LFA – Aula 01

Apresentação

Horário de aula

- segunda-feira: 14h00-15h40
- terça-feira: 16h00-17h40
- local: Lab. 5b Central

Celso Olivete Júnior

olivete@fct.unesp.br

Professor Celso Olivete Júnior

- Bacharelado em Ciência da Computação (Unoeste-2002)
- Mestrado e Doutorado em Engenharia Elétrica – Área: Visão Computacional (USP-SC-2005/2009)
- Áreas de interesse e atuação:

  Visão Computacional

  Processamento de Imagens Médicas

  Desenvolvimento para Web e Dispositivos Móveis

  Compiladores

Site do Curso

www.fct.unesp.br/docentes/dmec/olivete/lfa

- slides, exercícios, notas e demais materiais estarão disponíveis no site

- Envio de trabalhos e dúvidas através do email

olivete@fct.unesp.br

A disciplina LFA

• Linguagens formais e autômatos é uma disciplina fundamental dos cursos superiores da área de computação, especialmente daqueles que apresentam ênfase na formação científica do aluno, como é o caso dos cursos de bacharelado em Ciência da Computação e de vários cursos de Engenharia de Computação. Ela faz parte do núcleo denominado "Fundamentos da Computação" (conforme o currículo de referência da Sociedade Brasileira de Computação - `www.sbc.org.br`);

A disciplina LFA

- Áreas de Aplicação relacionadas com o aprendizado adquirido nesta disciplina:
  - Inteligência Artificial
    - Gramáticas
    - Autômatos Finitos (Linguagens Regulares)
  - Compiladores e Interpretadores
    - Linguagens Livres de Contexto
    - Linguagens Sensíveis ao Contexto...

A disciplina LFA

- utilização na disciplina de Compiladores

```
public class LFA {
    public static void main(String[] args) {
        int variavel = 20;
        for (int i = 0; i < 10; i++) {
            System.out.println("Alo mundo 10 vezes!!!");
        }
        System.out.println( variavel );
        System.out.println( variavel2 );
        System.out.println( "Fim do alo mundo!!!";
    }
}
```

Programa

1. **Introdução e fundamentos matemáticos (conjuntos, funções e relações, cadeias e linguagens**

2. **Linguagens regulares: expressões regulares, autômatos finitos, gramáticas regulares,**

3. **Propriedades das linguagens regulares**

4. **Autômato finito com saída: Máquinas de Moore e Mealy**

5. **Linguagens livre de contexto: BNF e EBNF e autômato com pilha**

6. **Propriedades das linguagens livre de contexto**

Objetivo da disciplina LFA

- Fornecer meios para um correto entendimento e aplicação dos conceitos de procedimento efetivo, computabilidade e solucionabilidade de problemas.

Avaliação

- Avaliação 1: 27-28/04 Avaliação 2: 22-23/06 Exame: 06-07/07
- A cada bimestre
  - Uma prova: NP
  - Trabalhos e projetos: MT
  - MB = (7*NP + 3*MT)/10 **SE E SOMENTE SE** (NP>=5 **E** MT>=5)
    - Caso contrário (MT<5 OU NP<5)
      - MB = Menor Nota (NP ou MT)
  - Onde:
  - NP = Nota da prova
  - MT = Média dos trabalhos
  - MB = Média do Bimestre
- A nota final (NF) do aluno no curso será a média das notas obtidas nos 2 bimestres
- Caso o aluno não obtenha a nota mínima para aprovação, será oferecida uma terceira avaliação (exame)

Avaliação

- A "cola" ou plágio em provas, exercícios ou atividades práticas implicará na atribuição de nota zero para todos os envolvidos. Dependendo da gravidade do incidente, o caso será levado ao conhecimento da Coordenação e do Conselho do Departamento, para as providências cabíveis. Na dúvida do que é considerado cópia ou plágio, o aluno deve consultar o professor antes de entregar um trabalho.

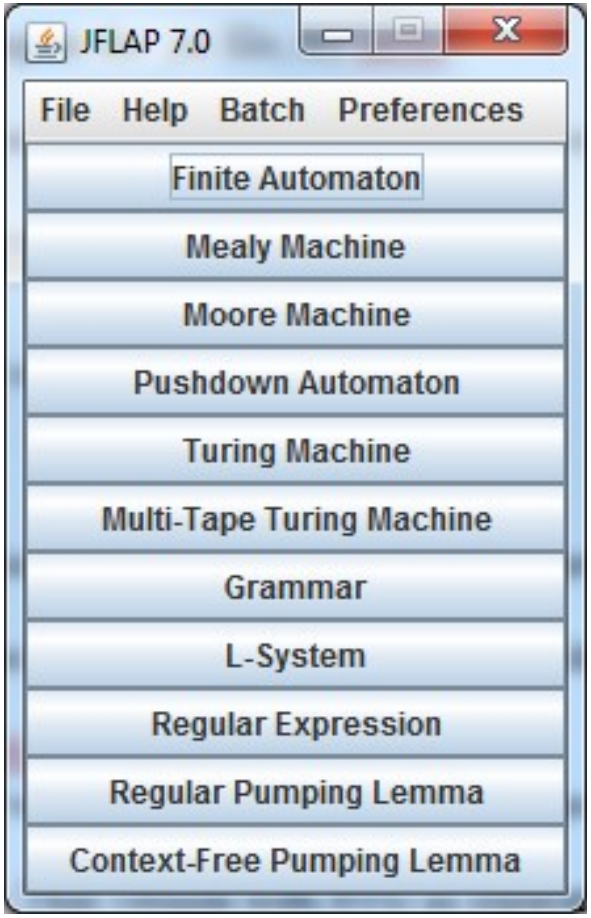
Projeto (em duplas)

- Parte 1: especificar e simular autômatos finitos através de diagramas de transições; especificar e simular expressões regulares e especificar e simular gramáticas regulares
  - Maior detalhamento no site da disciplina.

Projeto (em duplas)

- Parte 2:
  - Permitir a conversão entre os modelos:
    - AF ⇔ GR;
    - AF ⇔ ER
  - especificar e simular autômatos finitos com saída (Máquina de Moore ou Mealy) através de diagramas de transições;
    - Maior detalhamento no site da disciplina.

Ferramenta exemplo: JFlap



http://www.jflap.org/

Visão Geral da Disciplina

Linguagem Formal

- Uma Linguagem Formal possui:

  - Sintaxe bem definida: Dada uma sentença, é possível sempre saber se ela pertence ou não a uma linguagem;

  - Semântica precisa: De modo que não contenha sentenças sem significado ou ambíguas;

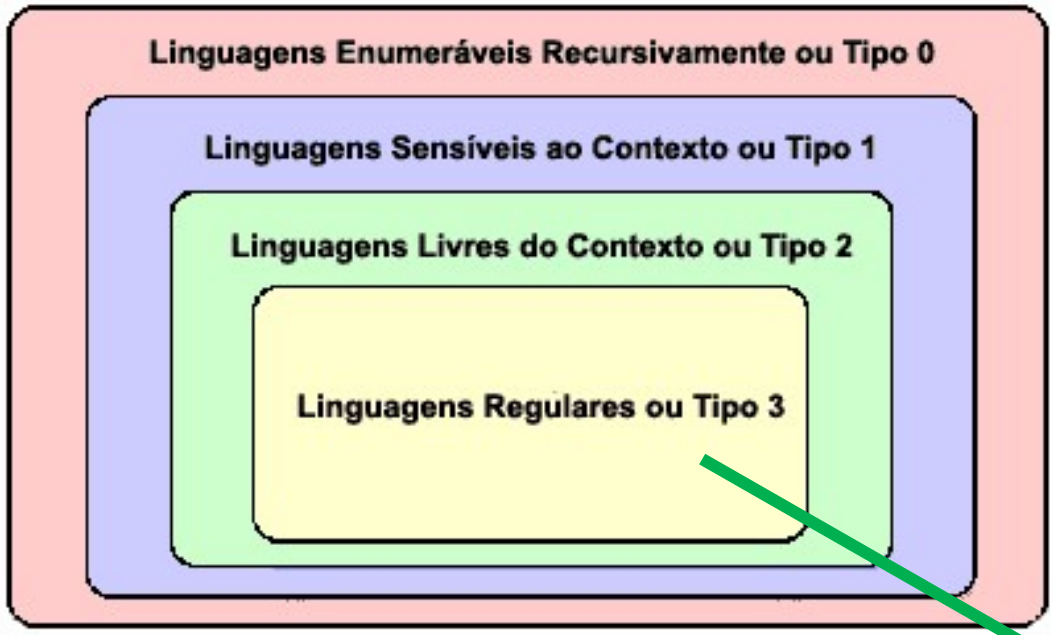
Linguagem Formal: exemplos na Computação

- Java, C, Pascal, HTML, Basic, C#, VB.net,...

- Ao projetar um sistema, o programador precisa estabelecer uma **linguagem formal** de comunicação com o usuário final – linguagem de comunicação complexa = sistema mal projetado

Fundamental em COMPILADORES

Classificação das Linguagens: qual será abordada considerando a Hierarquia de Chomsky?

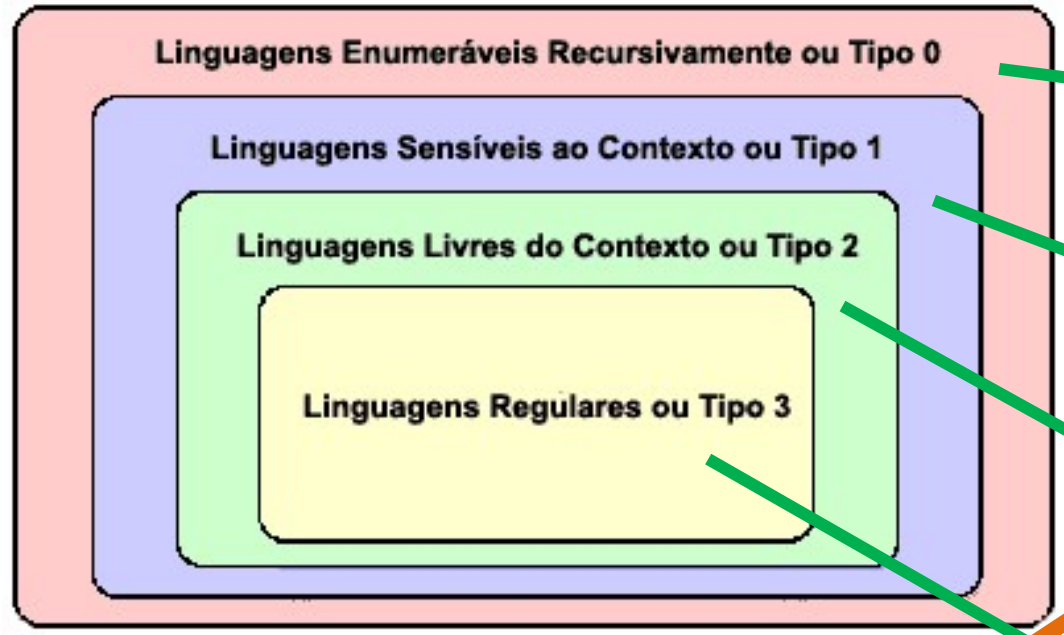


Programa do curso

Expressões regulares e linguagens
Propriedades das linguagens regulares
Gramáticas e Linguagens
Autômatos Finitos
Linguagens Livre de Contexto

Autômatos finitos
Expressões regulares
Gramáticas regulares

Classificação das Linguagens: qual será abordada considerando a Hierarquia de Chomsky?



Máquina de Turing

Máquina de Turing com fita limitada

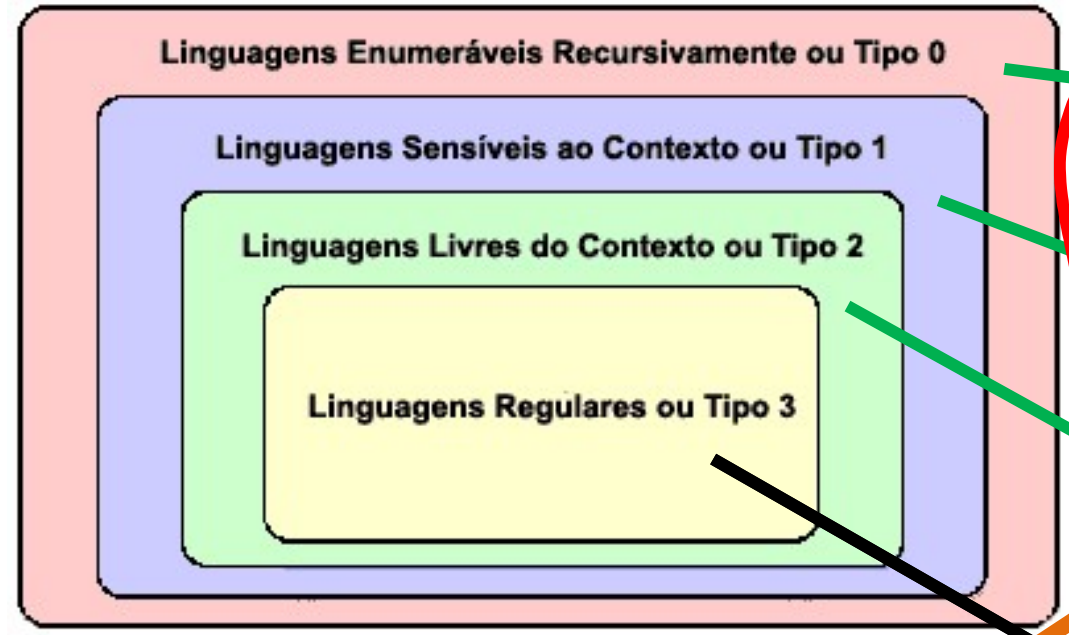
Autômato à pilha
Gramáticas livre de contexto

Autômatos finitos
Expressões regulares
Gramáticas regulares

Programa do curso

Expressões regulares e linguagens
Propriedades das linguagens regulares
Gramáticas e Linguagens
Autômatos Finitos
Linguagens Livre de Contexto

Classificação das Linguagens: qual será abordada considerando a Hierarquia de Chomsky?



Disciplina - Teoria da Computação

- Máquina de Turing
- Máquina de Turing com fita limitada
- Autômato à pilha
  Gramáticas livre de contexto
- Autômatos finitos
  Expressões regulares
  Gramáticas regulares

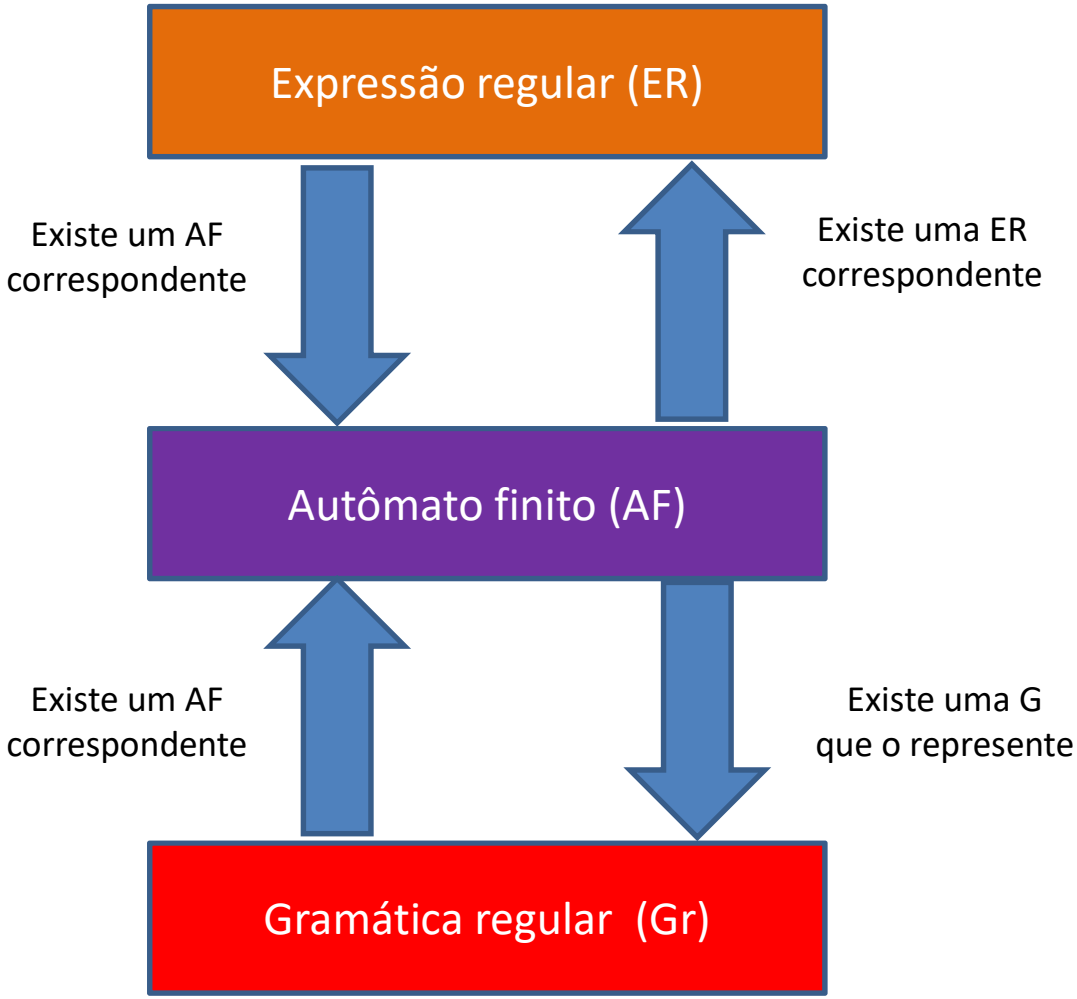
Programa do curso

Expressões regulares e linguagens
Propriedades das linguagens regulares
Gramáticas e Linguagens
Autômatos Finitos
Linguagens Livre de Contexto

Linguagem

- Um linguagem é um conjunto de strings;
- Uma string é uma sequência de símbolos;
- Estes símbolos estão definidos em um alfabeto finito;
  - Ex: Linguagem C ou Pascal, etc.
- Linguagem regular - Formas de representação: expressão regular (ER), autômato finito (AF) e gramática regular (GR)
  - **ER:** Servem para verificar se uma string está ou não em uma linguagem

Equivalências entre linguagens regulares



Visão geral - Expressão regular (ER)

- Uma ER é definida a partir de conjuntos básicos e operações de concatenação e união e oferece um modo declarativo de expressar as cadeias que queremos aceitar.
- desenvolvidas a partir de :
  1. símbolos do alfabeto Σ
  2. operadores de:
     - **união** : representado por $\cup$ ou + ou | ➔ conjunto de cadeias que está em um conjunto **ou** em outro
     - **concatenação**: representado por • ou sem ponto ➔ junção dos elementos
     - **fecho-estrela (fechamento)**: representado por * ➔ cadeias que podem ser formadas tomando qualquer número de repetição do elemento do conjunto (inclusive nenhuma vez)
  3. parênteses

Visão geral - Expressão regular

- Exemplos

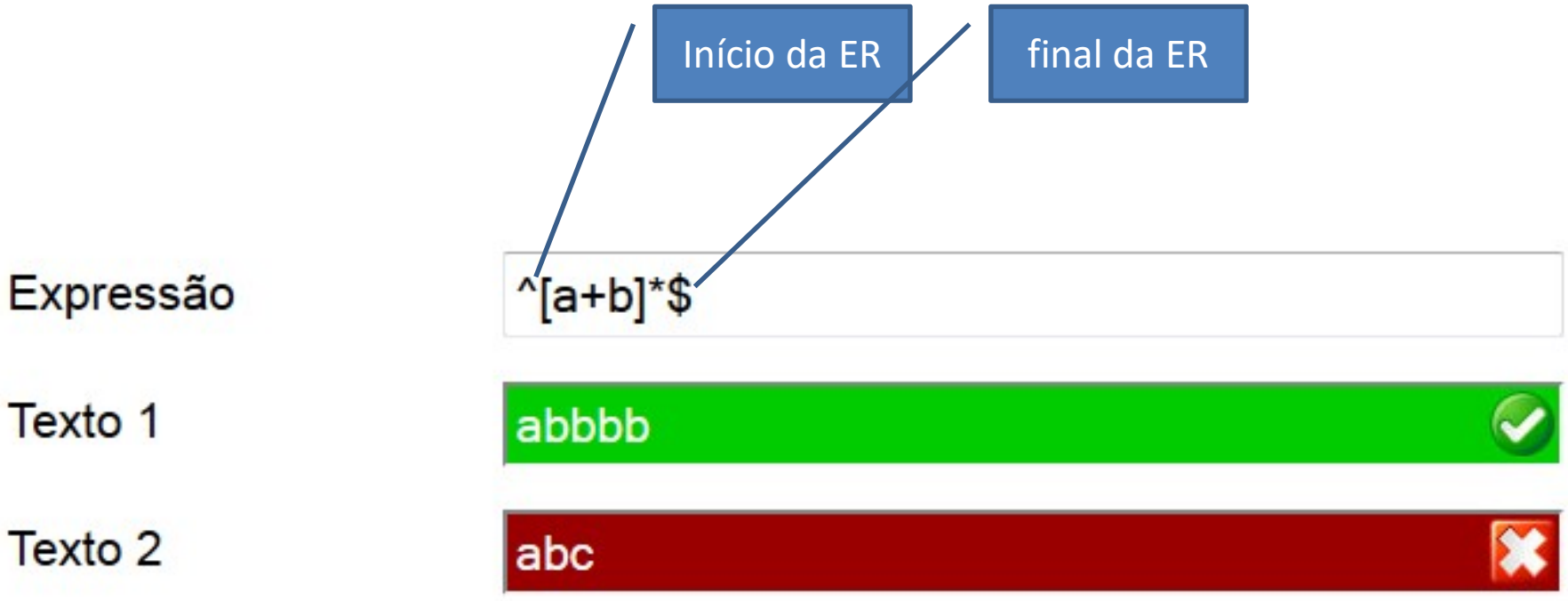
ER	Linguagem reconhecida
aa	??
ba*	??
(a+b)	??
(a+b)*	??
(a+b)*aa(a+b)*	??
a*ba*ba*	??
(a+b)*(aa+bb)	??

Simulador de ER

http://tools.lymas.com.br/regexp_br.php#

Visão geral - Expressão regular (ER)

- Exemplos



Simulador de ER

http://tools.lymas.com.br/regexp_br.php#

Visão geral - Expressão regular

- Exemplos

ER	Linguagem reconhecida
aa	somente a palavra **aa**
ba*	iniciam com b seguido por 0 **ou** mais ocorrências de a
(a+b)	formada por a **ou** por b
(a+b)*	formada por qualquer quantidade de a **ou** b, inclusive nenhuma vez
(a+b)*aa(a+b)*	??
a*ba*ba*	??
(a+b)*(aa+bb)	??

Simulador de ER

http://tools.lymas.com.br/regexp_br.php#

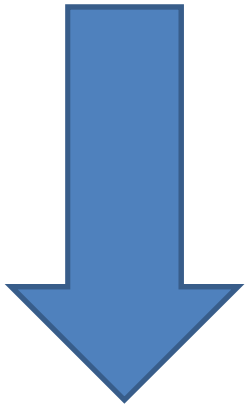
Visão geral - Expressão regular

- Exemplos

ER	Linguagem reconhecida
aa	somente a palavra aa
ba*	iniciam com b seguido por 0 ou mais a
(a+b)	formada por a **ou** por b
(a+b)*	formada por qualquer quantidade de a ou b, inclusive nenhuma vez
(a+b)*aa(a+b)*	contém **aa** como elemento
a*ba*ba*	todas as palavras contendo exatamente 2 b´s
(a+b)*(aa+bb)	terminam com aa ou bb

Defina uma ER que reconheça:
- um identificador em Java
- um número inteiro

Visão geral - Expressão regular: atuam na especificação de uma linguagem



Autômatos finitos: é um formalismo que pode convertido em um programa de computador

Visão Geral - Autômatos finitos são formados por:

- Um conjunto finito de estados
- Arestas levando de um estado a outro, anotada com um símbolo
- Um estado inicial
- Um ou mais estados finais
- Normalmente os estados são numerados ou nomeados para facilitar a manipulação e discussão

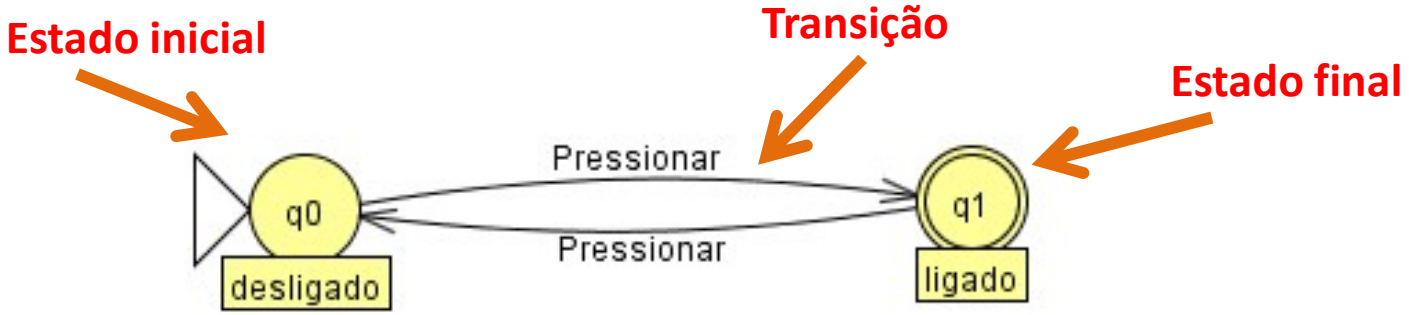
Visão Geral - Autômatos finitos: constituem um modelo útil para muitos elementos de hardware e software. Ex:

1. Controlar um interruptor: dispositivo para acender ou apagar uma lâmpada
2. software para projetar e verificar o comportamento de circuitos digitais;
3. analisador léxico de um compilador
4. descrever o processo de reconhecimento de padrões em cadeias de entrada, e assim podem ser utilizados para construir sistemas de busca;

Autômatos finitos: o que é e o que ele faz?

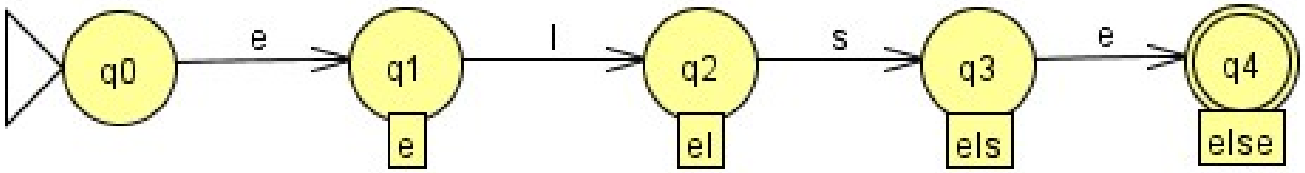
- Em cada um exemplos citados, podemos citar o autômato como estando em um de um número finito de "estados"
  - O estado permite "lembrar" o passado relevante do sistema (ele chegou lá após um certo número de transições/passos, que são o seu passado) e para onde pode seguir.
  - Se o número de estados de um sistema é finito, pode-se representá-lo com uma quantidade limitada de recursos.

Autômato finito: exemplo clássico do interruptor



- o dispositivo memoriza se está no estado "ligado" ou no estado "desligado", e permite ao usuário pressionar um botão diferente do estado atual

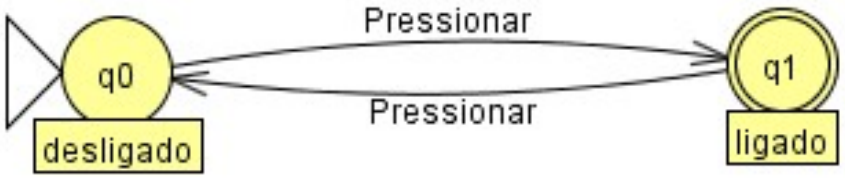
Autômato finito: parte de um analisador léxico – reconhece a palavra reservada *else*



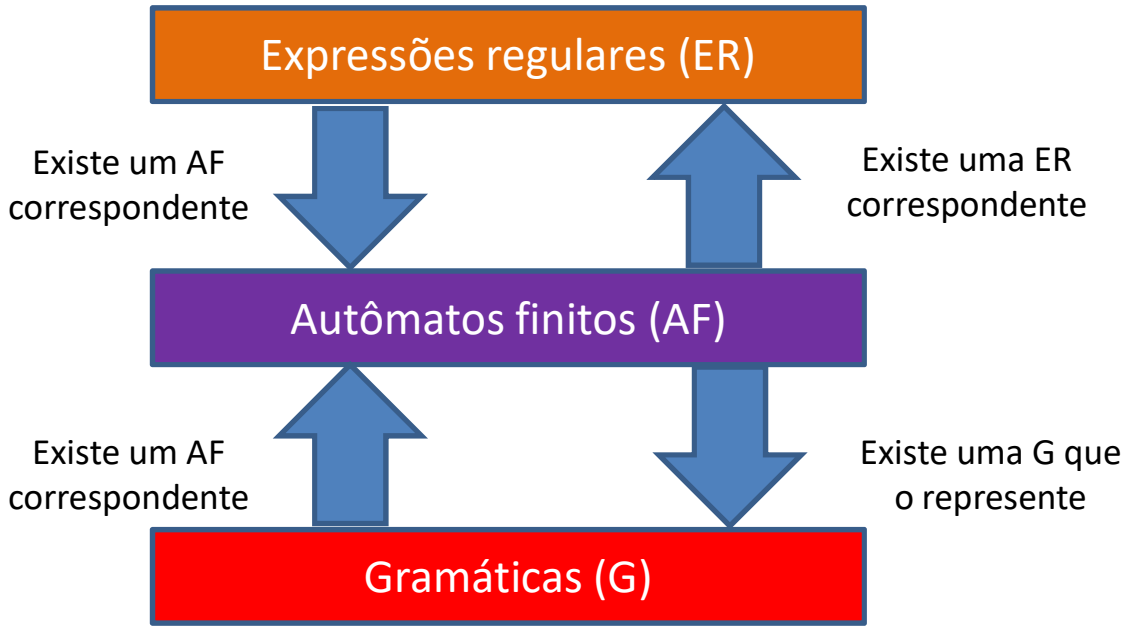
- cada estado memoriza o que já foi reconhecido – da palavra vazia até a palavra completa ***else***
- parte do analisador léxico, representado pelo autômato, examina letra a letra do arquivo fonte que está sendo compilado até atingir o estado q4 (reconhece o ***else***)

Autômatos finitos: principais características

- Memória Limitada.
- Trabalha apenas sobre o estado atual.
- Memória limitada pela quantidade de estados.



Autômatos finitos (AF) versus Expressões Regulares (ER) – como transformar?

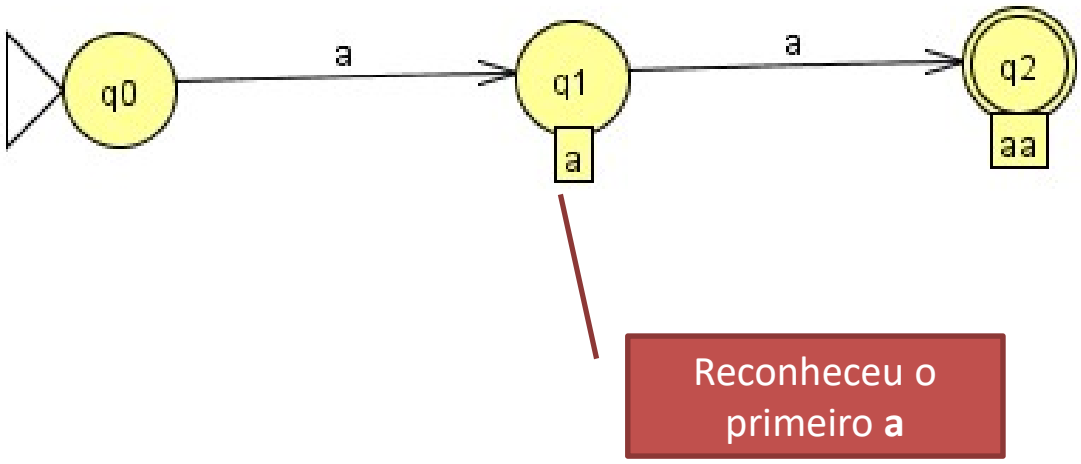


Autômatos finitos (AF) *versus* Expressões Regulares (ER) – como transformar?

ER	Linguagem gerada
aa	somente a palavra aa

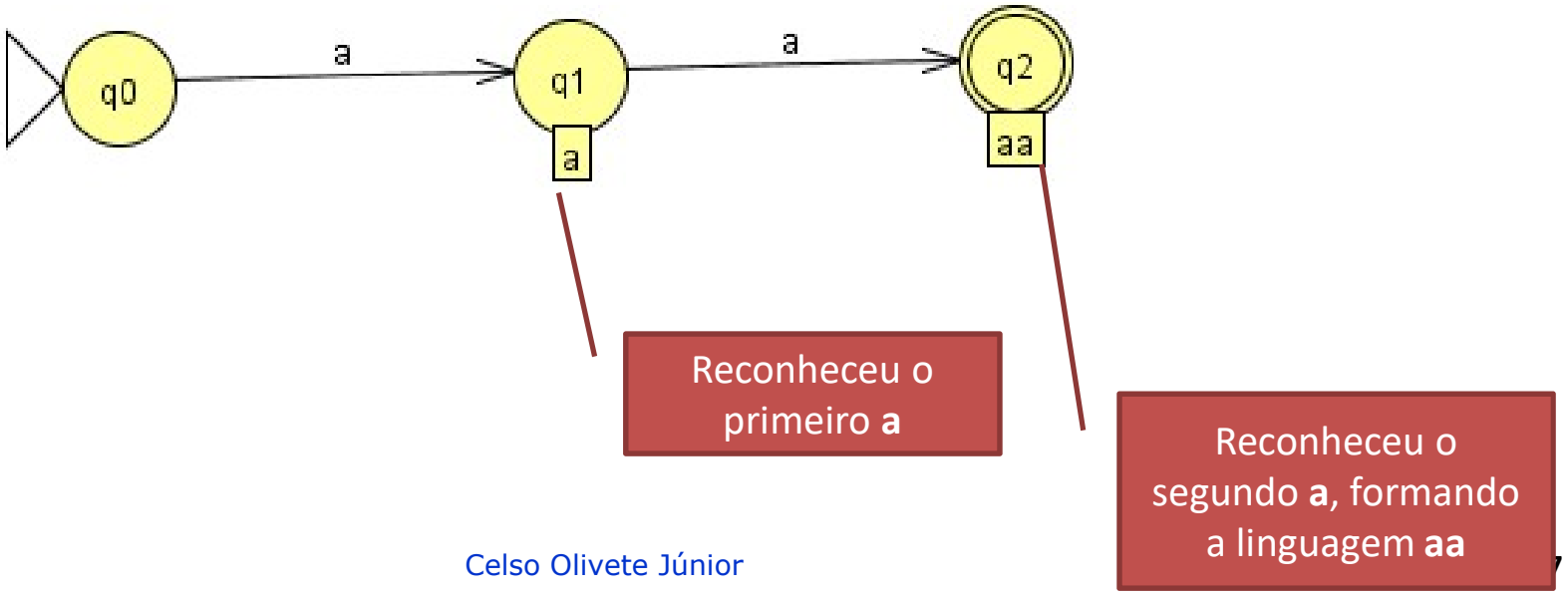
Autômatos finitos (AF) *versus* Expressões Regulares (ER) – como transformar?

ER	Linguagem gerada
aa	somente a palavra aa



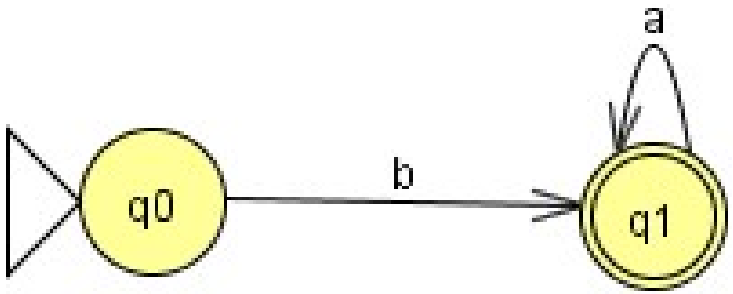
Autômatos finitos (AF) *versus* Expressões Regulares (ER) – como transformar?

ER	Linguagem gerada
aa	somente a palavra aa



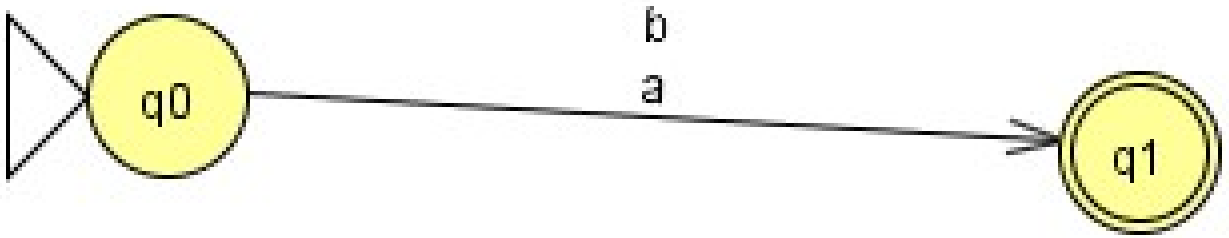
Autômatos finitos (AF) *versus* Expressões Regulares (ER) – como transformar?

ER	Linguagem gerada
ba*	iniciam com b seguido por 0 ou mais a



Autômatos finitos (AF) *versus* Expressões Regulares (ER) – como transformar?

ER	Linguagem gerada
(a+b)	formada por a **ou** por b



Visão Geral - Gramática

- A gramática é um formalismo projetado para a definição de linguagens
- Uma gramática mostra como gerar as palavras de uma linguagem
- Um elemento fundamental das gramáticas é denominado regra

Visão Geral – Gramática

- Exemplo de regra

S → aB
B → b

ou

S → Ba
B → b

- qual a linguagem gerada?
- ideia: "substituia" o símbolo que está em maiúsculo no lado direito da regra por seu significado correspondente encontrado no lado esquerdo:
  - B pode ser substituído por b
  - logo a linguagem gerada é representada pela ER = ab ou ba

Visão Geral – Gramática

S → aB

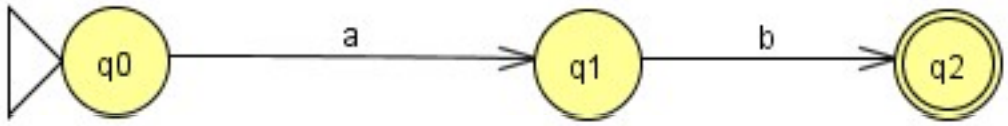
B → b

•linguagem gerada

é representada pela

ER = ab

AF correspondente



Gramática:

- Exemplo 2

$S \rightarrow aS \mid bA \mid \varepsilon$

$A \rightarrow c$

- qual a linguagem gerada?

Gramática:

- Exemplo 2

$S \rightarrow aS \mid bA \mid \varepsilon$

$A \rightarrow c$

AF correspondente

- qual a linguagem gerada?
  - qualquer quantidade de a **ou** bc
- ER = a* + bc

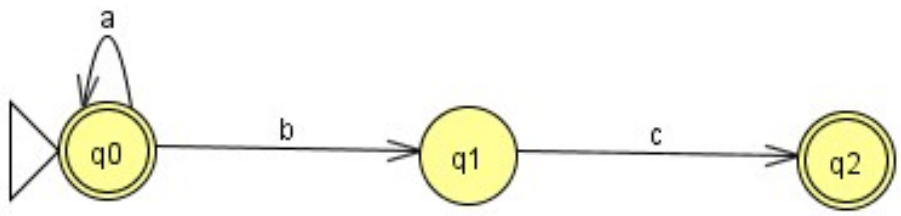
Gramática:

- Exemplo 2

S → aS | bA | ε

A → c

AF correspondente



- qual a linguagem gerada?
  - qualquer quantidade de a **ou** bc
- ER = a* + bc

Gramática: aplicação em compiladores: reconhecimento de números inteiros com sinal

Int → +Dig | -Dig

Dig → 0Dig | 1Dig |... 9Dig | 0 | 1 | ... 9

Gramática: regras que definem o comando WHILE (Pascal)

```
comando    → comandoWhile
comandoWhile → WHILE expr_bool DO comando;
expr_bool → expr_arit < expr_arit
          | expr_arit > expr_arit
          | ...
expr_arit → expr_arit * termo
          | termo
          | ...
termo → expr_arit
      | NÚMERO
      | IDENTIFICADOR
```

Próxima aula:

- Linguagens regulares:
  - Alfabeto
  - String
  - Concatenação
  - Comprimento de uma cadeia
  - Operações: união, concatenação, reverso, fechamento....
  - Expressões regulares
- Projeto:
  - Definição dos grupos