Introdução às Linguagens e aos Autômatos de Estados Finitos

Autômatos, Linguagens e Computação

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizagem

- Definir formalmente linguagem.
- Compreender a relação entre linguagem e problema computacional.
- Definir formalmente o conceito de autômatos de estados finitos.

Conceitos Fundamentais

- Alfabeto Σ: Conjunto finito de **símbolos** (ex.: $\{0, 1\}$).
- Cadeia $w = w_1 w_2 \ldots w_k,\ w_i \in \Sigma$: Sequência finita de símbolos.
- Fecho de Kleene de um alfabeto Σ, denotado Σ^*, é o conjunto de tadas 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:
  - $L_0 = \{w \in \{0, 1, \ldots, 9\}^* : \text{se } w \text{ representa um número, } w \text{ é primo}\}$
    - 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

Breve Histórico

Ano	Contribuição	Autor(es)
1929	Formalização de máquinas abstratas	A. M. Turing (máquina de Turing)
1936	Autômatos finitos simples	S. C. Kleene (expressões regulares)
1945	Autômato determinístico e não-determinístico	M. Rabin & S. Scott
1960s	Linguagens formais na computação	Aho, Ullman, Hopcroft

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.

Autômatos Finitos Determinísticos (AFD)

Processadores de cadeias

Reconhecedor de linguagens

Aceita ou **Rejeita** uma cadeia

Detecção de padrões

Processa um símbolo por vez

Exemplo 1: Diagrama de Estados

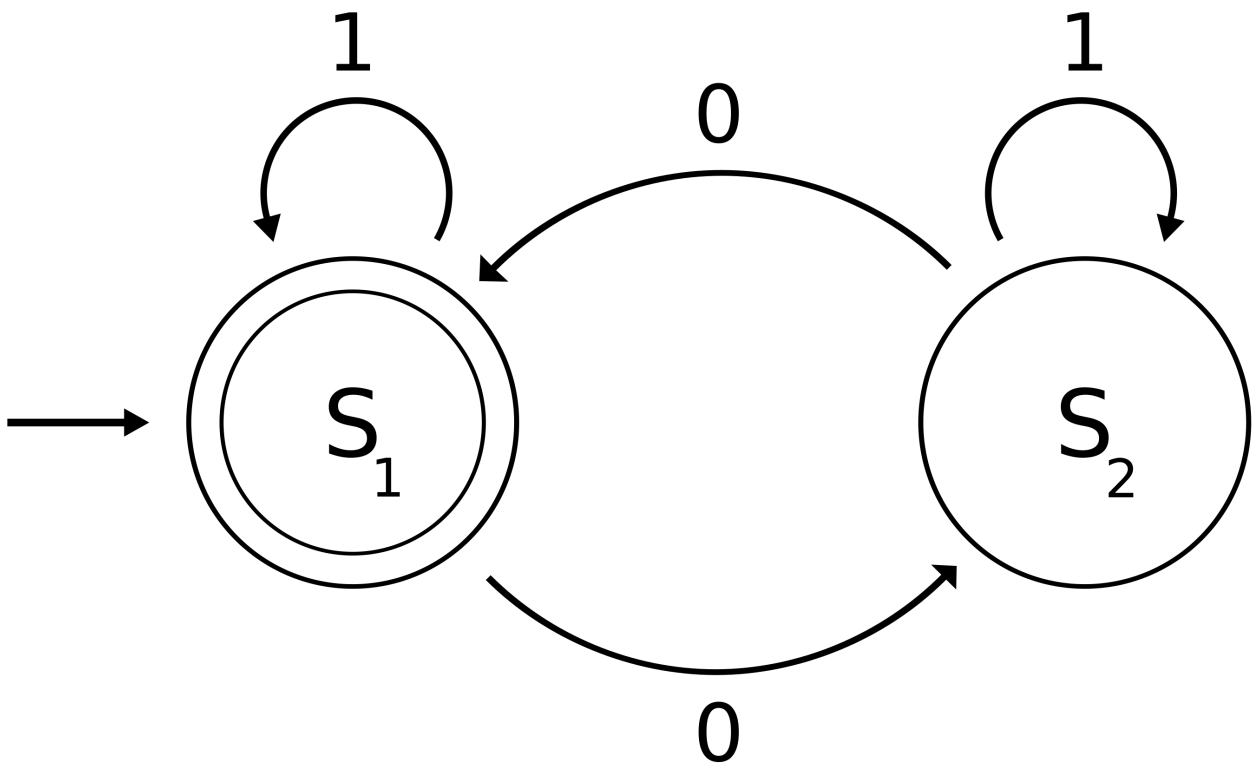


Diagrama de estados para um AFD simples. Fonte: Wikipedia.

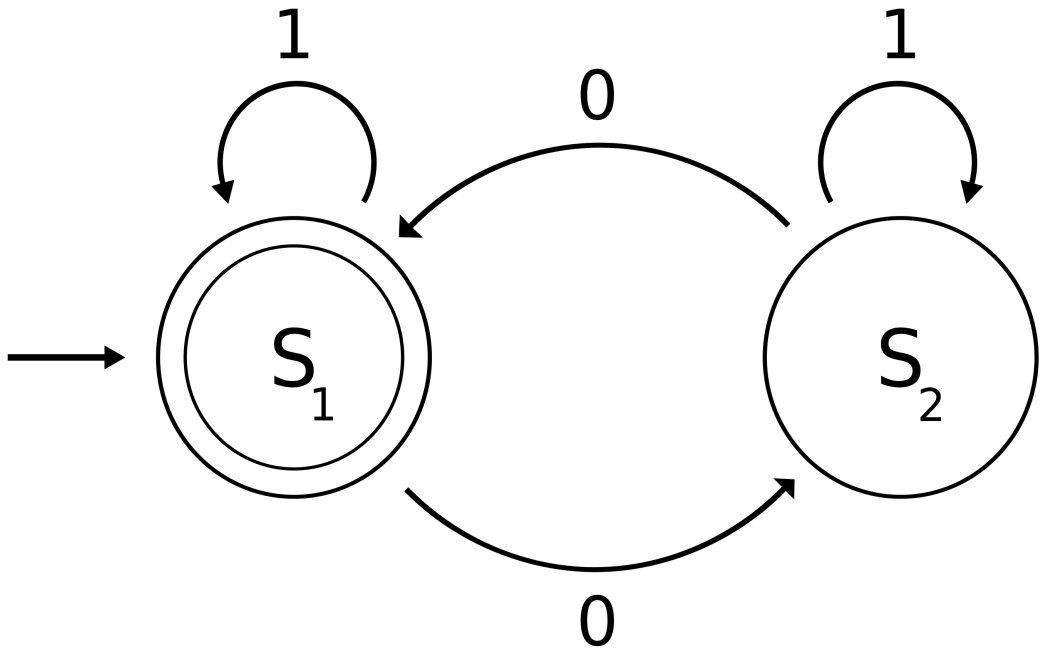
Autômato Finito Determinístico (AFD)

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).
- $\delta : Q \times \Sigma \to Q$ é uma função de transição determinística entre estados.
- $q_0 \in Q$ 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: $Q = \{S_1, S_2\}$, S_1 inicial.
- 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 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.

Exemplo 2

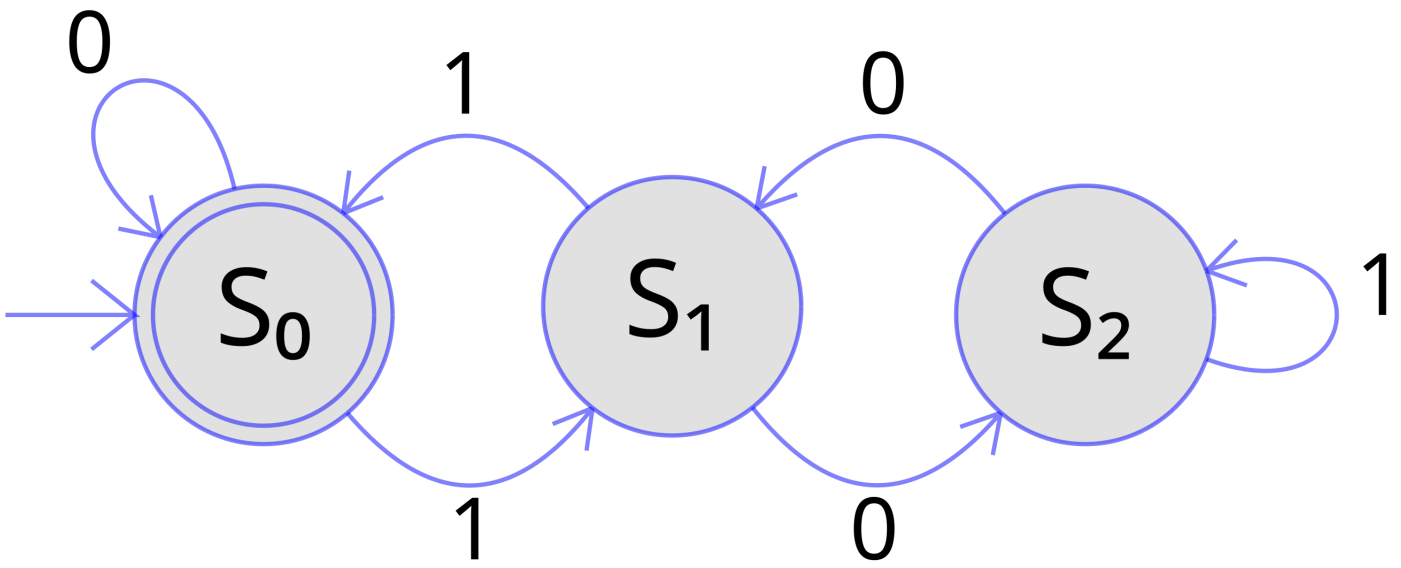
$M_2 = (Q, \Sigma, \delta, q_0, F)$, onde:

- 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_1 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.

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: $L_1 = \{w \in \{0,1\}^* \mid w \text{ conté m nú mero par de } 1\}$
2. Padrão final fixo: $L_2 = \{w \in \{a,b,c\}^* \mid w \text{ termina em } \textit{//bc//}\}$.
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.