



UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

COORDENAÇÃO DO CURSO EM ENGENHARIA DA COMPUTAÇÃO

LINGUAGENS FORMAIS E AUTÔMATOS
ANTÔNIO FIALHO NETO
ISABEL SILVA DE ARAÚJO
JUSTINO FELIPE NUNES

PROF. DR. THALES VALENTE

APLICAÇÃO DE GRAMÁTICA REGULARES NA CRIAÇÃO DE TOKENS DE SEGURANÇA

SÃO LUÍS - MA
2024



SÚMARIO

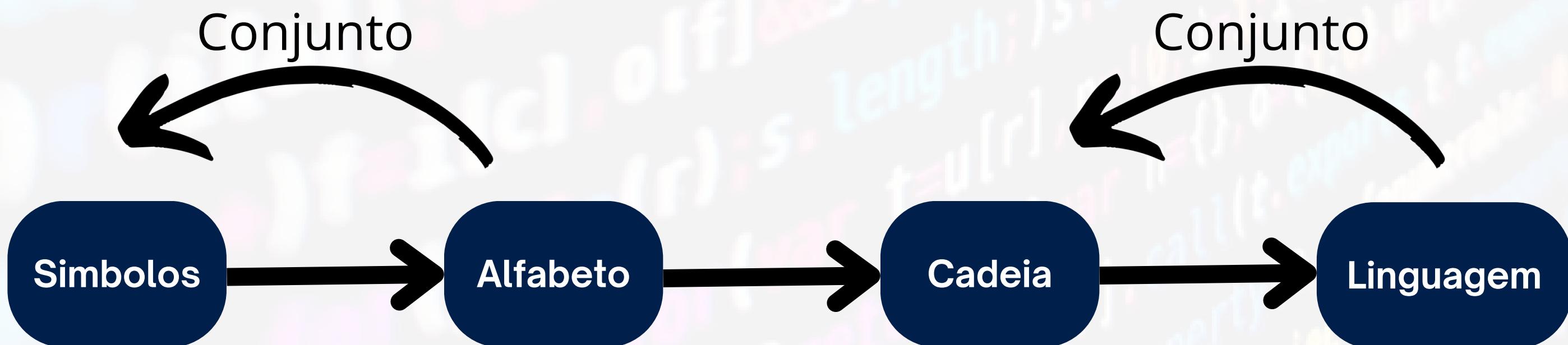
- 1. INTRODUÇÃO**
 - 2. IMPLEMENTAÇÃO**
 - 2.1 GRAMÁTICAS**
 - 2.2 RECONHECEDORES**
 - 3. LINGUAGENS**
 - 4. APLICAÇÃO**
 - 5. CONCLUSÃO**
- REFERÊNCIAS**



1. INTRODUÇÃO

1. INTRODUÇÃO

- O que é uma Linguagem Formal?
 - É um conjunto, finito ou infinito, de cadeias de comprimento finito, formadas pela concatenação de elementos de um alfabeto finito e não-vazio.



1. INTRODUÇÃO

- Unidades Básicas da Linguagem.
 - Símbolos: Entidades básicas e indivisíveis. Ex: a, 5, 0, 1, ab (Dependendo do contexto) - (a,b,c... ou 0,1)
 - Alfabetos: Conjuntos finitos não-vazios de símbolos. Ex: {a,b,c}, {0,1}, {ab,cd} - (Σ , Γ , Δ ,...)
 - Cadeias: Sequencias finitas de símbolos de um alfabeto. Ex: abc, 0011, abcd - (r, s, t,...ou a, β , γ ,...)



1. INTRODUÇÃO

- Unidades Básicas da Linguagem.
 - Ex: A linguagem de todas as cadeias que consistem em n valores 0 seguidos por n valores 1, $n \geq 0$:
 - Símbolo: 0,1
 - Alfabeto: $\Sigma = \{0,1\}$
 - Cadeias: $a = \{\epsilon, 01, 0011, 000111, \dots\}$



1. INTRODUÇÃO

- Concatenação:

- Operação associativa binária realizada sobre duas cadeias α e β (elementares ou não) que resulta em uma nova cadeia, formada pela juxtaposição ordenada dos símbolos.
- Ex: $\Sigma = \{a, b, c, d\}$, $\alpha = abc$, $\beta = dbaca$: $\alpha\beta = abcd\beta$, $\beta\alpha = dbacaabc$. OBS: X e Y são Linguagens.



$$XY = \{xy \mid x \in X, y \in Y\}$$

1. INTRODUÇÃO

- Propriedades:

- **Concatenação;**
- **Fechamento Reflexivo e transitivo;**
- **Fechamento transitivo;**
- **Complementação;**
- **Reversão**
- **Prefixo/Sufixo Próprio**
- **Quociente;**
- **Substituição.**





2. IMPLEMENTAÇÃO

2. IMPLEMENTAÇÃO

- Representação finita de linguagens:
 - Gramáticas
 - Reconhecedores
 - Enumeradores



2. IMPLEMENTAÇÃO

- Representação finita de linguagens:
 - Gramáticas: Especificações finitas de dispositivos de geração de cadeias ou dispositivos geradores de cadeias.



2. IMPLEMENTAÇÃO

- Representação finita de linguagens:
 - Reconhecedores: Correspondem a especificações finitas de dispositivos de aceitação de cadeias (Autômatos) ou dispositivos reconhecedores de cadeias.



2. IMPLEMENTAÇÃO

- Representação finita de linguagens:
 - Enumeradores: são listas explícitas e completas de todas as cadeias que pertencem a uma linguagem.





2.1 GRAMÁTICAS

2.1 GRAMÁTICAS

- Gramáticas:

- Formalmente é umas quadrupla:

$$G = (V, \Sigma, P, S)$$

- **V: Vocabulário - símbolos;**
 - **Σ : Símbolos terminais, é o alfabeto.**
 - **P: Produções ou regras de substituição ;**
 - **S: Raiz da gramática, $S \in V$.**
 - **$N = V - \Sigma$: Símbolos não terminais, classes sintáticas.**



2.1 GRAMÁTICAS

- Exemplo:

- $G_1 = (V_1, \Sigma_1, P_1, S)$, com:
 - $V_1 = \{a,b,c, S, B,C\}$
 - $\Sigma_1 = \{a,b,c\}$
 - $N_1 = \{S, A\}$
 - $P_1 = \{S \rightarrow aSbb, S \rightarrow A, A \rightarrow c, A \rightarrow \epsilon\}$
- Cadeias:
 - $S \rightarrow aSbb \rightarrow aAbb \rightarrow a\epsilon bb \rightarrow abb$
 - $S \rightarrow aSbb \rightarrow aAbb \rightarrow acbb$
 - $S \rightarrow aSbb \rightarrow aaSbbbb \rightarrow aa\epsilon bbbb \rightarrow aabb$
 - $S \rightarrow aSbb \rightarrow aaSbbbb \rightarrow aaAbbbb \rightarrow aacbbbb$



2.1 GRAMÁTICAS

- Gramáticas:

- Forma sentencial: é qualquer cadeia obtida pela aplicação recorrente das seguintes regras de substituição:
 - A raiz S é uma forma sentencial.
 - Seja $\alpha\beta$ uma forma sentencial, substituindo a ocorrência de ρ por γ , produz uma nova forma sentencial $\alpha\gamma\beta$ ou derivação direta.
- Derivação;
- Derivação Não trivial;
- Sentença.



2.1 GRAMÁTICAS

- Linguagem definida pela Gramática:
 - É o conjunto de todas as sentenças w geradas por uma gramática G.
 - Exemplo:
 - $G2 = (V2, \Sigma2, P2, S)$:
 - $V2 = \{a,b,c, S, B, C\}$
 - $\Sigma2 = \{a,b,c\}$
 - $N2= \{S, B, C\}$
 - $P2 = \{S \rightarrow aSBC, S \rightarrow abC, CB \rightarrow BC, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$
 - $L1 (G2): \{a^n b^n c^n | n \geq 1\}$ Código de Verificação.
- Código:
<https://colab.research.google.com/drive/1tt1K7kpTsz893dwbrSNI7A465Z1T6ErB?authuser=0#scrollTo=IDBCUYa1wcIZ>



2.1 GRAMÁTICAS

- Linguagem definida pela Gramática:
 - $P_2 = \{S \rightarrow aSBC, S \rightarrow abC, CB \rightarrow BC, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$
 - $S \rightarrow aSBC \rightarrow aabC \rightarrow aabc$
 - $S \rightarrow aSBC \rightarrow aabCBC \rightarrow aabBCC \rightarrow aabbCC \rightarrow aabbcC \rightarrow aabbcc$



2.1 GRAMÁTICAS

- Gramáticas equivalentes:

- Quando uma mesma linguagem pode ser definida por duas ou mais gramáticas.

- $G1 = (\{S,A,B\}, \{0,1\}, P1, S)$

- $P1 = \{S \rightarrow 0S \mid 1A \mid \epsilon, A \rightarrow 0B \mid 1A \mid \epsilon, B \rightarrow 0B \mid 1A \mid \epsilon\}$

-

- $G2 = (\{S,X\}, \{0,1\}, P2, S)$

- $P2 = \{S \rightarrow 0S \mid 1X \mid \epsilon, X \rightarrow 0S \mid 1X \mid \epsilon\}$

- $L = \{0^n 1^m \mid n, m \geq 0\}$





2.2 RECONHECEDORES

2.2 RECONHECEDORES

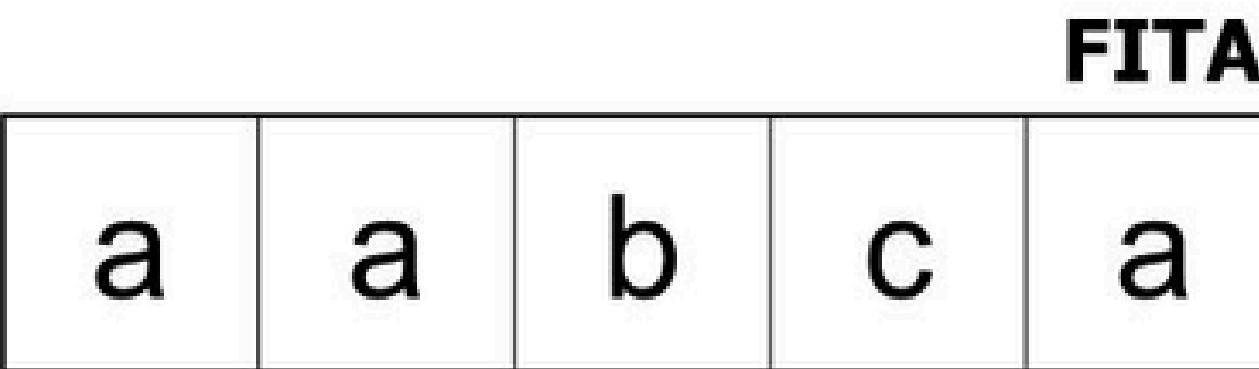
- São sistemas formais capazes de aceitar todas as sentenças que pertençam a uma determinada linguagem, rejeitando todas as demais.
- Quatro componentes:
 - Fita de entrada;
 - Cursor;
 - Maquina de estados;
 - Memória auxiliar.



2.2 RECONHECEDORES

- Fita de entrada:

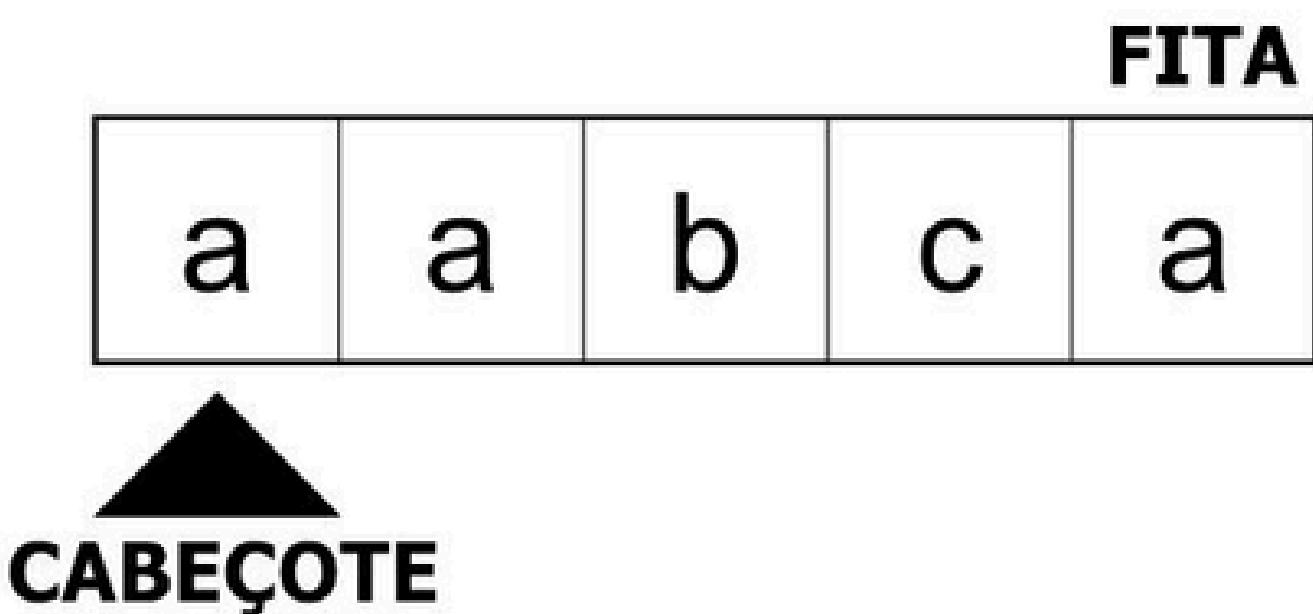
- Contém a cadeia a ser analisada pelo reconhecedor.
- A cadeia é disposta da esquerda para a direita, sendo o primeiro símbolo colocado na célula mais à esquerda da fita.



2.2 RECONHECEDORES

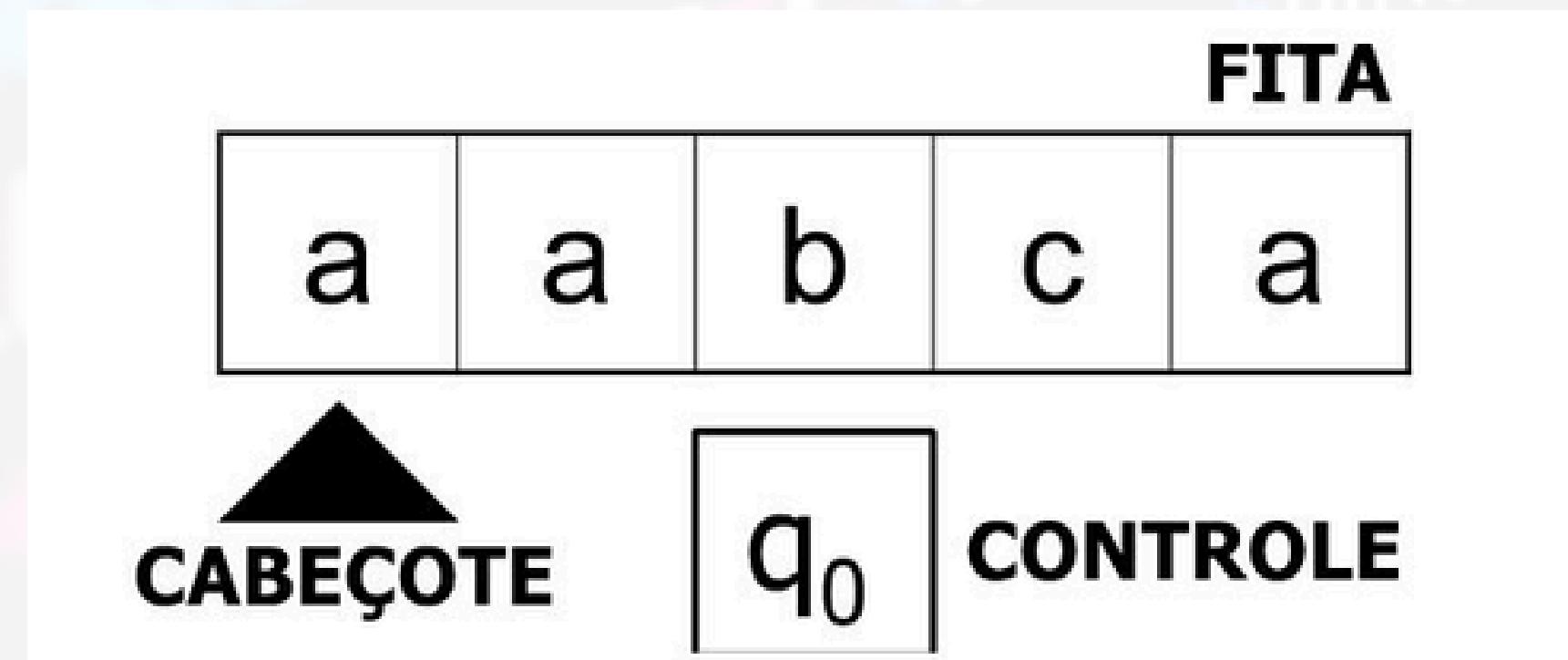
- Cursor:

- A leitura é feita através de um cabeçote de acesso, ou cursor, o qual sempre aponta para o símbolo da cadeia a ser processado.



2.2 RECONHECEDORES

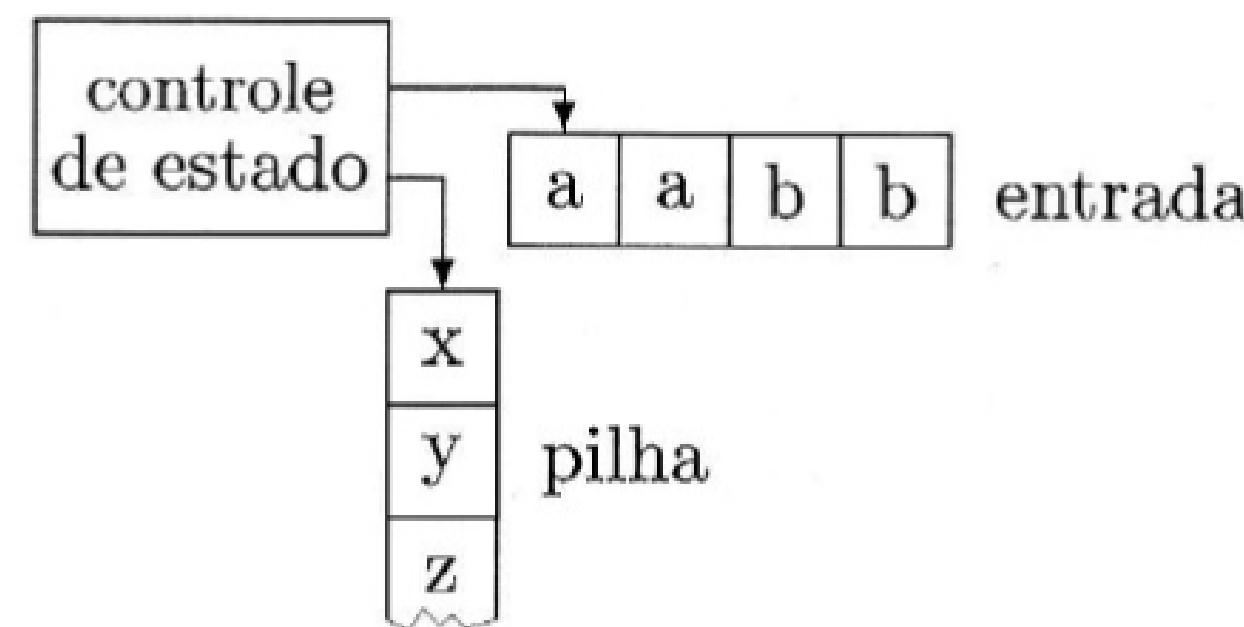
- Maquina de estados:
 - Funciona como um controlador central do reconhecedor, e contem uma coleção finita de estados.



2.2 RECONHECEDORES

- **Memoria auxiliar:**

- É opcional, e torna-se necessária apenas em reconhecedores de linguagens que apresentam uma certa complexidade.
- Assume a forma de uma estrutura de dados de baixa complexidade, como, por exemplo, uma pilha.

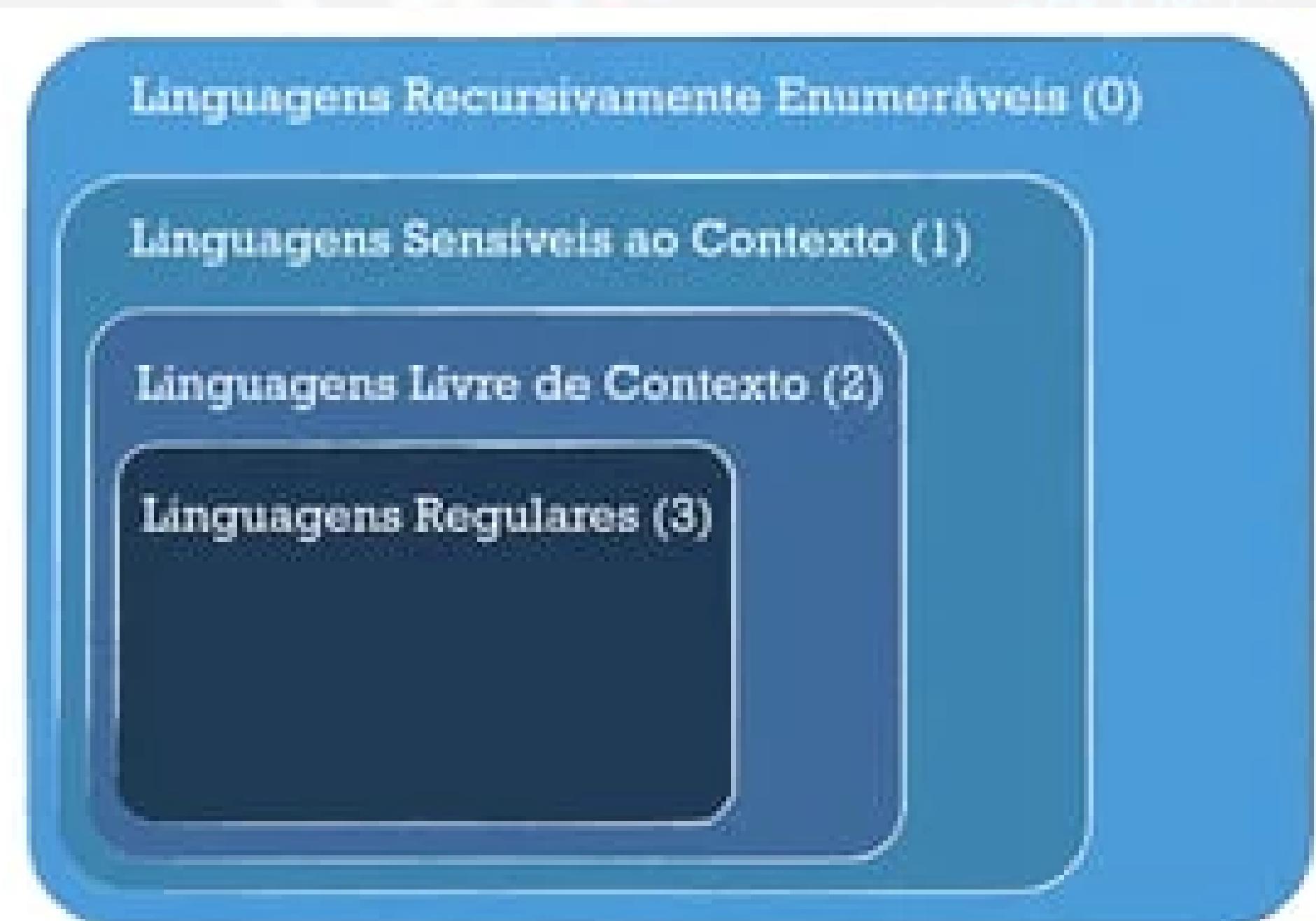




3. LINGUAGENS

3. LINGUAGENS

- Classificação das linguagens:
 - Conhecida como Hierarquia de Chomsky.



3. LINGUAGENS

- Linguagens irrestritas ou Tipo 0:
 - As produções não têm restrições específicas quanto ao seu formato, exceto pelo fato de que o lado esquerdo sempre contém pelo menos um símbolo não-terminal.
 - $\alpha \in V^*NV^*$
 - $\beta \in V^*$
 - Não se exige a validade de qualquer relação restritiva entre $|\beta|$ e $|\alpha|$



3. LINGUAGENS

- Linguagens Sensíveis ao Contexto ou Tipo 1:
 - Produções apresentam o comprimento da cadeia do lado direito igual ou maior do que o comprimento da cadeia do lado esquerdo.
 - $\alpha \in V^*NV^*$
 - $\beta \in V^*$
 - $|\beta| \geq |\alpha|$



3. LINGUAGENS

- Linguagens Livres de Contexto ou Tipo 2:
 - Cujas produções têm apenas um símbolo não-terminal no lado esquerdo e uma combinação de símbolos terminais e não-terminais no lado direito.
 - Toda G do tipo 3 também é do tipo 2.
 - $\alpha \in N$
 - $\beta \in V^*$



3. LINGUAGENS

- Linguagens Regulares ou Tipo 3:
 - Aquelas geradas por gramáticas lineares à direita ou por gramáticas lineares à esquerda.
 - Direita: caso todas suas regras de produção obedecam:
 - $\alpha \in N$
 - $\beta \in \Sigma, \beta \in N, \beta \in \Sigma^N$ ou $\beta = \epsilon$, de forma não exclusiva.



3. LINGUAGENS

- Linguagens Regulares ou Tipo 3:
 - Aquelas geradas por gramáticas lineares à direita ou por gramáticas lineares à esquerda.
 - Esquerda: caso todas suas regras de produção obedecam:
 - $\alpha \in N$
 - $\beta \in \Sigma$, $\beta \in N$, $\beta \in N\Sigma$ ou $\beta = \epsilon$, de forma não exclusiva.



3. LINGUAGENS

- Exemplo Linear a Esquerda:

- $V = \{S\}$
- $\Sigma = \{0,1\}$
- $S = S$
- $P = \{S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$
- Derivação de cadeia **1101**
- $S \rightarrow S1 \rightarrow S01 \rightarrow S101 \rightarrow 1101$

- Exemplo Linear a Direita:

- $V = \{S\}$
- $\Sigma = \{0,1\}$
- $S = S$
- $P = \{S \rightarrow 0S, S \rightarrow 1S, S \rightarrow 0, S \rightarrow 1\}$
- Derivação de cadeia **1101**
- $S \rightarrow 1S \rightarrow 11S \rightarrow 110S \rightarrow 1101$



3. LINGUAGENS

- Exemplo Linguagens Regulares ou Tipo 3(linear a direita):

- $G = (V, \Sigma, P, S)$
- $V = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$
- $\Sigma = \{a, c, g, t\}$
- $S = S_1$

atga
atgg
atta
aaga
cgag

$$\begin{aligned}P = & \{ S_1 \rightarrow aS_2 \mid cS_3 \\& S_2 \rightarrow tS_4 \mid aS_5 \\& S_4 \rightarrow gS_6 \mid tS_7 \\& S_6 \rightarrow a \mid g\end{aligned}$$



3. LINGUAGENS

- Exemplo Linguagens Regulares ou Tipo 3(linear a direita):

- $G = (V, \Sigma, P, S)$
- $V = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$
- $\Sigma = \{a, c, g, t\}$
- $S = S_1$

atga

atgg

atta

aaga

cgag

$P = \{ S_1 \rightarrow aS_2 \mid cS_3 \}$

$S_2 \rightarrow tS_4 \mid aS_5$

$S_4 \rightarrow gS_6 \mid tS_7$

$S_6 \rightarrow a \mid g$

$S_7 \rightarrow a$

$S_5 \rightarrow gS_8$

$S_8 \rightarrow a$

$S_3 \rightarrow gS_9$

$S_9 \rightarrow aS_{10}$

$S_{10} \rightarrow g\}$





4. APLICAÇÃO

4. APLICAÇÕES

- Criação de tokens de segurança utilizando como base gramáticas regulares.
 - Cenário:
 - Os tokens devem ter 8 dígitos.
 - No máximo só dois tipos de letras.
 - Podem ser constituídos apenas por números, apenas por caracteres ou por uma combinação dos dois. No entanto, o primeiro caractere sempre deve ser precedido pelo segundo caractere.



4. APLICAÇÕES

- Criação de tokens de segurança utilizando como base gramáticas regulares.
 - Demonstrando a Gramática geral:
 - $G = \{V \Sigma S P\}$
 - $\Sigma = (L1, L2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$
 - $V = (S1, S2, S8)$
 - $S = S1$
 - $P = \{$
 - $S1 \rightarrow [1l]S2 \mid 0S1 \mid 1S1 \mid 2S1 \mid 3S1 \mid 4S1 \mid 5S1 \mid 6S1 \mid 7S1 \mid 8S1 \mid 9S1 \mid 0S8 \mid 1S8 \mid 2S8 \mid 3S8 \mid 4S8 \mid 5S8 \mid 6S8 \mid 7S8 \mid 8S8 \mid 9S8$
 - $S2 \rightarrow [2l]S8 \mid [2l]S1 \mid [2l]$
 - $S8 \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 - }

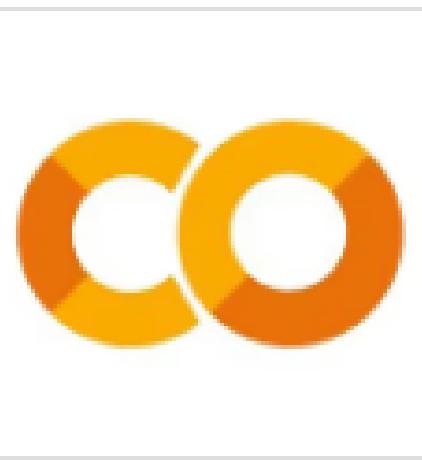




4. APLICAÇÕES

- Criação de tokens de segurança utilizando como base gramaticas regulares.

Código



Google Colaboratory
co google.com





5. CONCLUSÃO

5. CONCLUSÃO

A conclusão desse trabalho mostra uma aplicabilidade direta dos conceitos de gramáticas regulares na geração de tokens. Aqui, os tokens são construídos com base em regras específicas, que definem não apenas os caracteres que podem ser utilizados, mas também as possíveis combinações entre eles.



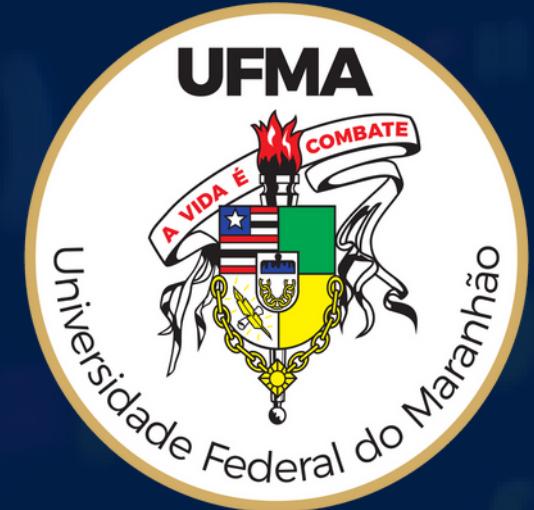


REFERÊNCIAS

REFERÊNCIAS

- RAMOS, Marcus V. M. Linguagens formais: teoria, modelagem e implementação. 1a ed. Porto Alegre: Bookman, 2009.
- MENEZES, Paulo B. Linguagens formais e autômatos. 6a ed. Porto Alegre: Bookman, 2011.
- VIEIRA, Jose N. Introdução aos Fundamentos da Computação: Linguagens e Máquinas. 1a ed. Rio de Janeiro: Thompson, 2006.
- PRADO, Simone das G. D. Apostila 02: Linguagens regulares. 1a ed. Bauru: UNESP, 2011.





DÚVIDAS ?



OBRIGADO !