Aspectos Teóricos da Computação

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

Cronograma da Aula

- Introdução a autômatos
- Introdução aos autômatos finitos
- Os conceitos centrais da teoria de autômatos

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.

Introdução a Autômatos

- 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 a Autômatos

- Em 1969 S. Cook estendeu o estudo de Turing do que podia e do que não podia ser calculado.

- Cook conseguiu separar os problemas que podem ser resolvidos de forma eficiente por computadores daqueles problemas que podem em princípio ser resolvidos mas que, na prática, levam tanto tempo que os computadores são inúteis para solucionar todas as instâncias do problema, exceto aquelas muito pequenas.

Introdução a Autômatos

- Os problemas dessa última classe são chamados “intratáveis” ou “NP-difíceis” (NP-hard).

- É altamente improvável que até mesmo a melhoria exponencial na velocidade de computação que o hardware de computadores vem alcançando (“Lei de Moore”) tenha impacto significativo sobre nossa habilidade para resolver grandes instâncias de problemas intratáveis.

Porque estudar a teoria de autômatos?

- **Fundamentos da Computação:**

  - A teoria de autômatos fornece a base teórica para entender o que é computação.

  - Ela ajuda a definir formalmente o que significa "calcular" e quais problemas podem ser resolvidos usando um computador.

Porque estudar a teoria de autômatos?

- **Classificação de Problemas:**

  - Através da teoria de autômatos, podemos classificar problemas com base em sua complexidade e determinar a classe de autômatos necessária para resolver cada tipo de problema.

Porque estudar a teoria de autômatos?

- **Projeto de Linguagens de Programação:**
  - Compreender autômatos é essencial para o design e análise de linguagens de programação.
  - A teoria fornece as ferramentas necessárias para construir analisadores e interpretadores.

Porque estudar a teoria de autômatos?

- **Desenvolvimento de Compiladores:**

  - A teoria de autômatos é crucial no desenvolvimento de compiladores, especialmente na construção de análises léxicas e sintáticas, que são as primeiras fases da compilação.

Porque estudar a teoria de autômatos?

- **Eficiência Computacional:**
  - Estudar autômatos permite compreender como otimizar algoritmos para que sejam tão eficientes quanto possível, considerando tempo e espaço.

Porque estudar a teoria de autômatos?

- **Entendimento de Linguagens Formais:**
  - A teoria de autômatos nos ensina sobre linguagens formais e como elas são estruturadas, o que é importante para a análise de dados e o processamento de linguagem natural.

Porque estudar a teoria de autômatos?

- **Aplicações Práticas:**

  - A teoria de autômatos tem aplicações em diversas áreas, como expressões regulares usadas em busca e substituição de texto em editores de texto e engenharia de software, onde é usada para modelar estados de um sistema.

Porque estudar a teoria de autômatos?

- **Base para Computação Avançada:**

  - Ela serve como alicerce para estudos mais avançados em teoria da computação, como a complexidade computacional, a teoria dos jogos computacionais e a criptografia.

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.

Introdução aos autômatos finitos

- Existem muitos sistemas ou componentes, como aqueles enumerados anteriormente, que podemos considerar como estando, a todo momento, em um de um número finito de “estados”.

- O propósito de um estado é memorizar a porção relevante da história do sistema.

- Tendo em vista que existe apenas um número finito de estados, a história inteira em geral não pode ser memorizada, e assim o sistema tem de ser projetado com cuidado, a fim de memorizar o que é importante e esquecer o que não é.

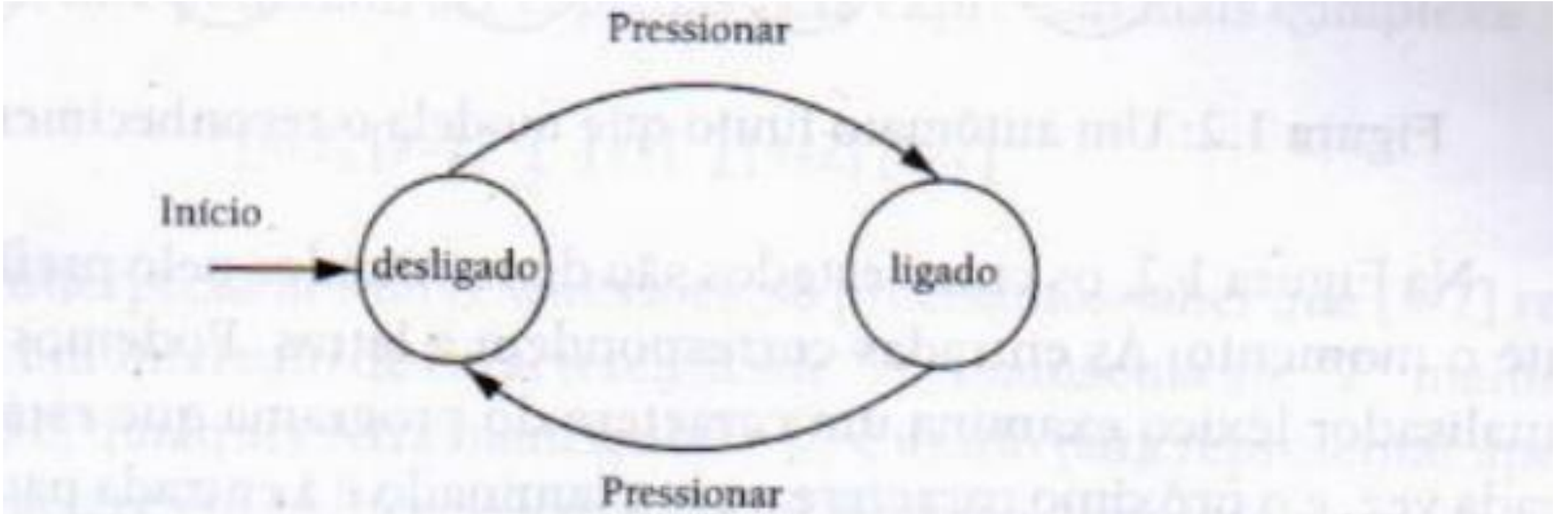
Introdução aos autômatos finitos

- A vantagem de ser apenas um número finito de estados é que podemos implementar o sistema com um conjunto fixo de recursos.

- Por exemplo, poderíamos implementá-lo em hardware como um circuito, ou como uma forma simples de programa que possa tomar decisões examinando somente uma quantidade limitada de dados ou usando a posição no próprio código para tomar a decisão.

Introdução aos autômatos finitos

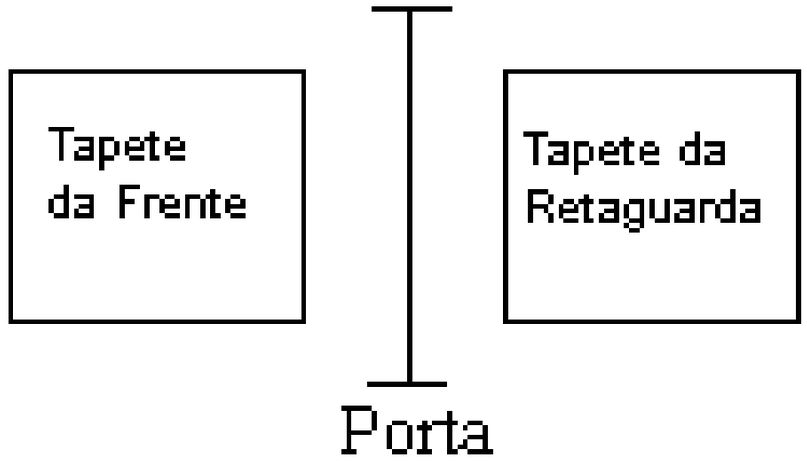
- O autômato finito não trivial mais simples é um interruptor liga/desliga.



Um autômato finito que modela um interruptor liga/desliga

Introdução aos autômatos finitos

- **Exemplo 2**



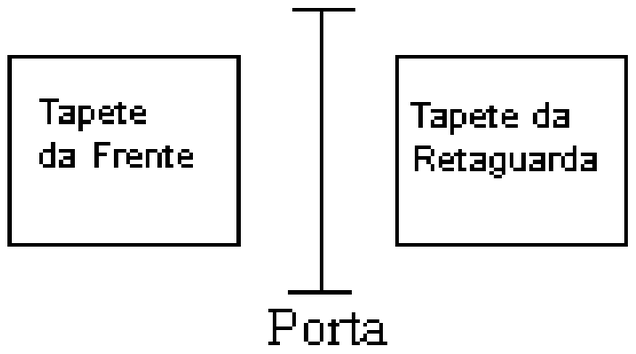
Visão aérea de uma porta automática

Introdução aos autômatos finitos

- **Exemplo 2**

  - O controlador da porta pode estar em 2 estados:
  - aberto (significando porta aberta)
  - fechado (significando porta fechada)

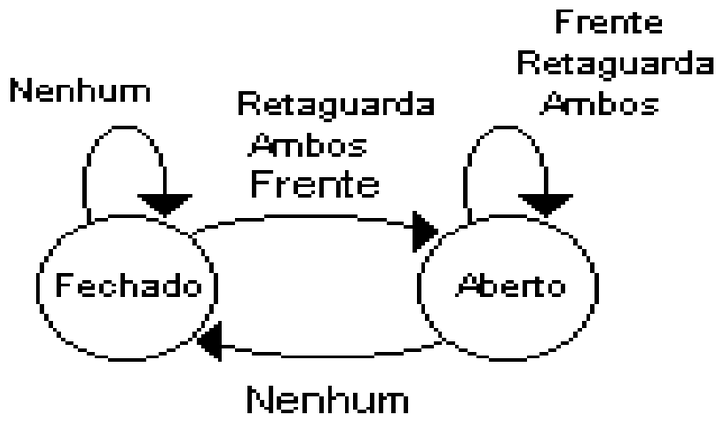
  O controlador passa de um estado para outro dependendo do estímulo (entrada) que recebe:



Visão aérea de uma porta automática

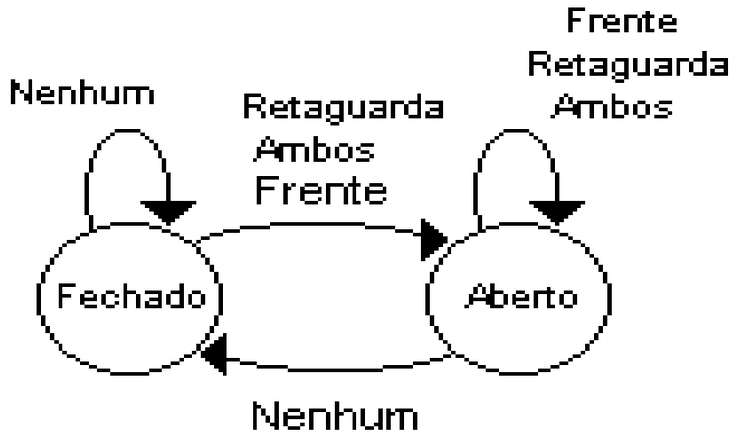
Introdução aos autômatos finitos

- **Exemplo 2**
  - **Entradas:** Existem 4 condições de entradas possíveis



Introdução aos autômatos finitos

- **Exemplo 2**
- **Tabela de Transição**



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

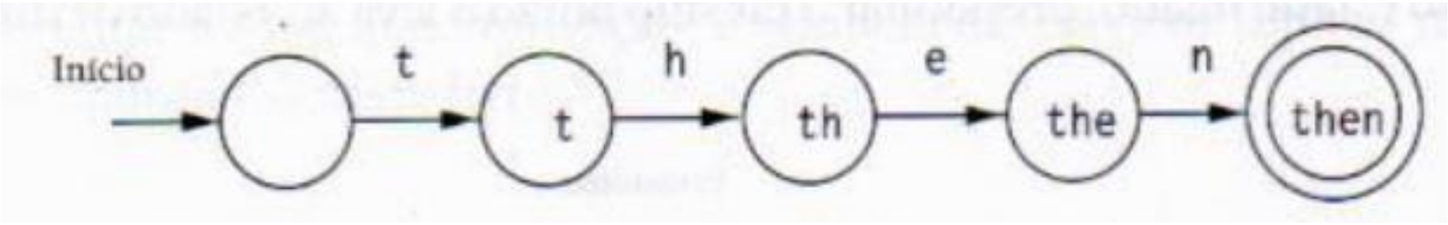
Introdução aos autômatos finitos

- Às vezes, o que é memorizado por um estado pode ser muito mais complexo que uma escolha entre ligado/desligado. A figura abaixo mostra outro autômato finito que poderia fazer parte de um analisador léxico.

- O trabalho desse autômato é reconhecer a palavra **then**.

- Desse modo, ele precisa de cinco estados, cada um dos quais representa uma posição diferente na palavra then, conforme o que foi atingido até o momento.

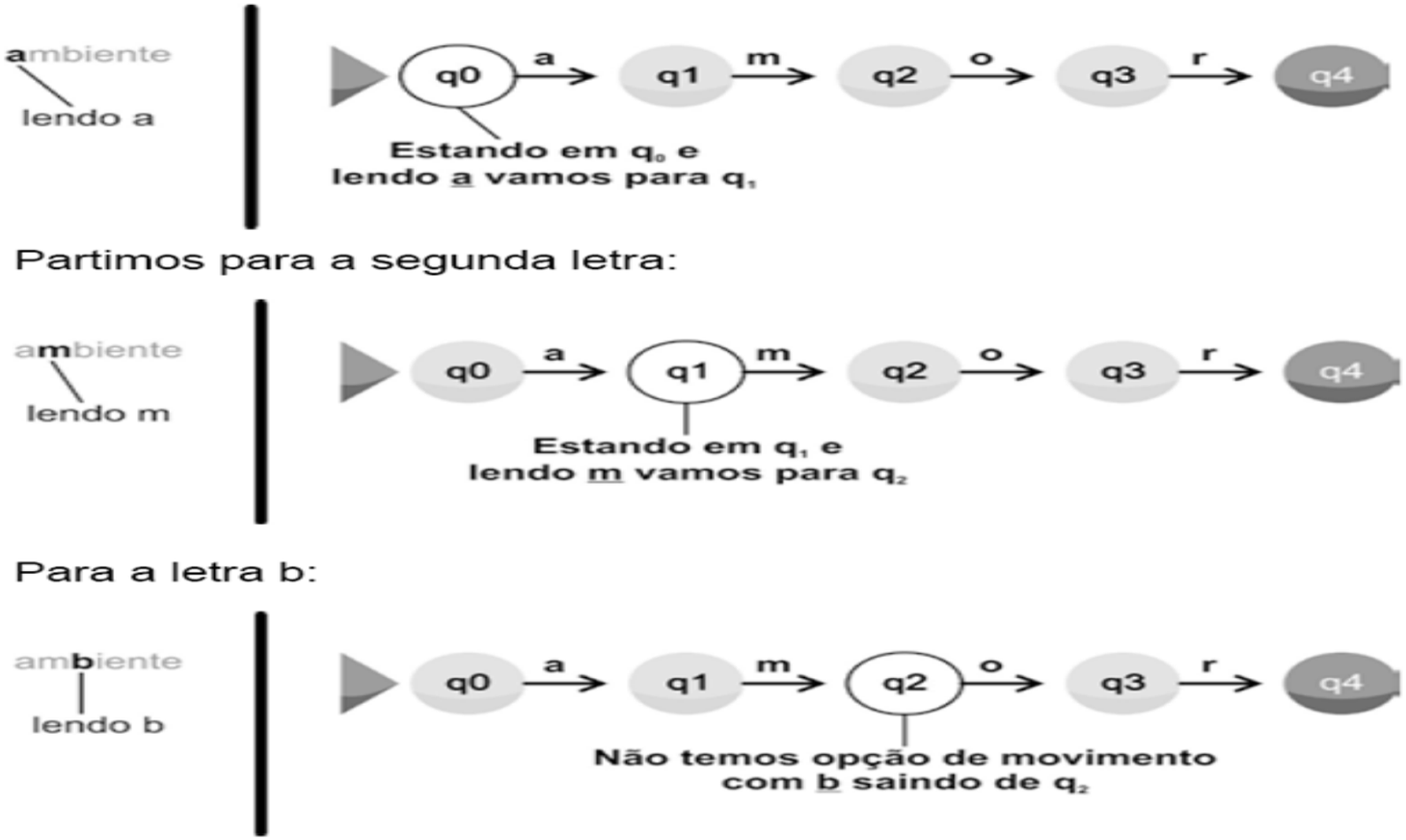
- Essas posições correspondem aos prefixos da palavra, variando desde o string vazio (isto é, nenhuma parte da palavra foi atingida até agora) até a palavra completa.



Introdução aos autômatos finitos

- **Exemplo 3**

- Tentativa de leitura da palavra **ambiente**

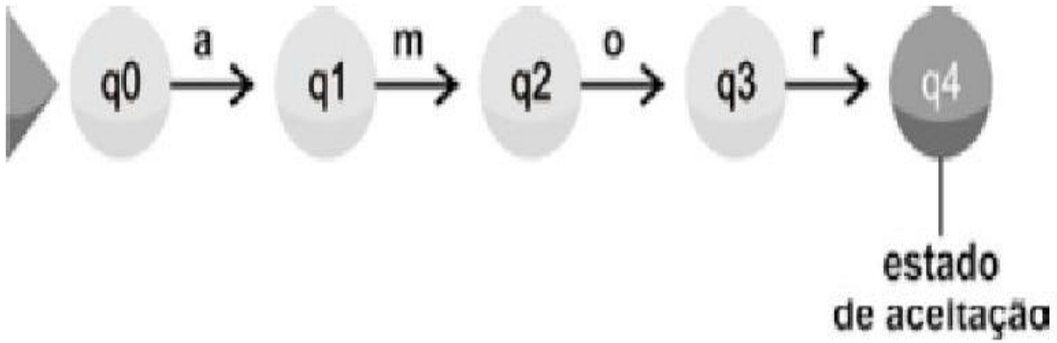


Introdução aos autômatos finitos

- **Exemplo 4**

- Palavra AMOR

- Qualquer outra palavra deve ser descartada



Representações estruturais

- Há dois formalismos importantes que não são semelhantes a autômatos, mas que desempenham um importante papel no estudo de autômatos e suas aplicações.

1. As gramáticas são modelos úteis quando se projeta software que processa dados com uma estrutura recursiva.

    - O exemplo mais conhecido é um analisador sintático (parser), o componente de um compilador que lida com as características recursivamente aninhadas de uma linguagem programação, como as expressões aritméticas, condicionais e assim por diante.

Representações estruturais

- Há dois formalismos importantes que não são semelhantes a autômatos, mas que desempenham um importante papel no estudo de autômatos e suas aplicações.

2. As Expressões regulares são padrões que correspondem a conjuntos de strings e são usadas para especificar de maneira concisa as linguagens regulares, que são o tipo de linguagem reconhecida por autômatos finitos. São uma ferramenta poderosa para descrever e identificar padrões de texto, e são amplamente utilizadas em busca e substituição de texto em editores de texto, validação de entrada de dados e análise léxica em compiladores.

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 ∑ para um alfabeto. Os alfabetos comuns incluem:

1. ∑ = {0, 1}, o alfabeto binário.

2. ∑ = {a, b, ..., z}, o conjunto de todas as letras minúsculas.

3. 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 string 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 ∑k = {f}, independente de qual alfabeto seja ∑ .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 ∑ e ∑1
- O primeiro ∑ é um alfabeto, seus elementos 0 e 1 são símbolos.
- O outro ∑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 ∑ é denotado convencionalmente por ∑* .
- Por exemplo:
- {0,1}* = {ε, 0, 1, 00, 01, 10, 11, 000, ...}.
- Em outros termos:

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

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: $xy = a_1a_2 \ldots a_i\, b_1b_2 \ldots b_j$.

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, ...}

Formadores de conjuntos como um modo de definir linguagens

- É comum descrever uma linguagem usando um “formador de conjuntos”:

$$\{ w \mid \text{algo sobre } w \}$$

- Essa expressão é lida como “o conjunto de palavras w tais que (seja o que for dito sobre w à direita da barra vertical)”

Formadores de conjuntos como um modo de definir linguagens

- Exemplos:

1. {w | w consiste em um número igual de 0’s e 1’s}. {01,0011,000111}
2. {w | w consiste em um número inteiro binário primo}. {10,11,101,111,1011}
3. {w | w é um programa em C sintaticamente correto}.

Formadores de conjuntos como um modo de definir linguagens

- Também é comum substituir w por alguma expressão com parâmetros e descrever os strings na linguagem declarando condições sobre os parâmetros.

- Exemplos:

- $\{0^n1^n \mid n>=1\}$ {01,0011,000111}

Nesta aula vimos

- Breve abordagem sobre a história e desenvolvimento da teoria de autômatos, começando com as máquinas de Turing de Alan Turing nos anos 1930, seguido pelo trabalho de Chomsky em gramáticas formais nos anos 1950, e as contribuições de S. Cook em 1969 sobre problemas computacionais intratáveis ou NP-difíceis.

Nesta aula vimos

- O material destacou a importância do estudo de autômatos para compreender a base teórica da computação, a classificação de problemas, o projeto de linguagens de programação, o desenvolvimento de compiladores, a eficiência computacional, o entendimento de linguagens formais, suas aplicações práticas, e como base para estudos avançados em teoria da computação.

Nesta aula vimos

- Foram discutidos conceitos chave como alfabetos, strings, linguagens, e a estrutura e função dos autômatos finito.

- O material também abordou representações estruturais importantes como gramáticas e expressões regulares, e forneceu uma visão sobre como os autômatos finitos são usados para analisar e projetar sistemas com um número finito de estados.

Pesquisa

- Pesquise e escreva ao menos uma página sobre a Lei de Moore.

  - Descrever o que é a Lei de Moore.
  - Discutir o contexto histórico e a declaração original feita por Gordon Moore em 1965.

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