[Aula 03] Linguagens regulares – Autômato finito determinístico (AFD)

Prof. João F. Mari
joaof.mari@ufv.br

BIBLIOGRAFIA

- MENEZES, P. B. Linguagens formais e autômatos, 6. ed., Bookman, 2011.
  - Capítulo 2.
  - + Slides disponibilizados pelo autor do livro.

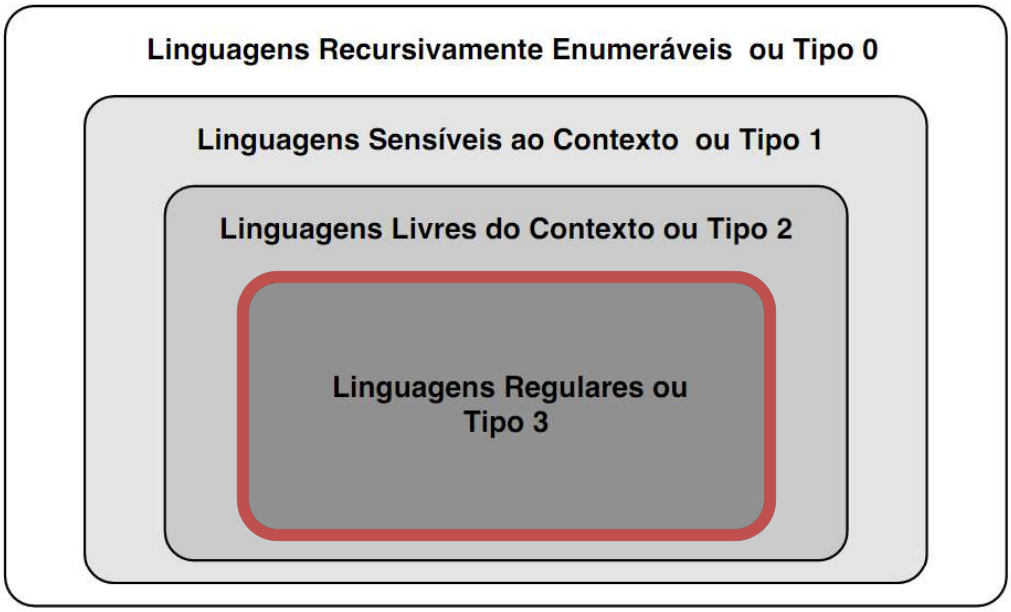
ROTEIRO

- Linguagens regulares
  - Hierarquia de Chomsky
- Autômato finito
  - Autômato finito determinístico (AFD)
  - **[EX]** aa ou bb como subpalavra
  - Função programa estendida, computação
  - **[EX]** Função programa estendida, computação
  - Linguagem aceita e rejeitada
  - Autômatos finitos equivalentes
  - Linguagem regular linguagem do tipo 3
  - **[EX]** Linguagem vazia e todas as palavras
  - **[EX]** Número par de cada símbolo
  - Algoritmo de reconhecimento AFD

Linguagens regulares

- **Linguagens Regulares** ou **Tipo 3**:
  - Formalismos:
    - **Autômato Finito:**
      - Formalismo operacional ou reconhecedor
      - Basicamente, um sistema de estados finitos (máquina de estados finitos)
    - Expressão Regular:
      - Formalismo denotacional ou gerador;
      - Conjuntos (linguagens) básicos + concatenação e união.
    - Gramática Regular:
      - Formalismo axiomático ou gerador;
      - Gramática com restrições da forma das regras de produção.

Hierarquia de Chomsky



Linguagens Regulares

- Hierarquia de Chomsky:
  - Classe de linguagens mais **simples.**
  - Algoritmos de reconhecimento, geração ou conversão entre formalismos:
    - Pouca complexidade;
    - Grande eficiência;
    - Fácil implementação.

```
int main( ) {
   printf("Olá mundo!!!");
}
```

- Fortes limitações de expressividade:
  - **[EX]**
    - **Duplo balanceamento** não é regular.
    - Linguagens de programação em geral são não regulares.

Linguagens Regulares

- Complexidade de algoritmos - autômatos finitos:
  - Classe de algoritmos mais eficientes;
    - Tempo de processamento.
  - Qualquer autômato finito é igualmente eficiente.
  - Qualquer solução é ótima:
    - A menos de eventual redundância de estados.
    - Redundância de estados não influi no tempo:
      - Pode ser facilmente eliminada: Autômato Finito Mínimo.

Linguagens Regulares

- Importantes propriedades - podem ser usadas para:
  - Construir novas linguagens regulares a partir de linguagens regulares conhecidas
  - Provar propriedades.
  - Construir algoritmos.

- Se um problema tiver uma solução regular
  - Considerar preferencialmente a qualquer outra não regular;
    - Eficiência e simplicidade dos algoritmos.

Linguagens Regulares

- Universo de aplicações das linguagens regulares:
  - Muito grande.
  - Constantemente ampliado.

- Exemplo típico e simples:
  - Análise léxica.
  - Compiladores e Interpretadores de linguagens de programação.

- Exemplos mais recentes:
  - Sistemas de animação;
  - Hipertextos;
  - Hipermídias.

AUTÔMATO FINITO

Autômato finito

- Autômato Finito: sistema de estados finitos
  - Número finito e predefinido de estados;
  - Modelo computacional comum em diversos estudos teórico-formais:
    - Linguagens Formais;
    - Compiladores;
    - Modelos para Concorrência.

Autômato finito

- Formalismo operacional/**reconhecedor**.

- Tipos de Autômatos Finitos:
  - **Autômato Finito Determinístico (AFD)**
    - Dependendo do estado corrente e do símbolo lido pode assumir **um único estado**.
  - **Autômato Finito Não Determinístico (AFN)**
    - Dependendo do estado corrente e do símbolo lido pode assumir **um conjunto de estados** alternativos.
  - **Autômato Finito com Movimentos Vazios (AFN-vazio)**
    - Dependendo do estado corrente e **sem ler qualquer símbolo.**
    - Pode assumir um conjunto de estados:
      - Portanto é não determinístico.

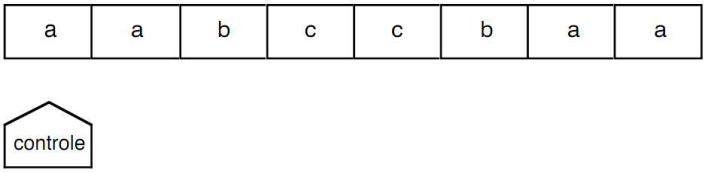
Autômato finito

- AFD, AFN e AFN-vazio são equivalentes em termos de poder computacional.

AUTÔMATO FINITO DETERMINÍSTICO (AFD)

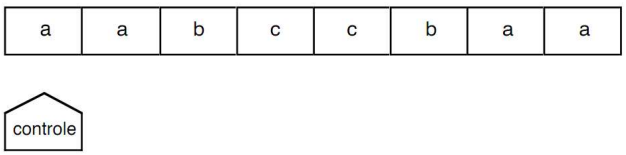
Autômato Finito Determinístico (AFD)

- Máquina constituída por:
  - **Fita:** dispositivo de entrada
    - contém informação **(palavra)** a ser processada.
  - **Unidade de Controle:** reflete o estado corrente da máquina
    - Possui **unidade de leitura** (cabeça da fita);
    - Acessa uma célula da fita de cada vez;
    - Movimenta-se **exclusivamente para a direita.**
  - **Programa, Função Programa ou Função de Transição**
    - Comanda as leituras;
    - Define o estado da máquina.



Autômato Finito Determinístico (AFD)

- Fita é finita
  - Dividida em células e cada célula armazena um símbolo
  - Símbolos pertencem a um alfabeto de entrada
  - Não é possível gravar sobre a fita (não existe memória auxiliar)
  - A palavra a ser processada ocupa toda a fita
- Unidade de Controle
  - Número finito e predefinido de estados
- Leitura
  - Lê o símbolo de uma célula de cada vez
  - Move a cabeça da fita uma célula para a direita
  - Posição inicial da cabeça célula mais à esquerda da fita
- Função Programa
  - Função Parcial
  - Dependendo do estado corrente e do símbolo lido determina o novo estado do autômato.



Autômato Finito Determinístico (AFD)

$$M = (\Sigma, Q, \delta, q_0, F)$$

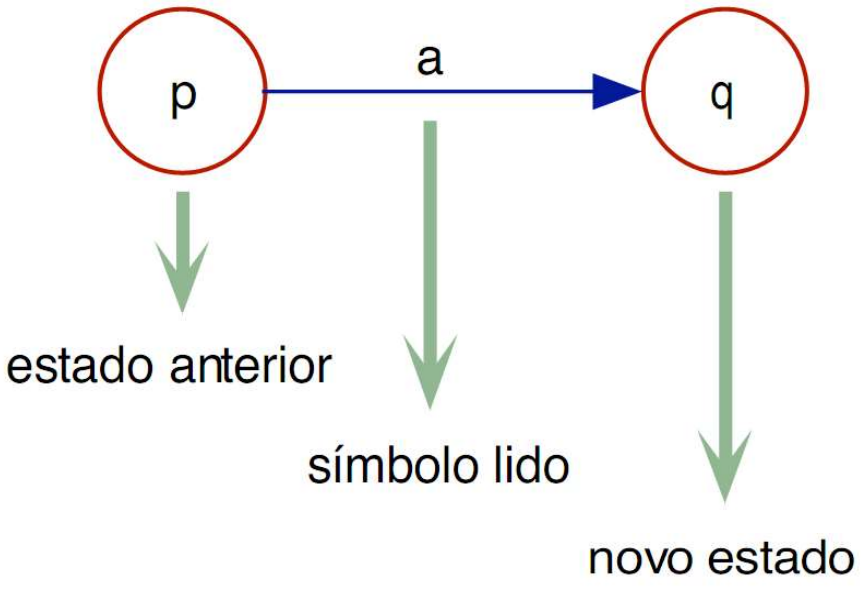
- **Σ**: Alfabeto (de símbolos) de entrada;
- **Q**: Um conjunto de estados possíveis do autômato (finito);
- **δ**: Programa ou Função de Transição (função parcial);

$$\delta: Q \times \Sigma \rightarrow Q$$

  - Transição do autômato: $\delta(p, a) = q$;
- q_0: Estado inicial (é um elemento distinguido de Q);
- **F**: Conjunto de estados finais (é um subconjunto de Q).

Autômato Finito Determinístico (AFD)

- Autômato finito como um diagrama: $\delta(p, a) = q$



Autômato Finito Determinístico (AFD)

- Estados iniciais e finais:

- Transições paralelas: **δ(p, a) = q** e **δ(p, b) = q**

Autômato Finito Determinístico (AFD)

- Função programa como uma tabela de dupla entrada:

$$\delta(p, a) = q$$

δ	a	...
p	q	...
q	...	...

Autômato Finito Determinístico (AFD)

- Computação de um autômato finito:
  - Sucessiva aplicação da **função programa**...
  - para cada símbolo da entrada (da esquerda para a direita)...
  - até ocorrer uma condição de parada.

- Lembre-se que um autômato finito:
  - Não possui memória de trabalho;
  - Para armazenar as informações passadas deve-se usar o conceito de estado;

[EX] aa ou bb como subpalavra

L_1 = { w | w possui aa ou bb como subpalavra }

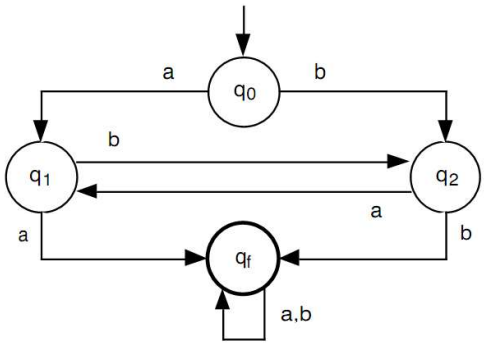
- Autômato finito:

$$M_1 = (\{ a, b \}, \{ q_0, q_1, q_2, q_f \}, \delta_1, q_0, \{ q_f \})$$

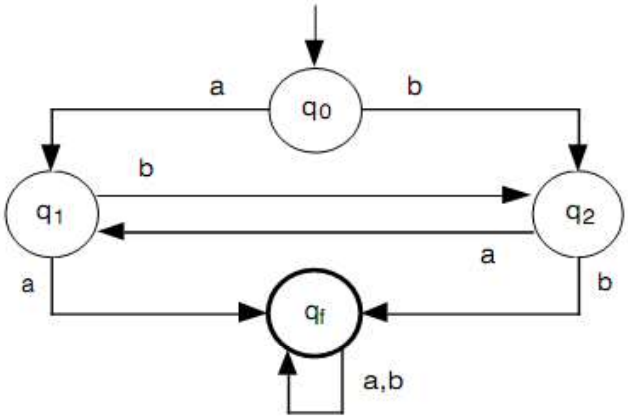
δ_1	a	b
q_0	q_1	q_2
q_1	q_f	q_2
q_2	q_1	q_f
q_f	q_f	q_f

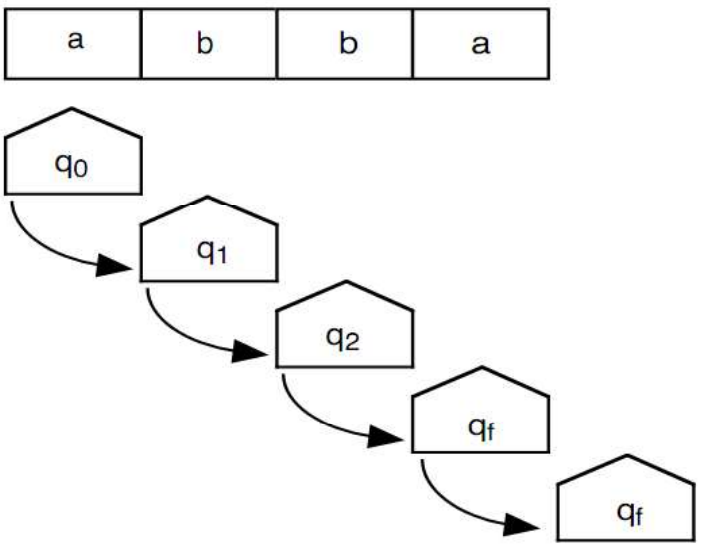
[EX] aa ou bb como subpalavra

- Qual a informação memorizada por q_1?
  - símbolo anterior é **a.**
- Qual a informação memorizada por q_2?
  - símbolo anterior é **b.**
- Qual a informação memorizada por q_0?
  - ???
- Qual a informação memorizada por q_f?
  - Após identificar **aa** ou **bb** q_f (final) varre o sufixo da entrada.



[EX] aa ou bb como subpalavra





Autômato Finito Determinístico (AFD)

- O Autômato Finito **sempre para**!

- Como?
  - Qualquer palavra é finita.
  - Um novo símbolo é lido a cada aplicação da função programa.
  - Não existe a possibilidade de ciclo (loop) infinito.

- Parada do processamento:
  - **Aceita a entrada**:
    - Após processar o último símbolo, assume um estado final.
  - **Rejeita a entrada** (duas possibilidades):
    - Após processar o último símbolo, assume um estado não-final;
    - Programa indefinido para argumento (estado e símbolo).

Função Programa Estendida, Computação

- M = (Σ, Q, δ, q0, F) autômato finito determinístico.

$$\delta^*: Q \times \Sigma^* \rightarrow Q$$

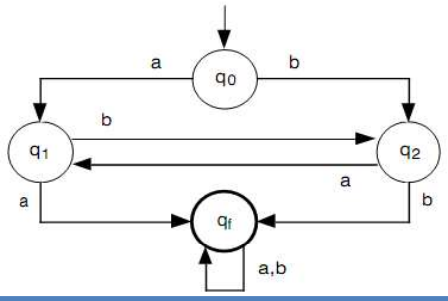
- É $\delta: Q \times \Sigma \rightarrow Q$ estendida para palavras - indutivamente definida
  - $\delta^*(q, \varepsilon) = q$
  - $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$

- Observe:
  - Sucessiva aplicação da função programa para cada símbolo da palavra a partir de um dado estado.
  - Se a entrada for vazia, fica parado.
  - Aceita/rejeita: função programa estendida a partir do estado inicial.

- Objetivando simplificar a notação:
  - δ e a sua extensão δ^* podem ser ambas denotadas por δ.

[EX] Função programa estendida

- $\delta^*(q_0, abaa)$ = função estendida sobre abaa
- $\delta^*(\delta(q_0, a), baa)$ = processa
- $\delta^*(q_1, baa)$ = função estendida sobre baa abaa
- $\delta^*(\delta(q_1, b), aa)$ = processa baa
- $\delta^*(q_2, aa)$ = função estendida sobre aa
- $\delta^*(\delta(q_2, a), a)$ = processa aa
- $\delta^*(q_1, a)$ = função estendida sobre a
- $\delta^*(\delta(q_1, a), \varepsilon)$ = processa a
- $\delta^*(q_f, \varepsilon)$ = $\mathbf{q_f}$ função estendida sobre ε: fim da indução; ACEITA



Linguagem Aceita, Linguagem Rejeitada

- M = (Σ, Q, δ, q0, F) autômato finito determinístico.

- **Linguagem Aceita** ou Linguagem Reconhecida por M

  $L(M) = ACEITA(M) = \{ w \mid \delta^*(q0, w) \in F \}$

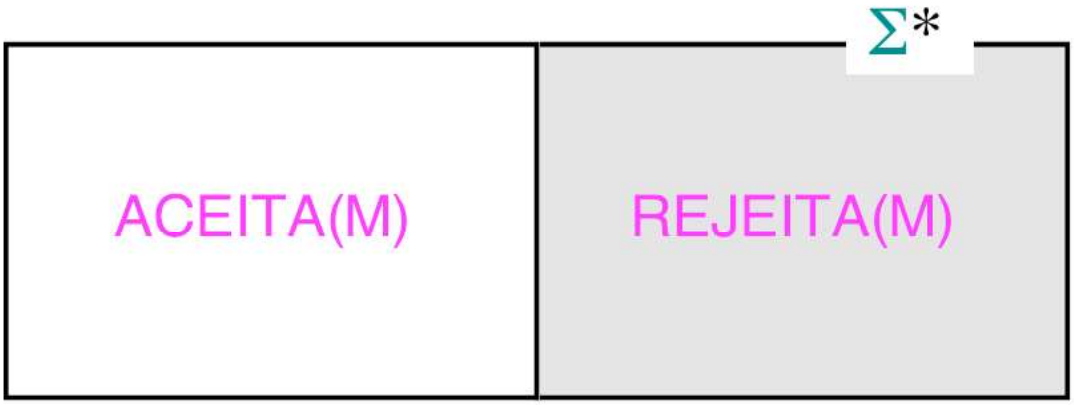
- **Linguagem Rejeitada** por M:

  $REJEITA(M) = \{ w \mid \delta^*(q0, w) \notin F$ ou $\delta^*(q0, w)$ é indefinida $\}$

- Supondo que Σ* é o conjunto universo:
  - $ACEITA(M) \cap REJEITA(M) = \emptyset$
  - $ACEITA(M) \cup REJEITA(M) = \Sigma^*$
  - ~ACEITA(M) = REJEITA(M)
  - ~REJEITA(M) = ACEITA(M)

Linguagem Aceita, Linguagem Rejeitada

- Cada autômato finito M sobre Σ induz uma partição de Σ* em duas classes de equivalência.



Autômatos Finitos Equivalentes

- Diferentes autômatos finitos podem aceitar uma mesma linguagem.

- M_1 e M_2 são Autômatos Finitos Equivalentes se e somente se:

$$ACEITA(M_1) = ACEITA(M_2)$$

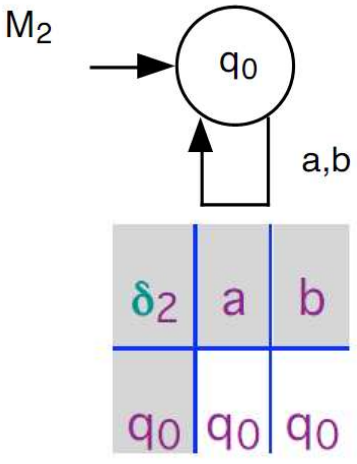
Linguagem Regular, Linguagem Tipo 3

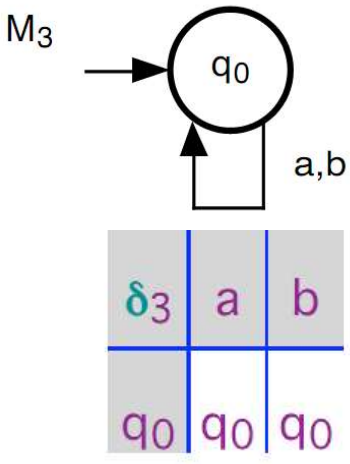
- **L** é uma Linguagem Regular ou Linguagem Tipo 3:
  - existe pelo menos um autômato finito determinístico que aceita **L**.

[EX] Linguagem vazia e todas as palavras

- Linguagens sobre o alfabeto { a, b }

$$L_2 = \emptyset \quad e \quad L_3 = \Sigma^*$$

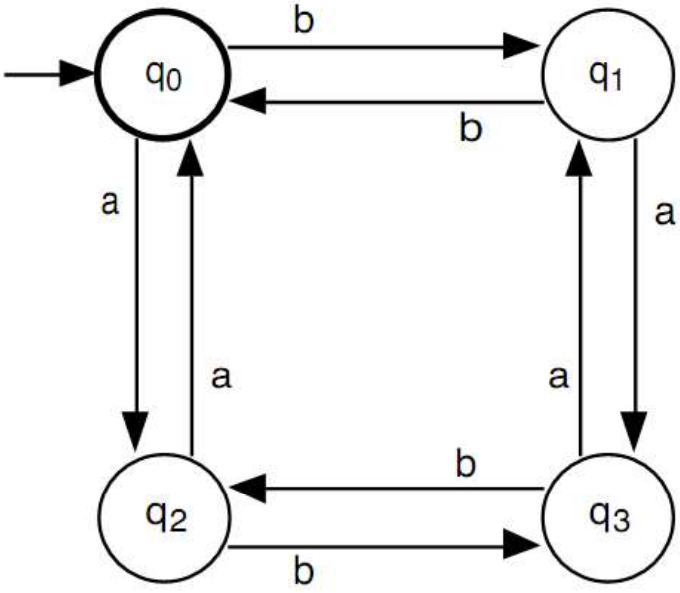




  - Qual a diferença entre δ_2 e δ_3?
  - O que, exatamente, diferencia M_2 de M_3?

[EX] Número par de cada símbolo

- L4 = { w | w possui um número par de a e um número par de b }



[EX] Número par de cada símbolo

- Para pensar:
  - Como seria para aceitar um número ímpar de cada símbolo?

Algoritmo de reconhecimento AFD

Início

Estado Atual ← Estado Inicial;

PARA I variar do Símbolo inicial da fita até o símbolo final FAÇA

SE Existe δ (Estado Atual, I) ENTÃO

Estado Atual ← δ (Estado Atual, I);

SENÃO

REJEITA;

SE Estado Atual é estado final ENTÃO

ACEITA;

SENÃO

REJEITA;

Fim

DESAFIO

- PROBLEMA: Um **Homem** quer atravessar um rio levando consigo um **Lobo**, uma **Cabra** e um **Repolho** e no bote só cabem **ele** e mais um dos outros três.
  - Não podem ficar sozinhos (sem o Homem) em nenhuma das margens: a Cabra com o Repolho e o Lobo com a Cabra. Os motivos são óbvios!

- Exemplos de possíveis estados do sistema:
  - **<HLCR-0>** - todos na margem esquerda
  - **<L-HCR>** - lobo na margem esquerda, cabra e repolho na direita

- Entradas do sistema:
  - ***h*** - homem atravessa o rio sozinho
  - ***l*** - homem atravessa o rio com o lobo
  - ***c*** - homem atravessa o rio com a cabra
  - ***r*** - homem atravessa o rio com o repolho

[FIM]

- FIM:
  - **[AULA 03]** LINGUAGENS REGULARES – Autômato Finito Determinístico
- Próxima aula:
  - **[AULA 04]** LINGUAGENS REGULARES – Autômato Finito Não Determinístico