

LINGUAGENS FORMAIS E AUTÔMATOS

EXPRESSÕES E GRAMÁTICAS REGULARES

Autor: Me. Pedro Henrique Chagas Freitas

Revisor: Edilson José Rodrigues

Tempo de leitura do conteúdo estimado em 1 hora.

Introdução

Olá, caro(a) estudante!

É com entusiasmo que convido você para a leitura desta unidade. Você sabia que as expressões e gramáticas regulares são de fundamental importância para o **desenvolvimento das linguagens de programação** ? Isso ocorre porque as expressões e gramáticas regulares são mecanismos geradores de conjuntos básicos que formam as linguagens. A formação de expressões e de gramáticas vale tanto para a comunicação entre humanos como para a interpretação da comunicação humano e máquina.

Nesta unidade, veremos a definição e as operações por de trás das expressões regulares e a definição e os componentes que formam as gramáticas regulares, exploraremos, também, o teorema do bombeamento para linguagens regulares e, por fim, veremos o funcionamento da hierarquia de Chomsky.

Boa leitura!

Definição e Operações de Expressões Regulares

Caro(a) estudante, é de seu conhecimento que a teoria das linguagens formais foi desenvolvida com o intuito de aproximar as linguagens humanas, bem como sua estrutura e formação, com as **linguagens de programação**, ou seja, com as linguagens regulares e os autômatos, que têm na sua base as expressões e as gramáticas regulares.

Atualmente, o campo da **inteligência artificial** tem beneficiado-se do emprego das expressões e gramáticas regulares para interpretação de problemas práticos, como conduzir um carro sem motorista, por exemplo.

Desde o início da computação, já era possível prever que, em algum momento, o processamento e o armazenamento de dados levariam-nos a implementar ideias que, baseadas nas expressões e nas gramáticas regulares, eram até então, campo exclusivo da **ficção científica**.

Historicamente, toda linguagem regular pode ser descrita por uma expressão denominada expressão regular. Trata-se de um formalismo denotacional, também considerado gerador, pois se pode inferir como construir ('gerar') as palavras de uma linguagem. Uma expressão regular é definida a partir de conjuntos (linguagens) básicos e operações de concatenação e de união. As expressões regulares são consideradas adequadas para a comunicação humano com humano e, principalmente, para a comunicação humano com máquina (MENEZES, 2015, p. 96).

Vamos iniciar esse tema apresentando a definição de Expressão Regular (ER), antes de abordar as operações de **união** e **concatenação**, vejamos:

A **expressão** \emptyset é uma expressão regular e denota a linguagem vazia: \emptyset . Dito isso, a expressão ϵ é uma expressão regular e denota a linguagem que contém exclusivamente a palavra vazia: $\{\epsilon\}$.

Logo, para qualquer símbolo $x \in \Sigma$, a expressão x é uma expressão regular e denota a linguagem que contém exclusivamente a palavra constituída pelo símbolo x : $\{x\}$.

Destrinchando, então, temos que uma expressão regular (ER) é derivada de um alfabeto (A), que tem por definição:

A **expressão** (EX) = expressão regular, como uma linguagem vazia e algum símbolo, como x , por exemplo, em que x pertence ao conjunto da expressão (ER), logo, nesse exemplo, ER é uma expressão regular e contém exclusivamente a palavra $\{X\}$, que, por sua vez, é constituída por meio de um símbolo: x .

Dito isso, caso tivéssemos (r) e (s) como expressões regulares, poderíamos ter as linguagens R e S, respectivamente, logo, quando quiséssemos realizar a operação de **união**, teríamos a expressão: $(r+s)$. Quando desejássemos realizar a operação de concatenação, teríamos a expressão (rs) , mas, se a **concatenação** fosse sucessiva, teríamos (r^*) , logo, se (r), dado nosso exemplo, fosse uma expressão regular, a correspondente linguagem derivada

por r seria representada por: $L(r)$.

Logo, na união, temos a expressão: $(r + s)$, que é uma expressão regular e denota a linguagem: $R \cup S$. Já na concatenação, temos a expressão (rs) , que é uma expressão regular e denota a linguagem: $RS = \{ uv \mid u \in R \text{ e } v \in S \}$.

Vamos observar um exemplo para ficar mais claro. No Quadro 2.1, temos as expressões regulares e as correspondentes linguagens geradas.

**Expressão
regular****Linguagens geradas** aa Somente com a palavra aa ba^* Todas as palavras que iniciam por b , seguido por zero ou mais a $(a+b)^*$ Todas as palavras sobre $\{a, b\}$ $(a+b)^*aa(a+b)^*$ Todas as palavras contendo aa como subpalavra a^*ba^*ba Todas as palavras contendo exatamente dois b $(A+b)^*(aa+bb)$ Todas as palavras que terminam com aa ou bb

Quadro 2.1 – Expressões regulares e correspondentes linguagens geradas a partir das operações apresentadas

Fonte: Menezes (2015, p. 105).

#PraCegoVer : o quadro é dividido em sete linhas e duas colunas. Seguindo a ordem de cima para baixo, apresenta, na primeira coluna, o quadro “Expressões regulares” e, na segunda coluna, as “linguagens geradas”. Na segunda linha da primeira coluna, temos as operações e, na segunda linha da segunda coluna, temos as descrições das linguagens. Temos, na segunda linha da primeira coluna e sucessivamente, as expressões regulares: aa , ba^* , $(a+b)^*$, $(a+b)^*aa(a+b)^*$, a^*ba^*ba e $(A+b)^*(aa+bb)$. Temos, na segunda linha da segunda coluna e sucessivamente, a descrição das linguagens geradas, são elas: somente com a palavra aa , todas as palavras que iniciam por b , seguido por zero ou mais a , todas as palavras $\{a,b\}$, todas as palavras contendo aa como subpalavra, todas as palavras contendo exatamente dois b e todas as palavras que terminam com aa ou bb .

Detalhando a **linguagem gerada pela expressão** $(a + b)^*(aa + bb)$, vale que:

a e b denotam $\{a\}$ e $\{b\}$, respectivamente

$a + b$ denota $\{a\} \cup \{b\} = \{a, b\}$

$(a + b)^*$ denota $\{a, b\}^*$

aa e bb denotam $\{a\}\{a\} = \{aa\}$ e $\{b\}\{b\} = \{bb\}$, respectivamente

$(aa + bb)$ denota $\{aa\} \cup \{bb\} = \{aa, bb\}$

$(a + b)^*(aa + bb)$ denota $\{a, b\}^* \{aa, bb\}$

Portanto, $GERA((a + b)^*(aa + bb))$ corresponde à seguinte linguagem:

$\{aa, bb, aaa, abb, baa, bbb, aaaa, aabb, abaa, abbb, baaa, babb, bbba, bbbb, \dots\}$



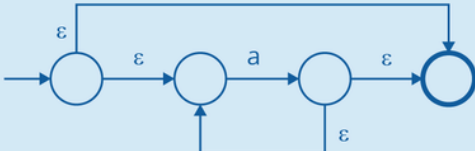


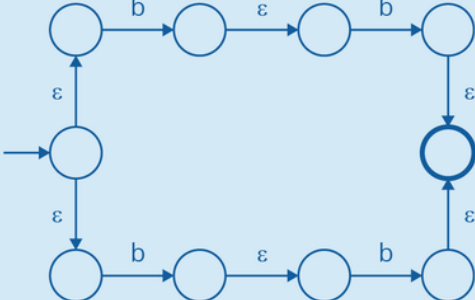
Podemos verificar que, por **definição**, uma linguagem é **regular** se, e somente se, for possível construir um **autômato finito** (determinístico, não determinístico ou com movimentos vazios), que reconheça essa linguagem.

Assim, é necessário mostrar que, dada uma expressão regular r qualquer, é possível construir um autômato finito M tal que:

$$\text{ACEITA}(M) = \text{GERA}(r)$$

Dessa forma, é necessário perceber que, dada uma expressão regular (r) qualquer, é possível construir um autômato finito M .

Precisamos aprofundar mais esse assunto! Concorda? Vamos lá!

Expressão regular	Autômato finito correspondente
a	
b	
a*	
aa	
bb	
(aa+bb)	

© VG Educacional

Fonte: Menezes (2015, p. 125).

#PraCegoVer : o infográfico estático apresenta forma de quadro com duas colunas e sete linhas, sendo que, nas duas primeiras linhas, em cada coluna, respectivamente, está escrito “Expressão regular” e “Autômato finito correspondente”. A partir da segunda linha, em cada coluna, respectivamente, há uma expressão e seu desenho a ilustrar. Os conteúdos são: 2ª linha: “a”, com a imagem na coluna ao lado de uma seta apontada para a direita, que vai até um círculo vazado (branco dentro e contorno preto) e, em seguida, há outra seta, com a letra “a” em cima, que vai desse círculo até o círculo seguinte, que está com o contorno mais destacado em preto. 3ª linha: “b”, com a imagem na coluna ao lado de uma seta apontada para a direita, que vai até um círculo vazado (branco dentro

e contorno preto) e, em seguida, há outra seta, com a letra “b” em cima, que vai desse círculo até o círculo seguinte, que está com o contorno mais destacado em preto. 4ª linha: “a*”, com a imagem na coluna ao lado de quatro bolinhas alinhadas, sendo a última do lado direito com o contorno em preto mais destacado, e setas que ligam a primeira à última bolinha, além de uma seta que liga as duas bolinhas do meio e uma seta que interliga a primeira e a última bolinha. A seta que vai da primeira bolinha para a segunda tem como símbolo “ε”; a seta que vai da segunda para a terceira bolinha tem como símbolo “a”; e a seta que vai da terceira para a quarta bolinha tem como símbolo novamente “ε”, assim como as setas que vão da primeira para a última bolinha e a que interliga as duas bolinhas do meio. 5ª linha: “aa”, com a imagem de quatro bolinhas, sendo a última com contorno mais destacado em preto, e setas que as ligam da primeira para a última. A seta que liga a primeira à segunda bolinha tem como símbolo “a”; a seta que liga a segunda à terceira bolinha tem como símbolo “ε”; e a seta que liga a terceira à quarta bolinha tem como símbolo “a”. 6ª linha: “bb” com a imagem de quatro bolinhas, sendo a última com contorno mais destacado em preto, e setas que as ligam da primeira para a última. A seta que liga a primeira à segunda bolinha tem como símbolo “b”; a seta que liga a segunda à terceira bolinha tem como símbolo “ε”; e a seta que liga a terceira à quarta bolinha tem como símbolo “b”. 7ª linha: “(aa+bb)”, com a imagem de 10 bolinhas de modo que, todas juntas, formam um retângulo com quatro bolinhas em cima, quatro embaixo e duas ao meio, uma em cada lado. O esquema de setas se inicia na primeira bolinha do meio, do lado esquerdo, que leva à bolinha de cima e à bolinha de baixo com os símbolos “ε”, onde se ligam à terceira bolinha, tanto em cima quanto embaixo, com o símbolo “b” e, então, se conectam à próxima com o símbolo “ε”, para, depois, se conectarem à próxima bolinha com o símbolo “b”, e finalizar se conectando à última bolinha com o símbolo “ε”.

Vejamos um exemplo:

Seja r uma expressão regular com zero operadores. Então, r só pode ser da forma:

$$r = \emptyset$$

$r = \varepsilon$

$r = x$ (x pertencente a Σ)

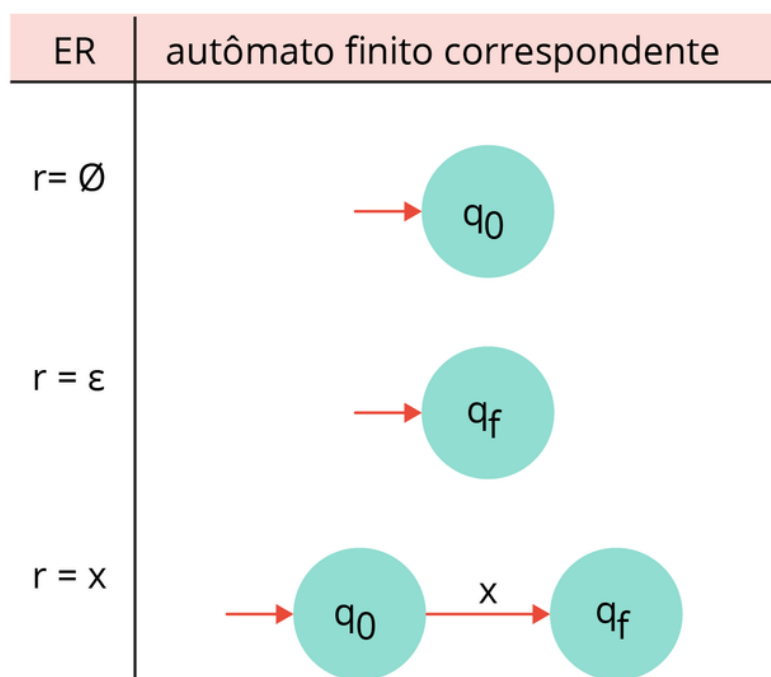
Os autômatos finitos:

$M1 = (\emptyset, \{q_0\}, \delta_1, q_0, \emptyset)$

$M2 = (\emptyset, \{q_f\}, \delta_2, q_f, \{q_f\})$

$M3 = (\{x\}, \{q_0, q_f\}, \delta_3, q_0, \{q_f\})$

A seguir, temos a ilustração dos ER e seus autômatos finitos correspondentes:



© VG Educacional

Figura 2.1 – Autômatos finitos correspondentes às expressões regulares com zero operadores

Fonte: Menezes (2015, p. 125).

#PraCegoVer : na ilustração, temos expressões regulares e seus respectivos autômatos finitos a partir de zero operadores. Na coluna à esquerda, temos as expressões regulares e, na coluna à direita, temos os autômatos finitos correspondentes. Temos, na coluna à esquerda, as respectivas expressões regulares, a partir de r com zero operadores, na primeira linha após o título, é apresentado $r = \emptyset$, na segunda linha, temos $r = \epsilon$ e, na terceira linha, temos $r = x$ (x pertencente a Σ), respectivamente, na coluna à direita, contendo os autômatos finitos correspondentes às expressões regulares, temos, na primeira linha, $M1 = (\emptyset, \{q_0\}, \delta_1, q_0, \emptyset)$, $M2 = (\emptyset, \{q_f\}, \delta_2, q_f, \{q_f\})$ e $M3 = (\{x\}, \{q_0, q_f\}, \delta_3, q_0, \{q_f\})$.

Por fim, prezado(a) estudante, em uma linguagem formal, as expressões são **responsáveis** pelo encadeamento lógico do comportamento da **linguagem**, conforme o conjunto gerado a partir das operações realizadas. Por sua vez, perceba que essas operações vão derivar os respectivos autômatos finitos

correspondentes e as gramáticas regulares, conforme veremos a seguir.

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

Leia o trecho a seguir:

“Uma expressão regular é definida a partir de conjuntos básicos e operações de concatenação e de união. Por exemplo, na expressão regular (W) , como uma linguagem vazia teremos, por exemplo, x , em que x pertence ao conjunto da expressão”.

DIVERIO, T. A.; MENEZES, P. F. B. **Teoria da computação** : máquinas universais e computabilidade. 3. ed. São Paulo: Bookman, 2011.

Considerando as definições de expressões regulares apresentadas, assinale a alternativa que representa concatenação da linguagem gerada, a partir da expressão regular $(a+b)^*$.

- ☐ **a)** Linguagem gerada com somente a palavra aa .
- ☐ **b)** Linguagem gerada com somente a palavra bb .
- ☐ **c)** Linguagem gerada com todas as palavras que iniciam com a .
- ☐ **d)** Linguagem gerada com todas as palavras sobre a e b .
- ☐ **e)** Linguagem gerada com todas as palavras contendo dois b .

Definição e Componentes de Gramáticas Regulares

Conforme já abordado anteriormente, o **conceito** de gramática regular **permeia** a própria formação das linguagens regulares, logo, a partir das gramáticas, é possível **definir** tanto as linguagens regulares como as linguagens não regulares. Todavia, como as gramáticas estabelecem **restrições** às regras de produção das linguagens, é natural que se apresente as gramáticas por classes.

Destaca-se que é possível perceber nas gramáticas lineares uma forte restrição no formato das produções, no caso, o lado esquerdo possui exatamente uma variável, já o lado direito de uma produção é constituído por, no máximo, uma variável. Adicionalmente, esta variável, se existir, sempre antecede (linear à esquerda) ou sucede (linear à direita) qualquer subpalavra (DIVERIO, MENEZES, 2011, p. 114).

Existe, então, mais de uma maneira de restringir as regras de produção das linguagens, de forma a definir uma gramática regular. Veremos quatro formas de análise de gramáticas lineares, que é o tipo mais comum de gramática regular.

Gramáticas lineares: seja $G = (V, T, P, S)$ uma gramática. Temos os elementos A e B como elementos V e w de uma palavra T^* .

Logo, G é uma gramática linear se todas as duas produções (A e B) encontram-se em uma, e em somente uma, das seguintes formas, conforme demonstrado a seguir:

Gramáticas lineares:

Gramática linear à direita (GLD)

Gramática linear à esquerda (GLE)

Gramática linear unitária à direita (GLUD)

Gramática linear unitária à esquerda (GLUE)

todas as regras de produção são da forma:

$$A \rightarrow wB \text{ ou } A \rightarrow w$$

Dito isso, vale ressaltar que uma gramática G é dita uma **gramática regular**, eventualmente abreviada por GR, se G for uma gramática linear. Vejamos, então, de forma algébrica, a derivação da linguagem gerada a partir de G :

Seja $G = (V, T, P, S)$ uma gramática. A linguagem gerada pela gramática G será $L(G)$, tal que: $L(G) = \{ w \in T^* \mid S \Rightarrow^+ w \}$

Vamos verificar um exemplo para deixar mais claro:

Dada a gramática regular: $a(ba)^*$, esta é gerada por quais gramáticas regulares?

A linguagem $a(ba)^*$ é gerada pelas seguintes gramáticas regulares:

a) **Linear à direita** . $G = (\{ S, A \}, \{ a, b \}, P, S)$, e P possui as seguintes regras de produção:

$$S \rightarrow aA$$

$$A \rightarrow baA \mid \varepsilon$$

b) **Linear à esquerda** . $G = (\{ S \}, \{ a, b \}, P, S)$, e P possui as seguintes regras de produção:

$$S \rightarrow Sba \mid a$$

c) **Linear unitária à direita** . $G = (\{ S, A, B \}, \{ a, b \}, P, S)$, e P possui as seguintes regras de produção:

$$S \rightarrow aA$$

$$A \rightarrow bB \mid \varepsilon$$

$$B \rightarrow aA$$

d) **Linear unitária à esquerda** . $G = (\{ S, A \}, \{ a, b \}, P, S)$, e P possui as seguintes regras de produção:

$$S \rightarrow Aa \mid a$$

$$A \rightarrow Sb$$

Por fim, vamos o exemplo da construção de um autômato finito não determinístico (AFN) a partir de uma gramática regular.

Considere a seguinte gramática linear unitária à direita:

$$G = (\{ S, A, B \}, \{ a, b \}, P, S)$$

Em que P é tal que:

$$S \rightarrow aA$$

$$A \rightarrow bB \mid \varepsilon$$

$$B \rightarrow aA$$

O AFN que reconhece a linguagem gerada pela gramática G é:

$$M = (\{ a, b \}, \{ S, A, B, qf \}, \delta, S, \{ qf \})$$

Assim, teremos as respectivas produções e transições:

Produção	Transição
$S \rightarrow aA$	$\delta(S, a) = \{A\}$
$A \rightarrow bB$	$\delta(A, b) = \{B\}$
$A \rightarrow \varepsilon$	$\delta(A, \varepsilon) = \{qf\}$
$B \rightarrow aA$	$\delta(B, a) = \{A\}$

Tabela 2.1 – Produções e transições oriundas da linguagem gerada pela gramática G

Fonte: Elaborada pelo autor.

#PraCegoVer : na primeira coluna, temos as produções derivadas da linguagem gerada pela gramática G , são elas: na segunda linha $S \rightarrow aA$, $A \rightarrow bB$, $A \rightarrow \varepsilon$ e $B \rightarrow aA$, sucessivamente. Na segunda coluna e na segunda linha, temos, respectivamente, as transições: $\delta(S, a) = \{A\}$, $\delta(A, b) = \{B\}$, $\delta(A, \varepsilon) = \{qf\}$ e $\delta(B, a) = \{A\}$.

Veja, então, que o autômato finito construído a partir da gramática regular apresentada será:

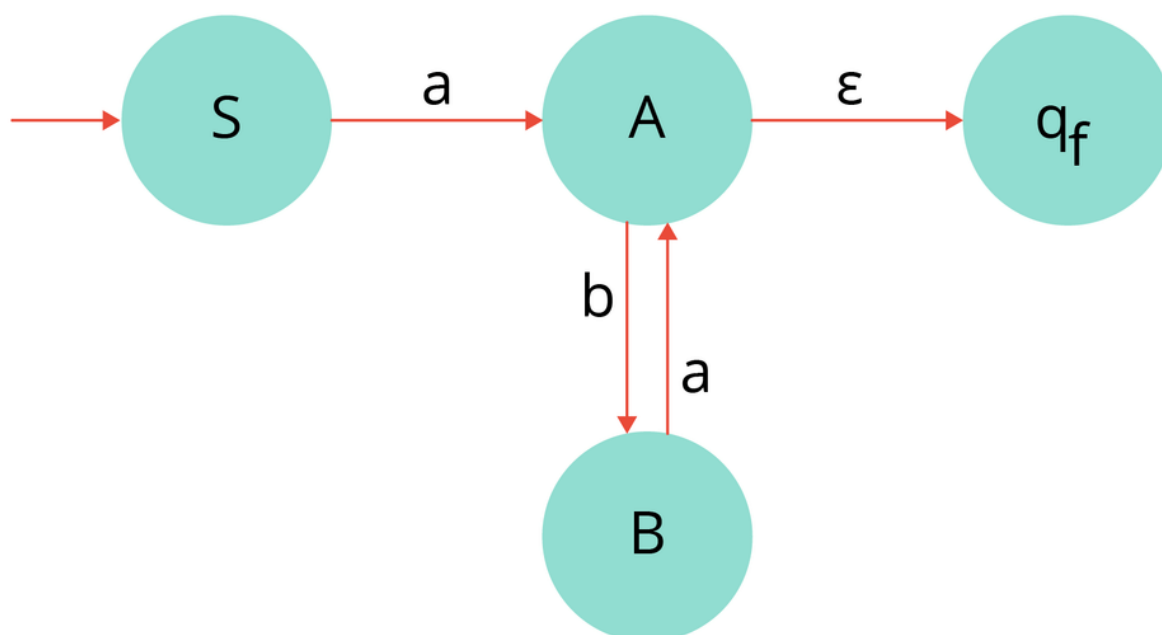


Figura 2.2 – Autômato finito construído a partir de uma gramática regular
Fonte: Adaptada de Menezes (2015).

#PraCegoVer : na figura, temos um autômato finito construído a partir da gramática G , sendo $M = (\{a, b\}, \{S, A, B, q_f\}, \delta, S, \{q_f\})$, na qual $G = (\{S, A, B\}, \{a, b\}, P, S)$. O respectivo autômato apresenta três conexões, à direita, q_f , à esquerda, S e, embaixo, B .

Na teoria da computação, é comum o emprego de autômatos finitos construídos a partir de gramáticas regulares, pois a própria elaboração de linguagens regulares permeia o emprego das gramáticas, logo, é importante perceber que a gramática é fundamental para implementação e construção do autômato finito.

SAIBA MAIS

É possível haver uma gramática linear à esquerda e linear à direita?

Sim. Suponha $|w| \geq 1$. Se uma gramática tiver, simultaneamente, produções do tipo $A \rightarrow wB$ (linear à direita) e do tipo $A \rightarrow Bw$ (linear à esquerda), então, a correspondente linguagem gerada poderá não ser regular, ou seja, essa não é uma gramática regular (MENEZES, 2015, p. 102).

Para aprofundar, assista ao vídeo do professor Francisco Anderson, que trata com profundidade a temática.

Fonte: Aula... (2020).

ASSISTIR

Podemos, então, concluir o assunto dizendo que as gramáticas regulares são fundamentais para derivar as restrições que irão compor as linguagens regulares que serão criadas, ou seja, não cabe falar de linguagem regular sem gramática, tendo em vista que as regras ou restrições coordenam o encadeamento lógico das linguagens. Vejamos mais uma atividade. Vamos lá!

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

As gramáticas lineares apresentam uma forte restrição no formato das produções, no caso, o lado esquerdo possui exatamente uma variável, já o lado direito de uma produção é constituído por, no máximo, uma variável.

DIVERIO, T. A.; MENEZES, P. F. B. **Teoria da computação** : máquinas universais e computabilidade. 3. ed. São Paulo: Bookman, 2011.

Diante das definições de gramáticas lineares apresentadas, seja G uma gramática linear, contendo os elementos A e B , assinale a alternativa que representa as regras de produção da gramática linear unitária à esquerda (GLUE).

- ☐ **a)** Todas as regras de produção são da forma: $A \rightarrow wB$ ou $A \rightarrow w$.
- ☐ **b)** Todas as regras de produção são da forma: $A \rightarrow Bw$ ou $A \rightarrow w$.
- ☐ **c)** Todas as regras de produção são da forma: $A \rightarrow wB$ ou $A \rightarrow w$ e, adicionalmente: $|w| \leq 1$.
- ☐ **d)** Todas as regras de produção são da forma: $A \rightarrow Bw$ ou $A \rightarrow w$ e, adicionalmente: $|w| \leq 1$.
- ☐ **e)** Todas as regras de produção são da forma: $AB \rightarrow wB$ ou $A \rightarrow w$ e, adicionalmente: $|w| = 1$.

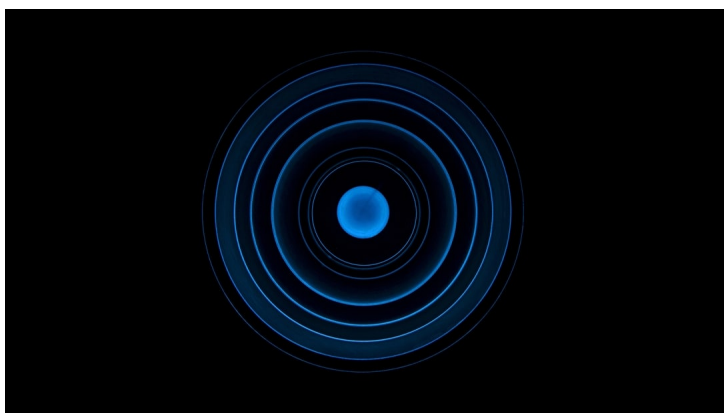
Teorema do Bombeamento para Linguagens Regulares

Caro(a) estudante, para aprofundar um pouco mais o conhecimento obtido até aqui, vamos aprender sobre as linguagens regulares, as quais são representadas por formalismos muito simples, sendo possível desenvolver algoritmos de **pouca complexidade**, de **grande eficiência** e de **fácil implementação** (DIVERIO; MENEZES, 2011). Entretanto, têm fortes limitações de expressividade, conforme vimos. A partir disso, surge o teorema do bombeamento para linguagens regulares.

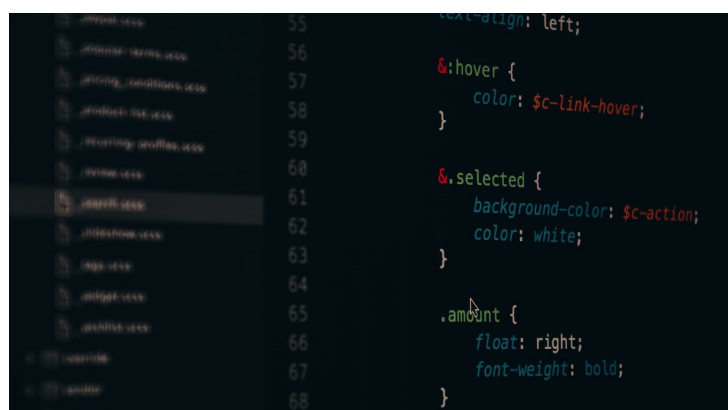
Umas das principais características das linguagens regulares é o fato de serem representadas por formalismos de pouca complexidade, grande eficiência e fácil implementação. A partir dessa lógica, você verá que nasce o teorema do bombeamento para as linguagens regulares, que adota duas variáveis por definição: (q_0) **estado inicial** e (q_f) **estado final**, em que as premissas a seguir são adotadas (MENEZES, 2015):



Se uma linguagem é regular, então, é aceita por um autômato finito determinístico, o qual possui um número finito e predefinido de (n) estados. Se o autômato reconhece uma entrada (w) de comprimento maior ou igual ao número de estados (n) , obrigatoriamente, o autômato assume algum estado (q) mais de uma vez, portanto, existe um ciclo na função programa que passa por (q) ;



logo, w pode ser dividido em três subpalavras $w = u v z$ tal que $|u v| \leq n$, $|v| \geq 1$, em que (v) é a parte de (w) reconhecida pelo ciclo. Claramente, então, tal ciclo pode ser executado, a execução desse ciclo é o “bombeamento”, ou seja é executado ou zero ou mais vezes. Portanto, para qualquer $i \geq 0$, temos a aceitação pelo autômato, ou seja, temos uma palavra da linguagem;



dito isso, se (L) é uma linguagem regular, então, existe uma constante (n) tal que, para qualquer palavra (w) de (L) , em que $|w| \geq n$, assim (w) pode ser definido como: $w = uvz$, em que: $|uv| \leq n$ e $|v| \geq 1$, sendo que, para todo $i \geq 0$, $u v^i z$ terão uma palavra de (L) .

Fontes: Ivan Kish / 123RF, Pixabay / Pexels, Pixabay / Pexels.

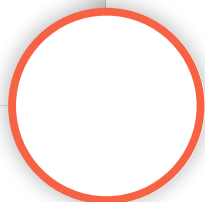
Por fim, vale destacar que, ao observar o **teorema do bombeamento**, temos que ele garante que os formalismos regulares são capazes de expressar diversos tipos de bombeamento.

Por exemplo:

Duplo bombeamento: $\{ a b \mid n \geq 0 \text{ e } m \geq 0 \}$

Triplo bombeamento: $\{ a b a \mid n \geq 0, m \geq 0 \text{ e } r \geq 0 \}$

REFLITA



O **teorema do bombeamento** também é conhecido como lema do bombeamento para linguagens regulares e descreve as propriedades essenciais de todas as linguagens regulares, isto é, todas as cadeias de uma linguagem regular que podem ser executadas ou bombeadas.

Fonte: Adaptado de Diverio e Menezes (2011).

O teorema do bombeamento foi fundamental para o desenvolvimento da computação como a conhecemos hoje, logo, a criação de bibliotecas de gramáticas das linguagens de programação passa diretamente pelo emprego dos conceitos apresentados sobre o teorema do bombeamento.

atividade

Atividade

Dentre as características principais das linguagens regulares, temos os formalismos de pouca complexidade, grande eficiência e fácil

implementação. A partir dessa lógica, nasce o teorema do bombeamento para as linguagens regulares, que adota duas variáveis por definição: (q_0) estado inicial e (q_f) estado final.

Portanto, estudante, ao analisar a teoria do bombeamento para linguagens regulares, se (L) é uma linguagem regular, então, existe uma constante (n) tal que, para qualquer palavra (w) de (L), em que $|w| \geq n$. Assim (w) pode ser definido como: $w = uvz$, em que: $|uv| \leq n$ e $|v| \geq 1$.

Diante do princípio algébrico apresentado, discorra sobre o comportamento a partir de $|v| \geq 0$.

Hierarquia de Chomsky

Com base em toda conjuntura temática apresentada até aqui, convidamos você a seguir nessa jornada de conhecimento, a fim de ampliar a esfera de fundamentos sobre linguagens formais e autômatos. Vamos lá?

As classes das linguagens regulares, livres do contexto, sensíveis ao contexto e recursivamente enumeráveis e suas inclusões próprias constituem a

hierarquia de Chomsky .

Esse é um assunto muito **rico em detalhes** e vamos buscar, ao longo do tópico, explicar todos os fatores que permeiam a temática em questão. Se observarmos a criação das linguagens regulares, veremos que vários são os problemas que ainda permeiam o desenvolvimento das linguagens regulares e dos autômatos. A criação de gramáticas regulares em grafos é uma das formas conhecidas de se generalizar o que conhecemos por gramáticas de Chomsky, que, por sua vez, têm demonstrado um imenso potencial para aplicações computacionais avançadas, como linguagens de interpretativa de inteligência artificial (MENEZES, 2015).

Veja, a seguir, quais são as **linguagens estudadas** na hierarquia de Chomsky.

Na hierarquia de Chomsky, são estudadas quatro classes de linguagens, são elas:

- *linguagens regulares ou tipo 3;*
- *linguagens livres de contexto ou tipo 2;*
- *linguagens sensíveis ao contexto ou tipo 1;*
- *linguagens recursivamente enumeráveis ou tipo 0.*

Logo, caro(a) estudante, as respectivas linguagens, como demonstrado na figura a seguir, constituem o que chamamos de **hierarquia de Chomsky** , ou seja, Noam Chomsky definiu essas classes como (potenciais) modelos para

linguagens naturais.

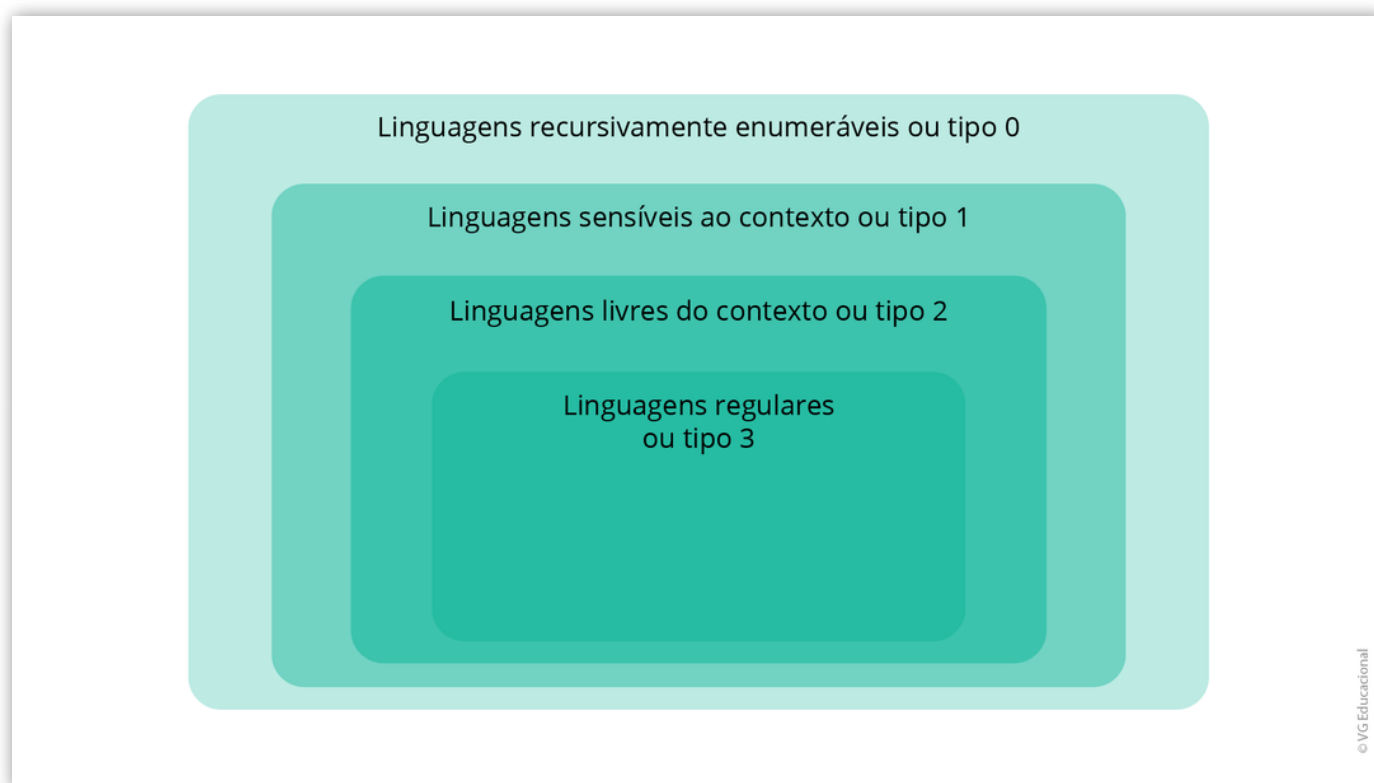


Figura 2.3 – Tipos de linguagens
Fonte: Adaptada de Menezes (2015).

#PraCegoVer : a figura mostra a hierarquia de Chomsky composta pelas linguagens regulares, ou tipo 3, pelas linguagens livres do contexto, ou tipo 2, pelas linguagens sensíveis ao contexto, ou tipo 1, e pelas linguagens recursivamente enumeráveis, ou tipo 0. Cada uma das linguagens está inserida na maior abstração da linguagem subsequente, no caso, temos as linguagens recursivamente enumeráveis ou tipo 0, como base, nelas, estão inseridas as linguagens sensíveis ao contexto ou tipo 1, já nas linguagens tipo 1 estão inseridas as linguagens livres do contexto ou tipo 2 e, por fim, nas linguagens tipo 2, estão inseridas as linguagens regulares ou tipo 3.

Dada a **complexidade** da temática, já podemos adiantar que nem sempre as linguagens de programação são tratadas adequadamente na hierarquia de Chomsky. Existem linguagens que não são livres do contexto, para as quais o poder dos formalismos sensíveis ao contexto é excessivo, sendo inadequados, principalmente no que se refere à complexidade (**quantidade de**

recursos, computacionais como tempo e espaço).

Observe que, adicionalmente, o conhecimento das linguagens sensíveis ao contexto é relativamente limitado, o que dificulta seu tratamento. Alguns exemplos de problemas que não podem ser tratados na classe de linguagens livres do contexto são os seguintes (DIVERIO; MENEZES, 2011):

- 1) múltiplas ocorrências de um mesmo trecho de programa, como a declaração de um identificador e suas referências de uso (problema análogo ao da linguagem $\{ w cw \mid w \text{ é palavra de } \{a, b\}^* \}$, a qual não é livre do contexto);
- 2) alguns casos de validação de expressões com variáveis de tipos diferentes;
- 3) a associação de um significado (semântica) de um trecho de programa, que dependeria da análise de um conjunto de informações (dependentes de contextos), como identificadores, ambientes, tipos de dados, localização, sequências de operações etc.

Por outro lado, prezado(a) estudante, para algumas linguagens de programação, a classe das linguagens livres do contexto é excessiva, e a classe das linguagens regulares, insuficiente. Por exemplo, o formalismo autômato com pilha possui a facilidade de não determinismo.

Entretanto, se a **linguagem** pode ser denotada por um autômato com pilha determinístico (ou seja, se é uma linguagem livre do contexto determinístico), então, é possível **implementar** (facilmente) um reconhecedor com tempo de processamento proporcional a $2n$ (n é o tamanho da entrada), o que é muito mais eficiente que o melhor **algoritmo** conhecido para as linguagens livres do contexto.

De qualquer forma, tanto para linguagens artificiais como para linguagens naturais, o estudo da classe das linguagens livres do contexto tem sido de especial interesse, pois elas permitem uma representação simples da sintaxe, adequada tanto para a estruturação formal como para a análise computacional.

Entretanto, estudante, note que o estudo dessa classe tem mostrado **problemas não solucionáveis**, como, por exemplo:

- 1) determinar se uma gramática livre do contexto é ambígua (ou seja, se existem duas ou mais árvores de derivação distintas para uma mesma palavra);
- 2) não existe um algoritmo que verifique a igualdade de duas linguagens livres do contexto, o que dificulta a otimização e o teste de processadores de linguagens.

Portanto, dependendo da linguagem e dos objetivos do trabalho, alguns estudos específicos, eventualmente fora da hierarquia de Chomsky, são recomendados ou necessários.

Nesse sentido, você concorda que vale a pena mencionarmos o conceito de **gramáticas de grafos**, tendo em vista que esse é o principal conceito análogo ao das gramáticas de Chomsky? As gramáticas de grafos têm duas ideias fundamentais: regras de produção são pares (mas de grafos), e uma derivação é a substituição de um subgrafo de acordo com uma regra de produção.

Portanto, as gramáticas de grafos constituem um **caso particular das gramáticas categoriais** (nenhum conceito de teoria das categorias é formalmente introduzido nessa publicação). A ideia básica é substituir palavras por grafos no conceito de gramática de Chomsky.

Mas, observe que, na realidade, a **noção categorial de gramática** é bem mais **expressiva**, pois:

- 1) gramáticas categoriais podem ser usadas sobre palavras, grafos, conjuntos parcialmente ordenados, redes, autômatos, máquinas, linguagens de programação e outros tipos de objetos, desde que sejam satisfeitas determinadas condições;
- 2) as derivações (aplicações de regras de uma gramática) são generalizadas.

Para facilitar a implementação dos conceitos apresentados, vamos utilizar a ilustração do jogo de videogame Pac-Man, amplamente conhecido mundialmente, principalmente nos anos 1990. Vejamos o exemplo.

O **jogo Pac-Man** (simplificado) possui um tabuleiro juntamente com algumas entidades especiais, como um Pac-Man e dois conjuntos, um de fantasmas e outro de maçãs.

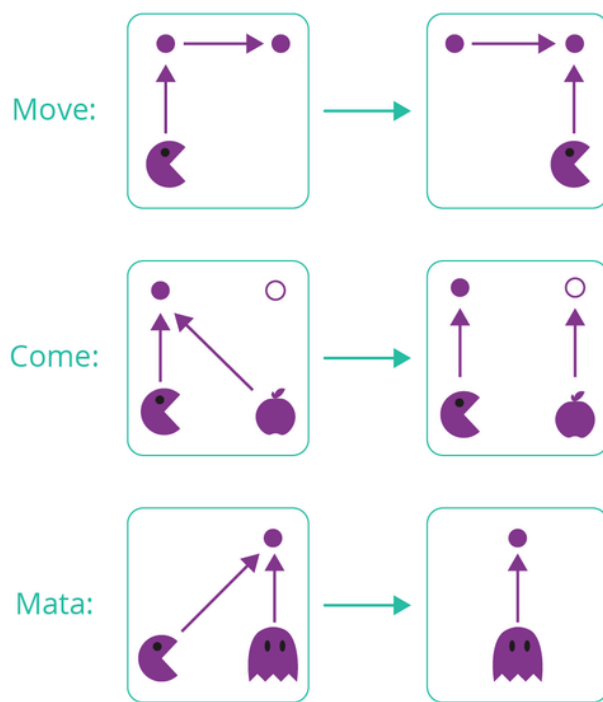
A Figura 2.5 **ilustra um grafo** representando um estado do sistema no qual, para facilitar a visualização, as entidades Pac-Man, fantasmas e maçãs são representadas por nodos com simbologia própria. Na figura, vê-se que:

- 1) os nodos pretos representam os lugares do tabuleiro, e as correspondentes arestas, os caminhos possíveis entre dois lugares;
- 2) para as entidades PacMan, fantasmas e maçãs, os correspondentes arcos denotam o seu posicionamento no tabuleiro;
- 3) uma maçã cujo nodo destino é o nodo branco denota uma maçã já comida;
- 4) a aresta numerada com origem e destino no nodo branco identifica a fase em que se encontra o jogo (no caso, a segunda fase).

A Figura 2.4 **ilustra** as seguintes **regras de produção** :

- 1) move: o Pac-Man move-se para uma casa adjacente;
- 2) come: o Pac-Man come a maçã. A maçã comida é posicionada no nodo branco;
- 3) mata: o fantasma mata o Pac-Man. O Pac-Man é excluído.

Por fim, a Figura 2.5 **ilustra um possível grafo** resultante da aplicação da regra “move” ao grafo ilustrado na Figura 2.4. Observe que, no grafo transformado, o PacMan encontra-se em um lugar onde existe um fantasma. Assim, a próxima produção a ser aplicada pode ser “morre” ou “move”.

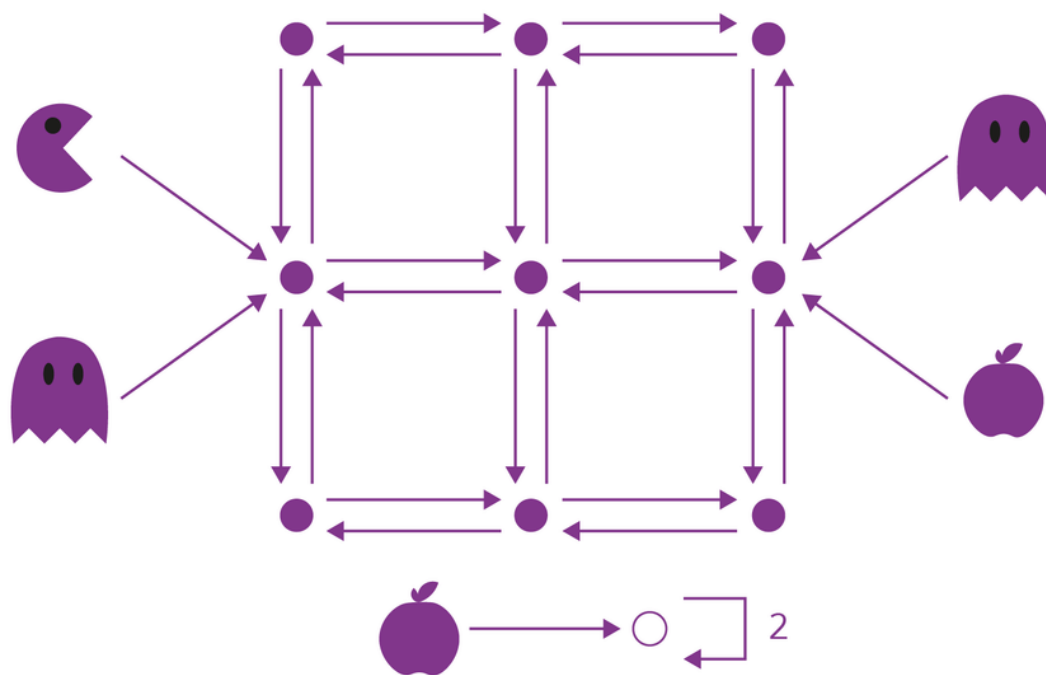


© VG Educacional

Figura 2.5 – Gramática de grafos: regras de produção do Pac-Man

Fonte: Menezes (2015, p. 240).

#PraCegoVer : na ilustração, é apresentado um grafo ilustrativo do jogo Pac-Man, no qual são apresentadas três possíveis decisões do personagem (Pac-Man), são elas: matar o adversário, comer o alimento e se mover para frente, para trás, para a esquerda e para a direita.



© VIG Educacional

Figura 2.6 – Grafo resultante da aplicação da regra “move”

Fonte: Menezes (2015, p. 241).

#PraCegoVer : na ilustração, é apresentado um grafo ilustrativo do jogo Pac-Man, no qual são apresentadas as arestas com os alimentos para o personagem (Pac-Man) comer e, respectivamente, os movimento assumidos pelo personagem Pac-Man, isto é, mover-se em direção ao alimento.

Por fim, o sucesso por trás do jogo PacMan, muito comum nas décadas de 1980 e 1990, reforça a aplicação prática dos conceitos aqui estudados, em especial, a hierarquia de Chomsky, que propiciou a criação de um dos jogos mais jogados no mundo, além de ser fundamental para a teoria da computação.

atividade

Atividade

Estudante, é possível perceber, nas gramáticas lineares, uma forte restrição no formato das produções, no caso, o lado esquerdo possui exatamente uma variável, já o lado direito de uma produção é constituído por, no máximo, uma variável. Adicionalmente, essa variável, se existir, sempre antecede (linear à esquerda) ou sucede (linear à direita) qualquer subpalavra.

Caso tivéssemos a gramática regular: $(a+b)^*(aa+bb)$, quais gramáticas regulares seriam geradas?

Material Complementar



FILME

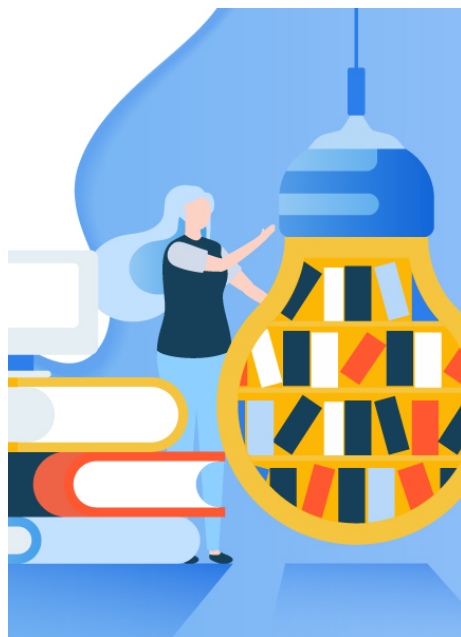
Ex Machina

Ano : 2014

Comentário : No filme, é possível verificar a importância das gramáticas regulares no aprendizado de máquina (*machine learning*), bem como sua fundamental utilização nos dias atuais, nos estudos sobre inteligência artificial. Confira o filme e verifique como os assuntos tratados ao longo da unidade são fundamentais para a evolução da computação nos próximos anos.

Para conhecer mais sobre o filme, acesse o *trailer* disponível em:

TRAILER



LIVRO

Estrutura de dados: algoritmos, análise da complexidade e implementações em Java e C/C++

Editora : Pearson Prentice Hall

Autor : Ana F. Gomes Ascencio

ISBN : 8576058812

Comentário : O livro apresenta exemplos práticos com implementação em Java e C/C++ dos conceitos aqui estudados, isto é, expressões regulares e gramáticas regulares, bem como o desenvolvimento do jogo Pac-Man. Vale a pena conferir a fim de aprimorar os conhecimentos teóricos e práticos de programação, com o objetivo de implementar os conceitos vistos ao longo da unidade.

Querido(a) estudante!

Chegamos ao fim da unidade, e, após a leitura, podemos concluir que o estudo das expressões regulares e das gramáticas regulares é **fundamental** para os campos mais avançados da computação, em especial, o campo da aprendizagem de máquina (*machine learning*).

Destaca-se, também, que, desde os **primórdios da computação** , tem-se verificado a importância de se manter sempre em constante evolução a criação de expressões e gramáticas regulares, tendo em vista o avanço dos desafios e da evolução computacional. Foi bom tê-lo(a) até aqui conosco! Até breve!

Referências



AULA 14 – Gramáticas regulares (GLD, GLE, GLUD e GLUE). [S. l.: s. n.], 2020. 1 vídeo (13 min 32 s). Publicado pelo canal Prof. Francisco Anderson.

Disponível em:

<https://www.youtube.com/watch?v=X1Bd1-sjDY0> . Acesso em: 23 maio 2021.

DIVERIO, T. A.; MENEZES, P. F. B. **Teoria da computação** : máquinas universais e

computabilidade. 3. ed. São Paulo: Bookman, 2011.

EX_Machina – Trailer Oficial Legendado (Portugal) HD. [S. l.: s. n.], 2015. 1 vídeo (2 min 35 s). Publicado pelo canal Universal Pictures Portugal. Disponível em:

<https://www.youtube.com/watch?v=54nzsdLPs9I> . Acesso em: 23 maio 2021.

MENEZES, P. F. B. **Linguagens formais e autômatos** . Porto Alegre: Grupo A, 2015.