0{,}6$

Linguagens Regulares

- As linguagens regulares são fundamentais no estudo da teoria da computação. Elas têm aplicações em várias áreas, desde a programação até a análise de linguagem natural.

- Uma linguagem regular é uma coleção de palavras formadas por um alfabeto específico e seguindo regras bem definidas.

Linguagens Regulares

- Os autômatos finitos são uma das principais formas de representação das linguagens regulares. Eles podem ser determinísticos ou não determinísticos.

- As expressões regulares são uma forma concisa de representar as linguagens regulares, permitindo busca e manipulação eficientes de texto.

Linguagens Regulares – Autômatos Finitos Determinísticos

Conceito:

- Um AFD é um modelo matemático que representa uma máquina de estados finitos, na qual cada estado possui uma transição única para um próximo estado.

Funcionamento:

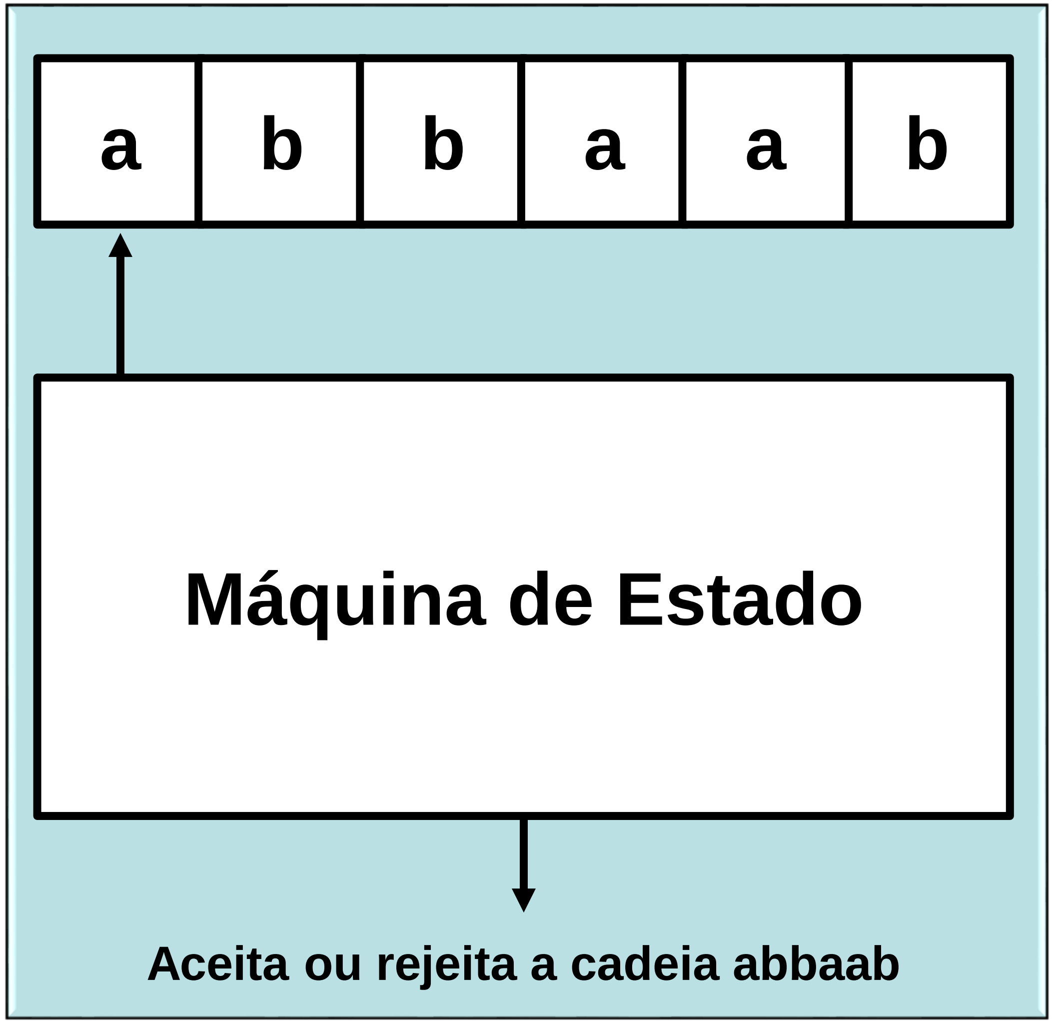
- Os AFDs seguem um processo determinístico, ou seja, o próximo estado é sempre determinado pelo estado atual e pelo símbolo de entrada.

Aplicações:

- Os AFDs são amplamente aplicados na construção de compiladores, processamento de linguagens formais e análise léxica.

Linguagens Regulares – Autômatos Finitos Determinísticos

Os autômatos finitos consistem em dois componentes principais:



Fonte: autoria própria.

Linguagens Regulares – Autômatos Finitos Determinísticos

Formalmente, um AFD é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$ onde:

- Conjunto de estados (Q): Conjunto finito de estados nos quais o AFD pode estar.
- Alfabeto (Σ): Conjunto finito de símbolos que são aceitos como entrada pelo autômato.
- Função de transição (δ): Função que mapeia um estado atual e um símbolo de entrada para o próximo estado. Matematicamente, $\delta = Q \times \Sigma \longrightarrow Q$.
- Estado inicial (q_0): É o estado em que o AFD se encontra antes de processar qualquer entrada. Matematicamente $q_0 \in Q$.
- Conjunto de estados finais (F): É um conjunto de estados que indicam que uma determinada sequência de entrada foi reconhecida pelo autômato. Matematicamente $F \subseteq Q$.

Linguagens Regulares – Autômatos Finitos Determinísticos

Exemplo: Seja M um autômato finito determinístico $(Q, \Sigma, \delta, q_0, F)$, onde:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{x, y, z\}$
- q_0 é o estado inicial
- $F = \{q_2\}$
- $\delta = \{((q_0, x), q_1), ((q_1, y), q_2), ((q_2, z), q_2)\}$
- Verificar se as palavras "xyzz", "xyyz" e "x" pertencem à linguagem L reconhecida pelo autômato finito determinístico M.

Linguagens Regulares – Autômatos Finitos Determinísticos

- $\delta = \{((q_0, x), q_1), ((q_1, y), q_2), ((q_2, z), q_2)\}$

1º –

x	y	z	z
⇑			
q_0			

2º –

x	y	z	z
	⇑		
	q_1		

3º –

x	y	z	z
		⇑	
		q_2	

4º –

x	y	z	z
			⇑
			q_2

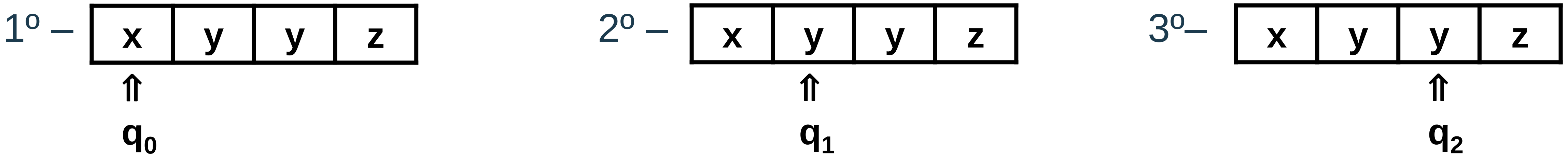
5º–

x	y	z	z	
				⇑
				q_2

- Como o cursor está à direita do último símbolo da fita e no estado final q_2, significa que o AFD alcançou sua configuração final e que o autômato finito reconhece a cadeia de entrada, ou seja, a cadeia de entrada pertence à linguagem reconhecida pelo autômato.

Linguagens Regulares – Autômatos Finitos Determinísticos

- $\delta = \{((q_0, x), q_1), ((q_1, y), q_2), ((q_2, z), q_2)\}$



A leitura não consegue prosseguir ao atingir o segundo “y” e o AFD atinge a configuração final com o cursor não estando à direita do último símbolo da fita. Logo, a palavra xyyz não pertence à linguagem reconhecida pelo AFD M. Dizemos então que M rejeita a cadeia xyyz.

Linguagens Regulares – Autômatos Finitos Determinísticos

- $\delta = \{((q_0, x), q_1), ((q_1, y), q_2), ((q_2, z), q_2)\}$

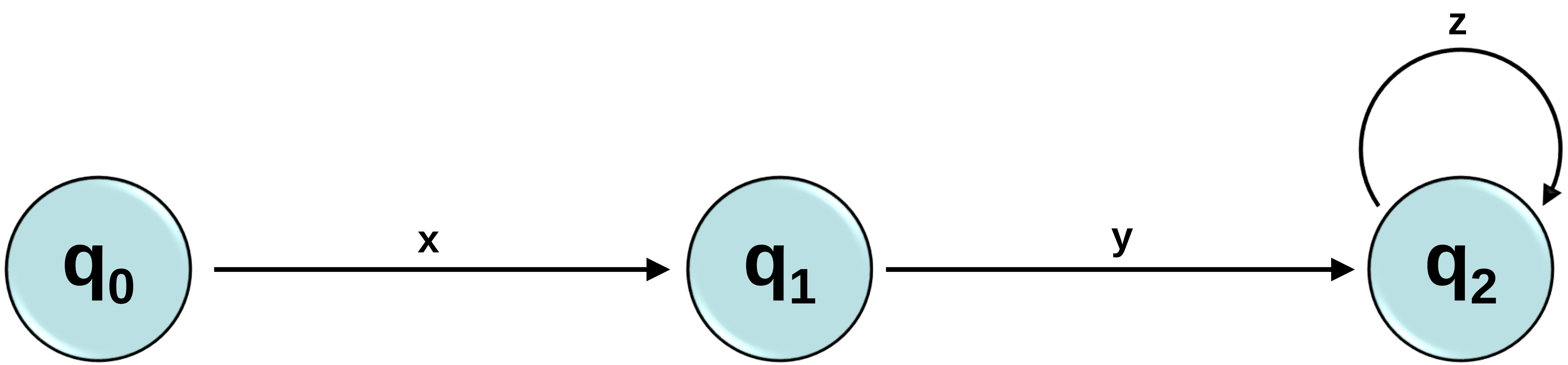
1º – $\boxed{\mathbf{x}}$
$\Uparrow$
$\mathbf{q_0}$

2º – $\boxed{\mathbf{x}}$
$\Uparrow$
$\mathbf{q_1}$

- O AFD M atinge a configuração final, como o cursor à direita do último símbolo da fita. Entretanto, M rejeita a cadeia de entrada, pois ele não alcançou o estado final q_2. Logo, a palavra “x” não pertence à linguagem reconhecida pelo autômato M.

Linguagens Regulares – Autômatos Finitos Determinísticos

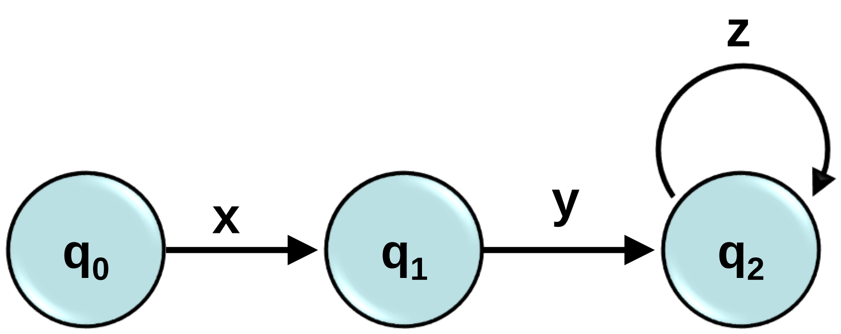
Podemos, também, representar o AFD M através de um grafo orientado. Assim:



Fonte: autoria própria.

Dessa forma, qual é a linguagem reconhecida pelo AFD M?

Linguagens Regulares – Autômatos Finitos Determinísticos



Fonte: autoria própria.

O processamento das palavras pertencentes à linguagem L é:

1) Estando, inicialmente, o AFD M no estado inicial q_0 e lendo o símbolo ‘x’, ele alcança o estado q_1;
2) Estando o AFD no estado q_1 e lendo o símbolo ‘y’, ele alcança o estado q_2;
3) Estando o AFD no estado q_2 e lendo o símbolo ‘z’, ele permanece em q_2.

- Note que no estado q_2, temos um laço, ou seja, o 3º passo pode se repetir zero ou mais vezes e, portanto, M reconhece qualquer palavra iniciada com os símbolos ‘xy’ e finalizada com zero ou mais símbolo ‘z’.
- Temos que a linguagem L reconhecida pelo autômato M é $L = \{\omega \mid \omega = xyz^n \text{ e } n \geq 0\}$.

Linguagens Regulares – Operações Regulares

São operações que se aplicam às linguagens regulares:

- União (ou soma): dadas duas linguagens regulares L1 e L2, a união de L1 e L2, denotada por $L1 \cup L2$ ou $L1 + L2$, é a linguagem que contém todas as palavras que pertencem a L1 ou a L2 (ou ambas).

Exemplo:

- Seja $L1 = \{x, y, z\}$ e $L2 = \{\varepsilon, a, b\}$ então $L1 \cup L2 = \{\varepsilon, x, y, z, a, b\}$.

Linguagens Regulares – Operações Regulares

- Concatenação: dadas duas linguagens regulares L1 e L2, a concatenação de L1 e L2, denotada por L1 · L2 ou L1L2, é a linguagem formada pela concatenação de cada palavra de L1 com cada palavra de L2.

Exemplo:

- Se a linguagem L1 = {x, y, z} e a linguagem L2 = {**ε, a**} então,
  L1L2 = {x**ε**, y**ε**, z**ε**, x**a**, y**a**, z**a**} = {x, y, z, x**a**, y**a**, z**a**}.

Linguagens Regulares – Operações Regulares

- Fechamento de Kleene (ou estrela): O fechamento de Kleene aplicado a uma linguagem L resulta em uma nova linguagem L*, que consiste em todas as possíveis concatenações de zero ou mais strings pertencentes à linguagem L. Em outras palavras, L* inclui todas as strings que podem ser formadas pela concatenação de qualquer número (incluindo zero) de strings em L.

- Por exemplo, se tivermos a linguagem L = {a, b}, então L* incluirá todas as combinações possíveis de sequências formadas por 'a' e 'b', incluindo strings vazias. Assim, L* seria {ε, a, b, aa, ab, ba, bb, aaa, ...}.

Linguagens Regulares – Operações Regulares

- Fechamento transitivo: É o conjunto formado por todas as cadeias do alfabeto Σ com exceção da cadeia vazia. Representa-se o fechamento transitivo por L+.

Exemplo: Seja o alfabeto unário Σ = { 0 }, tem-se que:

- L+ = {0, 00, 000, 0000, 00000, ...}

Linguagens Regulares – Autômatos Finitos Não Determinísticos

Diferenças:

- Os AFNs permitem que um estado possua múltiplas transições para diferentes estados, proporcionando maior flexibilidade.

Potencial Computacional:

- Os AFNs possuem um potencial computacional maior que os AFDs, pois podem simular estruturas paralelas durante o processamento.

Conversão:

- AFNs podem ser convertidos em AFDs equivalentes, permitindo a implementação de algoritmos mais eficientes em alguns casos.

Linguagens Regulares – Expressões Regulares

- As expressões regulares (ER) constituem-se em uma alternativa para representar as linguagens regulares. Assim, usando as operações regulares (união concatenação e estrela), podemos fazer uma representação algébrica das linguagens regulares. Portanto, o valor de uma ER é uma linguagem regular.

Linguagens Regulares – Expressões Regulares

- As operações contidas nas ER, obedecem a uma ordem de precedência, respectivamente: estrela, concatenação e, por fim, união; a utilização de parênteses indica a operação prioritária, mudando a ordem estabelecida. Vejamos agora um exemplo.

Linguagens Regulares – Expressões Regulares

Determine a linguagem descrita pela expressão regular abaixo:

- $R = a(b \cup c)a^*$

- Esta expressão regular descreve todas as cadeias formadas pela concatenação do símbolo “a” com a união entre o símbolo “b” e “c” concatenada com “a”*.

Isso quer dizer que todas as cadeias inicializarão com o símbolo “a” e finalizarão com zero ou mais símbolos “a”. Entre o início e o fim da cadeia terá que ter um “b” ou um “c”. Portanto:

- $L(R) = \{ab, aba, abaa, abaaa, ..., ac, aca, acaa, acaaa, ...\}$

Interatividade

Qual das seguintes afirmações sobre autômatos finitos (AFs) está correta?

a) Autômatos finitos podem descrever ou reconhecer linguagens livres de contexto.
b) Autômatos finitos possuem uma pilha para armazenar dados temporariamente.
c) Autômatos finitos podem descrever ou reconhecer linguagens regulares.
d) Autômatos finitos têm memória ilimitada.
e) Autômatos finitos podem reconhecer qualquer linguagem, independentemente de sua complexidade.

Resposta

Qual das seguintes afirmações sobre autômatos finitos (AFs) está correta?

a) Autômatos finitos podem descrever ou reconhecer linguagens livres de contexto.
b) Autômatos finitos possuem uma pilha para armazenar dados temporariamente.
c) Autômatos finitos podem descrever ou reconhecer linguagens regulares.
d) Autômatos finitos têm memória ilimitada.
e) Autômatos finitos podem reconhecer qualquer linguagem, independentemente de sua complexidade.

Linguagens Livres de Contexto

- Nesta apresentação, vamos explorar as linguagens livres de contexto, suas gramáticas e os autômatos de pilha.

Linguagens Livres de Contexto

O que são as linguagens livres de contexto?

- As linguagens livres de contexto são um conjunto de regras que permitem a construção de frases em uma linguagem.

- Elas são geradas pelas gramáticas livre de contexto.

Linguagens Livres de Contexto

Uma gramática livre de contexto (GLC) é formalmente definida por uma quádrupla
$G = (V, \Sigma, R, S)$, onde:

- V é um conjunto finito de símbolos não terminais.
- Σ é um conjunto finito de símbolos terminais, onde $V \cap \Sigma = \emptyset$ (não há símbolos comuns).
- R é um conjunto finito de regras de produção, em que cada regra é da forma $A \rightarrow \alpha$, onde A é um símbolo não terminal e α é uma cadeia de símbolos terminais ou não terminais onde $\alpha \in (\Sigma \cup V)^*$.
- S é o símbolo inicial da gramática, onde S pertence a V.

Linguagens Livres de Contexto

- Dessa forma, as palavras de uma LLC são geradas a partir de derivações.

- Formalmente, uma derivação em uma gramática livre de contexto é uma sequência finita de substituições de símbolos, começando com o símbolo inicial S e aplicando as regras de produção para substituir os não terminais até que se obtenha uma cadeia de símbolos terminais.

- Portanto, do símbolo S derivam todas as cadeias de uma gramática livre de contexto.

Linguagens Livres de Contexto

Exemplo: A gramática geradora da linguagem $L = \{\omega \mid \omega = a^n b^n, n \geq 0\}$ é $G = (V, \Sigma, R, S)$, onde:

- $V = \{S\}$ (conjunto de símbolos não terminais);
- $\Sigma = \{a, b\}$ (alfabeto composto de símbolos terminais);
- $R = \{S \rightarrow aSb \mid \varepsilon\}$ (conjunto de regras de produção);
- S é o símbolo não terminal inicial da derivação.

Determine se a linguagem L é livre de contexto:

Linguagens Livres de Contexto

- $R = \{S \rightarrow aSb \mid \varepsilon\}$ (conjunto de regras de produção)

- $S \Rightarrow \varepsilon$
- $S \Rightarrow aSb \Rightarrow a\varepsilon b = ab$
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\varepsilon bb = aabb$
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaa\varepsilon bbb = aaabbb$
- Portanto, $L = \{\varepsilon, ab, aabb, aaabbb, aaaabbbb, \ldots\}$

Linguagens Livres de Contexto

Observação

- Uma gramática livre de contexto é ambígua quando existem duas ou mais sequências de substituições que podem gerar a mesma cadeia a partir do símbolo inicial da gramática.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- Autômatos de pilha são máquinas abstratas capazes de reconhecer linguagens livres de contexto ao utilizarem uma pilha para armazenar informações sobre a estrutura da cadeia em análise.

- O autômato de pilha pode ser utilizado para reconhecer a linguagem de palíndromos, que não pode ser reconhecida por um autômato finito.

- Permitem a criação de compiladores mais eficientes e precisos para linguagens de programação complexas.

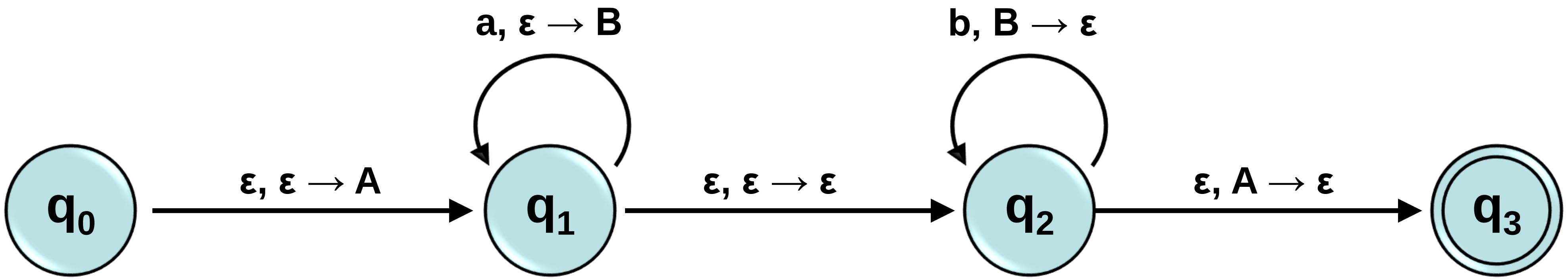
Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

A definição formal de um autômato de pilha é dada por uma 6-tupla $M = (Q, \Sigma, \Gamma, \delta, q0, F)$, onde:

- Q é um conjunto finito de estados.
- Σ é o alfabeto de entrada, ou seja, o conjunto de símbolos de entrada que a máquina lê.
- Γ(gama) é o alfabeto da pilha, ou seja, o conjunto de símbolos que podem ser colocados na pilha.
- δ é a função de transição, dado por Q "×" ("Σ∪" {"ε" })"×"("Γ∪" {"ε" })"→P" ("Q×" ("Γ∪" {"ε" }))
- $q0 \in Q$ é o estado inicial.
- $F \subseteq Q$ é o conjunto de estados finais.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- Exemplo: Determine se o APND, representado pelo grafo abaixo, reconhece a cadeia de entrada aaabbb:

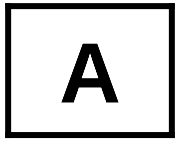


Fonte: autoria própria.

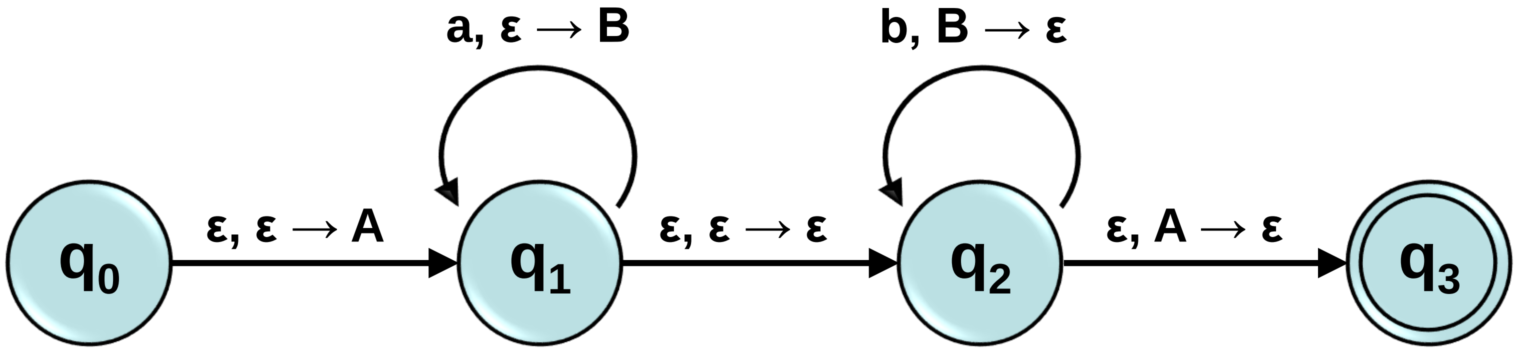
Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Para a leitura de qualquer cadeia, o APND inicializa no estado inicial q_0. Neste estado, a função de transição “ε, ε → A” determina que: lendo nenhum símbolo da cadeia de entrada, não se desempilha nenhum símbolo da pilha, empilha-se o símbolo “A”, avança-se para o q_1.



Pilha

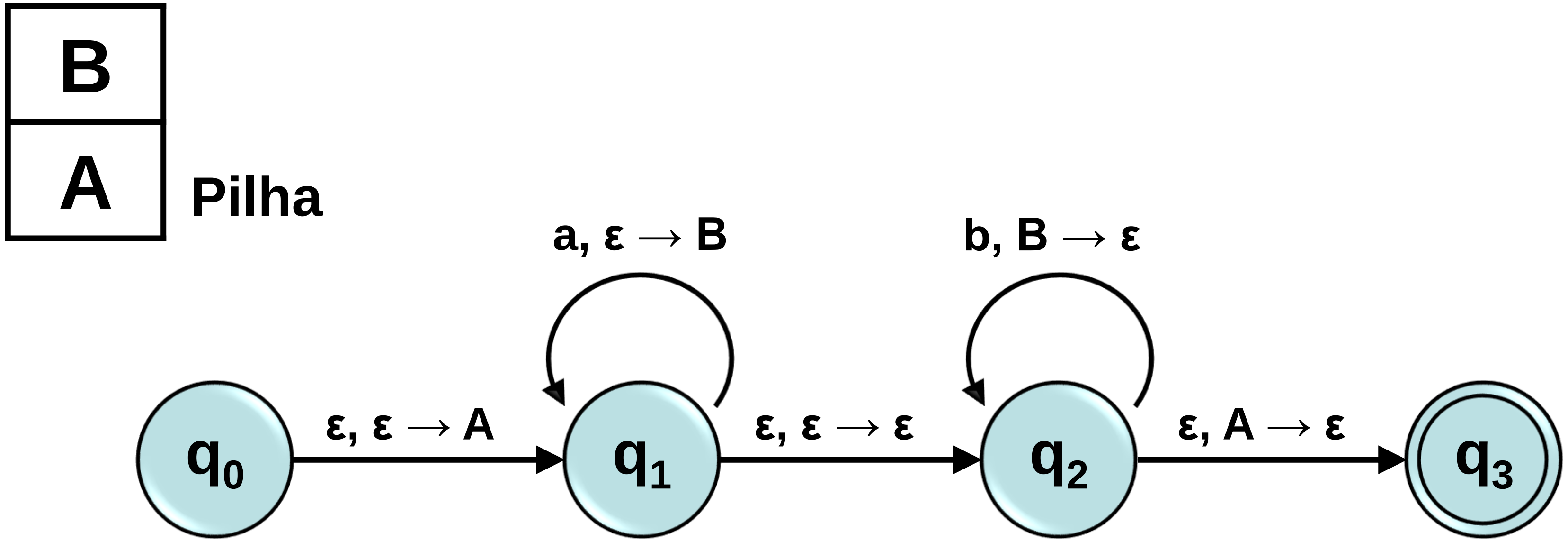


Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Como o autômato de pilha é não determinístico, note que em q_1 há duas transições possíveis. Todavia, como queremos ler a cadeia aaabbb, optaremos pela transição "a, ε → B". Desse modo, estando o APND no estado q_1, lendo o primeiro símbolo "a" da cadeia de entrada, não se desempilha nenhum símbolo da pilha, empilha-se o símbolo "B" e permanece-se em q_1.

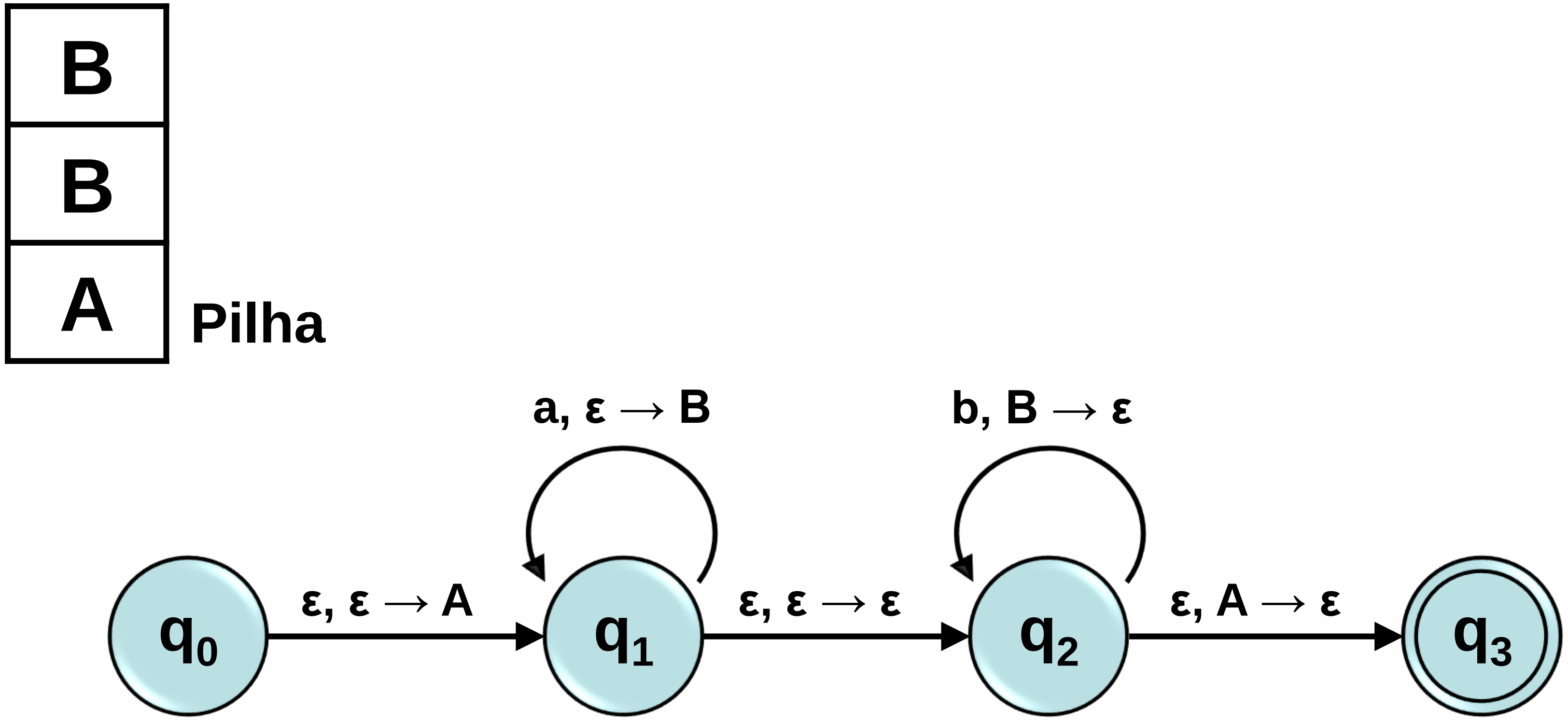


Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Ainda na transição "a, ε → B" , estando o APND no estado q_1, lendo o segundo símbolo “a” da cadeia de entrada, não se desempilha nenhum símbolo da pilha, empilha-se o símbolo ”B” e permanece-se em q_1.

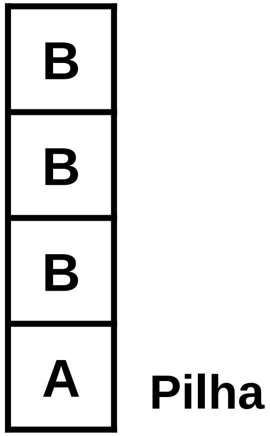


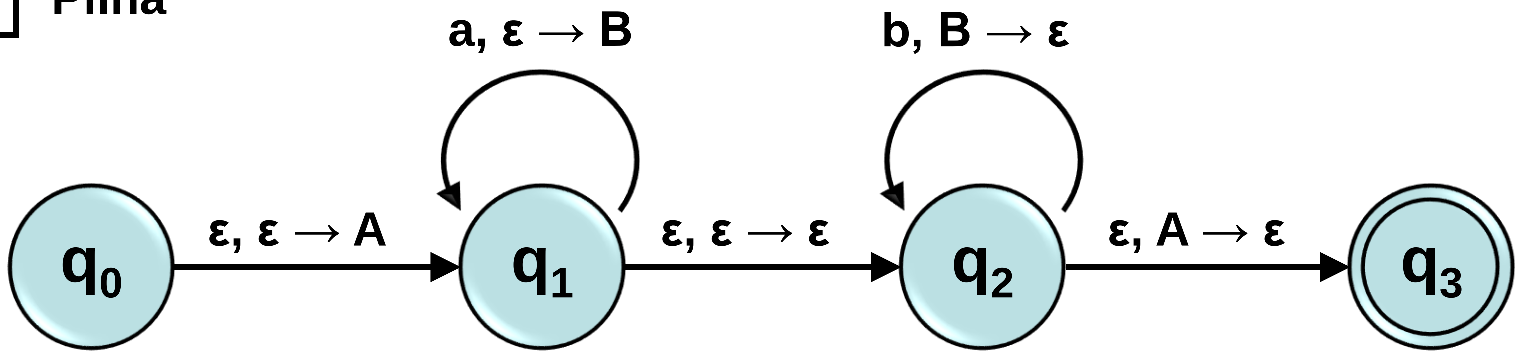
Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Ainda na transição "a, ε → B" , estando o APND no estado q_1, lendo o terceiro símbolo “a” da cadeia de entrada, não se desempilha nenhum símbolo da pilha, empilha-se o símbolo “B” e permanece-se em q_1.



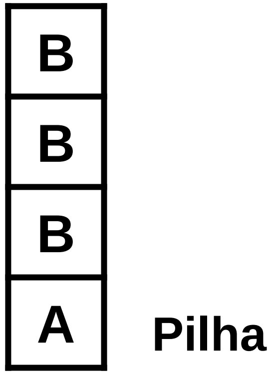


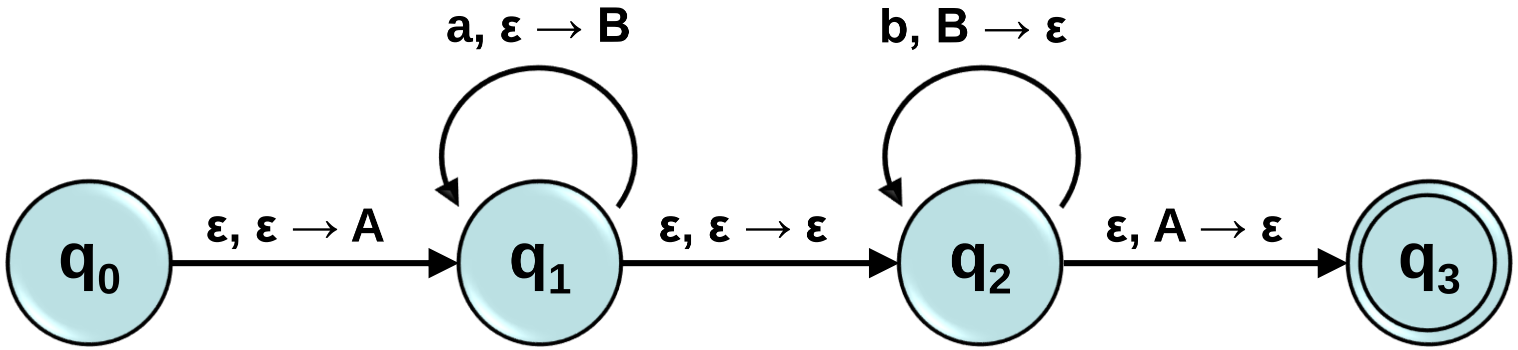
Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Lidos todos os símbolos “a”, podemos usar a transição “ε, ε → ε” . Dessa forma, estando o APND no estado q_1, não lendo nenhum símbolo da cadeia de entrada, não desempilhando nenhum símbolo da pilha, não empilhando nenhum símbolo, avança-se para o estado q_2.



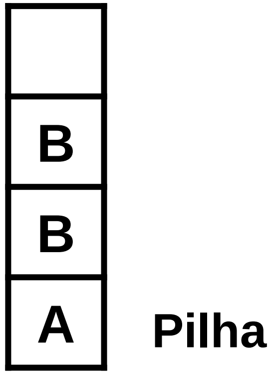


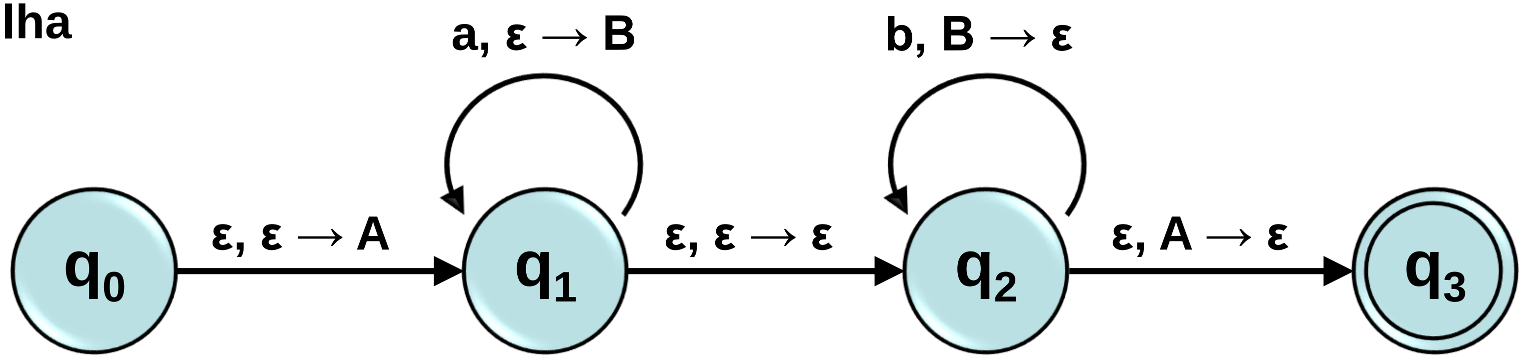
Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Note que em q_2, temos duas transições possíveis. Entretanto, como queremos ler os três símbolos "b" restantes da cadeia de entrada, optaremos pela transição "b, B → ε". Desse modo, estando o APND no estado q_2, lendo o primeiro símbolo "b" da cadeia de entrada, desempilha-se o símbolo "B", não empilhando nenhum símbolo e permanece-se em q_2.



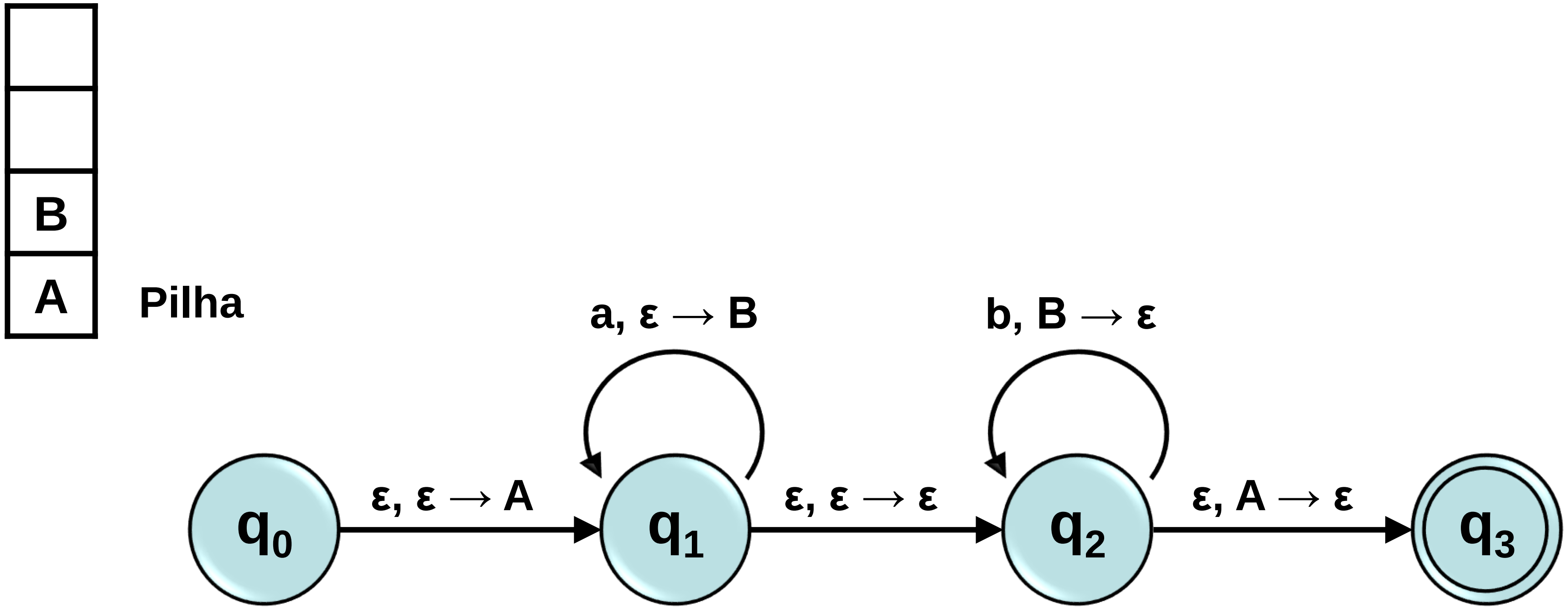


Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Ainda na transição "b, B → ε" , estando o APND no estado q_2, lendo o segundo símbolo “b” da cadeia de entrada, desempilha-se o símbolo “B”, não empilhando nenhum símbolo e permanece-se em q_2.

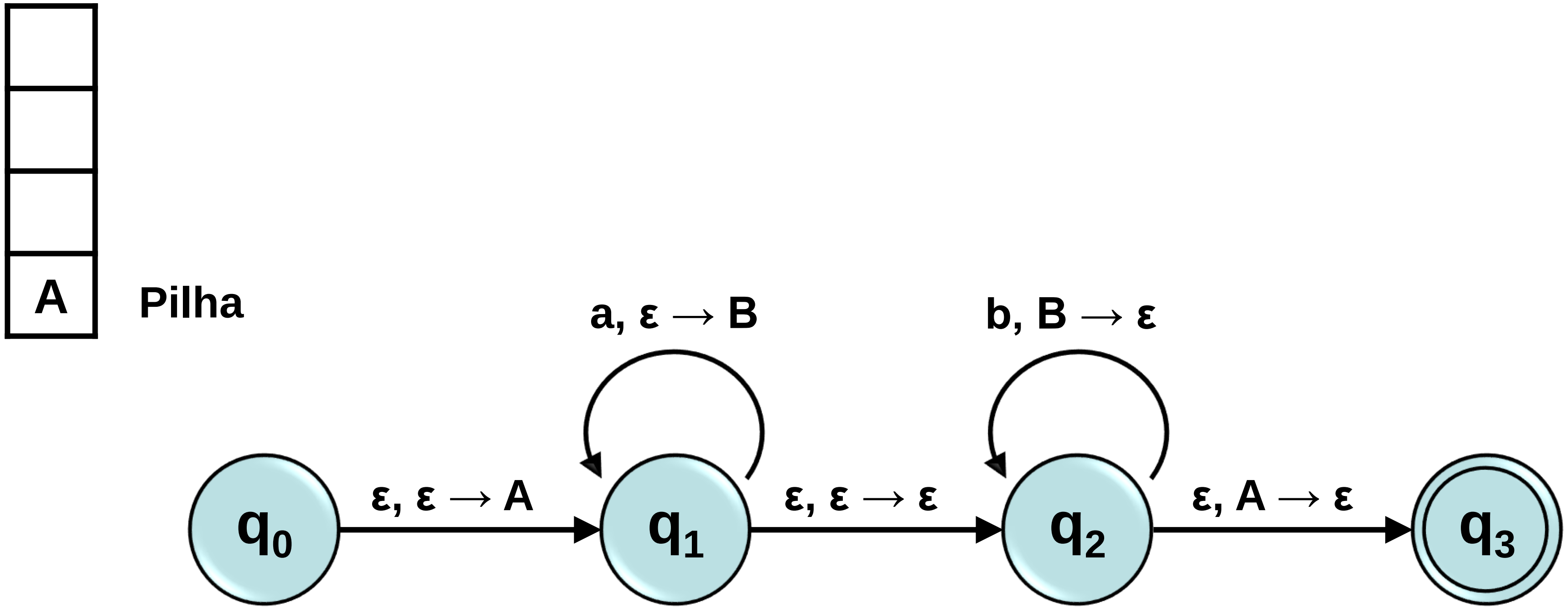


Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Ainda na transição “b, B → ε” , estando o APND no estado q_2, lendo o terceiro símbolo “b” da cadeia de entrada, desempilha-se o símbolo “B”, não empilhando nenhum símbolo e permanece-se em q_2.

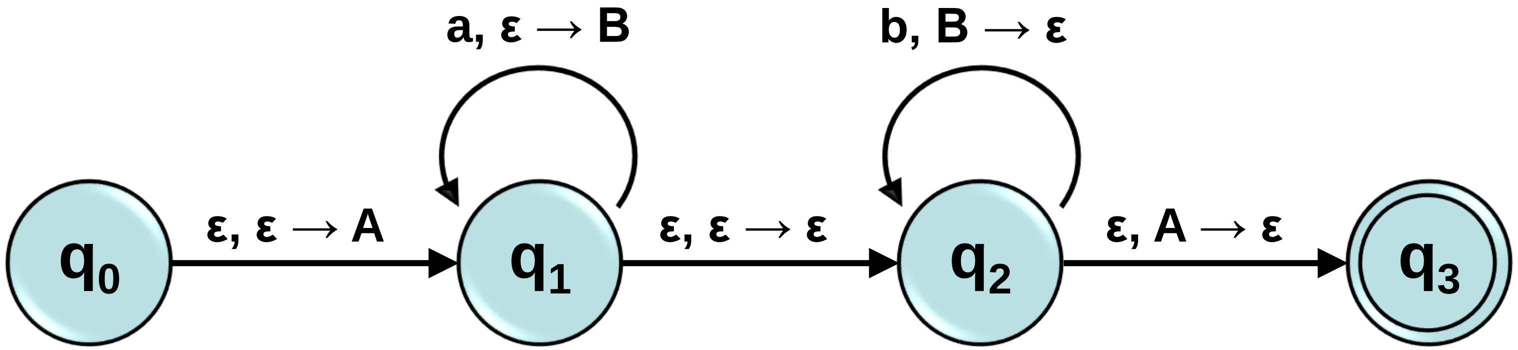


Fonte: autoria própria.

Linguagens Livres de Contexto – Autômato de Pilha Não Determinístico

- aaabbb

- Vamos agora para a outra transição existente em q_2, “ε, A → ε” . Estando o APND no estado q_2 lendo nenhum símbolo da cadeia de entrada, desempilha-se o símbolo “A”, não se empilha nenhum símbolo e avança-se para o estado final q_3. Como a cadeia foi lida em sua totalidade, o APND encontra-se no estado final e a pilha está vazia, conclui-se que o APND aceitou (reconheceu) a cadeia de entrada aaabbb.



Fonte: autoria própria.

Interatividade

Qual das seguintes afirmações sobre linguagens livres de contexto está correta?

a) Linguagens livres de contexto podem ser reconhecidas por autômatos finitos determinísticos (AFDs).
b) Linguagens livres de contexto não podem representar estruturas de dados aninháveis.
c) Linguagens livres de contexto são mais poderosas do que linguagens recursivamente enumeráveis.
d) Linguagens livres de contexto não podem conter gramáticas ambíguas.
e) Toda linguagem regular é uma linguagem livre de contexto.

Resposta

Qual das seguintes afirmações sobre linguagens livres de contexto está correta?

a) Linguagens livres de contexto podem ser reconhecidas por autômatos finitos determinísticos (AFDs).
b) Linguagens livres de contexto não podem representar estruturas de dados aninháveis.
c) Linguagens livres de contexto são mais poderosas do que linguagens recursivamente enumeráveis.
d) Linguagens livres de contexto não podem conter gramáticas ambíguas.
e) Toda linguagem regular é uma linguagem livre de contexto.

Computabilidade

- Exploraremos os fundamentos da computabilidade, incluindo a tese de Church-Turing, máquinas de Turing, linguagens recursivas e recursivamente enumeráveis além de variações dessas máquinas.

Definição:

- A computabilidade é a teoria que estuda os limites e possibilidades dos problemas solucionáveis por computadores.

Computabilidade

Importância:

- Compreender a computabilidade é fundamental para desenvolver algoritmos eficientes e identificar problemas insolúveis.

- Um exemplo de problema insolúvel é o “problema da parada”, que estudaremos na próxima aula, que verifica se um programa terminará sua execução ou entrará em um loop infinito.

Computabilidade – Tese de Church-Turing

- A tese de Church-Turing estabelece uma base teórica para a definição formal e estudo da computabilidade.

- Ela é fundamental porque envolve a classificação dos problemas em termos de sua solubilidade. Através dessa tese, podemos entender que qualquer algoritmo ou problema computacional pode ser representado por uma máquina de Turing, que é uma máquina abstrata capaz de realizar qualquer cálculo computacional.

Computabilidade – Tese de Church-Turing

- Essa relação com a computabilidade nos permite definir formalmente o que é um problema computacionalmente solúvel (computável) e distinguir entre problemas que podem ser resolvidos de forma eficiente e aqueles que são intrinsecamente difíceis ou insolúveis. Além disso, a tese de Church-Turing também nos ajuda a estabelecer limites teóricos para a computação.

- Por exemplo, ela mostra que existem problemas que são insolúveis, ou seja, não há um algoritmo ou máquina de Turing que possa resolvê-los em tempo finito.

Computabilidade – Máquina de Turing

- A máquina de Turing é um modelo teórico de um dispositivo capaz de armazenar e manipular informações em uma fita ilimitada à direita.

- Consiste em um cabeçote de leitura/escrita que se move pela fita, alterando o estado interno da máquina de acordo com regras predefinidas.

- As máquinas de Turing são usadas para modelar e analisar a computabilidade de problemas e expressar algoritmos.

Computabilidade – Máquina de Turing

- O cabeçote ou cursor da máquina de Turing é capaz de escrever e sobrescrever sobre a fita, além de poder mover-se à direita e a esquerda.

Vejamos a definição formal de uma máquina de Turing:

Computabilidade – Máquina de Turing

Uma máquina de Turing (MT) é uma 7-upla $(Q, \Sigma , \Gamma , \delta , q_0, q_a, q_r)$, onde:

- Q é o conjunto finito de estados;
- Σ é o alfabeto de entrada, onde $\beta \notin \Sigma$;
- Γ é o alfabeto da fita, tal que $\Sigma \subseteq \Gamma$ e $\beta \in \Gamma$;
- δ é a função de transição, com $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E,D\}$;
- $q_0 \in Q$ é o estado inicial;
- q_a é o estado de aceitação;
- q_r é o estado de rejeição, $q_r \neq q_a$.

Computabilidade – Máquina de Turing

- δ é a função de transição, com $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E,D\}$

Observe a função de transição δ. Por se tratar de uma função, o conjunto de partida é um par ordenado que contém um determinado estado e um símbolo do alfabeto da fita, por exemplo, (q_1, x) e o conjunto de chegada é uma 3-upla contendo um estado, um símbolo do alfabeto da fita e a direção a seguir pelo cabeçote da fita, por exemplo (q_2, y, D). Dessa forma, temos por exemplo:

- $\delta (q_1, x) = (q_2, y, D)$

- Essa é a notação formal da função de transição, cuja interpretação é: “Estando a MT no estado q_1, lendo x, sobrescreva ‘y’, mova-se uma célula à direita e alcance o estado q_2”.

Computabilidade – Máquina de Turing

Exemplo: Execute o algoritmo, dada a máquina de Turing M = (Q, Σ , Γ , δ , q_0, q_a, q_r) e a fita:

•	**a**	**a**	**a**	**β**	**β ...**

- Q = {q_0, q_1, q_2, q_3, q_4, q_a, q_r};
- Σ = {a, b};
- Γ = {·, a, β};
- δ = ((q0, a) = (q1, β, D); (q1, a) = (q2, β, D); (q2, a) = (q3, β, D); (q3, β) = (q4, β, E); (q4, β) = = (q4, β, E); (q4, ·) = (qa, ·, E))
- q_0 é o estado inicial;
- q_a é o estado de aceitação;
- q_r é o estado de rejeição.

Computabilidade – Máquina de Turing

δ = ((q0, a) = (q1, β, D); (q1, a) = (q2, β, D); (q2, a) = (q3, β, D); (q3, β) = (q4, β, E); (q4, β) = (q4, β, E); (q4, ·) = (qa, ·, E))

•	a	a	a	β	β ...
	↑ q_0				

•	β	a	a	β	β ...
		↑ q_1			

•	β	β	a	β	β ...
			↑ q_2		

•	β	β	β	β	β ...
				↑ q_3	

•	β	β	β	β	β ...
			↑ q_4		

•	β	β	β	β	β ...
		↑ q_4			

•	β	β	β	β	β ...
	↑ q_4				

•	β	β	β	β	β ...
↑ q_4					

•	β	β	β	β	β ...
	↑ q_4				

Computabilidade – Linguagens Recursivas e Recursivamente Enumeráveis

- Recordando os autômatos finitos e os de pilha, dizíamos que ele aceitou a cadeia quando ele alcançava o estado final.

No caso das máquinas de Turing, como se têm dois estados especiais, qa e qr, insere-se nova nomenclatura. Assim, dada uma máquina de Turing (MT) e uma cadeia dizemos que:

- MT aceita a cadeia se, partindo da configuração inicial e efetuando zero ou mais transições, alcança-se configuração de aceitação; e

- MT decide a cadeia se, partindo da configura inicial e efetuando zero ou mais transições, alcançam-se a configuração de aceitação ou a configuração de rejeição.

Computabilidade – Linguagens Recursivas e Recursivamente Enumeráveis

Uma linguagem L é recursiva, ou Turing – decidível, se existe uma MT que a decide:

- Dado $\omega \in \Sigma^*$, a MT decide se $\omega \in L$ ou $\omega \notin L$.

- A linguagem L é recursivamente enumerável, ou Turing – reconhecível, se existe um MT que a reconhece.

  - Dado $\omega \in \Sigma^*$, a MT aceita se $\omega \in L$ e rejeita ou entra em loop se $\omega \notin L$.

  - Se uma linguagem é recursiva, então também é recursivamente enumerável.

Computabilidade – Linguagens Recursivas e Recursivamente Enumeráveis

Observe a figura:

Fonte: autoria própria.

Computabilidade – Variações da Máquina de Turing

Nas máquinas de Turing é possível implementar diversas variações, entre elas:

- Máquina de Turing cujo cabeçote de leitura pode não se mover.
- Máquina de Turing com múltiplas (K) fitas.
- Apesar de mais eficientes, o poder computacional não se altera, ou seja, tudo que é computável para as variações também é computável para a MT tradicional.

Interatividade

Sobre a classe das linguagens recursivas:

a) Está contida propriamente na classe das linguagens enumeráveis recursivamente.
b) Não pode ser reconhecida por uma máquina de Turing.
c) Não pode ser reconhecida por uma máquina de Turing que sempre para, qualquer que seja a entrada.
d) É sempre reconhecida por um autômato finito.
e) É sempre reconhecida por um autômato de pilha.

Resposta

Sobre a classe das linguagens recursivas:

a) Está contida propriamente na classe das linguagens enumeráveis recursivamente.
b) Não pode ser reconhecida por uma máquina de Turing.
c) Não pode ser reconhecida por uma máquina de Turing que sempre para, qualquer que seja a entrada.
d) É sempre reconhecida por um autômato finito.
e) É sempre reconhecida por um autômato de pilha.

ATÉ A PRÓXIMA!