Aspectos Teóricos da Computação

Prof. Rodrigo Martins
rodrigo.martins@francomontoro.com.br

Cronograma da Aula

- Aula Inaugural
- Apresentação da disciplina
- Critérios de Avaliação
- Metodologia de Trabalho

Ementa

- Gramáticas, máquinas de estados finitos, autômatos, expressões regulares e suas aplicações em linguagens formais, compiladores, síntese de circuitos sequenciais, protocolos de comunicação.

Objetivos da Disciplina

- Proporcionar uma visão abrangente dos fundamentos da teoria dos processos efetivos (algoritmos e programas), por meio do estudo de vários modelos formais de computação, como base para a especificação e a análise de algoritmos.

- Dentre as aptidões, competências e habilidades esperadas vislumbra-se: projetar e implementar algoritmos básicos por meio de autômatos finitos; formular gramáticas como meio de geração e reconhecimento de linguagens; desenvolver árvores de derivação para reconhecimento de vários tipos de linguagem artificiais ou naturais; relacionar o tipo de gramática ao tipo de linguagem a ser produzido ou reconhecido

Critérios de Avaliação

T1 – Lista de Exercícios/Trabalhos/Projetos (30%)
P1 - Avaliação Bimestral (P1) (70%)

Media Bimestral: T1 * 0.30 + P1 * 0.70

T2 - Lista de Exercícios/Trabalhos/Projetos (30%)
P2 – Avaliação Bimestral (P2) (70%)

Média Final – (MB1 + MB2) / 2

Média final maior ou igual a 7,0 (sete) implicará em aprovação sem exame final;
Média final igual ou superior a 4,0 (quatro) e inferior a 7,0 (sete) dependerá de aprovação em exame final;
Média final de aproveitamento inferior a 4,0 (quatro) implicará em reprovação;
A aprovação em exame final será obtida se a média aritmética da média final de aproveitamento com a nota do exame final for igual ou superior a 5,0 (cinco).

Metodologia de Trabalho

- Aulas expositivas com exercícios.
- Uso do simulador JFLAP.
- Lista de exercícios para serem resolvidas para fixação dos assuntos abordados nas aulas expositivas.
- Uso de metodologias ativas com a const

Bibliografia Básica

- HOPCROFT, J.E.; MOTWANI, R.; ULLMAN, J.D. Introdução a Teoria de Autômatos, Linguagens e Computação. Campus, 2002.

- PAPADIMITRIOU, C., HARRY, L., Elementos de Teoria da Computação, Bookman, 2000.

- MENEZES, P. B. Linguagens Formais e Autômatos. Editora SagraLuzzato, 2000.

Bibliografia Complementar

- MENEZES, P. F.B., DIVERIO, T.A., Teoria da Computação, Sagra-Luzzatto, 1999.

Introdução a Autômatos

A teoria de autômatos é o estudo dos dispositivos de computação abstratos, ou “máquinas”. Antes de existirem os computadores, na década de 1930. Alan Turing estudou uma máquina abstrata que tinha todas as características dos computadores atuais, pelo menos no que se refere ao quanto eles poderiam calcular.

O objetivo de Turing era descrever com exatidão o limite entre o que uma máquina de computação podia fazer e aquilo que ela não podia fazer, suas conclusões se aplicam não apenas às suas máquinas de Turing abstratas, mas também às máquinas reais de hoje.

Introdução a Autômatos

Nas décadas de 1940 e 1950, tipos de máquinas mais simples, que hoje chamamos “autômatos finitos”.

Também no final dos anos 50, Chomsky iniciou o estudo de “gramaticas” formais, e hoje servem como base de alguns importantes componentes de software, como compiladores.

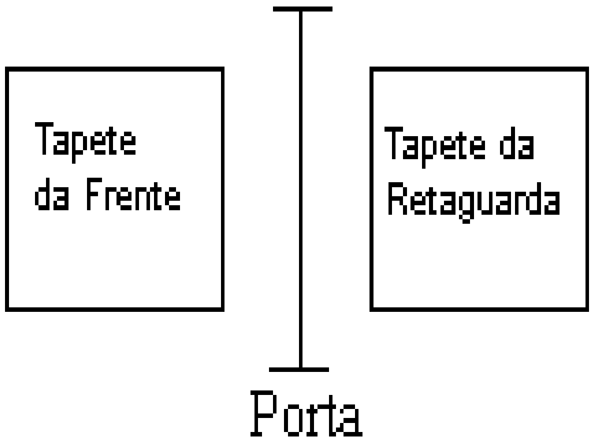
Introdução aos autômatos finitos

Os autômatos finitos constituem um modelo útil para muitos elementos importantes de hardware e software, como:

1. Software para projetar e verificar o comportamento de circuitos digitais.
2. O “analisador léxico” de um compilador típico.
3. Softwares para examinar grandes corpos de texto, como coleções de páginas web, a fim de encontrar ocorrências de palavras, frases ou outros padrões.
4. Software para verificar sistemas de todos os tipos que tem um número finito de estados distintos, como protocolos de comunicações ou protocolos para troca segura de informações.

Autômatos Finitos

Exemplo 1



Visão aérea de uma porta automática

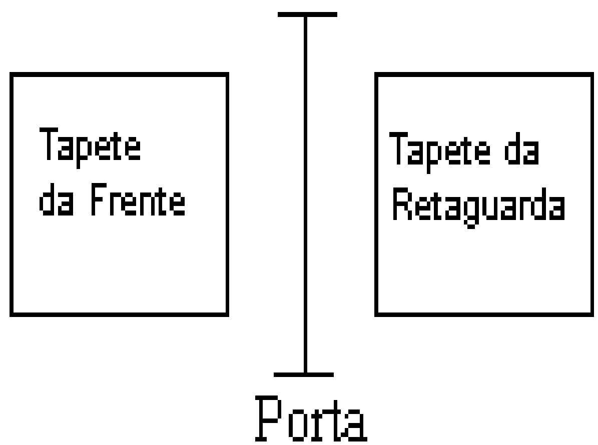
Autômatos Finitos

Exemplo 1

O controlador da porta pode estar em 2 estados:

- aberto (significando porta aberta)
- fechado (significando porta fechada)

O controlador passa de um estado para outro dependendo do estímulo (entrada) que recebe:



Autômatos Finitos

Exemplo 1

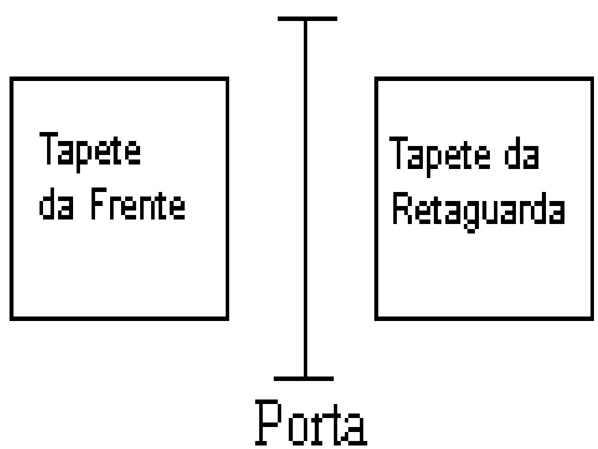
Entradas: Existem 4 condições de entradas possíveis

Frente: significando que uma pessoa está em pé sobre o tapete da frente;

Retaguarda: significando que uma pessoa está em pé sobre o tapete de dentro;

Ambos: significando que existem pessoas sobre os 2 tapetes;

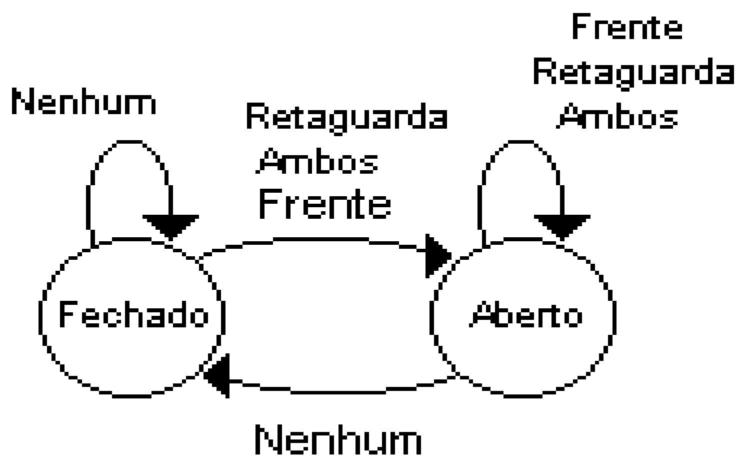
Nenhum: significando que ninguém está sobre os tapetes.



Autômatos Finitos

Exemplo 1

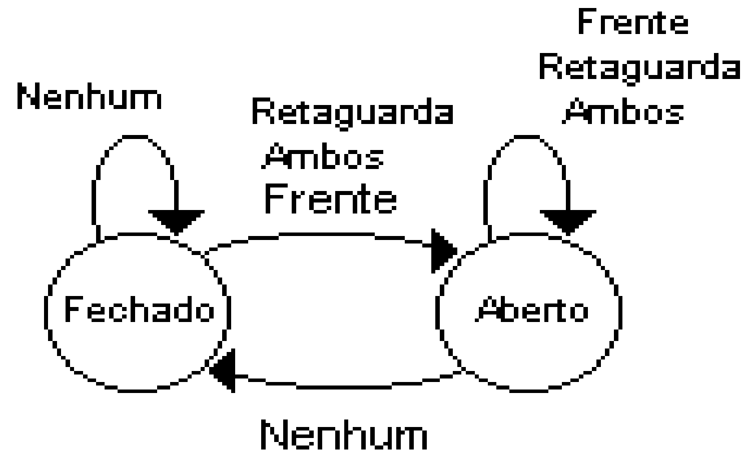
Entradas: Existem 4 condições de entradas possíveis



Autômatos Finitos

Exemplo 1

Tabela de Transição



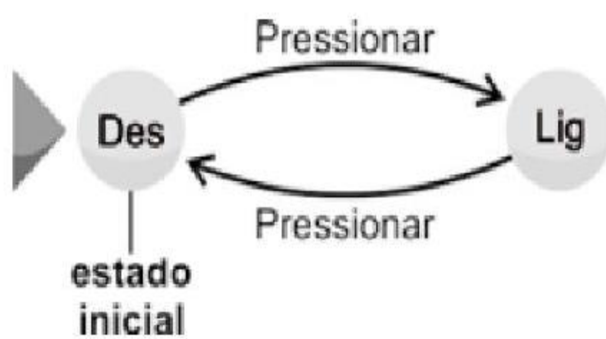
	NENHUM	FRENTE	RETAGUARDA	AMBOS
FECHADO	FECHADO	ABERTO	ABERTO	ABERTO
ABERTO	FECHADO	ABERTO	ABERTO	ABERTO

Autômatos Finitos

Exemplo 2

Interruptor

Estados: Ligado ou Desligado

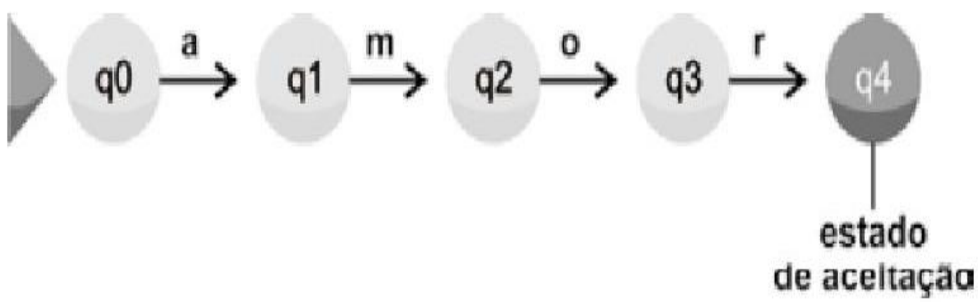


Autômatos Finitos

Exemplo 3

Palavra AMOR

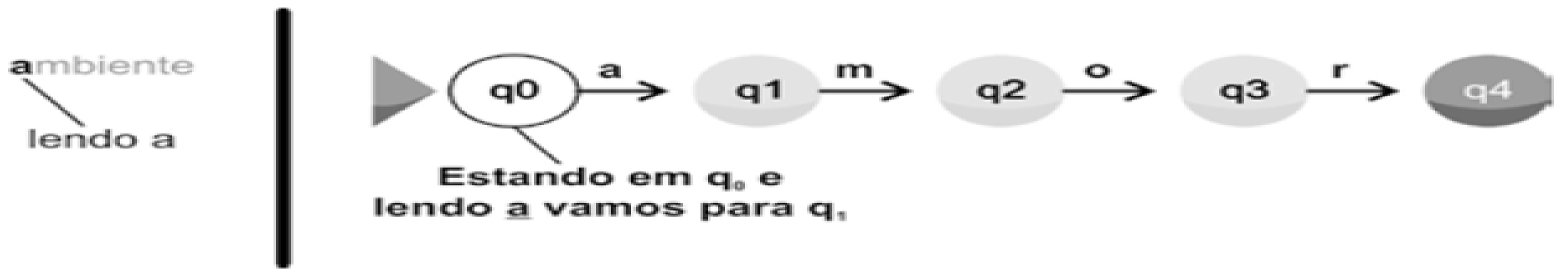
Qualquer outra palavra deve ser descartada



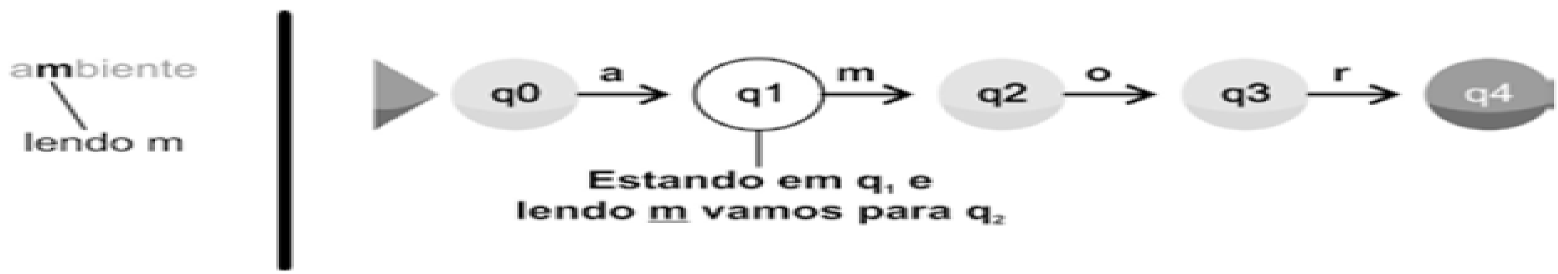
Autômatos Finitos

Exemplo 3

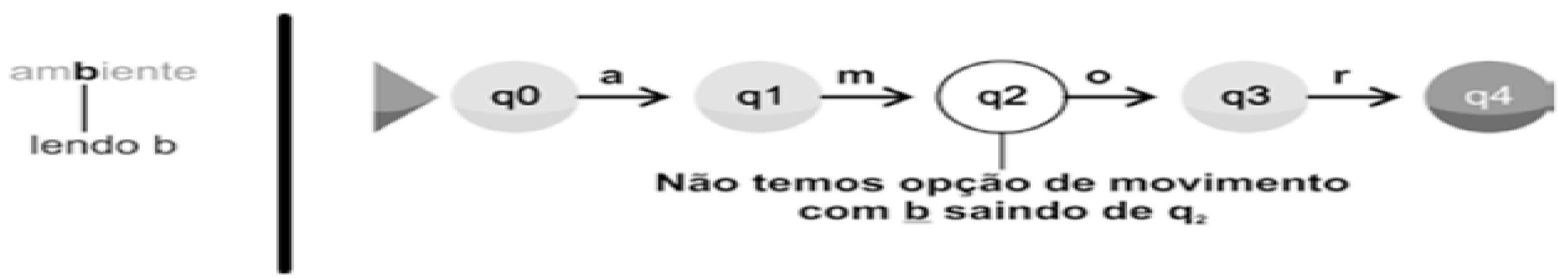
Tentativa de leitura da Palavra AMBIENTE



Partimos para a segunda letra:



Para a letra b:



Os conceitos centrais da teoria de autômatos

Introduziremos as definições mais importantes dos termos que permaneiam a teoria de autômatos.

Esses conceitos incluem o “alfabeto” (um conjunto de símbolos), “strings” (uma lista de símbolos de um alfabeto) e “linguagem” (um conjunto de strings de um mesmo alfabeto).

Alfabetos

Um alfabeto é um conjunto de símbolos finito e não-vazio. Convencionalmente, usamos o símbolo $\sum$ para um alfabeto. Os alfabetos comuns incluem:

- $\sum = \{0, 1\}$, o alfabeto binário.
- $\sum = \{a, b, ..., z\}$, o conjunto de todas as letras minúsculas.
- O conjunto de todos os caracteres ASCII, ou o conjunto de todos os caracteres ASCII imprimíveis.

Strings

Um string (ou às vezes **palavra** ou também **cadeia**) é uma sequencia finita de símbolos escolhidos de algum alfabeto. Por exemplo, 01101 é um string do alfabeto binário $\sum = \{0,1\}$. O string 111 é outro string escolhido nesse alfabeto.

O Strings vazio

O string vazio é o string com zero ocorrências de símbolos. Esse string, denotado por ε (épsilon), é um string que pode ser escolhido de qualquer alfabeto.

Comprimento de um String

Com frequência, é útil para classificar strings por seu **comprimento**, isto é, o número de posições para símbolos no string.

Por exemplo, 01101 tem comprimento 5.

A notação padrão para o comprimento de um string w é |w|.
Por exemplo:

$$|011| = 3$$

$$|\varepsilon| = 0$$

Potências de um alfabeto

Se $\sum$ é um alfabeto, podemos expressar o conjunto de todos os strings de um certo comprimento a parte desse alfabeto, usando uma notação exponencial.

Definimos $\sum^k$ como o conjunto de strings de comprimento k, e o símbolo de cada um deles está em $\sum$.

Potências de um alfabeto

Exemplo: Observe que $\sum^k$ = {f}, independente de qual alfabeto seja $\sum$ .Isto é, ε é o único string cujo comprimento é 0.

Se $\Sigma = \{0,1\}$, então $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$,

$$\Sigma^3 = \{000,001,010,011,100,101,110,111\}$$

Observe que existe uma ligeira confusão entre $\sum$ e $\sum^1$

O primeiro $\sum$ é um alfabeto, seus elementos 0 e 1 são símbolos.

O outro $\sum^1$ é um conjunto de strings, seus elementos são os strings 0 e 1, cada um dos quais tem comprimento 1.

Potências de um alfabeto

O conjunto de todos os strings sobre um alfabeto Σ é denotado convencionalmente por Σ^* .

Por exemplo:

$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$.

Em outros termos:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

Potências de um alfabeto

As vezes, desejamos excluir o string vazio do conjunto de strings. O conjunto de strings não vazios do alfabeto ∑ é denotado por $\sum^+$. Desse modo, duas equivalências apropriadas são:

$$\Sigma^+ = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots.$$

$$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}.$$

Concatenação de strings

Sejam os strings x e y. Então, xy denota a concatenação de x e y, isto é, o string formado tomando-se uma cópia de x e acrescentando-se a ela uma cópia de y. Mais precisamente, se x é o string composto de i símbolos $x = a_1a_2 \ldots a_i$ e y é o string composto de j símbolos $y = b_1b_2 \ldots b_j$, então xy é o string de comprimento i + j: $xy = a_1a_2 \ldots a_1b_1b_2 \ldots b_j$.

Concatenação de strings

Exemplo: Sejam x = 01101 e y = 110.

Então

xy = 01101110 e

yx = 11001101

Linguagens

Um conjunto de strings, todos escolhidos a partir de algum $\sum^*$, onde $\sum$ é um alfabeto específico, é chamado **linguagem**.

Linguagens comuns podem ser vistas como conjunto de strings. Um exemplo é o português, no qual uma coleção de palavra válidas em português é um conjunto de strings sobre o alfabeto que consiste em todas as letras.

Linguagens

Outro exemplo, são as linguagens de programação, na qual os programas válidos são um subconjunto dos strings possíveis que podem ser formados a partir do alfabeto da linguagem.

O alfabeto exato pode diferir entre diferentes linguagens de programação, por exemplo, as letras maiúsculas e minúsculas, os dígitos, a pontuação e símbolos matemáticos.

Linguagens

Contudo, também existem muitas outras linguagens, por exemplo:

1. A linguagem de todos os strings que consistem em n 0's seguidos por n valores 1, para algum n >=0. {ε, 01, 0011,000111, ...}.

2. O conjunto de strings de 0's e 1's com um número igual de cada um deles: {ε,01, 10, 0011, 0101, 1001, ...}

3. O conjunto de números binários cujo valor decimal é um número primo: {10, 11, 101, 111, 1011, ...}

Descrevendo uma linguagem

Podemos descrever uma linguagem seja ela finita ou infinita.

Formalismos matemáticos

- Existem três tipos de formalismos:

Tipos de Formalismos

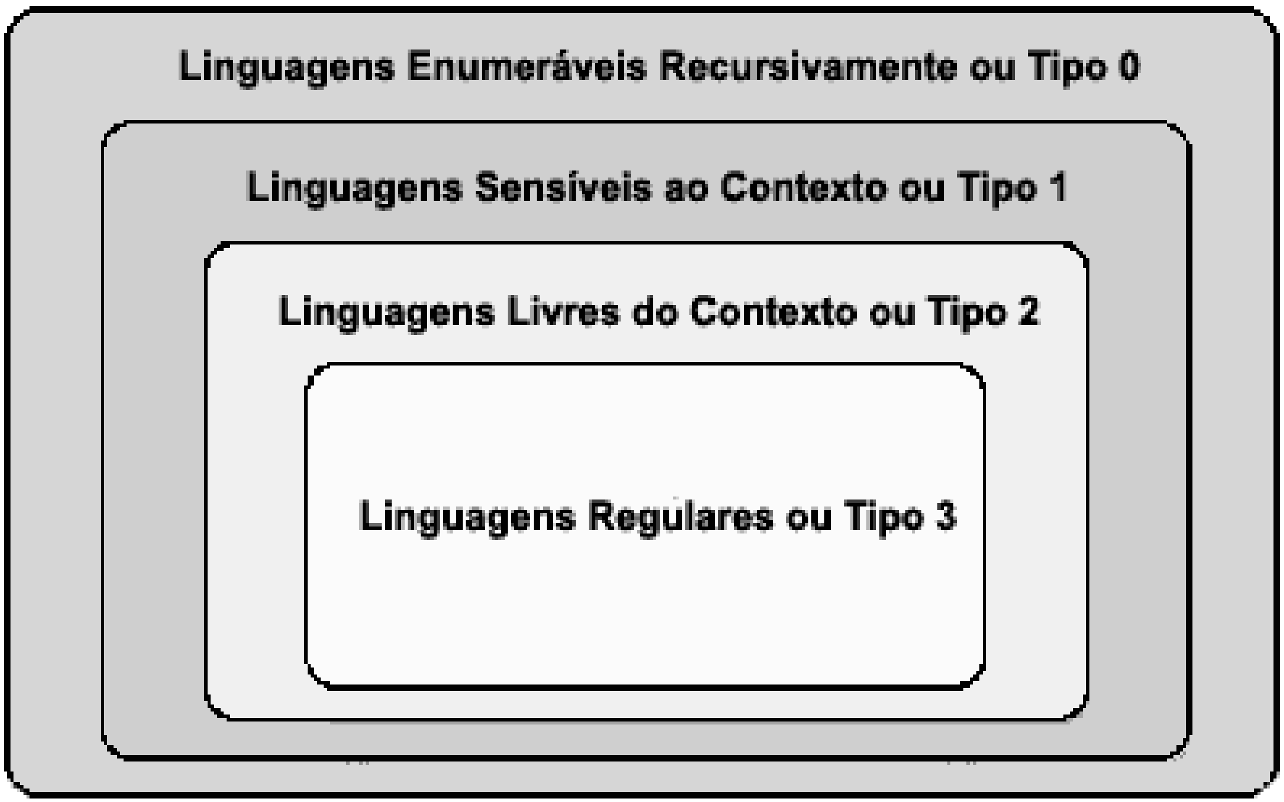
- Reconhecedores
  - Recebe uma palavra e retorna um valor para dizer se ela é ou não da linguagem
- Geradores
  - Define um conjunto de regras que podem ser combinadas para gerar palavras
- Denotacional
  - Uma expressão que denota de modo geral as palavras da linguagem

Classificação das Linguagens

- Hierarquia de Chomsky
  - Quatro categorias hierárquicas
  - Categorias superiores incluem todas as demais
  - Cada categoria é reconhecida por certos formalismos característicos

Classificação das Linguagens

- Hierarquia de Chomsky



Linguagens Regulares

- Chamadas de Linguagens Tipo 3
  - Classe mais simples e restrita

Linguagens Regulares

- Formalismo reconhecedor
  - Recebe uma palavra de entrada
  - Indica se ela é aceita ou rejeitada

- Baseado no conceito de “máquinas de estados finitas”

Exercício

Fazer um AF M tal que $L(M) = \{w \mid w \in \{0,1\}^*$ e possui um número par de ocorrências de 0´s e de 1´s $\}$

Referências desta aula

HOPCROFT, John E.; MOTWANI, Rajeey; ULLMAN, Jeffrey D. Introdução a teoria de autômatos, linguagens e computação. Rio de Janeiro: Campus, 2002.

FIM

Obrigado

Rodrigo