



Linguagens e Autômatos Finitos Determinísticos

Autômatos, Linguagens e Computação

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizagem

- Definir formalmente o conceito de linguagem.
- Compreender a relação entre linguagem e problema computacional.
- Definir formalmente o conceito de autômatos finitos determinísticos (AFDs).
- Compreender o processo de reconhecimento de cadeias através de AFDs.

Conceitos Fundamentais

- **Alfabeto** Σ : Conjunto finito de **símbolos** (ex.: $\{0, 1\}$).
- **Cadeia** $w = w_1 w_2 \dots w_k$, $w_i \in \Sigma$: Sequência finita de símbolos.
- **Fecho de Kleene** de um alfabeto Σ , denotado Σ^* , é o conjunto de todas as cadeias sobre Σ .
 - $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$, onde $\Sigma^0 = \{\varepsilon\}$.
- **Fecho Positivo** de um alfabeto Σ , denotado Σ^+ , é o conjunto de todas as cadeias **não vazias** sobre Σ .
 - $\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i$

Linguagem

Uma **Linguagem** L sobre um alfabeto Σ é um subconjunto de Σ^*

- $L \subseteq \Sigma^*$
- Conjunto de cadeias sobre Σ .
- Exemplo: $L_1 = \{010, 11, 0, 1011\}$ sobre $\Sigma = \{0, 1\}$
- Note que **linguagens podem ser infinitas**, mas alfabetos precisam ser finitos.

Linguagem e Problema

Um **problema** pode ser visto como um conjunto de instâncias que requerem uma resposta "sim" ou "não".

- Focaremos na pergunta: **Dados** $L \subseteq \Sigma^*$ **e** $w \in \Sigma$, $w \in L?$
- Problema de decisão: Cabe **decidir** se a cadeia w **pertence** à linguagem L .
- Dessa forma, qualquer problema computacional pode ser expresso por linguagens:
 - $L_0 = \{w \in \{0, 1, \dots, 9\}^* : \text{se } w \text{ representa um número, } w \text{ é primo}\}$
 - Ou seja, linguagem dos números primos.
 - $L_1 = \{w0 \in \{0, 1\}^* : w \in \{0, 1\}^*\}$
 - Ou seja, linguagem dos números pares em representação binária.

Mais exemplos de linguagens na lousa

Sistema de Estados Finitos

- Modelo (matemático) computacional simples
- Estrutura abstrata que descreve a execução de algoritmos e dispositivos computacionais através de **estados discretos** e **transições**.
- Cada **estado** representa uma configuração completa do sistema em um instante; transições são acionadas por **símbolos de entrada** (ou eventos) que modificam a configuração.
 - **Entrada** : cadeia w
 - **Saída** : pertence ou não pertence (i.e., uma **decisão**)
- Modelam desde circuitos digitais simples até protocolos de comunicação, linguagens formais e processos de controle industrial.

Autômato Finito Determinístico (AFD)

Máquina automática: Processador de cadeias

Finito: Memória limitada

Reconhecedor de linguagens

Aceita ou **Rejeita** uma cadeia

Detecção de padrões

Processa um símbolo por vez

Exemplo 1: Diagrama de Estados

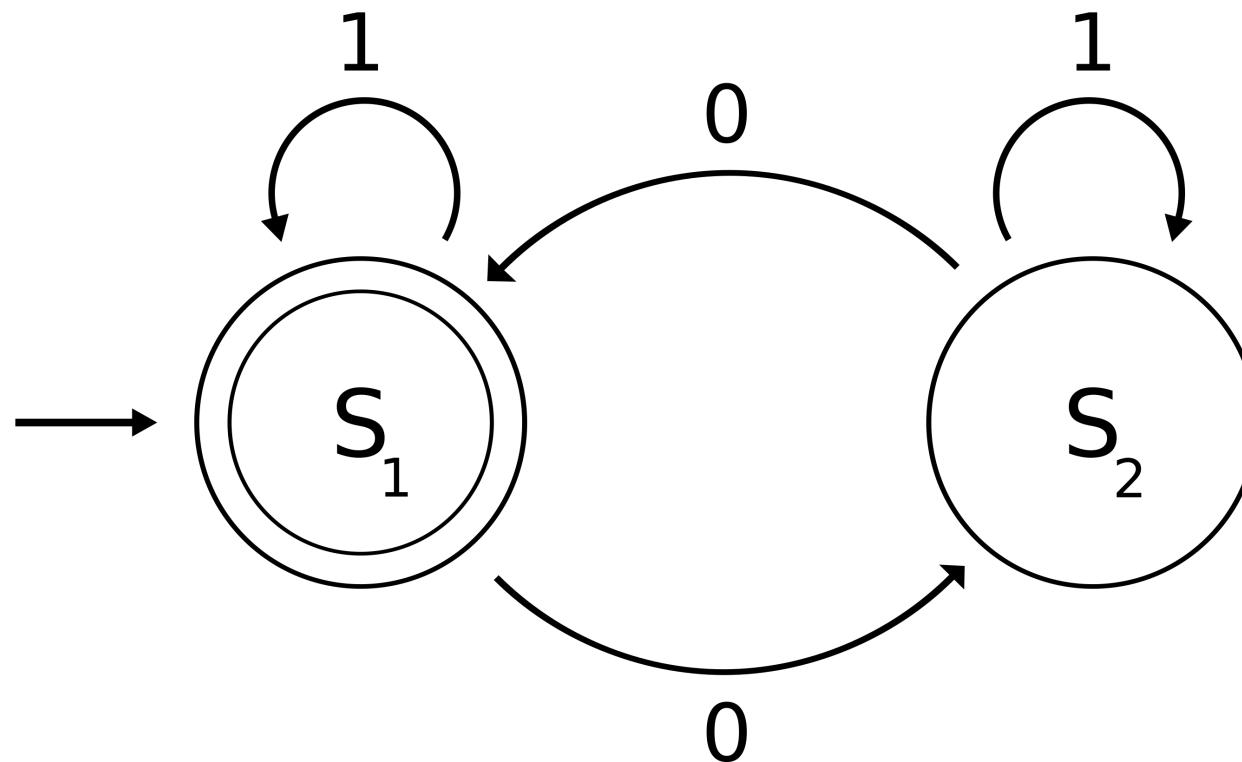


Diagrama de estados para um AFD simples. Fonte: [Wikipedia](#).

Prática: Processar as cadeias $w_1 = 0011$ e $w_2 = 1101$

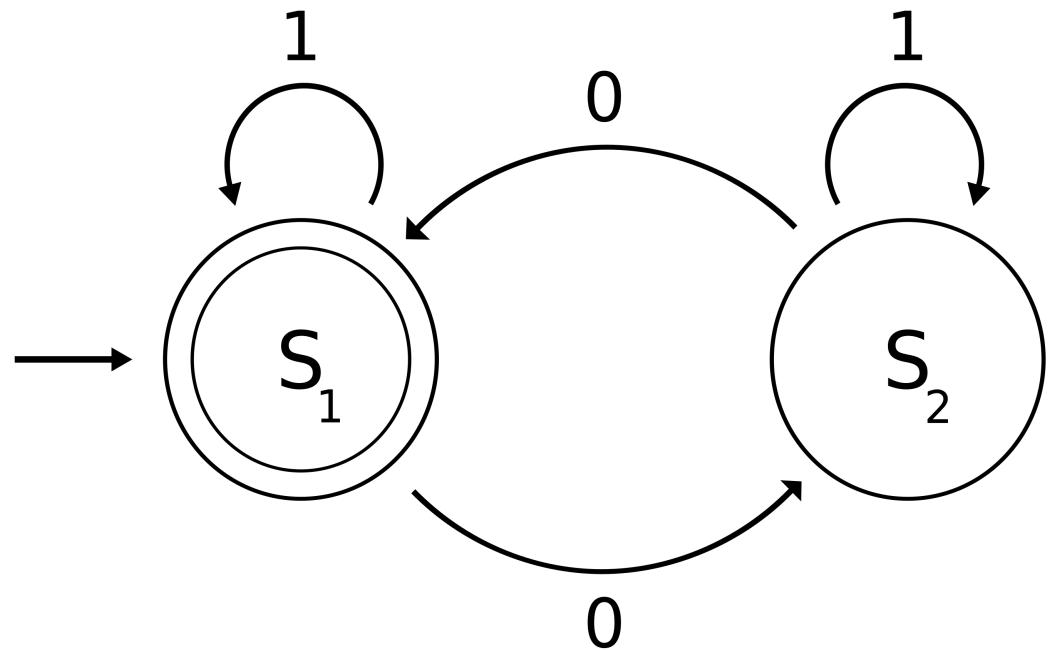
Autômato Finito Determinístico (AFD)

Definição formal: Um AFD é uma 5-upla $A = (Q, \Sigma, \delta, q_0, F)$, onde:

- Q é um conjunto finito de estados.
- Σ é um conjunto finito de símbolos (alfabeto).
- $\delta : Q \times \Sigma \rightarrow Q$ é uma função de transição determinística entre estados. Ou seja, $\delta(p, a) = q$.
- $q_0 \in Q$ é o estado inicial.
- $F \subseteq Q$ é o conjunto finito de estados finais (de aceite).
- **Determinismo:** Para cada símbolo do alfabeto existe **apenas um** estado possível para o qual o autômato pode transitar a partir do (único) estado atual.

Exemplo 1: Formalização

- $M_1 = (Q, \Sigma, \delta, q_0, F)$, onde:
- **Estados**: $Q = \{s_1, s_2\}$.
- **Alfabeto**: $\Sigma = \{0, 1\}$.
- **Função δ** :
 - $\delta(s_1, 0) = s_2, \delta(s_2, 0) = s_1$
 - $\delta(s_1, 1) = s_1, \delta(s_2, 1) = s_2$
- **Estado inicial**: $q_0 = s_1$.
- **Estado final**: $F = \{s_1\}$.
- **Pergunta 1**: Qual a propriedade das cadeias aceitas por M_1 ?
- **Pergunta 2**: Qual a forma tabular de δ ?
- **Pergunta 3**: Qual aplicação prática de M_1 ?



Exemplo de AFD simples. Fonte: [Wikipedia](#).

Reconhecimento de linguagem

Definição (Linguagem de uma Máquina): Seja X o conjunto de todas as cadeias aceitas por M , dizemos que X é a **linguagem aceita** por M , ou seja, $L(M) = X$.

- Um autômato reconhece **apenas uma** linguagem. Ou seja, a linguagem X é única.
 - Pode ser a linguagem vazia \emptyset .
- Pode reconhecer infinitas cadeias.

Observação importante sobre AFDs

Condições de parada: após processar o último símbolo de uma cadeia

$w = w_1 w_2 \dots w_n$,

- **Aceita** a cadeia ✓: se o AFD assume estado de aceitação;
- **Rejeita** a cadeia ✗: se o AFD não assume estado de aceitação **ou** δ não está definida para alguma transição $\delta(q_i, w_j) = \perp$.
- Sempre para. **Sem loop infinito.**

Exemplo 2

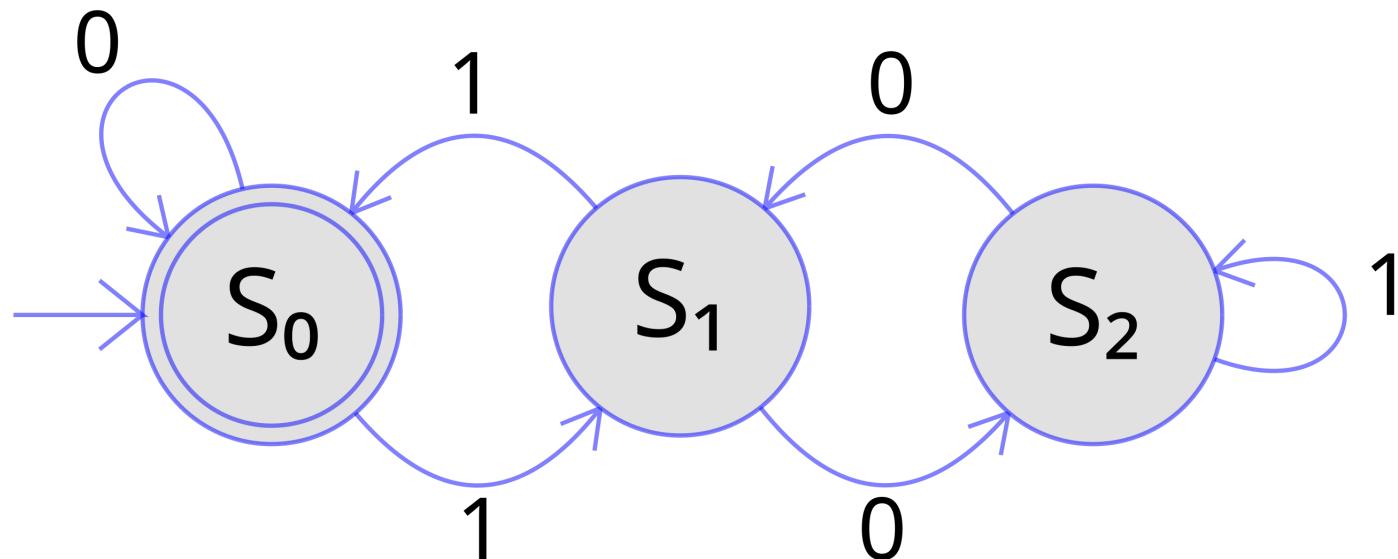
$M_2 = (Q, \Sigma, \delta, q_0, F)$, onde:

- **Estados:** $Q = \{q_0, q_1\}$, q_0 inicial.
- **Alfabeto:** $\Sigma = \{0, 1\}$.
- **Função δ :**
 - $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$
 - $\delta(q_1, 0) = q_0$, $\delta(q_1, 1) = q_1$
- **Estado final:** $F = \{q_1\}$.
- M_2 reconhece (uma linguagem de) cadeias com número ímpar de zeros.

Representação gráfica do AFD M_2 **na lousa**

Exercício Prático

- Dado o AFD M_3 abaixo que aceita apenas números binários múltiplos de 3:
 - Descreva formalmente $M_3 = (Q, \Sigma, \delta, q_0, F)$.
 - Especifique em formato tabular a função de transição δ .



Fonte: [Wikipedia](#).

Função de Transição Estendida

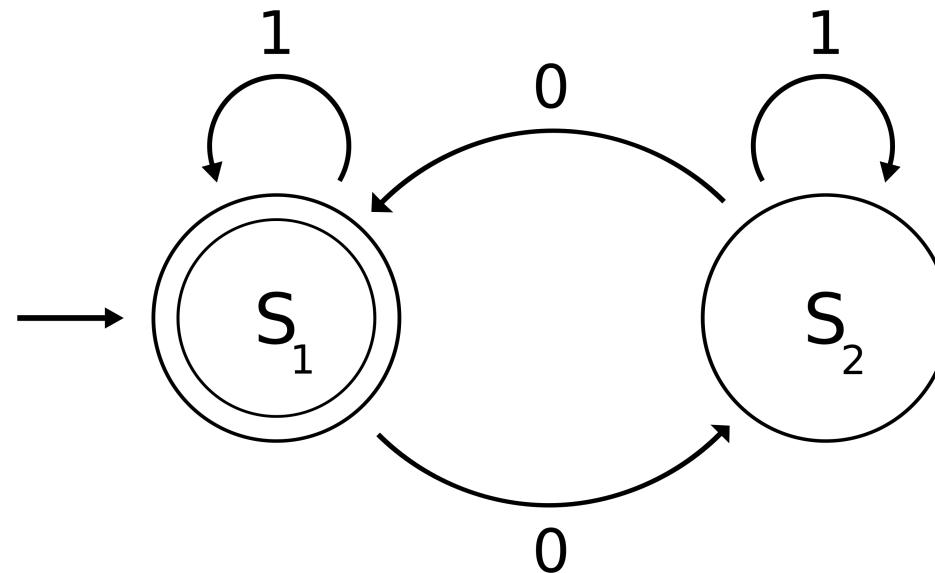
Definição formal: Dado um AFD $M = (Q, \Sigma, \delta, q_0, F)$, a função de transição estendida $\delta^* : Q \times \Sigma^* \rightarrow Q$, definida como

$$\delta^*(q, w) = \begin{cases} q & \text{se } w = \varepsilon \quad (\text{palavra vazia}) \\ \delta(\delta^*(q, u), a) & \text{se } w = ua \text{ com } u \in \Sigma^*, a \in \Sigma \end{cases}$$

para, $q \in Q$ e $w \in \Sigma^*$

- Definição recursiva.
- $\delta^*(q, u)$ é o estado de M **após processar toda a cadeia** w a partir do estado q .
- Permite descrever o estado final após leitura de uma cadeia completa sem repetir explicitamente cada transição.
- **Aceite:** Um AFD M aceita $w \in \Sigma^*$ se $\delta^*(q, w) \in F$, caso contrário, M rejeita w .

Função de Transição Estendida para M_1



Exemplo de AFD simples. Fonte: [Wikipedia](#).

- $\delta^*(s_1, 1001) = s_1$
- $\delta^*(s_1, 000) = s_2$
- $\delta^*(s_1, 100101) = ?$
- $\delta^*(s_1, 01110) = ?$

Linguagem de um AFD

A linguagem de um AFD $A = (Q, \Sigma, \delta, q_0, F)$, denotada por $L(A)$, é definida como

$$L(A) = \{w | \delta^*(q_0, w) \in F\}$$

Se L é a linguagem $L(A)$ para algum AFD A , então dizemos que
 L é uma **linguagem regular**.

Conceitos Chave

Termo	Definição	Observação
Estado	Ponto de posição da máquina.	Finito, discretos.
Transição	Movimento entre estados.	Determinística vs não-determinística.
Estado Inicial	Onde a computação começa.	Único.
Estado Final (Aceitante)	Reconhecimento bem-sucedido.	Pode haver zero ou mais.
Palavra/Entrada	Sequência de símbolos em Σ .	A máquina aceita se terminar em F .

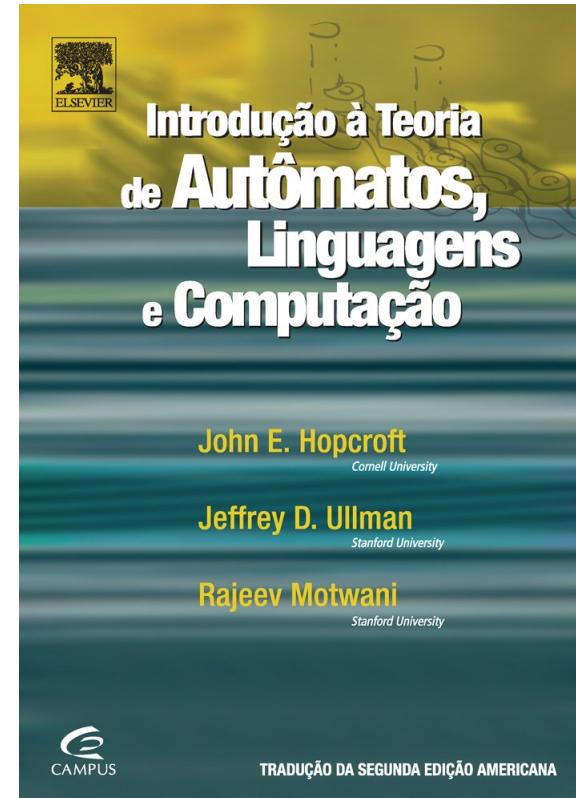
Exercícios Práticos

Construa AFDs que reconheça as linguagens:

1. **Paridade de 1:** $L_1 = \{w \in \{0, 1\}^* \mid w \text{ contém número par de } 1\}$
2. **Padrão final fixo:** $L_2 = \{w \in \{a, b, c\}^* \mid w \text{ termina em } bc\}.$
3. **Reconhecimento de subcadeia:** $L_3 = \{w \in \{0, 1\}^* \mid w \text{ possui subcadeia } 101\}.$

Resumo e Próximos Passos

- **Definição Formal:** $M = (Q, \Sigma, \delta, q_0, F)$
 - Começa em q_0 .
 - Para cada símbolo a_i da palavra, segue $\delta(q_{i-1}, a_i)$.
 - Palavra aceita se o último estado é um estado final (pertence a F).
- **Determinismo:** para qualquer $q \in Q$, $a \in \Sigma$, existe exatamente um destino.
- **Decidibilidade:** aceitação pode ser testada em tempo linear na extensão da palavra.
- **Aplicações Práticas:** Análise lexical em compiladores (identificação de tokens). Reconhecimento de padrões em sistemas embarcados e protocolos de comunicação.



Atividade recomendada: Leitura do capítulo 1 e das seções 2.1 e 2.2 do capítulo 2.

Perguntas e Discussão

1. Quais técnicas podem ser usadas para identificar e corrigir erros em um AFD construído manualmente?
2. Como o tamanho do conjunto de estados afeta a eficiência da execução? Em que situações um AFD com mais estados pode ser desejável apesar de maior custo de memória?
3. Quais são as limitações intrínsecas dos AFDs na modelagem de sistemas complexos? Cite exemplos onde o modelo determinístico não consegue capturar a dinâmica do problema.
4. Como podemos otimizar um AFD já construído sem alterar sua linguagem reconhecida?
5. Qual é o impacto da cardinalidade do alfabeto na complexidade de transição de um AFD? Exemplo: Alfabeto binário versus alfabeto de caracteres ASCII completo.