Módulo 1 - Conceitos Básicos - Expressões Regulares

Ricardo Vargas Dorneles

2 de março de 2023

Ementa

- Apresentação da terminologia e de conceitos básicos relativos a linguagens formais.
- Estudo e análise dos formalismos clássicos para especificação de linguagens.
- Utilização de formalismos para especificação de linguagens.
- Estudo e análise dos algoritmos para reconhecimento de linguagens.
- Aplicação de algoritmos a problemas de reconhecimento.

Objetivo

- Instrumentalizar o aluno para que ele possa implementar um analisador léxico-sintático para uma linguagem de programação simples.

Conteúdo Programado

1. Introdução: Alfabeto, Sentença, Linguagem, Geração e Reconhecimento de Linguagens, Gramáticas, Derivação
2. Linguagens Regulares

2.1 Gramáticas regulares

2.2 Expressões regulares

2.3 Autômato finito

2.3.1 Determinístico

2.3.2 Não determinístico

2.3.3 Com movimentos vazios

2.4 Conversão entre autômatos, gramáticas e expressões regulares

3. Introdução à Compilação
4. Análise Léxica

Conteúdo Programado

5. Linguagens Livres do Contexto

5.1 Gramática livre de contexto

5.2 Árvore de derivação

5.3 Ambigüidade

5.4 Operações com gramáticas (fatoração, eliminação de recursão etc.)

5.5 Autômato de pilha

5.6 Conversão entre autômatos de pilha e gramáticas

6. Análise Sintática

6.1 Análise Sintática Descendente Recursiva

6.2 Análise Sintática Descendente Preditiva

6.3 Análise Sintática Top Down

6.4 Análise Sintática Bottom Up

Metodologia

- As aulas serão teórico-expositivas com a resolução de diversos exercícios práticos durante as aulas. O professor orientará o aluno no desenvolvimento, ao longo do semestre, do projeto de um analisador léxico e sintático de alguma linguagem ou subconjunto de linguagem.

Avaliação

- O desempenho do aluno será avaliado no decorrer do semestre através de duas provas (P1 e P2), ambas com peso 1, e um trabalho prático (T1), também com peso 1 (média harmônica).
- Será considerado aprovado o aluno que atingir, no mínimo, média 6 (seis) nas atividades de avaliação e freqüência igual ou superior a 75%.
  - Se esta média não for alcançada, o aluno poderá recuperar uma das duas provas, P1 ou P2, na data prevista no cronograma.
  - A nota obtida na recuperação substituirá a nota anterior correspondente. Notas de trabalhos não serão recuperados.
- O conceito final será expresso segundo as normas regimentais da Instituição.

Bibliografia

- Básica
  - AHO, A.V. Compiladores - Princípios, Técnicas e Ferramentas. Rio de Janeiro: Livros Técnicos e Científicos, 1995.
  - MENEZES, P.B. Linguagens Formais e Autômatos. Porto Alegre: Editora Sagra-Luzzatto. 4ª ed. 2001.
- Bibliografia Complementar
  - HOPCROFT, John E.; ULLMAN, Jeffrey D. Introduction to automata theory, languages, and computation. Massachusetts: Addison-Wesley, 1979. 418 p.
  - AHO, A. V. et alii. Compilers principles, techniques and tools. New Jersey: Prentice-Hall, 1988.
  - AHO, A. V. and ULLMAN, J. The theory of parsing, translation and compiling. New Jersey: Prentice-Hall, 1972.
  - APPEL, A.W. Modern Compiler Implementation in C. Cambridge: Cambridge University Press. 1998.
  - WIRTH, N. Compiler Construction. Englewood Cliffs: Addison-Wesley, 1996.

Tipos de tradutores mais usados:

- Montador : linguagem assembly para linguagem de máquina
- Compilador : linguagem de alto nível para linguagem assembly ou de máquina
- Interpretador : Traduz e executa simultaneamente

Voltar para o índice

Fonte $\rightarrow$	Compilador	$\rightarrow$ Programa Objeto
		$\rightarrow$ Mensagens de Erro
		$\rightarrow$ Tabelas Auxiliares

Tabela: Compilador

Voltar para o índice

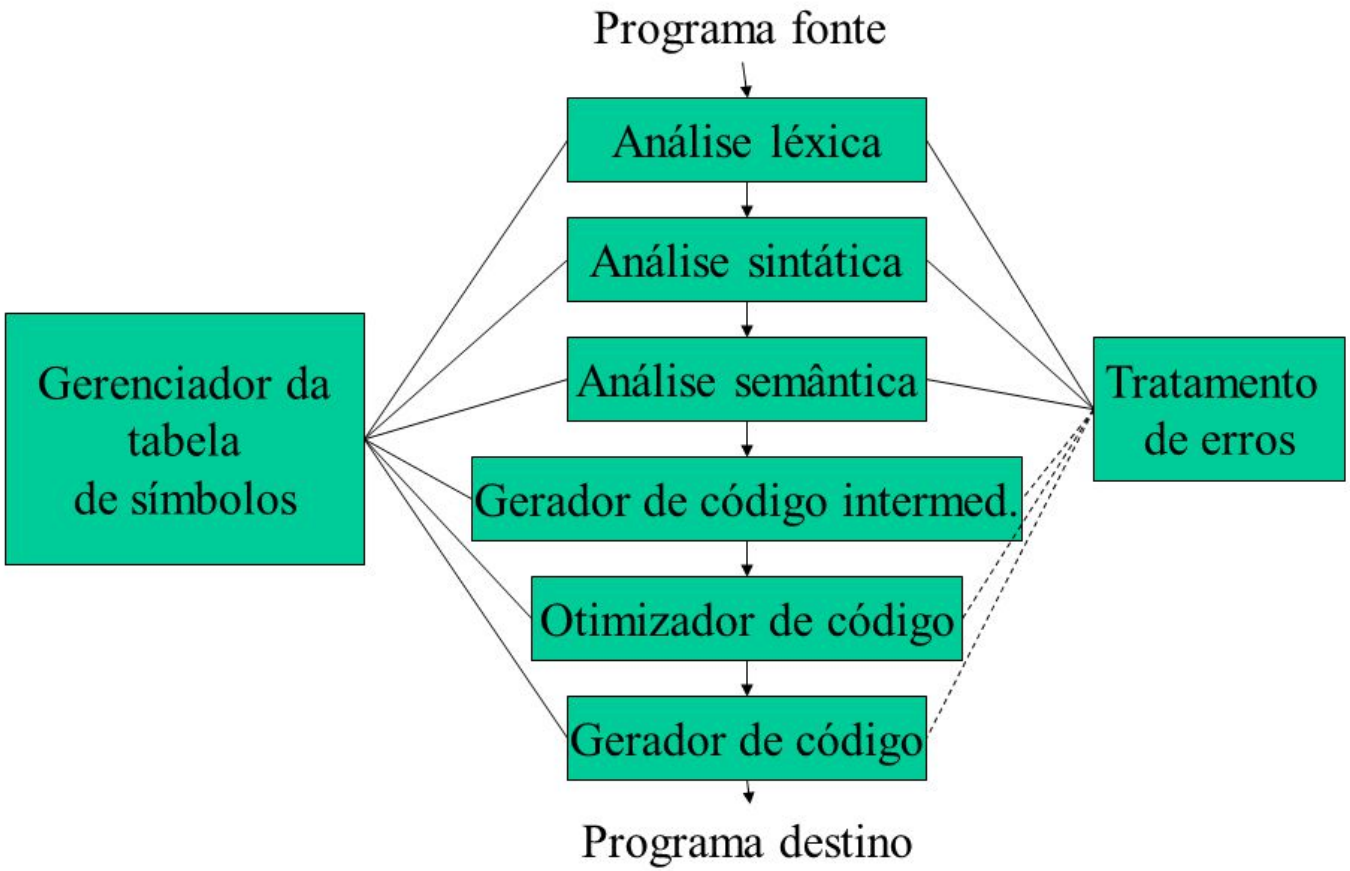
Modelo Análise x Síntese

A compilação pode ser dividida em duas partes, com tarefas distintas em cada uma : análise e síntese

- Análise:
  - Reconhece cada uma das partes componentes do programa fonte e sua estrutura
  - Cria uma representação interna
  - A sintaxe a ser analisada é descrita através de gramáticas ou autômatos
- Síntese
  - Gera um programa objeto a partir da representação interna
  - Depende da semântica do programa

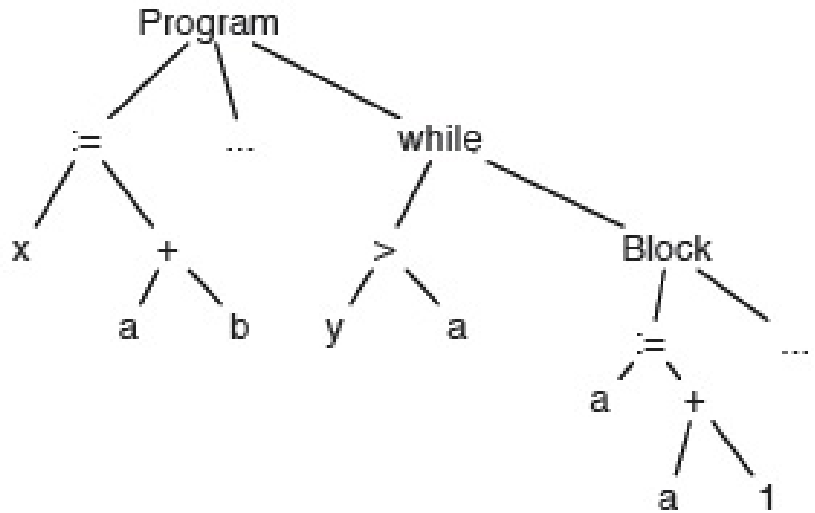
Voltar para o índice

Fases de um compilador (II)



- Exemplo de representação interna : Árvore Sintática

```
x := a + b;
y := a * b;
while (y > a) {
    a := a + 1;
    x := a + b
}
```



Voltar para o índice

- Sintaxe - descreve a forma de um comando (forma de escrita)
- Semântica - descreve o significado de um comando (o que o comando faz)

Voltar para o índice

- Análise do Programa Fonte é feita em três fases:
  - Análise **léxica**
  - Análise **sintática**
  - Análise **semântica**

Voltar para o índice

Análise léxica:

- Lê seqüências de caracteres que formam o programa fonte
- Agrupa os caracteres em tokens que são as unidades léxicas, os átomos que compõem o programa fonte (ex: números, identificadores, operadores)
- Elimina os caracteres não significativos, como o fim de linha e espaços em branco

Voltar para o índice

Ex:

elemento := inicial + fator * 60

$\Downarrow$

Análise léxica

$\Downarrow$

(id,1)(op,:=)(id,2)(op,+)(id,3)(op,*)(cte,60)

Voltar para o índice

- O analisador léxico normalmente trabalha "por demanda".
- As informações obtidas na análise léxica são armazenadas na tabela de símbolos.

Ex: Identificador
Tipo
Valor
Linha onde foi definido

Voltar para o índice

Análise Sintática

- Agrupa os tokens em sentenças da linguagem
- Verifica se os tokens formam sentenças corretas da linguagem de acordo com as regras de sintaxe da linguagem
  - Atrib $\rightarrow$ id = exp
  - Exp $\rightarrow$ exp + exp
  - Exp $\rightarrow$ cte | id
- Ex: x3 = y + 3

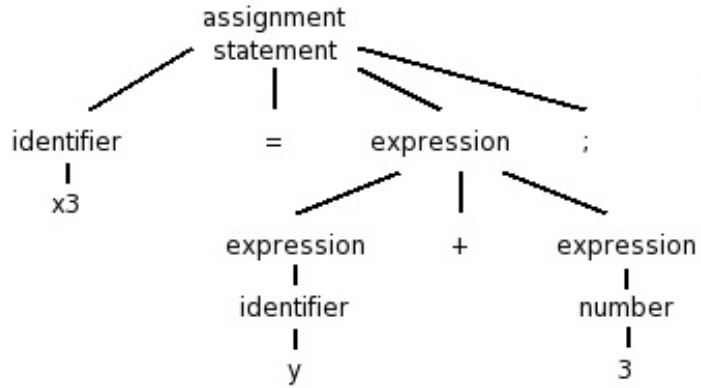


Figura: Árvore de derivação

Análise Semântica

- Efetua verificações que não podem ser feitas pela análise sintática
  - Verifica se as combinações entre operandos e operadores são válidas (posso aplicar o operador % a float?)
  - Verificação de tipo (posso atribuir um float a um char?)
  - Declarações (duplicadas, variáveis ainda não declaradas...)
  - Alguma coisa sobre estruturas de controle (esse break está dentro de um switch, for ou while?, entradas duplicadas em um switch...)
  - Complementa a tabela de símbolos

Voltar para o índice

Fases de um compilador

Programa Fonte $\rightarrow$ An. Léxica $\rightarrow$ An. Sintática $\rightarrow$ An. Semântica $\rightarrow$

ANÁLISE

Ger, de cód. intermediário $\rightarrow$ Otimização de Código $\rightarrow$ Ger. de código Objeto

SÍNTESE

Voltar para o índice

Ferramentas para a geração de compiladores

- Em C:
  - Geradores de analisadores léxicos : LEX
  - Geradores de analisadores sintáticos : YACC
- Em Java:
  - Javacc : gerador de analisadores léxicos e sintáticos

Voltar para o índice

Termos Básicos

- **Vocabulário** ou **alfabeto** : É o conjunto de símbolos que podem ser utilizados na construção de sentenças. Ex: {0,1}, {a,b,c...z}
- **Sentença** : Qualquer seqüência finita composta de zero ou mais símbolos. '0', '01', '001'
- **Linguagem** : É um conjunto qualquer de sentenças.
  - {1,100,1000} por enumeração
  - {01,0011,000111,...} 0^n1^n - define-se regras

Voltar para o índice

Operações com Sentenças

- Concatenação ou produto: Se x='01'e y='233' então xy='01233'
- Potenciação ou exponenciação: X^n é a concatenação de n ocorrências de x. Ex: se x='01' então x^2='0101'
- Tamanho da Sentença: $|x|$, representa o número de elementos da seqüência x. Ex: $|'01'| = 2$. Obs: a sentença vazia tem tamanho zero e é representada por ϵ.

Voltar para o índice

Operações com Linguagens

- União: $L \cup M$ é a linguagem formada pelas sentenças de L e M
- Diferença :$L - M$: sentenças de L que não estão em M
- Concatenação : LM : sentenças geradas pela concatenação de sentenças de L com sentenças de M. LM=$\{xy \mid x \in L \text{ e } y \in M\}$
- Potenciação da Linguagem: L^n é a linguagem resultante da concatenação de n sentenças de L
- Fecho de linguagens: L^* é a linguagem resultante da concatenação de qualquer número de sentenças de L. $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup ...$
- Fecho positivo: L^+ é igual a L^* mas sem a sentença vazia

Voltar para o índice

1. Considere as seguintes sentenças:

x=a
y=ab
z=bcd

Apresente o resultado das operações a seguir sobre as sentenças:

a) xy =
b) yx =
c) $(xy)z =$
d) $y^2 =$
e) $z^3 =$
f) $z^0 =$
g) $|xy| =$
h) $|z^3| =$
i) $|z^2| =$
j) $|z^1| =$
k) $|z^0| =$

Voltar para o índice

2) Considere as seguintes linguagens:

A={a,b}
B={aa,ab,bb}
C={b,bb,bbbb}

Apresente o resultado das operações que seguem sobre as linguagens(por enumeração ou por regra):

a)$A \cup B =$
b)$B \cup C =$
c)A - B =
d)(A-B) -C =
e)$A^3 =$
f)$B^2 =$
g)$A^* =$
h)$C^* =$

Voltar para o índice

Notação Algébrica para Linguagens

Estas operações podem ser usadas para especificar linguagens.

Ex: $\{ 10^n \mid n>=0\} = \{1, 10, 100, 1000\}$
$\{ 0^n1^n \mid n>=0\} = \{\epsilon, 01, 0011\}$

A linguagem sobre V=$\{0,1\}$

$\{ x00y \mid x \text{ e } y \in V^*\}$ (qualquer sentença que pertença a V^* e que contenha ao menos uma seqüência 00)

Voltar para o índice

Componentes de Sentenças

- **Prefixo** de x : É obtido removendo-se 0 ou mais símbolos do final de x.

Ex.: x = 001010 prefixos: ϵ, 0, 00, 001, 0010, 00101, 001010

- **Sufixo** de x : obtido removendo-se 0 ou mais símbolos do início de x.

Ex.: x = 001010 sufixos: ϵ, 0, 10, 010, 1010, 01010, 001010

- **Subcadeia** : obtida removendo-se um sufixo e um prefixo.

Ex.: x= 001010 algumas subcadeias: ϵ, 001, 010,101, 1010, etc.

- **Prefixo próprio**, **sufixo próprio** e **subcadeia própria**: são prefixos, sufixos e subcadeias diferentes de x.

Voltar para o índice

3. Considere as seguintes sentenças:

x = abba
y = abab
z = bcd

a)apresente os prefixos de x
b)apresente os sufixos de y
c)apresente as subcadeias de z
d)apresente os prefixos próprios de yz

Voltar para o índice

Expressões regulares

- Uma expressão regular sobre um alfabeto V é uma expressão construída a partir de símbolos deste alfabeto e da sentença vazia combinados a partir da união, concatenação e fecho. O conjunto de sentenças formadas a partir da expressão regular forma uma linguagem.

Para o alfabeto {0,1}

1)Os símbolos do alfabeto constituem uma expressão

Expressão	Linguagem
0	{ 0 }
1	{ 1 }
ϵ	{ ϵ }

Voltar para o índice

2)Duas expressões r e s podem ser concatenadas na forma rs:

Expressão	Linguagem
01	{ 01 }

3)União : (r | s)

Expressão	Linguagem
0 | 1	{0,1}
0 | 00 | 000	{0,00,000}
(0|1)(0|1)	{00,01,10,11} - expressão com concatenação e duas uniões

4)Fecho : r^*

$0^* = \{ \epsilon, 0, 00, 000,...\}$

Voltar para o índice

Precedência :

1. Fecho
2. Concatenação
3. União

Voltar para o índice

1)Escrever 4 sentenças (se houver) p/ cada uma das expressões abaixo sobre o alfabeto V={a,b}

a)(ab)b =
b)$\epsilon bb\epsilon a$ =
c)a|b|ab|ba =
d)(a|b)(b|a) =
e)(a|ϵ)bbb =
f)aa(b|ϵ)aa =
g)$(a|b)a^*$ =
h)a^*ba^* =
i)$(a^*)(b^*)$ =
j)$(ab)^*$ =
k)$(a|bb)^*$ =
l)$(b^*ab^*)^*$ =

Voltar para o índice

Reescrita de expressões regulares

É a forma pela qual uma ER gera sentenças. Tem a forma:
$r \rightarrow s$

onde a expressão r pode ser substituída pela expressão s.

As regras para reescrita são as seguintes:

1) $(r\epsilon) \rightarrow r$
2) $(\epsilon r) \rightarrow r$
3) $(r|s) \rightarrow r$
4) $(r|s) \rightarrow s$
5) $(r^*) \rightarrow \epsilon$
6) $(r^*) \rightarrow (r^*)r$

Voltar para o índice

A partir da expressão $0(0|1)^*$ gerar a sentença 0101.

$0(0|1)^* \rightarrow (6)$
$0(0|1)^*(0|1) \rightarrow (4)$
$0(0|1)^*1 \rightarrow (6)$
$0(0|1)^*(0|1)1 \rightarrow (3)$
$0(0|1)^*01 \rightarrow (6)$
$0(0|1)^*(0|1)01 \rightarrow (5)$
$0\epsilon\ (0|1)01 \rightarrow (4)$
$0\epsilon 101 \rightarrow (1)$
0101

Voltar para o índice

2)Considere a expressão regular $(ab)^*(a|b)^*$. Apresente duas seqüências de aplicação de regras de reescrita que devem ser aplicadas a esta expressão para gerar a sentença **abababaaab**.

Voltar para o índice

Regras algébricas para expressões regulares

- As expressões regulares são expressões algébricas sobre as quais se podem fazer algumas operações (união, concatenação, fecho).
- Essas operações devem obedecer a um certo número de regras.
- Podemos agrupá-las em regras comutativas, regras associativas, regras distributivas, identidades e anuladores, regras de fecho.
- Duas expressões regulares são equivalentes se elas denotam a mesma linguagem.
- Quando se simplifica uma expressão regular, obtêm-se sucessivamente expressões regulares equivalentes.

Voltar para o índice

Regras comutativas e associativas

- Sejam L, M e N expressões regulares.
  - A união de duas expressões regulares é comutativa
    $L \mid M = M \mid L$
  - A união de linguagens é associativa
    $(L \mid M) \mid N = L \mid (M \mid N)$
  - Se queremos fazer a união de três conjuntos, poderemos unir os primeiros dois e depois unir o resultante com o terceiro; ou podemos unir os dois últimos, unindo depois o primeiro com o conjunto resultante.

Voltar para o índice

Regras comutativas e associativas

- A concatenação de expressões regulares é associativa
- (LM)N = L(MN)
- A concatenação de três cadeias pode fazer-se indiferentemente de dois modos: concatenar as duas primeiras e depois a terceira à direita; ou concatenar as duas últimas e depois a primeira à esquerda.
- Mas não é comutativa
  - $LM \neq ML$.

Voltar para o índice

Regras distributivas

- A concatenação tem alguma analogia com o produto algébrico e a união com a adição.
- A concatenação é distributiva à esquerda em relação à união:

  $L\ (M \mid N) = LM \mid LN$
- Por exemplo aba(bba+abb)=ababba+abaabb. Atendendo ao significado de +, união, temos que concatenar o prefixo aba com cada uma das cadeias dentro do parêntese. O resultado é a união dessas concatenações.
- E também distributiva à direita

  $(M \mid N)L = ML \mid NL$

Voltar para o índice

Regras distributivas

- A união não é distributiva em relação à concatenação nem à direita nem à esquerda
  - $(MN)|L \neq (M|L)(N|L)$
  - $L|(MN) \neq (L|M)(L|N)$

Voltar para o índice

Identidades e anuladores (zeros)

- Numa certa álgebra, a identidade é o elemento que operado com qualquer outro elemento resulta nesse elemento, tal como 1 na multiplicação numérica.
  - O conjunto vazio, $\emptyset$, é a identidade para a união porque $\emptyset \mid L = L \mid \emptyset = L$
  - Já para a concatenação, a identidade é a cadeia vazia porque $\epsilon\ L = L\ \epsilon = L$
- O anulador é o elemento que operado com outro elemento dá sempre o conjunto vazio.
  - No caso da concatenação é o conjunto vazio $\emptyset\ L = L\emptyset = \emptyset$
  - Para a união não existe anulador.

Voltar para o índice

Regras do fecho

- O fecho é uma operação muito importante em expressões regulares quando se trata de linguagens infinitas.
- Ele possui as seguintes propriedades:

(i) $(L^*)^* = L^*$
(ii) $L^+ = LL^* = L^*L$
(iii) $L^* = L^+|\epsilon$
(iv) $\emptyset^* = \epsilon$
(v) $\emptyset^* = \emptyset$
(vi) $(L|M)^* = (L^*M^*)^*$

Voltar para o índice

Definições regulares

- Uma **definição regular** introduz um nome para uma expressão regular na forma
  - $D \rightarrow r$
- onde D é um nome

Ex: $Min \rightarrow a|b|c|d|e|...|z$
$Mai \rightarrow A|B|C|D|E|...|Z$
$L \rightarrow Min \mid Mai$

Voltar para o índice

Especificação léxica através de expressões regulares

Dig $\rightarrow$ 0|1|...|9
Letra $\rightarrow$ a|b|c|...|z|A|...|Z

-Identificador em Pascal

$IdPas \rightarrow Letra(Letra|Dig|\_)^*$
$IdC \rightarrow (Letra|\_)(Letra|Dig|\_)^*$

Um identificador ADA é semelhante ao de pascal mas não pode haver dois _'s seguidos e o identificador não pode terminar em _.

$IdADA \rightarrow Letra(Letra|Dig|\_(Letra|Dig))^*$

Voltar para o índice

- SimbEsp $\rightarrow$:= | = | < | <=
- PalReserv $\rightarrow$ AND | BEGIN | CONST | ...
- Número inteiro sem sinal : $Intsem \rightarrow DigDig^*$
- Número inteiro com sinal : $Intcom \rightarrow (+|-|\epsilon)DigDig^*$
- Número em ponto flutuante :
  - 3.2
  - 3E10
  - 3.2E10
  - +3.2

Voltar para o índice

- Um comentário em linguagem C é uma seqüência de caracteres delimitada por /* e */. Entre os delimitadores pode aparecer qualquer caracter, exceto a seqüência "/". Use o nome Y para representar "qualquer caracter exceto * e / ".

Coment $\rightarrow$

Obs : Tokens : são símbolos básicos (as "palavras") de um programa na linguagem.

Voltar para o índice

Exercícios

- Escreva uma expressão regular sobre o alfabeto $\{a, b, c\}$ que gere a linguagem cujas sentenças contenham no mínimo 1 e no máximo 3 símbolos **a**.
- Considere o alfabeto {a,b}. Construa expressões regulares sobre esse alfabeto que representem cada uma das linguagens abaixo:
  - a) Sentenças que contem no máximo uma ocorrência do substring bbb;
  - b) Contem exatamente uma ocorrência do substring aaa e uma ocorrência do substring bbb (em qualquer ordem)
- Escreva uma expressão regular que gere todas as sentenças em $\{0, 1\}^*$ em que o número de 0's é divisível por 3.

Voltar para o índice

- Desenvolva expressões regulares e gramáticas regulares que gerem as seguintes linguagens sobre $\Sigma = \{a, b\}$
  - a)$\{w \mid$ w tem no máximo um par de a's como subpalavra e no máximo um par de b's como subpalavra$\}$
  - b)$\{w \mid$ qualquer par de a's antecede qualquer par de b's$\}$
  - c)$\{w \mid$ w não possui aba como subpalavra$\}$
- Especifique expressão regular que gere todas as sentenças de $\{0, 1\}^*$ formadas por 0's e 1's alternados.
- Especifique uma expressão regular equivalente que gere todas as sentenças em $\{0,1\}$ com no mínimo três símbolos e que o terceiro da direita para a esquerda (antepenúltimo) seja 0.

Voltar para o índice

- Descreva em palavras as linguagens geradas pelas seguintes expressões regulares:
  - a)$(aa + b)^*(a + bb)$
  - b)$(b + ab)^*(\epsilon + a)$
  - c)$(aa + bb + (aa + bb)(ab + ba)(aa + bb))^*$

Voltar para o índice

- Construa expressões regulares que gerem as seguintes linguagens sobre o alfabeto {0,1}
  - a) O conjunto de todos strings que terminam em 00.
  - b) O conjunto de todos os strings com 3 0's consecutivos.
  - c) O conjunto de todos strings tal que a cada 3 caracteres consecutivos pelo menos dois são 0.
  - d) O conjunto de todos os strings tal que o penúltimo simbolo é 0.

Voltar para o índice

(2014 - Técnicas 68) Considere a expressão regular a seguir. $(c^*a[abc]^*b[abc]^*)|c^*$ Assinale a alternativa que descreve, corretamente, todas as cadeias geradas por essa expressão regular.

a) Cadeias sobre o alfabeto {a, b, c} onde o primeiro **a** precede o primeiro **b**.

b) Cadeias sobre o alfabeto {a, b, c} com um número par de **a**'s.

c) Cadeias sobre o alfabeto {a, b, c} contendo a substring **baa**.

d) Cadeias sobre o alfabeto {a, b, c} contendo um número ímpar de **c**'s.

e) Cadeias sobre o alfabeto {a, b, c} terminadas por **c**.

Voltar para o índice

(2012 - Fundamentos 42) Assinale a alternativa que apresenta, corretamente, uma expressão regular que denota todas as strings de a's e b's que têm pelo menos dois b's consecutivos.

a) $(a^* + bb)(a + ba)^*(a + b)^*$

b) $(a + ba)^* bb(ba + a)^*$

c) $(a + b)^* ba^* b(a + b)^*$

d) $(a + bb)^*(bb + a)^*$

e) $(a + ba)^* bb(a + b)^*$

Voltar para o índice

11) (2004 - Fund - 36) As seguintes expressões regulares denotam as linguagens P, Q, L e R, respectivamente:$(1+10)^*$, $(0+01)^*$, $(0+1)^*$, $0(11)^*+1(00)^*$. Não se pode afirmar que: (obs: o operador "\"representa a operação de diferença de conjuntos)

(a) $P \cap Q \neq \emptyset$

(b) $P \cup Q \neq L$

(c) $P \cap Q = \{\epsilon\}$

(d) $(1+0)^* \backslash P = Q$

(e) $R \subset L \backslash (P \cup Q)$

Voltar para o índice

10) (2004 - Fund - 21) Seja $\Sigma = \{a, b\}$. Uma expressão regular denotando a linguagem $L = \{w \in \Sigma^*$ tal que toda ocorrência de "a"em w é imediatamente seguida de "b"$\}$ é:

(a) $(a^*b)^*$

(b) $(b + ab)^*$

(c) a^*b

(d) $b + (ab)^*$

(e) $(ab)^*$

Voltar para o índice