Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologias
Engenharia da Computação

Thales L. A. Valente

Disciplina: Linguagens Formais e Autômatos
Código: EECP0020

15 de abril de 2024

Conteúdo programático

- Elementos de matemática discreta
- Conceitos básicos de linguagens
- Linguagens regulares e autômatos finitos
- Linguagens livres de contexto e autômatos de pilha
- Linguagens sensíveis ao contexto e Máquinas de Turing com fita limitada
- Linguagens recursivas e Máquinas de Turing com finta infinita
- Linguagens recursivamente enumeráveis

Sumário

Hierarquia de Chomsky

- De acordo com a complexidade relativa das linguagens, Chomsky definiu uma classificação que permite antecipar as propriedades fundamentais das linguagens e vislumbrar os modelos de implementação mais adequados. A **Hierarquia de Chomsky** é ilustrada abaixo:

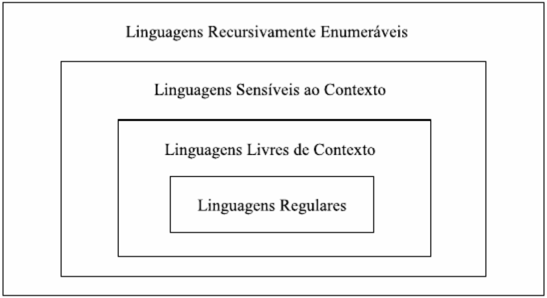


Figura: Hierarquia de Chomsky

Linguagens regulares ou do tipo 3

- O estudo de linguagens regulares pode ser abordado a partir de três formalismos básicos:
  - **Operacional ou reconhecedor**: representado por autômatos finitos.
  - **Axiomático ou gerador**: representado por gramáticas regulares.
  - **Denotacional**: representado por expressões regulares.

Definição

- Os **sistemas de estados finitos** são máquinas abstratas que capturam as partes essenciais de algumas máquinas concretas.
- Como exemplos de máquinas que podem ser modeladas matematicamente com essa abordagem citam-se máquinas de vender refrigerante, relógios digitais, elevadores, analisadores léxicos e computadores.
- Esses sistemas apresentam um número finito de estados possíveis, assim como entradas e saídas discretas.
- Uma característica fundamental de uma máquina de estados finitos é que sua memória é limitada e exclusivamente organizada em torno do conceito de estado.

Motivação: um quebra-cabeça

- Um homem, um leão, um coelho e um repolho devem atravessar um rio usando uma canoa, com a restrição de que o homem deve transportar no máximo um dos três de cada vez de uma margem à outra. Além disso, o leão não pode ficar na mesma margem que o coelho sem a presença do homem, e o coelho não pode ficar com o repolho sem a presença do homem. Pergunta-se: é possível fazer a travessia? Em caso afirmativo, forneça uma sequência de movimentações que a propicie.

Motivação: um quebra-cabeça

- Para a resolução do problema, devem-se abstrair detalhes e focar no essencial:
    1. Em um dado instante, a margem do rio onde estão o homem, o leão, o coelho e o repolho.
    2. A sequência de movimentações entre as margens que propiciou a situação indicada.

Motivação: um quebra-cabeça

- Para a resolução do problema, deve-se relevar detalhes e focar no essencial:
  1. Podem-se usar os símbolos **h**omem, **l**eão, **c**oelho e **r**epolho para explicitar em que margem estão os elementos. Por exemplo, $\{\boldsymbol{h}, \boldsymbol{c}, \boldsymbol{r}\}/\{\boldsymbol{l}\}$ indica que o homem, o coelho e o repolho estão na margem inicial e o leão na margem final. A uma determinado configuração do ambiente dá-se o nome de **estado**.
  2. A sequência de movimentações pode ser representada por uma palavra $w = a_1 a_2 \ldots a_n$, onde cada a_i pode ser **s**ozinho, **l**eão, **c**oelho ou **r**epolho, indicando quem atravessa o rio com o homem. Por exemplo, a palavra **csr** indica que o homem levou o coelho da margem inicial para a final, em seguida ele voltou sozinho para a margem inicial e retornou à margem final com o repolho. Cada símbolo da palavra especifica uma operação que propicia uma **transição** de um estado a outro.

Motivação: um quebra-cabeça

- O problema pode ser simplificado para "encontrar uma palavra que represente uma sequência de transições que leve do estado inicial $\{\boldsymbol{h}, \boldsymbol{l}, \boldsymbol{c}, \boldsymbol{r}\}/\varnothing$ ao estado final $\varnothing/\{\boldsymbol{h}, \boldsymbol{l}, \boldsymbol{c}, \boldsymbol{r}\}$". A figura abaixo ilustra o diagrama de estados para o problema.

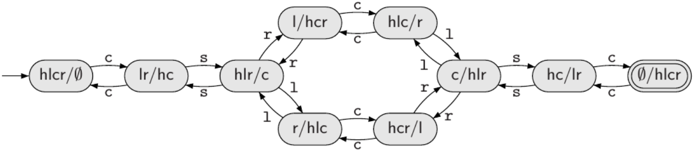


Figura: Diagrama de estados para o problema do quebra-cabeças.

- Elipses: representam os estados.
- Setas: representam as transições possíveis.
- Estado inicial: recebe uma seta isolada.
- Estado final: é uma elipse dupla.

Motivação: um quebra-cabeça

- Dada uma palavra $w \in \{s, l, c, r\}^*$, ela representa uma solução do problema se o seu caminho correspondente parte do estado inicial e alcança o estado final. Neste caso, diz-se que a palavra é reconhecida ou aceita. Caso contrário, ela é não-reconhecida ou rejeitada.
  - Para o problema do quebra-cabeças, que palavras são aceitas?

Motivação: um problema matemático

- Seja o problema de projetar uma máquina que, dada uma sequência de 0s e 1s, determine se o número representado por ela na base 2 é divisível por 6. O que se deseja é um projeto independente de implementação, ou seja, que capture apenas a essência de tal máquina, não importando se ela será mecânica, eletrônica, um programa ou o que quer que seja.

Motivação: um problema matemático

- Para a resolução do problema, devem-se observar os seguintes fatos:
  - Para um número x ser divisível por 6, tem-se que $\boldsymbol{x} \textbf{ mod } \mathbf{6} = \mathbf{0}$. Assim, dentre as 6 possibilidades de resto, apenas uma resulta em um número divisível por 6.
  - Seja n o número representado pela palavra w. Então $w0$ é $2n$ e $w1$ é $2n + 1$.

Motivação: um problema matemático

- Com base nos fatos anteriores, percebe-se que há 6 alternativas de resto possíveis. Assim, uma máquina com 6 estados poderia ser construída para solucionar o problema: se a representação binária do número alcançar o estado que denota o resto 0, tal número é divisível por 6; caso contrário, ele não o é. Tal máquina pode ser representada como segue:

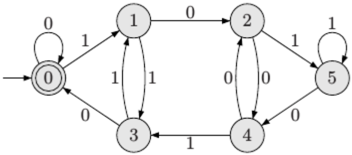


Figura: Diagrama de estados para o problema da divisibilidade por 6.

Definição

- Um **autômato finito determinístico (AFD)**, ou simplesmente **autômato finito**, pode ser visto como um sistema de estados finitos composto de, basicamente, três partes:
  1. **Fita**: componente que contém a informação a ser processada. É dividido em células, onde cada uma armazena um símbolo de um alfabeto de entrada.
  2. **Unidade de controle**: componente que contém um número finito e pré-definido de estados. Possui uma cabeça exclusivamente de leitura que lê uma célula da fita por vez e movimenta-se para a direita.
  3. **Função programa ou função de transição**: função parcial que dependendo do estado corrente e do símboblo lido, determina o novo estado.

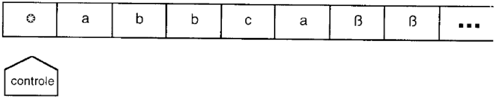


Figura: Autômato finito determinístico

Definição

- Formalmente, um **autômato finito determinístico M** corresponde a uma 5-tupla:

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

onde:

  - Σ: alfabeto de símbolos de entrada.
  - Q: conjunto finito de estados possíveis do autômato.
  - δ: função programa da forma $\delta : Q \times \Sigma \to Q$.
  - q_0: estado inicial, tal que $q_0 \in Q$.
  - F: conjunto de estados finais, tal que $F \subseteq Q$.

Definição

- A função programa pode ser representada por um grafo finito direcionado, como ilustrado abaixo:

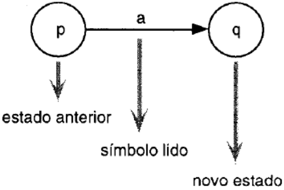


Figura: Função programa como um grafo

- Nesse grafo, os estados inicial e final são representados como segue:

Figura: Estados inicial e final do AFD

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

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

.

- Um autômato finito que reconhece a linguagem é definido por:

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

onde δ_1 é definida pela tabela abaixo:

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

Figura: Definição de δ_1

Exemplos

- O AFD M_1 pode ser representado também pelo grafo da figura abaixo:

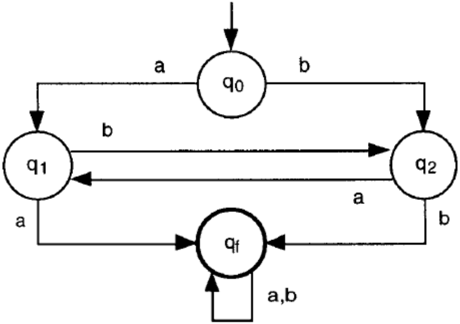


Figura: Grafo do AFD M_1

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_2 = \{w|w \text{ possui número par de a e um número par de b}\}$$

.

- Um autômato finito que reconhece a linguagem é definido por:

$$M_2 = (\{a, b\}, \{q_0, q_1, q_2, q_3\}, \delta_2, q_0, \{q_0\})$$

com o grafo correspondente abaixo:

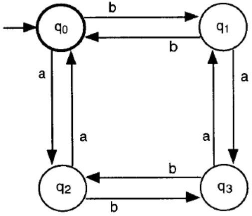


Figura: Grafo do AFD M_2

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_3 = \{\}$$

.

- Um autômato finito que reconhece a linguagem é definido por:

$$M_3 = (\{a, b\}, \{q_0\}, \delta_3, q_0, \{\})$$

com o grafo correspondente abaixo:

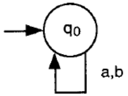


Figura: Grafo do AFD M_3

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_4 = \Sigma^*$$

.

- Um autômato finito que reconhece a linguagem é definido por:

$$M_4 = (\{a, b\}, \{q_0\}, \delta_4, q_0, \{q_0\})$$

com o grafo correspondente abaixo:

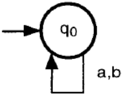


Figura: Grafo do AFD M_4

Exemplos

- Perceba que, um AFD sempre pára seu processamento, uma vez que a palavra de entrada é finita e um novo símbolo de entrada é lido a cada aplicação da função programa.
- A parada de um processamento pode ser de duas formas: aceitando ou rejeitando uma entrada w. As condições de parada são as seguintes:
  - Após processar o último símbolo da fita, o AFD assume um estado final. Neste caso, o AFD **aceita** a palavra w.
  - Após processar o último símbolo da fita, o AFD assume um estado não-final. Neste caso, o AFD **rejeita** a palavra w.
  - A função programa é indefinida para o argumento (estado corrente e símbolo lido). Neste caso, o AFD pára e **rejeita** a palavra w.

Função programa estendida

- Seja $M = (\Sigma, Q, \delta, q_0, F)$ um autômato finito determinístico. A função programa estendida denotada por

$$\underline{\delta} : Q \times \Sigma^* \to Q$$

é a função programa $\delta : Q \times \Sigma \to Q$ estendida para palavras. Ela é indutivamente definida como segue:
$\underline{\delta}(q, \epsilon) = q$
$\underline{\delta}(q, aw) = \underline{\delta}(\delta(q, a), w)$

Função programa estendida

- Como exemplo, considere o AFD M_1 definido anteriormente. Então, a função programa estendida $\underline{\delta}$[1] aplicada à palavra *abaa* a apartir do estado inicial q_0 é como segue:

$\underline{\delta}(q_0, abaa) =$	função estendida sobre abaa
$\underline{\delta}(\delta(q_0, a), baa) =$	processa $\underline{a}$baa
$\underline{\delta}(q_1, baa) =$	função estendida sobre baa
$\underline{\delta}(\delta(q_1, b), aa) =$	processa $\underline{b}$aa
$\underline{\delta}(q_2, aa) =$	função estendida sobre aa
$\underline{\delta}(\delta(q_2, a), a) =$	processa $\underline{a}$a
$\underline{\delta}(q_1, a) =$	função estendida sobre a
$\underline{\delta}(\delta(q_1, a), \varepsilon) =$	processa aba$\underline{a}$
$\underline{\delta}(q_f, \varepsilon) = q_f$	função estendida sobre ε: fim da indução

Figura: Definição de δ_1

[1] Por simplicidade, vai-se denotar no futuro tanto a função programa δ quanto a função programa estendida $\underline{\delta}$ por δ.

Função programa estendida

- A linguagem aceita por por um AFD $M = (\Sigma, Q, \delta, q_0, F)$, denotada por $ACEITA(M)$ ou $L(M)$, é o conjunto de todas as palavras pertencentes a Σ^* aceitas por M, ou seja,

$$ACEITA(M) = \{w | \delta(q_0, w) \in F\}$$

- Analogamente, $REJEITA(M)$ é o conjunto de todas as palavras que pertencem a Σ^* rejeitadas por M.
- Neste contexto, as seguinte afirmações são verdadeiras:
  - $ACEITA(M) \cap REJEITA(M) = \varnothing$
  - $ACEITA(M) \cup REJEITA(M) = \Sigma^*$
  - O complemento de $ACEITA(M)$ é $REJEITA(M)$.
  - O complemento de $REJEITA(M)$ é $ACEITA(M)$.

Autômatos finitos equivalentes

- Dois AFDs M_1 e M_2 são ditos **autômatos finitos equivalentes** se, e somente se
$$ACEITA(M_1) = ACEITA(M_2)$$
- Por exemplo, os dois AFD representados abaixo por seus grafos são equivalentes:

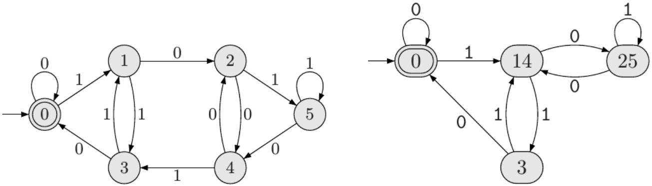


Figura: AFDs para reconhecer divisibilidade por 6

Definição

- Uma linguagem aceita por um AFD é chamada de regular ou do tipo 3.

Exemplos

- As seguintes linguagens são regulares. Construa ADFs que as reconheça.
  - $L = \{w \in \{0,1\}^* | w \text{ tem tamanho par}\}$.
  - $L = \{a^n b | n \geq 0\}$.
  - $L = \{w \in \{a,b\}^* | w \text{ inicia com prefixo ab}\}$.
  - $L = \{w \in \{0,1\}^* | w \text{ não contem a subpalavra } 001\}$.

Definição

- **Autômatos finitos não-determinísticos (AFN)** são uma generalização dos autômatos finitos determinísticos. Enquanto nos AFDs, para cada par (símbolo, estado) há transição para apenas um estado, nos AFN é possível haver transição para dois ou mais estados. Com isso, pode haver várias computações para a mesma palavra.
- Abaixo, tem-se um exemplo de um AFN e as computações possíveis para a palavra 1010:

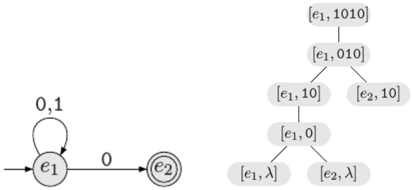


Figura: AFN e as computações possíveis para a palavra 1010

- Que linguagem o AFN acima reconhece?

Definição

- Como pode haver várias computações para uma mesma palavra, os AFNs adotam o seguinte critério para o reconhecimento de uma palavra:
  "Uma palavra é reconhecida se, e somente se, existe uma computação que a consome e que termina em um estado final".
- Note que os AFNs executam todas as computações de maneira paralela e independente.

Definição

- Formalmente, um **autômato finito não-determinístico M** corresponde a uma 5-tupla:

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

onde:

  - Σ: alfabeto de símbolos de entrada.
  - Q: conjunto finito de estados possíveis do autômato.
  - δ: função programa da forma $\delta : Q \times \Sigma \rightarrow 2^Q$.
  - q_0: estado inicial, tal que $q_0 \in Q$.
  - F: conjunto de estados finais, tal que $F \subseteq Q$.

Definição

- A função programa pode ser representada por um grafo finito direcionado, como ilustrado abaixo:

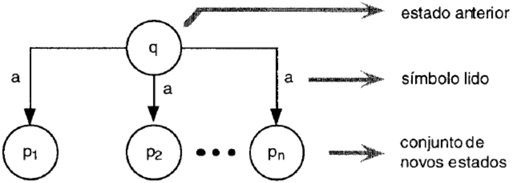


Figura: Função programa como um grafo

Função programa estendida

- Seja $M = (\Sigma, Q, \delta, q_0, F)$ um autômato finito não-determinístico. A função programa estendida denotada por

$$\underline{\delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$$

é a função programa $\delta : Q \times \Sigma \rightarrow 2^Q$ estendida para palavras. Ela é indutivamente definida como segue:
$\underline{\delta}(P, \epsilon) = P$
$\underline{\delta}(P, aw) = \underline{\delta}(\cup_{q \in P} \delta(q, a), w)$

- Assim, tem-se que, para um dado conjunto de estados $\{q_1, q_2, \ldots, q_n\}$ e para um dado símbolo a:

$$\underline{\delta}(\{q_1, q_2, \ldots, q_n\}, a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \cdots \cup \delta(q_n, a)$$

Função programa estendida

- A linguagem aceita por por um AFN $M = (\Sigma, Q, \delta, q_0, F)$, denotada por $ACEITA(M)$ ou $L(M)$, é o conjunto de todas as palavras pertencentes a Σ^* tais que existe pelo menos um caminho alternativo que aceita a palavra, ou seja,

$$ACEITA(M) = \{w| \text{ existe } q \in \delta(q_0, w) \text{ tal que } q \in F\}$$

- Analogamente, $REJEITA(M)$ é o conjunto de todas as palavras que pertencem a Σ^* rejeitadas por por todos os caminhos alternativos de M (a partir de q_0).

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_5 = \{w | w \text{ possui aa ou bb como subpalavra}\}$$

.

- Um autômato finito que reconhece a linguagem é definido por:

$$M_5 = (\{a, b\}, \{q_0, q_1, q_2, q_f\}, \delta_5, q_0, \{q_f\})$$

onde δ_5 é definida pela tabela abaixo:

δ_5	a	b
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_f\}$	-
q_2	-	$\{q_f\}$
q_f	$\{q_f\}$	$\{q_f\}$

Figura: Definição de δ_1

Exemplos

- O AFN M_5 pode ser representado também pelo grafo da figura abaixo:

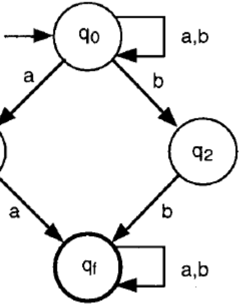


Figura: Grafo do AFN M_5

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_6 = \{w|w \text{ possui aaa como sufixo}\}$$

.

- Um autômato finito que reconhece a linguagem é definido por:

$$M_6 = (\{a, b\}, \{q_0, q_1, q_2, q_3\}, \delta_2, q_6, \{q_0\})$$

com o grafo correspondente abaixo:

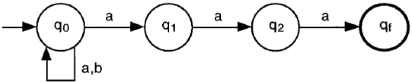


Figura: Grafo do AFN M_6

Exemplos

- Construa um AFD e um AFN que aceitem a linguagem $\{0, 1\}^*1010$.

Exemplos

- Construa um AFD e um AFN que aceite a linguagem $\{0,1\}^*1010$.

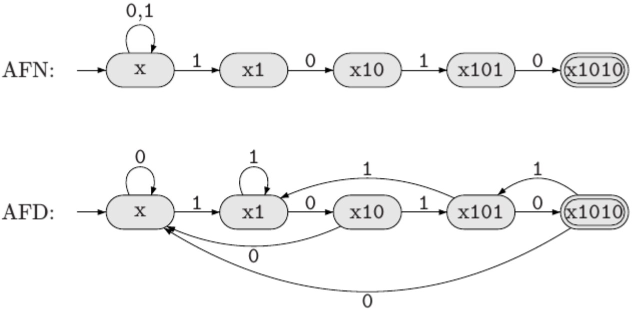


Figura: AFD e AFN que aceitam a linguagem

Exemplos

- Construa um AFD e um AFN que aceitem a linguagem abaixo:

  $\{w \in \{0,1\}^* \mid |w| \geq 3 \text{ e o } 3^{\underline{o}} \text{ símbolo da direita para a esquerda é } 1\}$

Exemplos

- Construa um AFD e um AFN que aceitem a linguagem abaixo:

$$\{w \in \{0,1\}^* || w| \geq 3 \text{ e o 3º símbolo da direita para a esquerda é } 1\}$$

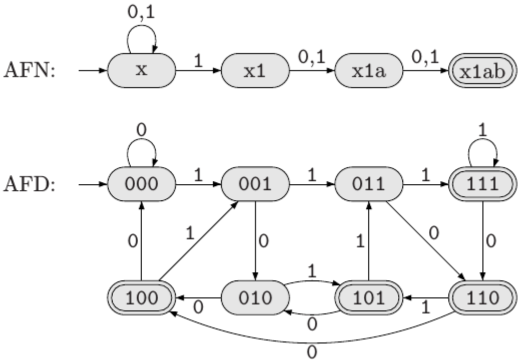


Figura: AFD e AFN que aceitam a linguagem

Equivalência entre AFNs e AFDs

- Embora a facilidade de não-determinismo forneça um aparente avanço aos Autômatos Finitos, ela não aumento seu poder computacional. De fato, para cada AFN é possível construir um AFD capaz de realizar o mesmo processamento. Assim, diz-se que **AFNs e AFDs são equivalentes**.

Equivalência entre AFNs e AFDs

- Seja $M = (\Sigma, Q, \delta, q_0, F)$ um AFN qualquer. Seja $M' = (\Sigma, Q', \delta', \langle q_0 \rangle, F')$ um AFD construído a partir de M como segue:
  - $\boldsymbol{Q'}$: conjunto de todas as combinações de estados de Q, com as diversas cardinalidades. As combinações são denotadas por $\langle q_1 q_2 \dots q_n \rangle$, onde $q_i \in Q$, para $i = 1, 2, \dots n$;
  - $\boldsymbol{\delta'}$: tal que $\delta'(\langle q_1 \dots q_2 \rangle, a) = \langle p_1 \dots p_n \rangle$ se, e somente se, $\delta(\{q_1, \dots, q_n\}, a) = \{p_1, \dots, p_m\}$;
  - $\langle \boldsymbol{q_0} \rangle$: estado inicial;
  - $\boldsymbol{F'}$: conjunto de todos os estados $\langle q_1 q_2 \dots q_n \rangle$ pertencentes a Q' tal que alguma componente q_i pertença a F.

Equivalência entre AFNs e AFDs

- Por exemplo, considerando o AFN abaixo, construa o AFD correspondente.

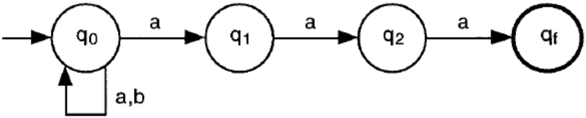


Figura: Grafo do AFN

Equivalência entre AFNs e AFDs

- Por exemplo, considerando o AFN M abaixo, construa o AFD M' correspondente.

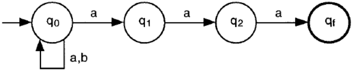


Figura: Grafo do AFN

- O AFD $M' = (\{a, b\}, Q', \delta', \langle q_0 \rangle, F')$ é como segue:
  - $Q' = \{\langle q_0 \rangle \langle q_1 \rangle, \langle q_2 \rangle, \langle q_f \rangle, \langle q_0 q_1 \rangle, \langle q_0 q_2 \rangle, \ldots, \langle q_0 q_1 q_2 q_f \rangle\}$
  - $F' = \{\langle q_f \rangle, \langle q_0 q_f \rangle, \langle q_1 q_f \rangle, \langle q_0 q_1 q_2 q_f \rangle\}$
  - δ' é definido na tabela abaixo:

δ_6'	a	b
$\langle q_0 \rangle$	$\langle q_0 q_1 \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 \rangle$	$\langle q_0 q_1 q_2 \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 q_2 \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$

Figura: Definição de δ'

Equivalência entre AFNs e AFDs

- Dados os AFNs abaixo, construa os AFDs correspondentes:
  - $M = (\{0, 1\}, \{q_0, q_1, q_f\}, \delta, q_0, \{q_f\})$
    $\delta(q_0, 0) = \{q_0, q_f\};\ \delta(q_f, 0) = \{q_1\};\ \delta(q_1, 1) = \{q_1\}$
    $\delta(q_0, 1) = \{q_f\};\ \delta(q_f, 1) = \{q_1\}$
  - $M = (\{0, 1\}, \{q_0, q_1, q_2, q_3, q_f\}, \delta, q_0, \{q_f\})$
    $\delta(q_0, 0) = \{q_1\};\ \delta(q_1, 0) = \{q_0\};\ \delta(q_2, 0) = \{q_3\};\ \delta(q_3, 0) = \{q_2\}$
    $\delta(q_0, 1) = \{q_2, q_f\};\ \delta(q_1, 1) = \{q_3, q_f\};\ \delta(q_2, 1) =$
    $\{q_1, q_f\};\ \delta(q_3, 1) = \{q_1, q_f\}$
  - $M = (\{a, b\}, \{q_0, q_1, q_2, q_f\}, \delta, q_0, \{q_f\})$
    $\delta(q_0, a) = \{q_0, q_1\};\ \delta(q_1, a) = \{q_f\};\ \delta(q_f, a) = \{q_f\}$
    $\delta(q_0, b) = \{q_0, q_2\};\ \delta(q_2, b) = \{q_f\};\ \delta(q_f, b) = \{q_f\}$

Definição

- Um **movimento vazio** é uma transição de um estado a outro sem que haja leitura de símbolo algum da fita.
- É entendido como um determinismo interno do autômato, onde o único efeito aparente é a mudança de estado.
- Autômatos finitos com movimentos vazios são utilizados para facilitar algumas construções e demonstrações relacionadas a autômatos.
- É importante notar que a facilidade de movimentos vazio **não** aumenta o poder de reconhecimento do autômato.

Definição

- Formalmente, um **autômato finito não-determinístico e com movimentos vazios (AFNϵ)** ou simplesmente **autômato finito com movimentos vazios (AFϵ) M** corresponde a uma 5-tupla:

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

onde:

- Σ: alfabeto de símbolos de entrada.
- Q: conjunto finito de estados possíveis do autômato.
- δ: função programa da forma $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$.
- q_0: estado inicial, tal que $q_0 \in Q$.
- F: conjunto de estados finais, tal que $F \subseteq Q$.

Definição

- A função programa com movimentos vazios pode ser interpretada como um grafo finito direto:

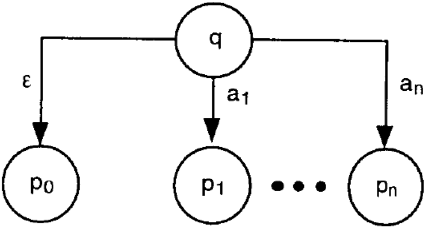


Figura: Grafo da função programa com movimentos vazios

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b\}$:

$$L_7 = \{w| \text{ qualquer simbolo a antecede qualquer simbolo b}\}$$

.

- Um autômato finito com movimentos vazios que reconhece a linguagem é definido por:

$$M_7 = (\{a, b\}, \{q_0, q_f\}, \delta_7, q_0, \{q_f\})$$

onde δ_7 é definida pela tabela abaixo:

δ_7	a	b	ε
q_0	$\{q_0\}$	-	$\{q_f\}$
q_f	-	$\{q_f\}$	

Figura: Definição de δ_7

Exemplos

- O AFϵ M_7 pode ser representado também pelo grafo da figura abaixo:

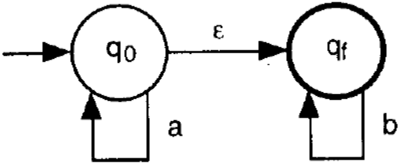


Figura: Grafo do AFϵ M_7

Exemplos

- Considere a linguagem abaixo, definida sobre o alfabeto $\Sigma = \{a, b, c\}$:

$$L_8 = \{w | w \text{ possui como sufixo a ou bb ou ccc}\}$$

.

- Um autômato finito com movimentos vazios que reconhece a linguagem é definido por:

$$M_8 = (\{a, b, c\}, \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f\}, \delta_8, q_0, \{q_f\})$$

.

Exemplos

- O AFϵ M_8 pode ser representado também pelo grafo da figura abaixo:

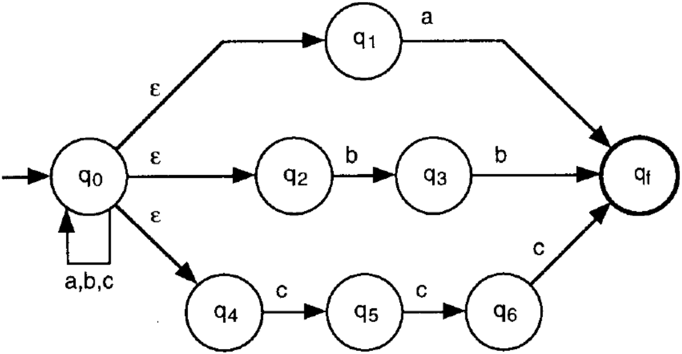


Figura: Grafo do AFϵ M_8

Equivalência entre AFN e AFϵ

- A classe de autômatos finitos com movimentos vazios é equivalente à classe de autômatos finitos não-determinísticos. Para provar tal afirmação, deve-se construir um AFN que realize o mesmo processamento que o AFϵ correspondente.
- Assim, seja $M = (\Sigma, Q, \delta, q_0, F)$ um AFϵ qualquer. Seja $M' = (\Sigma, Q, \delta',$ $q_0, F')$ um AFN construído a partir de M como segue:
  - O AFN M' tem o mesmo número de estados do AFϵ M.
  - As transições de M' acontecem da seguinte forma: para cada estado de M, considere que estados são alcancáveis exclusivamente por meio de transições vazias, quando tomados todos os símbolos do alfabeto Σ.
  - O estado final de M' é todo estado em Q que pode alcançar um estado final de M por meio de transiçÕes em vazio.

Exemplos

- Seja o AFϵ $M_9 = (\{a, b\}, \{q_0, q_1, q_2\}, \delta_9, q_0, \{q_2\})$ ilustrado pelo grafo abaixo:

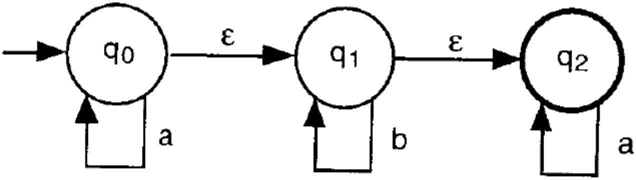


Figura 2.15 Grafo do Autômato Finito com Movimentos Vazios

Figura: Grafo do AFϵ de M_9

- Pergunta-se: que linguagem M_9 reconhece?

Exemplos

- O AFN correspondente a M_9 é ilustrado pelo grafo abaixo:

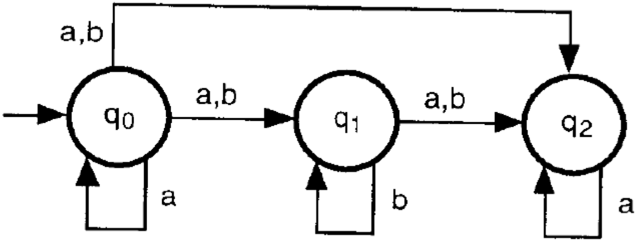


Figura: Grafo do AFN de M_9

Minimização de Autômatos Finitos

- Uma vez que o Autômato Ffoi modelado (usando AFD, AFN ou AF_ϵ), pode-se proceder seu processo de minimização.
- Um **Autômato Mínimo** de uma linguagem regular L é um AFD $M = (\Sigma, Q, \lambda, q_0, F)$ tal que $ACEITA(M) = L$ e que, para qualquer outro AFD $M' = (\Sigma, Q', \lambda', q_0', F')$ tal que $ACEITA(M') = L$, tem-se que $\#Q' \geq \#Q$.

Minimização de Autômatos Finitos

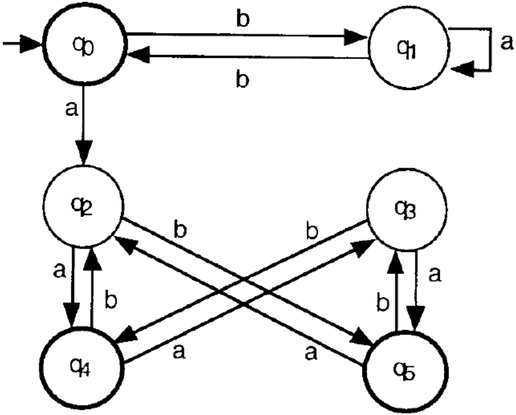
- Para proceder a minimização do autômato, os seguintes pré-requisitos devem ser observados:
  1. Ele deve ser determinístico.
  2. Não pode ter estados inacessíveis.
  3. A função programa deve ser total.
- Tais pré-requisitos podem ser satisfefitos das setuintes formas, caso necessário:
  1. Transformar o AFN ou o AF_ϵ em AFD.
  2. Eliminar estados inacessíveis.
  3. Para transformar uma função parcial em total, basta introduzir um novo estado não-final d e incluir as transiçÕes não-previstas, tendo d como estado destino.

Minimização de Autômatos Finitos

- Suponha que um AFD $M = (\Sigma, Q, \lambda, q_0, F)$ que fatisfaça os pré-requisitos anteriores. O **algoritmo de minimização** é apresentado abaixo:
  1. Contrução de **tabela de não-equivalentes**.
  2. Marcação de estados **trivialmente não-equivalentes**.
  3. Marcação de **estados não-equivalentes**.
  4. **Unificação** de estados equivalentes.
  5. **Exclusão** de estados inúteis.

Minimização de Autômatos Finitos

- Para exemplififcar a aplicaçÇão do algoritmo de minimização, considere o AFD abaixo:



Minimização de Autômatos Finitos

- **PASSO 1**: Contrução de **tabela de não-equivalentes**.

q_1					
q_2					
...					
q_n					
d					
	q_0	q_1	...	q_{n-1}	q_n

Minimização de Autômatos Finitos

- **PASSO 2**: Marcação de estados **trivialmente não-equivalentes**.

q_1	X				
q_2	X				
q_3	X				
q_4		X	X	X	
q_5		X	X	X	
	q_0	q_1	q_2	q_3	q_4

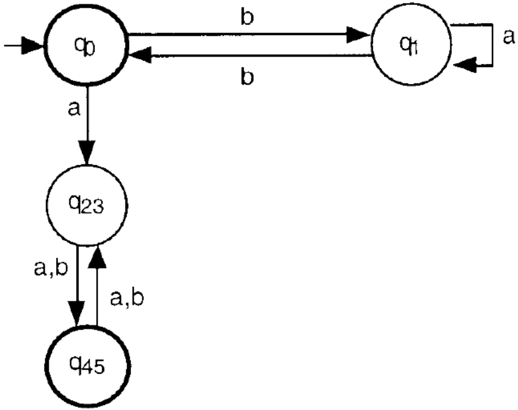
Minimização de Autômatos Finitos

- **PASSO 3**: Marcação de **estados não-equivalentes**.
  - Considere o par (q_0, q_4).
    - Para o símbolo a: $\lambda(q_0, a) = q_2$ e $\lambda(q_4, a) = q_3$.
    - Para o símbolo b: $\lambda(q_0, b) = q_1$ e $\lambda(q_4, b) = q_2$.
    - Se $q_2 \not\equiv q_1$, então $q_0 \not\equiv q_4$.
    - Se $q_3 \not\equiv q_2$, então $q_0 \not\equiv q_4$.

q_1	×				
q_2	×				
q_3	×				
q_4		×	×	×	
q_5		×	×	×	
	q_0	q_1	q_2	q_3	q_4

Minimização de Autômatos Finitos

- **PASSO 4**: **Unificação** de estados equivalentes.



Minimização de Autômatos Finitos

- **PASSO 5**: **Exclusão** de estados inúteis.
  - Neste exemplo, não há estados inúteis. Se houvesse, eles poderiam ser deletados.

Bibliografia

1. MENEZES, Paulo B. Linguagens formais e autômatos. 6ª ed. Porto Alegre: Bookman, 2011.
    - **Capítulos 3 e 4**.
2. VIEIRA, José N. Introdução aos Fundamentos da Computação: Linguagens e Máquinas. 1ª ed. Rio de Janeiro: Thompson, 2006.
    - **Capítulo 2**, disponível em `http://homepages.dcc.ufmg.br/~nvieira/cursos/ftc/livro/copcap2.pdf`.
3. PRADO, Simone das G. D. Apostila 02: Linguagens regulares. 1ª ed. Bauru: UNESP, 2011.
    - Notas de aula disponíveis em `http://wwwp.fc.unesp.br/~simonedp/zipados/TC02.pdf`

Dúvidas?