

Interpretação de linguagens artificiais

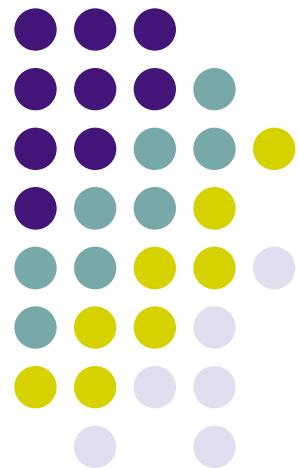
**Teoria de linguagens formais:
Linguagens e expressões regulares**

Aula 1

Gregory Moro Puppi Wanderley

Pontifícia Universidade Católica do Paraná (PUCPR)

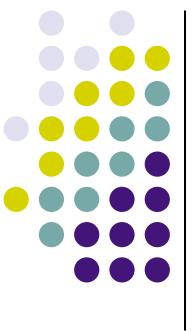
Bacharelado em Ciência da Computação – 4º Período





Apresentação do Professor

- Prof. Gregory Moro Puppi Wanderley
 - Graduação
 - Curso: Engenharia de Computação
 - Instituição: PUCPR
 - Mestrado
 - Dissertação: "FolksDialogue: Um Método para o Aprendizado Automático de Folksonomias a partir de Diálogo Orientado à Tarefa em Português do Brasil"
 - Instituição: PUCPR
 - Doutorado
 - Tese: "A Framework for Facilitating the Development of Systems of Systems"
 - Instituição: Université de Technologie de Compiègne - France



Disciplina

- Construção de Interpretadores
 - Reflexão: O que é?

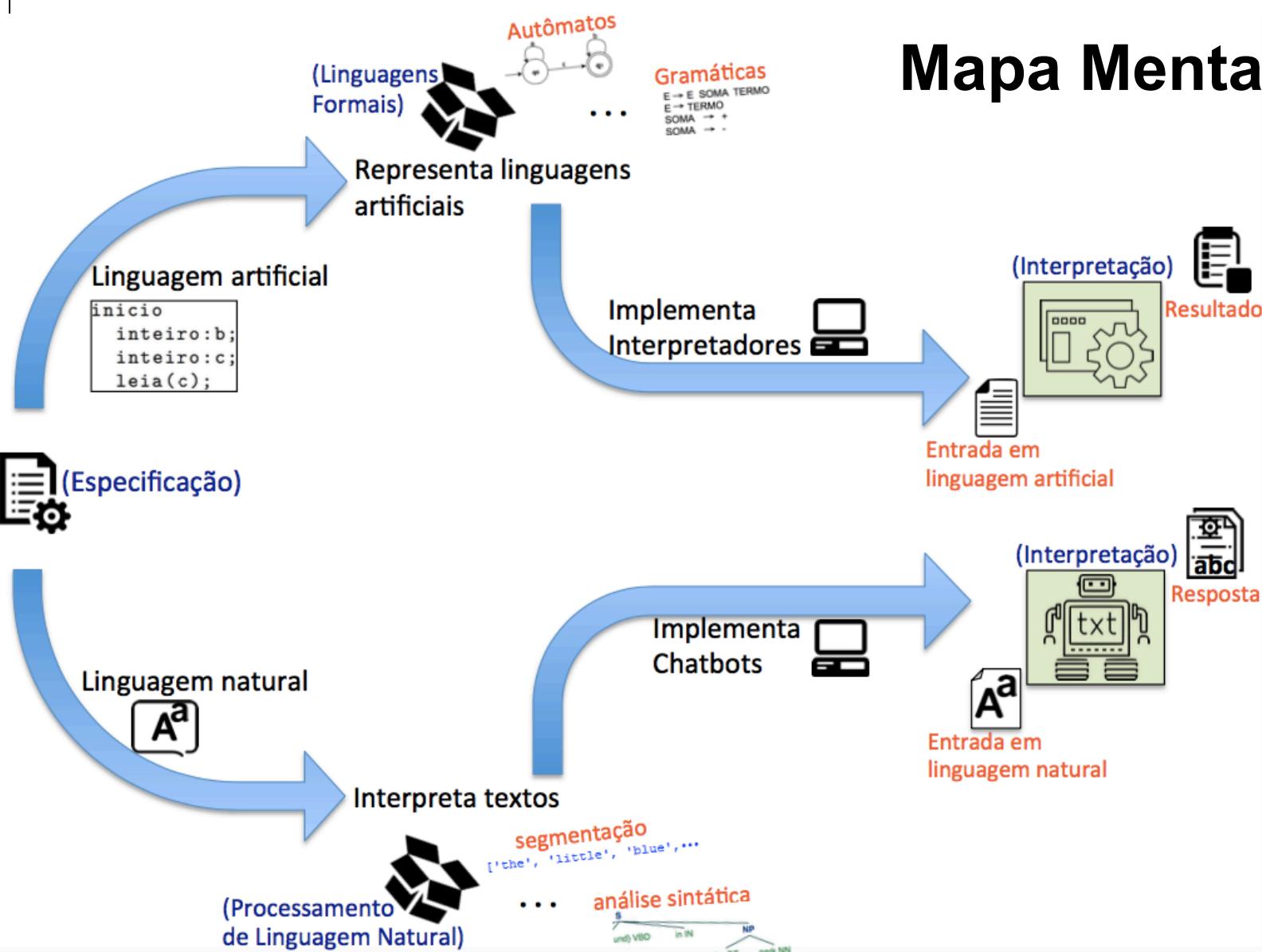


Plano de Ensino

- Disponível no Blackboard

Mapa Mental

I D
N E
T E L
R I P
R G E U
T A T
A G C
C E A
Ã N O S





Objetivos

- Aplicar conceitos da teoria de linguagens formais para construir interpretadores de linguagens artificiais.
- Aplicar técnicas de Processamento de Linguagem Natural (PLN) para construir interpretadores de linguagens naturais.



Temas de Estudo

- Processo de interpretação de linguagens
- Teoria de linguagens formais e autômatos
- Análise léxica
- Análise sintática
- Compilação de código-fonte
- Processamento de linguagem natural



Metodologia

- Material didático de apoio e orientações serão constantemente disponibilizados no **Blackboard**.
- Aulas expositivas dialogadas com uso de recursos de metodologias ativas.
- Desenvolvimento de **trabalhos** práticos para avaliação do rendimento do estudante.
- Desenvolvimento de **projetos** como parte da avaliação.
- **Avaliações individuais** de caráter teórico.



Resultados de Aprendizagem

Resultado de Aprendizagem

Temas de Estudo

RA1. Diferenciar o processo de interpretação de linguagens artificiais e naturais, de acordo com a especificação.

Processo de interpretação de linguagens
Análise léxica
Análise sintática
Compilação de código-fonte
Processamento de linguagem natural

RA2. Representar linguagens artificiais a partir de linguagens formais, em conformidade com a especificação.

Teoria de linguagens formais e autômatos

RA3. Construir interpretadores de linguagens artificiais, em conformidade com a especificação.

Análise léxica
Análise sintática

RA4. Interpretar textos em linguagem natural aplicando técnicas de processamento de linguagem natural, em conformidade com a especificação.

Processamento de linguagem natural

RA5. Construir chatbots através da aplicação de técnicas de processamento de linguagem natural, em conformidade com a especificação.

Processamento de linguagem natural



Avaliação

Atividade	Data
Trabalho 1	05/08/2019
Trabalho 2	12/08/2019
Trabalho 3	26/08/2019
Defesa 1 do Projeto I	02/09/2019
Trabalho 4	09/09/2019
Prova I ; Defesa 2 do Projeto I	16/09/2019
Trabalho 5	23/09/2019
Trabalho 6	30/09/2019
Trabalho 7	07/10/2019
Prova II	21/10/2019
Defesa 1 do Projeto II	11/11/2019
Defesa 2 do Projeto II	25/11/2019



Avaliação (cont.)

Item de avaliação	Resultados de Aprendizagem(RA)				
	RA1	RA2	RA3	RA4	RA5
Trabalho 1		0,10			
Trabalho 2		0,10			
Trabalho 3		0,10			
Trabalho 4		0,10			
Prova I	0,30	0,40			
Defesa 1 do Projeto I	0,10	0,10	0,40		
Defesa 2 do Projeto I	0,10	0,10	0,60		
Trabalho 5				0,10	
Trabalho 6				0,10	
Trabalho 7				0,10	
Prova II	0,30			0,40	
Defesa 1 do Projeto II	0,10			0,15	0,60
Defesa 2 do Projeto II	0,10			0,15	0,40
<i>nota do RA</i>	1,00	1,00	1,00	1,00	1,00
<i>peso do RA na média</i>	0,10	0,20	0,25	0,20	0,25
<i>média</i>	1,00				



Avaliação (cont.)

- Considerar-se-á aprovado o estudante que obter nota mínima de 7,0 (sete) em cada RA, além de uma frequência mínima de 75% de presença nas aulas.



Semana Estendida de Recuperação

- Atenção:
 - Semana Estendida de Recuperação (Prova): **02/12/2019**



Bibliografia

- Disponível na biblioteca da PUCPR



Bibliografia

- Bibliografia básica:
 - MENEZES, Paulo B.; Linguagens Formais e Autômatos, 5a Edição, 2005. Editora Sagra-Luzzato.
 - HOPCROFT, J.E.; MOTWANI, R.; ULLMAN, J.D. Introdução à teoria de autômatos, linguagens e computação. Ed. Campus, 2002.
 - LOUDEN, K. C.; Compiladores Princípios e práticas. Editora Thompson, 2004.
 - AHO, A. V.; Compiladores Princípios, Técnicas e Ferramentas. Editora Guanabara Koogan, 1995.



Bibliografia (cont.)

- **Bibliografia complementar:**

- LEWIS, R. L., PAPADIMITRIOU C. H.; Elementos de Teoria da Computação, 2a Edição. Editora Bookman, 2000.
- KOZEN, D. C.; Automata and Computability. Editora Springer, 1997.
- DIVERIO, T. A.; Teoria da Computação, 2a Edição. Editora Sagra Luzzatto, 2004.
- NETO, J. J.; Introdução à Compilação. Editora LTC, 1987.



Bibliografia (cont.)

- Bibliografia complementar:
 - SETZER, V. W.; A construção de um compilador, 2a Edição. Editora Campus, 1985.
 - SETHI, R.; Programming Languages Concepts & Constructs, 2a Edição. Editora Addison Wesley, 1997.
 - PRICE, A. M., TOSCANI S. S.; Implementação de Linguagens de Programação: Compiladores, 3a Edição. Editora Bookman, 2008.



Bibliografia (cont.)

- Bibliografia de apoio:
 - JURAFSKY, D., MARTIN, J.; Speech and Language Processing, Draft da 3a Edição (2018) disponível em <<https://web.stanford.edu/~jurafsky/slp3/>>. Acesso em 23/07/2019.
 - BIRD, S., KLEIN, E., LOPER, E.; Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. Disponível em <<http://www.nltk.org/book/>>. Acesso em 23/07/2019.



Horário e Local

- Horário:
 - Segundas-feiras das 07h50 às 11h10
- Local
 - Sala 005 (Bloco 5)



Atrasos

- Serão permitidos atrasos de 15 minutos nas primeiras semanas de aula.
- Atenção aos horários da disciplina.



Trabalhos e Projetos

- Trabalhos e Projetos serão especificados ao longo do semestre.
- Detalhes e orientações específicas serão fornecidos nos mesmos.



Aulas

- Não é permitido filmar, fotografar ou gravar em qualquer tipo de mídia as aulas, as avaliações e as atividades realizadas ou distribuir o material fornecido sem a autorização escrita do professor.





Política de Direitos Autorais

- Todo e qualquer artefato produzido pelos alunos poderá ser disponibilizado para acesso aberto.
- A produção de cada aluno será corrigida e, na indicação de cópia de material de terceiros, sem a devida referência de autoria, levará a atribuição da nota zero.
- Atenção: cópia é crime e não será tolerada nesta disciplina.





Orientações Gerais

- Comprometimento e responsabilidade do aluno.
- Fazer os trabalhos e os projetos estipulados.
- Se preparar para as provas.
- Se organizar com o cronograma (datas) ao longo do semestre.



Contato

- Todo o contato deve ser feito preferencialmente via email:
 - gregory.puppi@pucpr.br



Plano de Aula

- Contextualização
- Interpretação de linguagens artificiais
- Introdução à teoria de linguagens formais
- Linguagens e expressões regulares



Contextualização

- Linguagem artificial e linguagem natural
 - Reflexão: O que é?



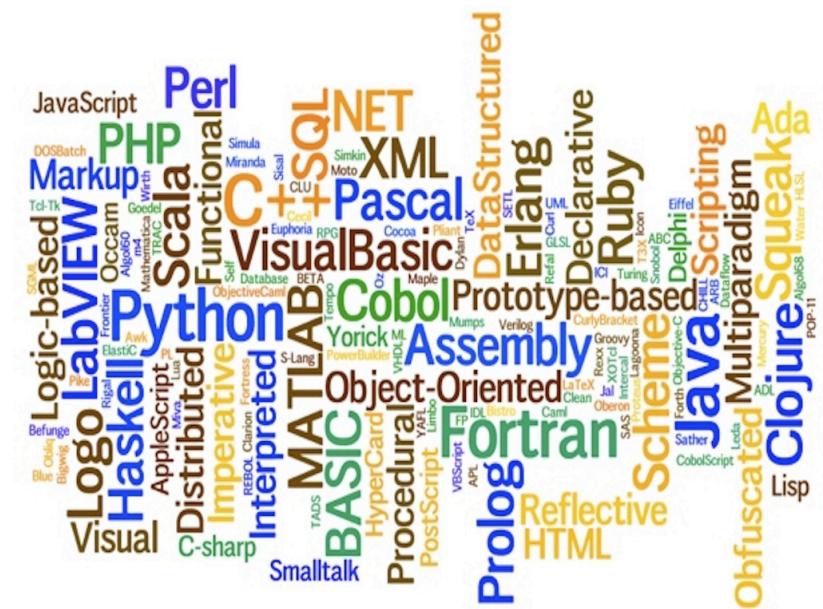
Contextualização

- Linguagem artificial
 - Linguagem **planejada e projetada de modo consciente.**
 - Gramática normalmente simplificada para facilitar o aprendizado e permitir uma análise mais eficiente por ferramentas computacionais, como compiladores e interpretadores. (Casalnuovo et al., 2018)



Contextualização

- Linguagem artificial
 - Linguagem **planejada e projetada de modo consciente**.
 - Gramática normalmente simplificada para facilitar o aprendizado e permitir uma análise mais eficiente por ferramentas computacionais, como compiladores e interpretadores. (Casalnuovo et al., 2018)
 - Exemplos
 - Linguagens de programação em geral.





Contextualização

- Linguagem natural
 - Linguagem que **evolui naturalmente**.
 - Gramáticas são modelos imperfeitos de fenômenos linguísticos que ocorrem naturalmente e, em geral, são mais complexos, não-determinísticos e ambíguos do que as gramáticas artificiais.
(Casalnuovo et al., 2018)

Contextualização

- Linguagem natural
 - Linguagem que **evolui naturalmente**.
 - Gramáticas são modelos imperfeitos de fenômenos linguísticos que ocorrem naturalmente e, em geral, são mais complexos, não-determinísticos e ambíguos do que as gramáticas artificiais.
(Casalnuovo et al., 2018)
 - Exemplo
 - Línguas faladas, como inglês, português, etc.





Plano de Aula

- Contextualização
- Interpretação de linguagens artificiais
- Introdução à teoria de linguagens formais
- Linguagens e expressões regulares



Interpretação de Linguagens Artificiais

- Processo de análise e processamento de uma linguagem artificial (ex.: de programação) através de uma ferramenta computacional denominada **interpretador**.
- O código é executado a medida em que vai sendo analisado e processado.



Interpretação de Linguagens Artificiais

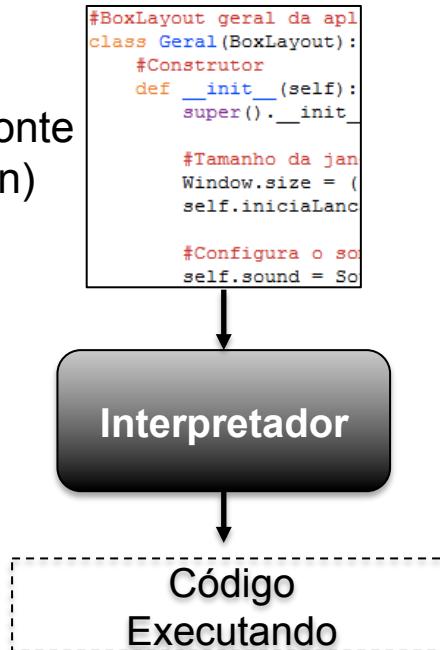
- Interpretador vs. Compilador
 - Reflexão: Diferença?



Interpretador vs. Compilador

- Um **interpretador** executa um programa-fonte de imediato, em oposição a um **compilador** que gera inicialmente um **código-objeto (máquina/binário)** o qual é então executado após o término da tradução.

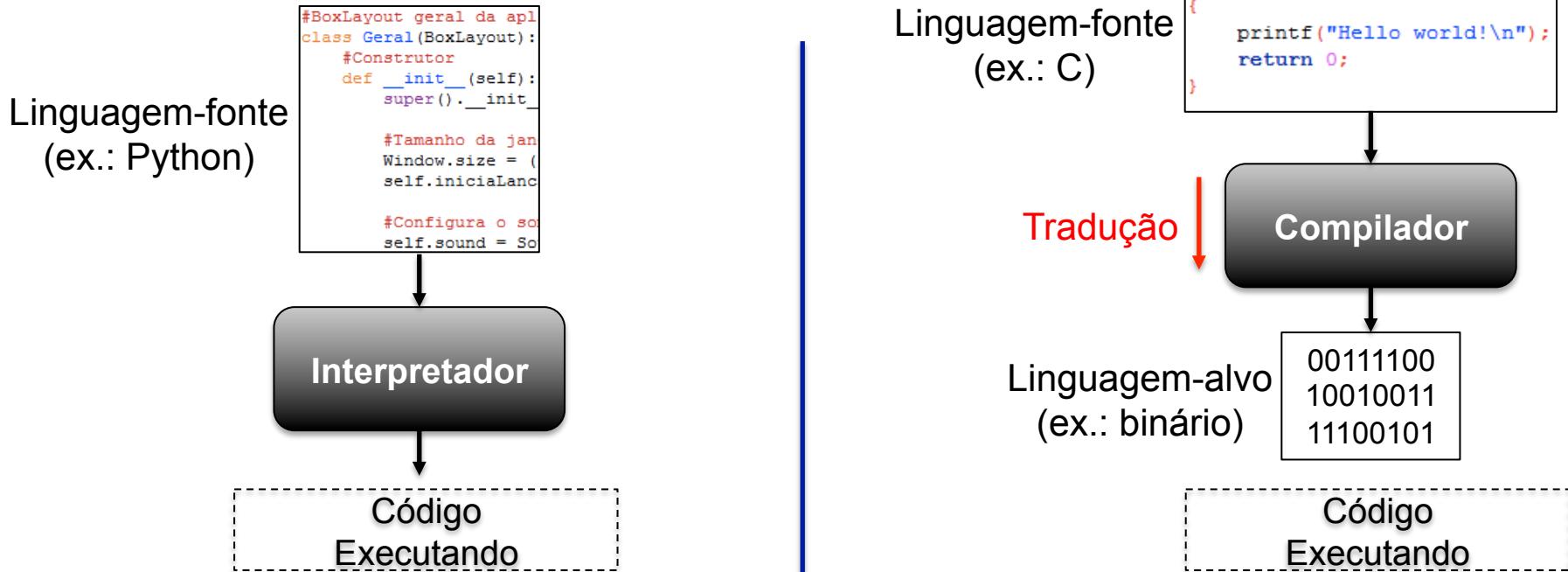
Linguagem-fonte
(ex.: Python)





Interpretador vs. Compilador

- Um **interpretador** executa um programa-fonte de imediato, em oposição a um **compilador** que gera inicialmente um **código-objeto (máquina/binário)** o qual é então executado após o término da tradução.





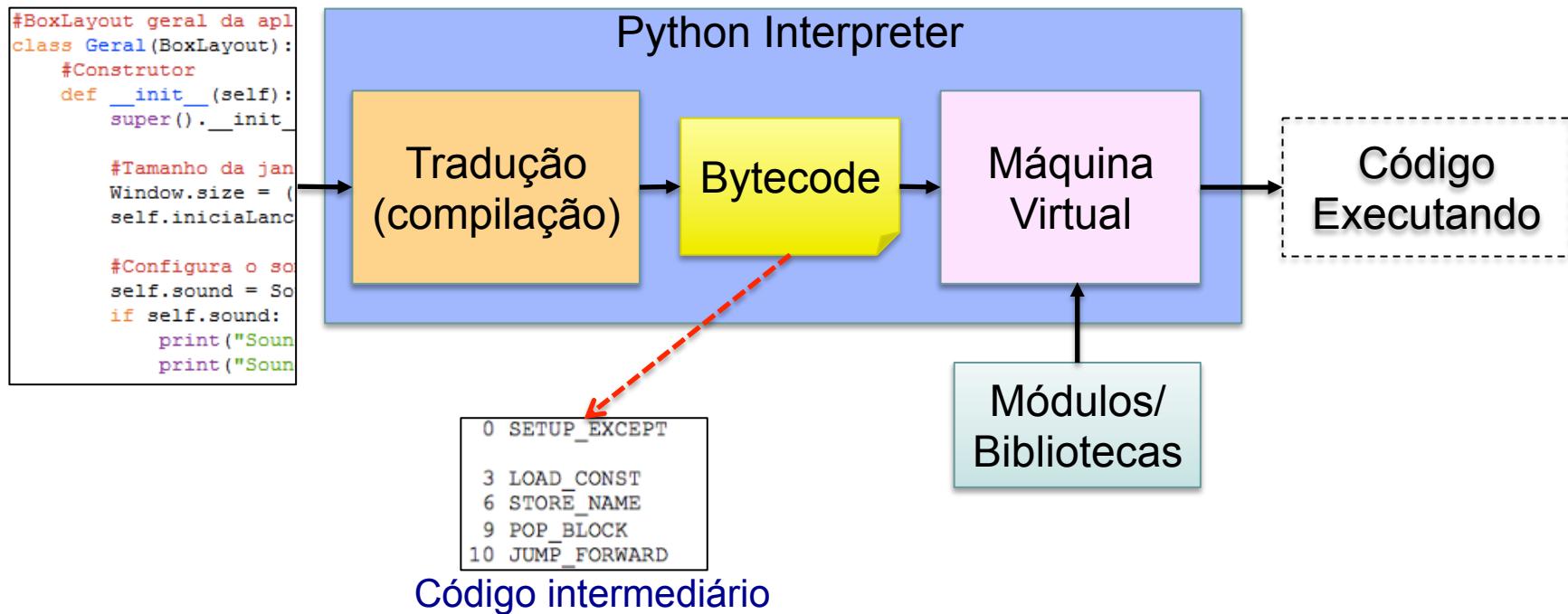
Interpretadores

- Podem inicialmente traduzir o código-fonte para um formato intermediário, o qual é então interpretado em tempo de execução, por exemplo por uma máquina virtual.



Interpretadores

- Podem inicialmente traduzir o código-fonte para um formato intermediário, o qual é então interpretado em tempo de execução, por exemplo por uma máquina virtual.





Interpretadores

- Principais Funcionalidades

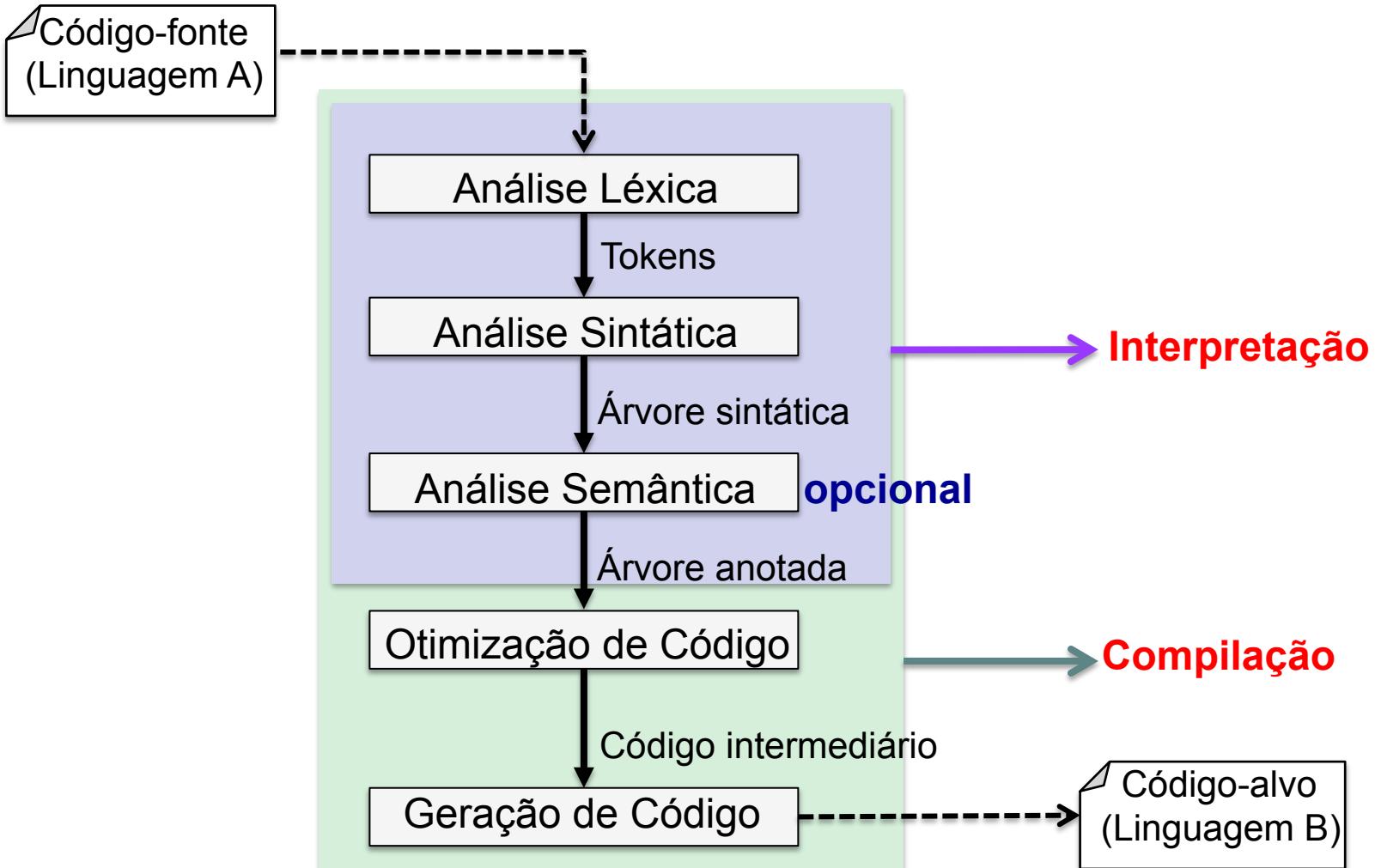
- Fornecer um meio de abstração para a programação.
 - Independência da arquitetura da máquina.
- Verificar certos tipos de erros no código.
- Permitir a execução do código.



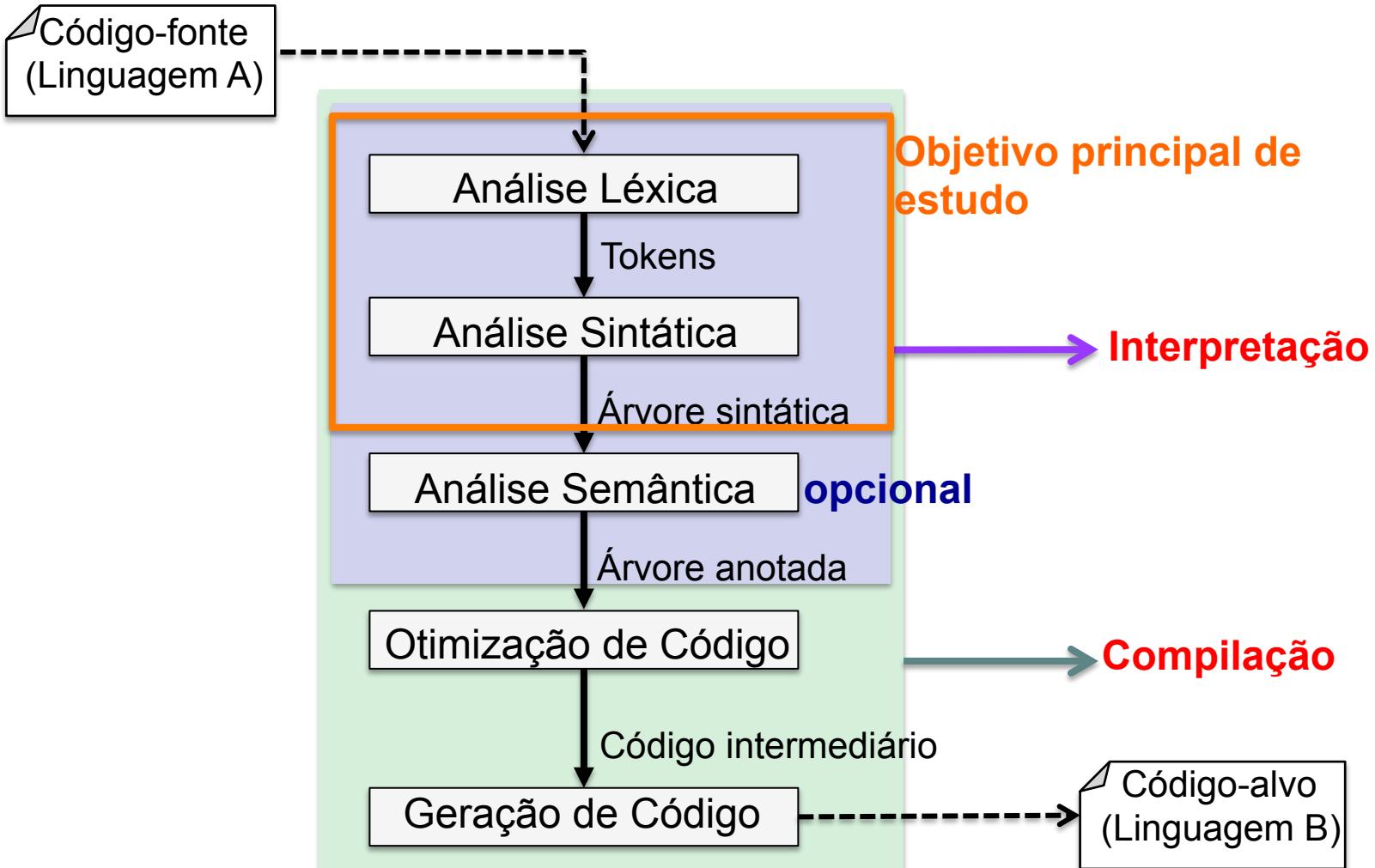
Interpretadores

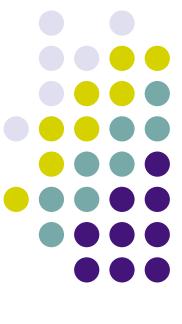
- Principais fases ou etapas de interpretação de código:
 - Análise léxica
 - Análise sintática
 - Opcionais:
 - Análise semântica
 - Etapas adicionais que caracterizam o processo de compilação:
 - Geração de código
 - Otimização de código

Processo de Interpretação



Processo de Interpretação





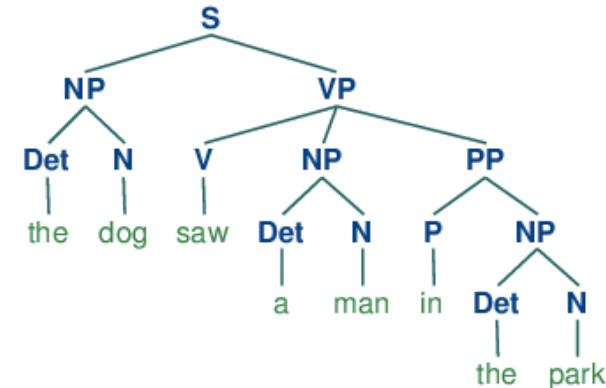
Plano de Aula

- Contextualização
- Interpretação de linguagens artificiais
- Introdução à teoria de linguagens formais
- Linguagens e expressões regulares



Linguagens Formais

- O que são linguagens formais?
 - Preocupa-se com o estudo e formalização de uma linguagem, tratando, dentre outros, de seus **elementos** e das **regras** que definem a estrutura entre eles.





Correlação

- Linguagens Formais e Interpretadores

- Interpretadores analisam os elementos, estruturas, e a semântica (de uma certa forma) de um programa escrito através de uma linguagem de programação.

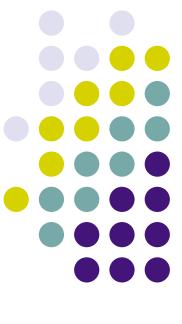
```
int main() {  
  
    int i, number, position;  
  
    printf("Initial array: ");  
    printTableau();  
  
    for(i = 0; i < MAX; i++) {  
  
        printf("\nType a number (int): ");  
        scanf("%d", &number); fflush(stdin);  
  
        position = recherchePosition(number);  
        printf("Returned position = %d\n", position);  
        insereEntier(number, position);  
        printTableau();  
    }  
  
    return 0;  
}
```



Linguagens formais

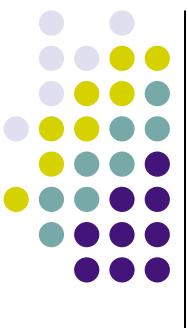
- Meio para modelar e especificar linguagens, bem como seus processos de análise e propriedades.
 - Regras para seus elementos.
 - Sintaxe (estrutura) bem definida.

Linguagens formais auxiliam no projeto de uma linguagem de programação.



Alfabeto

- Definição
 - Um **alfabeto** (Σ) é um conjunto finito de símbolos indivisíveis.
- Exemplos
 - $\Sigma = \{a, b, c\}$
 - $\Sigma = \{0, 1\}$



Alfabeto

- É alfabeto?

- \emptyset (conjunto vazio) 
- $\{\}$
- \mathbb{N} (conjunto dos números naturais) 
- $\{\}$
 - $\{0, 1, 2, 3, 4, 5, \dots\}$
- Conjunto dos dígitos hexadecimais 
- $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f\}$
- $\{a, b, aa, ab, ba, bb, aaa, \dots\}$ 



Palavra

- Definição

- Uma **palavra** é uma sequência finita de símbolos de um alfabeto.

- Exemplos

- barco (ex.: $\Sigma = \{a, b, c, d, e, f, \dots\}$)
- 0101 (ex.: $\Sigma = \{0, 1\}$)
- a^n (ex.: $\Sigma = \{a, b\}$)



Palavra

- **Tamanho** de uma palavra w : $|w|$
 - Número de símbolos que compõem a palavra.
 - Ex.: $w = abab ; |w| = 4$
- **Palavra vazia**: ϵ , sendo $|\epsilon| = 0$
- **Concatenação** de palavras
 - Operação binária que associa a um dado par de palavras uma palavra formada pela justaposição da primeira com a segunda.
 - Ex.: $w = abbba \ u = babab$
 - wu (ou $w.u$) = **abbababab**
 - uw (ou $u.w$) = **babababbba**



Palavra

- Concatenação sucessiva
 - Concatenação de uma palavra w com ela mesma.
 - w^n , sendo n o número de concatenações sucessivas. Exemplos:
 - $a^5 = \text{aaaaa}$
 - $a^0 = \epsilon$
 - $ab^1 = \text{ab}$
 - $ab^3 = \text{babab}$



Palavra

- **Subpalavra** de uma palavra
 - Qualquer sequência de símbolos contíguos da palavra.
 - Ex.: $w = abab$
 - Todas as subpalavras de w : ϵ , a, b, ab, ba, bab, abab



Palavra

- **Prefixo** de uma palavra
 - Qualquer sequência inicial de símbolos da palavra.
 - Ex.: $w = aabba$
 - Todos os prefixos de w : ϵ , a, aa, aab, aabb, aabba
- **Sufixo** de uma palavra
 - Qualquer sequência final de símbolos de uma palavra.
 - Ex: $w = aabba$
 - Todos os sufixos de w : ϵ , a, ba, bba, abba, aabba
- Qualquer prefixo ou sufixo de uma palavra é uma subpalavra.



Palavra

- O **reverso** de uma palavra w é a palavra w^r formada pelos símbolos de w na ordem inversa.
 - Ex.: Reverso de $w = abaab$; $u = abba$
 - $w^r = baaba$; $u^r = abba$ (palíndromo)



Palavra

- Conjunto de todas as palavras
 - Seja Σ um alfabeto, então Σ^* denota o **conjunto de todas as palavras** sobre o alfabeto.
 - * representa o fecho de Kleene ("0 ou mais repetições")
 - Ex.: se $\Sigma = \{a, b\}$, então:
 - $\Sigma^* = \{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$
 - Além disso, Σ^+ denota $\Sigma^* - \{\epsilon\}$
 - Ex.: se $\Sigma = \{a, b\}$, então:
 - $\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, \dots\}$



Linguagem

- Uma Linguagem Formal ou **Linguagem L** sobre um alfabeto Σ é um subconjunto de Σ^* .
 - $L \subseteq \Sigma^*$
 - Exemplos de linguagens sobre $\Sigma = \{a, b\}$:
 - $L_1 = \{aa, ab, ba, bb\}$ ou $\{w : |w| = 2\}$
 - $L_2 = \{a^n b^n \mid n \geq 0\}$ $L_2 = \{\epsilon, ab, aabb, aaabbb, \dots\}$
 - $L_3 = \{w \mid w = w^r\}$ $L_3 = \{\epsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, \dots\}$



Linguagem

- Operações sobre linguagens

- União: $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ ou } w \in L_2\}$
 - Ex.: $L_1 = \{a\}$ e $L_2 = \{b, aa\}$; $L_1 \cup L_2 = \{a, b, aa\}$
- Interseção: $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ e } w \in L_2\}$
 - Ex.: $L_1 = \{a, b\}$ e $L_2 = \{b, c\}$; $L_1 \cap L_2 = \{b\}$
- Diferença: $L_1 - L_2 = \{w \mid w \in L_1 \text{ e } w \notin L_2\}$
 - Ex.: $L_1 = \{a, b\}$ e $L_2 = \{b, d\}$; $L_1 - L_2 = \{a\}$
- Concatenação: $L_1 \cdot L_2 = \{uv \mid u \in L_1 \text{ e } v \in L_2\}$
 - Ex.: $L1 = \{a, b\}$ e $L2 = \{aa, bb\}$; $L1.L2 = \{aaa, abb, baa, bbb\}$



Trabalho 1

1^a parte

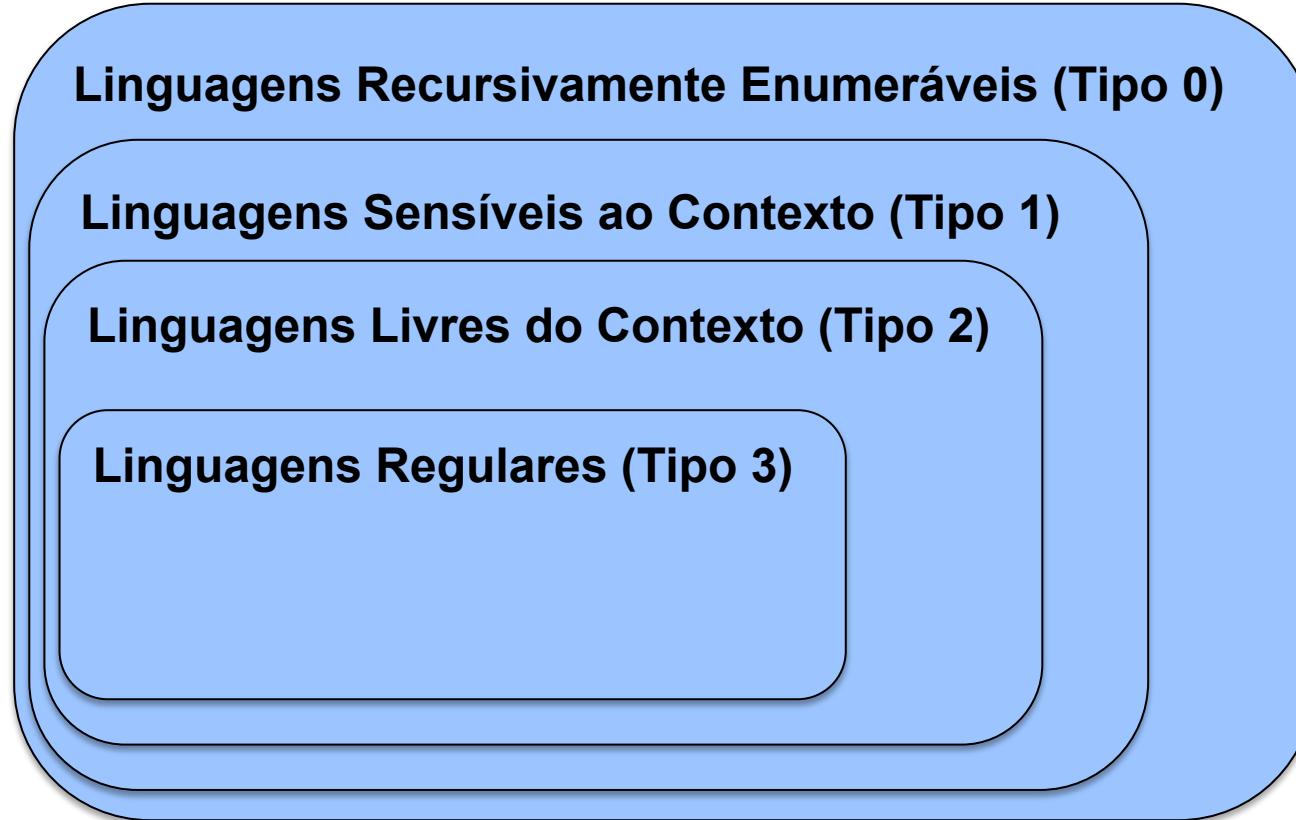


Hierarquia de Chomsky

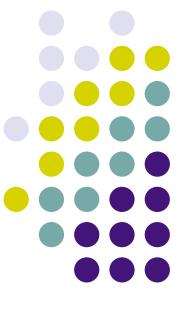
- Definida por Noam Chomsky em 1956.
- Classificação hierárquica dos tipos de linguagens conforme seus graus de complexidade.



Hierarquia de Chomsky

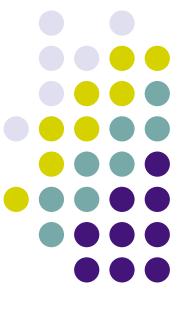


↑
C
O
M
P
L
E
X
I
D
A
D
E

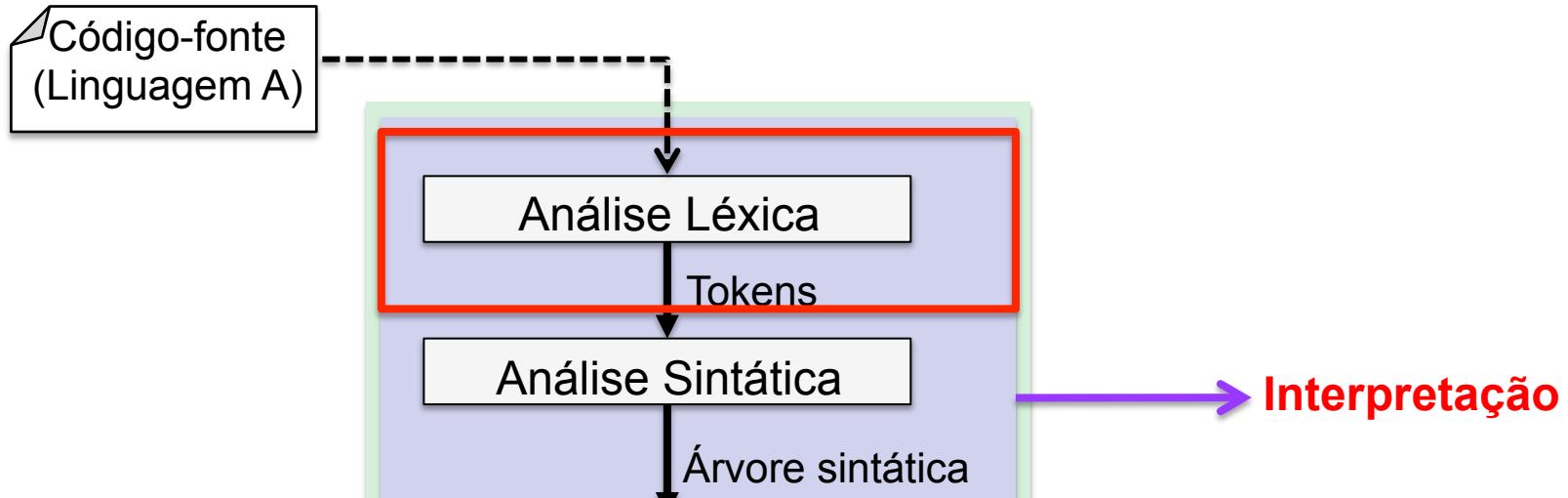


Plano de Aula

- Contextualização
- Interpretação de linguagens artificiais
- Introdução à teoria de linguagens formais
- Análise léxica: Linguagens e expressões regulares



Análise Léxica





Análise Léxica

- Lê caracteres do código tentando organizá-los em tokens.
 - **Tokens**: classes de símbolos, como palavras reservadas, delimitadores, identificadores, etc.



Análise Léxica

- Lê caracteres do código tentando organizá-los em tokens.
 - **Tokens**: classes de símbolos, como palavras reservadas, delimitadores, identificadores, etc.
- É um caso especial de reconhecimento de padrões.
 - Necessário o estudo de métodos específicos para:
 - Especificação dos padrões. (**O que deve ser reconhecido?**)
 - Reconhecimento dos padrões. (**Realizar o reconhecimento.**)



Análise Léxica

- Exemplo

Código-fonte

```
if (net > 0.0) total += net * (1.0 + tax / 100.0);
```

↓

```
if (net > 0.0) total += net * (1.0 + tax / 100.0);
```

Análise Léxica



Análise Léxica

- Exemplo

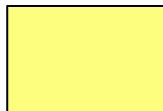
Código-fonte

```
if (net > 0.0) total += net * (1.0 + tax / 100.0);
```

↓

```
if (net > 0.0) total += net * (1.0 + tax / 100.0);
```

Análise Léxica



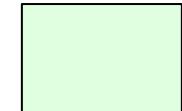
Palavras reservadas



Símbolos especiais



Identificadores



Números



Análise Léxica

- Diferentes cadeias de entrada podem ter o mesmo token. Exemplos:
 - Token: **NUM**
 - Pode representar infinitas cadeias.
 - Ex.: 1, 7, 23, 58, etc.



Análise Léxica

- Diferentes cadeias de entrada podem ter o mesmo token. Exemplos:
 - Token: **NUM**
 - Pode representar infinitas cadeias.
 - Ex.: 1, 7, 23, 58, etc.
 - Token: **ID**
 - Pode representar infinitas cadeias.
 - Ex.: a, nome, sala, etc.



Análise Léxica

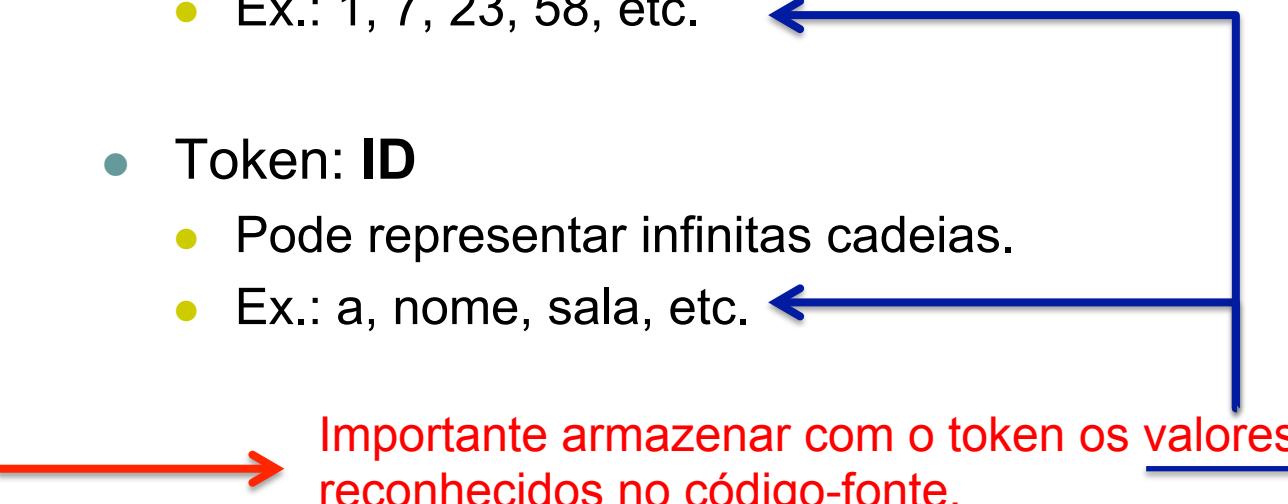
- Diferentes cadeias de entrada podem ter o mesmo token. Exemplos:
 - Token: **NUM**
 - Pode representar infinitas cadeias.
 - Ex.: 1, 7, 23, 58, etc.
 - Token: **ID**
 - Pode representar infinitas cadeias.
 - Ex.: a, nome, sala, etc.

Importante armazenar com o token os valores reconhecidos no código-fonte.

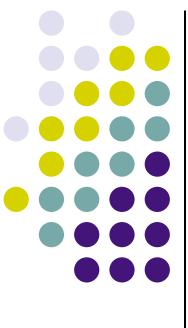


Análise Léxica

- Diferentes cadeias de entrada podem ter o mesmo token. Exemplos:
 - Token: **NUM**
 - Pode representar infinitas cadeias.
 - Ex.: 1, 7, 23, 58, etc.
 - Token: **ID**
 - Pode representar infinitas cadeias.
 - Ex.: a, nome, sala, etc.

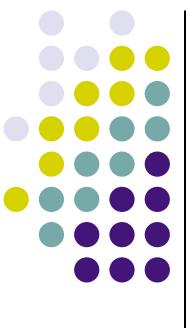


Importante armazenar com o token os valores reconhecidos no código-fonte.



Análise Léxica

- Padrão e Lexema
 - Padrão
 - **Regra** descrevendo um conjunto de cadeias de entrada que podem ser reconhecidas como um dado token.



Análise Léxica

- Padrão e Lexema

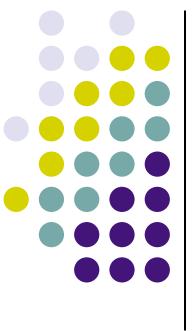
- Padrão
 - Regra descrevendo um conjunto de cadeias de entrada que podem ser reconhecidas como um dado token.
- Lexema ou valor
 - Conjunto de **caracteres reconhecidos no código-fonte** pelo padrão de um token.
 - Atributo do token.



Análise Léxica

- Exemplos de token, lexema e padrão

Token	Lexema (exemplo)	Padrão (informal)
WHILE	while	while
IF	if	if
ASSIGN	=	=
PLUS	+	+
MINUS	-	-
ID	a, nome, sala	Letra seguida por letras e/ou dígitos
NUM	23, 1.4142	Qualquer constante numérica



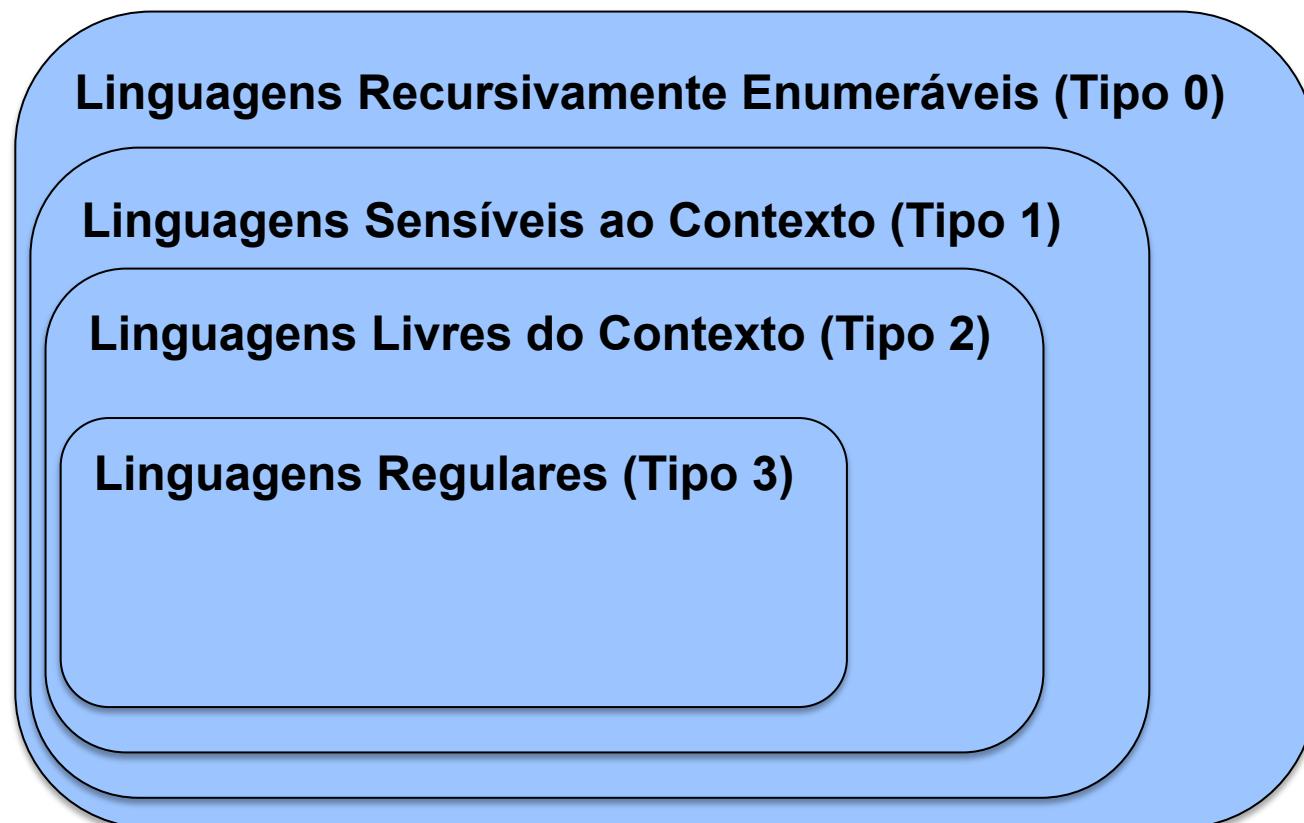
Análise Léxica

- Como especificar e reconhecer os tokens da análise léxica?



Linguagens Regulares

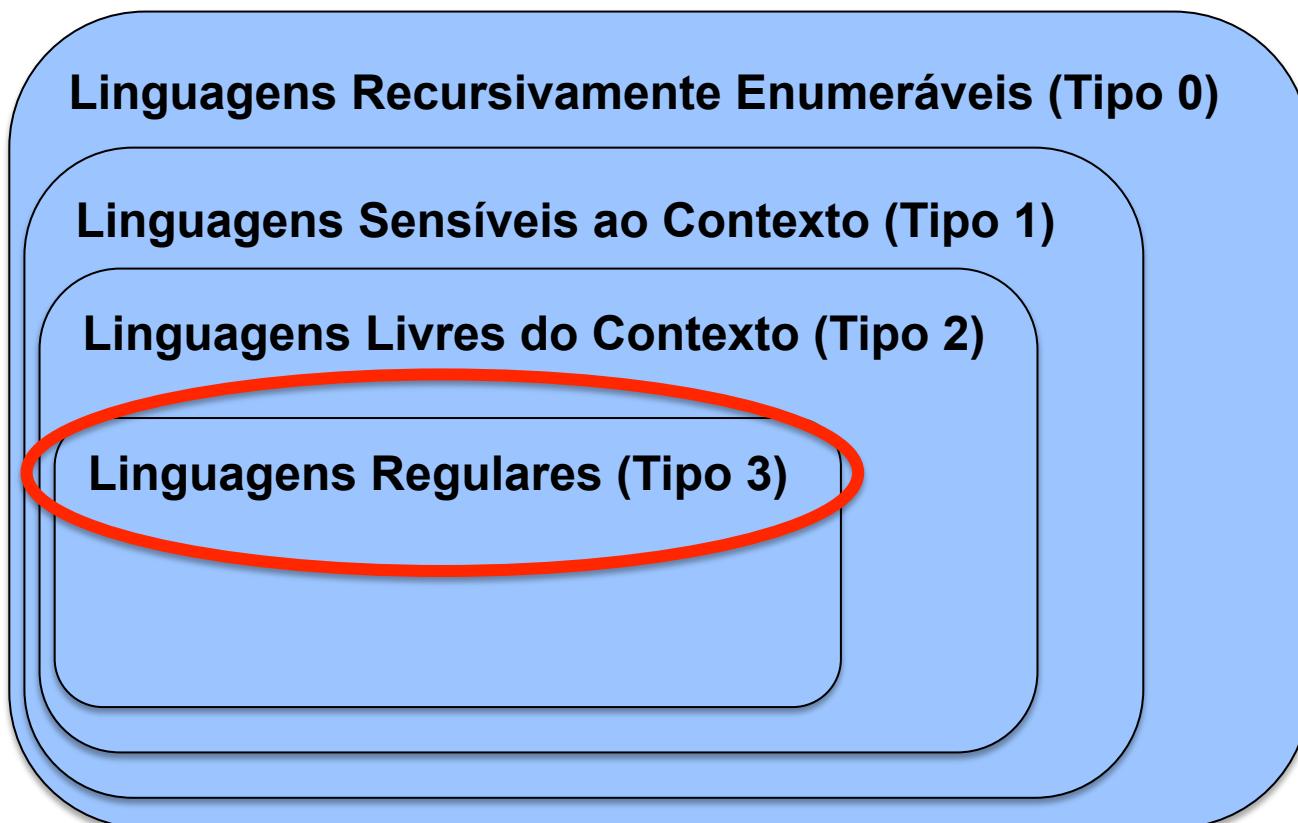
- Classes de linguagens mais simples (Hierarquia de Chomsky).





Linguagens Regulares

- Classes de linguagens mais simples (Hierarquia de Chomsky).





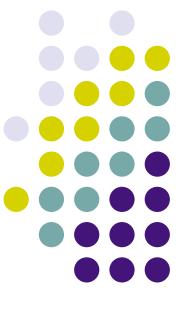
Linguagens Regulares

- Uma linguagem regular é a linguagem formal que pode ser expressa através de uma expressão regular.



Linguagens Regulares

- Se adaptam as necessidades mínimas da análise léxica de compiladores.
 - Especificação (descrição) e reconhecimento de tokens.



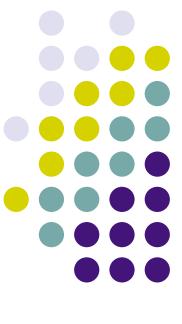
Linguagens Regulares

- Se adaptam às necessidades mínimas da análise léxica de compiladores.
 - Especificação (descrição) e reconhecimento de tokens.
- Possuem limitações de expressividade:
 - Uma linguagem com duplo balanceamento não é regular.
 - Ex.: Linguagens que utilizam parênteses平衡ados (abertura e fechamento), como C, C++, Pascal, LISP, Java, etc.



Expressão Regular

- Toda linguagem regular pode ser **descrita** (especificada) por uma *expressão regular*.



Expressão Regular

- Toda linguagem regular pode ser **descrita** (especificada) por uma *expressão regular*.
- Formalismo gerador, pois pode inferir-se como gerar as palavras de uma linguagem.



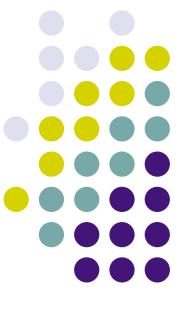
Expressão Regular

- Toda linguagem regular pode ser **descrita** (especificada) por uma *expressão regular*.
- Formalismo gerador, pois pode inferir-se como gerar as palavras de uma linguagem.
- Representa um **padrão** de cadeias de caracteres.



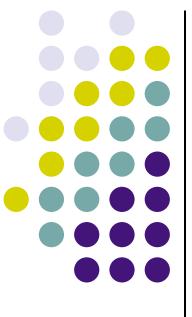
Expressão Regular

- Toda linguagem regular pode ser **descrita** (especificada) por uma *expressão regular*.
- Formalismo gerador, pois pode inferir-se como gerar as palavras de uma linguagem.
- Representa um **padrão** de cadeias de caracteres.
- Útil para descrever formalmente os tokens de uma linguagem.



Definição

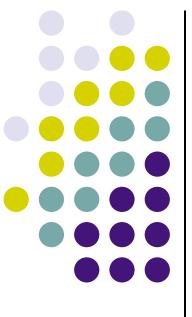
- As expressões regulares (ER) são definidas a partir de:
 - Conjuntos básicos
 - \emptyset é uma ER e representa a linguagem vazia.
 - ϵ é uma ER e representa a linguagem contendo exclusivamente a palavra vazia, i.e., $\{ \epsilon \}$
 - Para **qualquer símbolo x de Σ** , x é uma ER e representa a linguagem $\{ x \}$
 - Operações de **concatenação** ($.$) e **união** ($+$)
 - Fecho de **Kleene** ($*$)



Exemplo ER

- $\Sigma = \{a, b\}$
- ER: $aa(a+b)^*bb$

→ Qual a linguagem gerada?

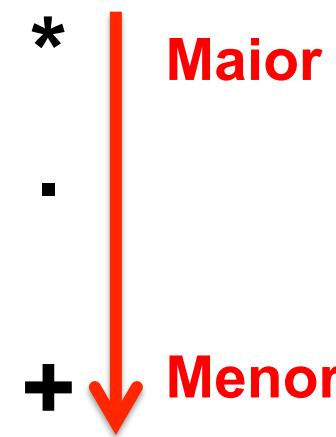


Exemplo ER

- $\Sigma = \{a, b\}$
- ER: $aa(a+b)^*bb$
- $L = \{aabb, aaabb, aabbb, aaabbb, aaaabb, \dots\}$



Precedência de operadores





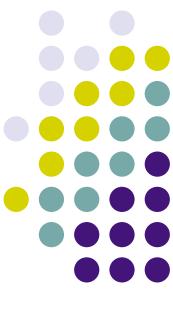
Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	
aab*	
a+b	
(a+b)*	
(a+b)*bb(a+b)*	
b*ab*	
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	
a+b	
(a+b)*	
(a+b)*bb(a+b)*	
b*ab*	
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	
(a+b)*	
(a+b)*bb(a+b)*	
b*ab*	
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	$L = \{a, b\}$
(a+b)*	
(a+b)*bb(a+b)*	
b*ab*	
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	$L = \{a, b\}$
(a+b)*	$L = \{\epsilon, a, b, aab, bbaa, abb, \dots\}$
(a+b)*bb(a+b)*	
b*ab*	
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	$L = \{a, b\}$
(a+b)*	$L = \{\epsilon, a, b, aab, bbaa, abb, \dots\}$
(a+b)*bb(a+b)*	Todas as palavras contendo bb como subpalavra
b*ab*	
(aa+bb)(a+b)*	



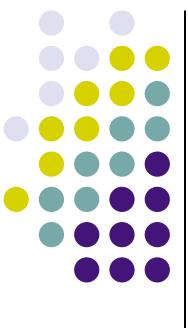
Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	$L = \{a, b\}$
(a+b)*	$L = \{\epsilon, a, b, aab, bbaa, abb, \dots\}$
(a+b)*bb(a+b)*	Todas as palavras contendo bb como subpalavra
b*ab*	Todas as palavras contendo exatamente um a
(aa+bb)(a+b)*	



Outros Exemplos ER

Expressão Regular	Linguagem Gerada
ab	$L = \{ab\}$
aab*	$L = \{aa, aab, aabb, aabbb, \dots\}$
a+b	$L = \{a, b\}$
(a+b)*	$L = \{\epsilon, a, b, aab, bbaa, abb, \dots\}$
(a+b)*bb(a+b)*	Todas as palavras contendo bb como subpalavra
b*ab*	Todas as palavras contendo exatamente um a
(aa+bb)(a+b)*	Todas as palavras que começam com aa ou bb



Notação de União

- Numa ER, o operador de + da União pode ser denotado também como |
 - No exemplo abaixo ambas as expressões são equivalentes:
 - $(a+c)^*(b+\varepsilon)(a+c)^*$
 - $(a|c)^*(b|\varepsilon)(a|c)^*$



Notação de União

- Numa ER, o operador de + da União pode ser denotado também como |
 - No exemplo abaixo ambas as expressões são equivalentes:
 - $(a+c)^*(b+\varepsilon)(a+c)^*$
 - $(a|c)^*(b|\varepsilon)(a|c)^*$

→ Qual a linguagem gerada?



Notação de União

- Numa ER, o operador de + da União pode ser denotado também como |
 - No exemplo abaixo ambas as expressões são equivalentes:
 - $(a+c)^*(b+\varepsilon)(a+c)^*$
 - $(a|c)^*(b|\varepsilon)(a|c)^*$

→ Qual a linguagem gerada?

Todas as palavras que contém no máximo um b.
 $L = \{\varepsilon, b, acccb, ccbaa, acac, \dots\}$



Exercícios

- Considere $\Sigma = \{a, b\}$. É possível escrever uma ER para a linguagem abaixo?
 - $L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, \dots\}$

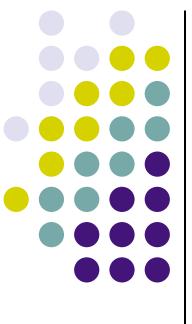


Exercícios

- Considere $\Sigma = \{a, b\}$. É possível escrever uma ER para a linguagem abaixo?
 - $L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, \dots\}$

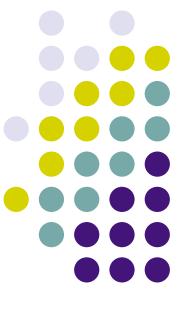


Não. Como garantir que o número de a's é igual ao número de b's ?



Exercícios

- Escreva uma ER para identificar os números inteiros.



Exercícios

- Escreva uma ER para identificar os números inteiros.
 - naturais = 0 | 1 | 2 | 3 | ... | 9
 - sinal = + | -
 - numeros_inteiros = sinal naturais naturais*



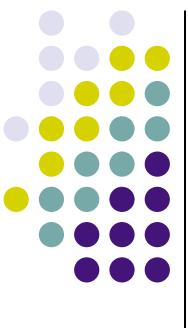
Exercícios

- Escreva uma ER para identificar os números inteiros.
 - naturais = 0 | 1 | 2 | 3 | ... | 9
 - sinal = + | -
 - numeros_inteiros = sinal naturais naturais*
 - naturais = [0-9]+ ← 1 ou mais ocorrências
 - numeros_inteiros = (+ | -)? naturais
- ↑
**0 ou 1
ocorrência**



Representação simplificada

- [0-9] (números naturais)
- [a-z] (alfabeto em minúsculo)
- [A-Z] (alfabeto em maiúsculo)
- [a-zA-Z] (alfabeto em minúsculo e maiúsculo)
- + (uma ou mais ocorrências do elemento precedente)
- * (zero ou mais ocorrências do elemento precedente)
- ? (zero ou uma ocorrência do elemento precedente)



Trabalho 1

2^a parte



Dúvidas?



Síntese da Aula

- Conceitos de linguagens formais
 - Alfabeto, palavra, linguagem, Hierarquia de Chomsky
- Análise léxica
 - Linguagens e expressões regulares



Próxima Aula

- **TDE01**
 - Estudo do conceito de Autômato Finito (Não-) Determinístico.
- **Trabalho 2**



Referências

- Casalnuovo, C., Sagae, K., & Devanbu, P. (2018). Studying the difference between natural and programming language corpora. *Empirical Software Engineering*, 1-46.