



SCC-205 - Capítulo 1

Linguagens Regulares e Autômatos Finitos

João Luís Garcia Rosa¹

¹Departamento de Ciências de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
<http://www.icmc.usp.br/~joaoluis>

2009

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Um Exemplo: Parênteses Casados

- *Parênteses casados* incluem todas as cadeias balanceadas de parênteses esquerdos e direitos - por exemplo, $(())$, $()()$ e $(()(())$. A seguinte especificação descreve a linguagem de parênteses casados intuitivamente:
 - 1 A cadeia $()$ é bem formada.
 - 2 Se a cadeia de símbolos A é bem formada, então a cadeia (A) também é.
 - 3 Se as cadeias A e B são bem formadas, então a cadeia AB também é.
- Pode-se agora seguir esta definição intuitiva para obter um sistema de re-escrita - uma **gramática** - que gera exatamente o conjunto de cadeias legais de parênteses casados:
 - 1 $S \rightarrow ()$
 - 2 $S \rightarrow (S)$
 - 3 $S \rightarrow SS$

Um Exemplo: Conceitos

- Estas três regras de re-escrita são chamadas de **produções**. Elas dizem “dada uma cadeia, pode-se formar uma nova cadeia substituindo um S por $()$ ou (S) ou SS ”. Para gerar $((())((())()$), ocorrem as seguintes substituições:

$$\begin{aligned} S &\Rightarrow SS \Rightarrow (S)S \Rightarrow ((())S \Rightarrow ((())(S) \Rightarrow ((())(SS) \Rightarrow \\ &\qquad\qquad\qquad ((())((()S \Rightarrow ((())((())() \end{aligned}$$

- Pode-se resumir a derivação anterior com a notação:

$$S \Rightarrow^* ((())((())()$$

- que significa que “ S deriva (em muitos passos) $((())((())()$ ”. Quando se deseja descrever $v \Rightarrow w$ em palavras, onde v e w são cadeias arbitrárias, deve-se dizer que “ v deriva diretamente w .” Então SS deriva diretamente $(S)S$.

Um Exemplo: Conceitos

- A gramática apresenta dois tipos diferentes de símbolos: os **não-terminais**, ou variáveis, que são os símbolos que podem aparecer em derivações mas não aparecem nas cadeias finais (no exemplo, S é a única variável); e os **terminais**, que são os símbolos que formam a cadeia que se deseja produzir (no exemplo, “(” e “)” são símbolos terminais).
- O estilo de linguagem de programação para escrever gramáticas, a chamada forma de Backus-Naur, ou BNF, usa uma notação levemente diferente. Na BNF as variáveis são fechadas em $\langle \rangle$, e \rightarrow é substituído pelo símbolo “ $::=$ ”. Portanto, a segunda produção do exemplo dos parênteses casados é escrita na notação BNF como:

$$\langle S \rangle ::= (\langle S \rangle)$$



Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- **Gramáticas e Linguagens**
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Alfabeto e Linguagem

- Seja Σ um conjunto finito não vazio de símbolos terminais, ou **alfabeto terminal**. Seja Σ^* o conjunto de todas as cadeias de comprimento finito sobre Σ . Este conjunto infinito de cadeias inclui a cadeia vazia λ .
- Uma linguagem L sobre Σ é qualquer subconjunto de Σ^* . Então a linguagem de parênteses casados, M , é uma linguagem porque $M \subseteq \{(,)\}^*$. Cada cadeia w em L tem um comprimento finito, $|w|$, com $|\lambda| = 0$. Se $w = x_1 x_2 \dots x_n$, cada $x_j \in \Sigma$, temos $|w| = n$. Podemos escrever Σ^+ para denotar a linguagem de cadeias não vazias sobre Σ . Se $w = x_1 x_2 \dots x_n$ e $w' = x'_1 x'_2 \dots x'_m$ são duas cadeias quaisquer, podemos concatená-las para obter

$$ww' = x_1 x_2 \dots x_n x'_1 x'_2 \dots x'_m$$

- Estabelece-se que $w\lambda = \lambda w = w$.

Gramática

- **Definição.** Uma **gramática de estrutura de frase** G é uma quádrupla (Σ, V, S, P) onde Σ e V são alfabetos finitos disjuntos e
 - 1 Σ é o alfabeto terminal para a gramática;
 - 2 V é o alfabeto não terminal ou alfabeto de variável para a gramática;
 - 3 S é o símbolo inicial para a gramática; e
 - 4 P é conjunto de produções da gramática. P é um conjunto de pares (v, w) com v uma cadeia em $(\Sigma \cup V)^*$ contendo no mínimo um símbolo não terminal, enquanto que w é um elemento arbitrário de $(\Sigma \cup V)^*$. Um elemento (v, w) de P é usualmente escrito como $v \rightarrow w$.
- A gramática para a linguagem de parênteses casados pode ser escrita como

$$G = (\{(),\}, \{S\}, S, \{S \rightarrow (), S \rightarrow (S), S \rightarrow SS\})$$

Linguagem

- **Definição.** Seja G uma gramática com símbolo inicial S . A **linguagem** gerada por G , $L(G)$, é definida como o conjunto de cadeias terminais deriváveis do símbolo inicial:

$$L(G) = \{ w \mid w \in \Sigma^* \text{ e } S \Rightarrow^* w \}$$

- Dado G , se $S \Rightarrow^* w$ (onde w é em geral construído tanto de símbolos terminais como não terminais) referimos a w como uma **forma sentencial**. Então $L(G)$ é o conjunto de formas sentenciais terminais.

Exemplos

- **Exemplo 1:** Sejam $\Sigma = \{a, b\}$, $V = \{S\}$,
 $P = \{S \rightarrow aSb, S \rightarrow ab\}$. Neste caso
 $L(G) = \{a^n b^n \mid n > 0\}$. Ou seja, $L(G)$ é o conjunto de todas as seguintes cadeias: um bloco não vazio de a 's, seguido por um bloco de b 's do mesmo comprimento.
- **Exemplo 2:** O conjunto de produções de uma gramática
 - $S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$gera o conjunto de todos os palíndromos sobre o alfabeto terminal $\{a, b\}$.
- **Exemplo 3:** A seguinte gramática gera todas as cadeias sobre o alfabeto terminal $\{0, 1\}$ com um número igual de 0's e 1's:

- $\Sigma = \{0, 1\}$
- $V = \{S, A, B\}$
- $P = \{S \rightarrow 0B \mid 1A, A \rightarrow 0 \mid 0S \mid 1AA, B \rightarrow 1 \mid 1S \mid 0BB\}$.

Gramática Linear a Direita - GLD

- **Definição:** Uma gramática $G = (\Sigma, V, S, P)$ é **linear a direita** (GLD) se toda produção é da forma

$$A \rightarrow bC \text{ ou } A \rightarrow b$$

onde $A, C \in V$ e $b \in \Sigma \cup \{\lambda\}$. Uma linguagem gerada por tal gramática é chamada de linguagem linear a direita.

- **Exemplo 4:** Considere a GLD G_1 com produções

$$S \rightarrow \lambda \mid 0 \mid 0S \mid 1 \mid 1S$$

Claramente, $L(G_1) = \{0, 1\}^*$, o conjunto de todas as cadeias binárias.

- **Exemplo 5:** Agora, considere a GLD G_2 :

$$\begin{aligned} S &\rightarrow \lambda \mid 0S \mid 1T, \\ T &\rightarrow 0T \mid 1S \end{aligned}$$

Pode-se provar que

$$L(G_2) = \{w \mid w \in \{0, 1\}^* \text{ e } w \text{ tem um número par de 1's}\}$$

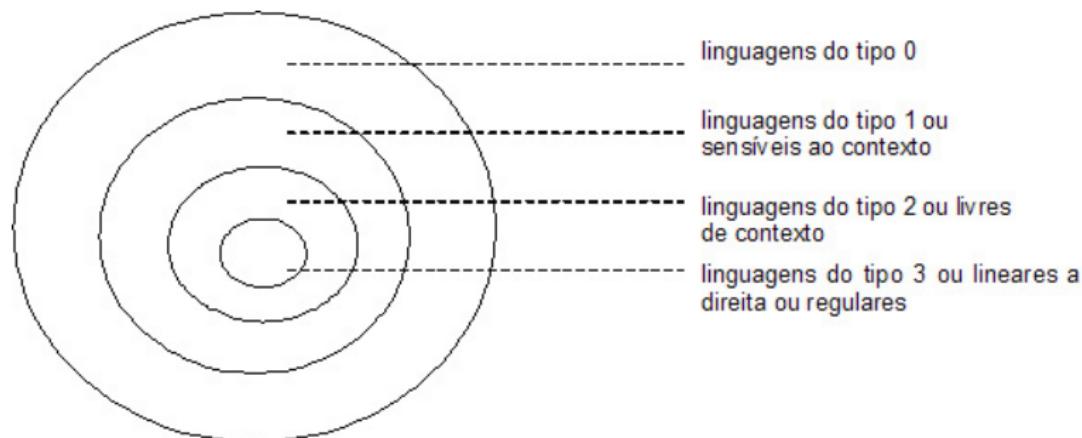
Tipos de Gramáticas e Linguagens

- **Definição:** Seja $G = (\Sigma, V, S, P)$ uma gramática. Então G é classificada como tipo i , $i = 0, 1, 2, 3$, de acordo com restrições na forma das regras de P :
 - Uma gramática é do **tipo 3** se toda produção de P for da forma $A \rightarrow bC$ ou $A \rightarrow b$, onde $A, C \in V$ e $b \in \Sigma$ ou $b = \lambda$. Tais gramáticas são as **gramáticas lineares a direita**.
 - Uma gramática é do **tipo 2** se toda produção de P for da forma $A \rightarrow w$, com $A \in V$ e $w \in (\Sigma \cup V)^*$. Estas são as **gramáticas livres de contexto**.
 - Uma gramática é do **tipo 1** se toda produção de P for da forma $vAw \rightarrow vzw$ onde $z \in (\Sigma \cup V)^+$ (isto é, $z \neq \lambda$). Em adição, permite-se a única regra- λ $S \rightarrow \lambda$ no caso onde S não aparece do lado direito de nenhuma produção. São as **gramáticas sensíveis ao contexto**.
 - Qualquer gramática é do **tipo 0**. Não existem restrições na forma das produções para as gramáticas desta classe. São chamadas de **gramáticas irrestritas**.

A Hierarquia de Chomsky

- Vai-se usar a notação \mathcal{L}_i para as linguagens de tipo i :
$$\mathcal{L}_i = \{L \in \Sigma^* | L = L(G) \text{ para alguma gramática } G \text{ de tipo } i\}$$
- **O Teorema da Hierarquia:** A hierarquia de Chomsky é uma hierarquia estrita de classes de linguagem:

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$$



Operações sobre Linguagens

- Os teóricos da linguagem estão interessados em formas nas quais linguagens dadas (conjuntos de palavras) podem ser combinadas para formar novas linguagens.
- **Definição:** Sejam L_1, L_2 duas linguagens sobre o mesmo alfabeto Σ (isto é, L_1 e L_2 são ambas subconjuntos de Σ^*). Então define-se

- ➊ A **união** de L_1 e L_2 é

$$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ ou } w \in L_2\}$$

- ➋ A **concatenação** de L_1 e L_2 (também chamada “ L_1 ponto L_2 ”) é

$$L_1 \cdot L_2 = \{w \mid w = w_1 w_2 \text{ para algum } w_1, w_2 \text{ com } w_1 \in L_1, w_2 \in L_2\}$$

- ➌ A **iteração** de L_1 (também chamada “ L_1 estrela”) é

$$L_1^* = \{w \mid w = w_1 w_2 \dots w_n \text{ para algum } n \geq 0 \text{ e cada } w_i \in L_1\} = \bigcup_{n \geq 0} L_1^n$$

Propriedades de Fechamento

- Na iteração, define-se as potências L_1^n de L_1 para $n \geq 0$ por
 - Passo Básico: $L_1^0 = \{\lambda\}$
 - Passo de Indução: $L_1^{n+1} = L_1^n \cdot L_1$ para $n \geq 0$
- Então, $L_1^1 = L_1$, $L_1^2 = L_1 \cdot L_1$, etc.
- Diz-se que uma classe de linguagens é **fechada** sob uma operação de linguagem se, quando se aplica esta operação às linguagens que pertencem a esta classe, obtêm-se uma outra linguagem pertencente à classe.
- **Teorema:** A família de linguagens de tipo i é fechada sob \cup , \cdot e $*$ para cada $i = 0, 1, 2, 3$.
- **Proposição:** As linguagens lineares a direita (tipo 3) são fechadas sob complemento e interseção.
- **Teorema:** As linguagens livres de contexto não são fechadas sob interseção ou complemento.

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- **Linguagens Regulares e de Estados Finitos**

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Linguagem Regular

- **Definição:** Seja Σ um alfabeto finito. Uma linguagem $L \subseteq \Sigma^*$ é **regular** se L for finita, ou L pode ser obtida indutivamente usando uma das seguintes operações:
 - 1 L é $L_1 \cup L_2$, a união de L_1 e L_2 , onde L_1 e L_2 são regulares; ou
 - 2 L é $L_1 \cdot L_2$, a concatenação de L_1 e L_2 , onde L_1 e L_2 são regulares; ou
 - 3 L é L_1^* , a iteração de L_1 , onde L_1 é regular.
- Note que o conjunto vazio \emptyset e o conjunto $\{\lambda\}$ são ambos regulares, sendo finitos. As linguagens
 - 1 $\{ab\}^* \cdot \{a\}^*$
 - 2 $\{a, b\}^*$são também regulares. Por que?

Expressão Regular

- **Proposição:** Todo conjunto regular é uma linguagem linear a direita.
- **Definição:** Seja Σ um alfabeto finito. Define-se expressões regulares R e suas denotações de conjuntos regulares $S(R)$ indutivamente como:
 - 1 \emptyset é uma expressão regular com $S(\emptyset) = \emptyset$, o conjunto vazio.
 - 2 λ é uma expressão regular com $S(\lambda) = \{\lambda\}$.
 - 3 Se $a \in \Sigma$, então a é uma expressão regular com $S(a) = \{a\}$.
 - 4 Se R_1 e R_2 são expressões então $(R_1 + R_2)$ é uma expressão regular com $S((R_1 + R_2)) = S(R_1) \cup S(R_2)$.
 - 5 Se R_1 e R_2 são expressões então $(R_1 \cdot R_2)$ é uma expressão regular com $S((R_1 \cdot R_2)) = S(R_1) \cdot S(R_2)$.
 - 6 Se R é regular então $(R)^*$ é regular com $S((R)^*) = (S(R))^*$.

Expressão Regular: Exemplos

- 1 $(a + b)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$
- 2 $(a^* \cdot b^*)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$ (Por que?)
- 3 $(aa + bb)^*$, que abrevia $((a \cdot a) + (b \cdot b))^*$, representa todas as cadeias finitas de a 's e b 's construídas pela concatenação de pares de a 's e pares de b 's.
- 4 $aa(b^* + aaa)c + (a + c)^*$ representa a linguagem que é a união de três sublinguagens:
 - 1 todas as cadeias começando com um a duplo, seguido por qualquer número de b 's, seguido por um c ;
 - 2 a linguagem cadeia $\{aaaaac\}$; e
 - 3 a linguagem construída por todas as cadeias de a 's e c 's.

Expressão Regular e Gramática

- **Exemplo:** Considere a expressão regular $(b^* + (ab)^*)$.

Para achar uma gramática linear a direita para esta expressão, primeiro forme uma gramática para b^* :

$S_1 \rightarrow bS_1 | \lambda$. Então forme uma gramática para (ab) :

$S_2 \rightarrow aB_2, B_2 \rightarrow b$. Isto pode ser estrelado adicionando $S_2 \rightarrow \lambda, B_2 \rightarrow bS_2$. Finalmente, a linguagem completa pode ser gerada adicionando $S \rightarrow bS_1|\lambda|aB_2$, levando a

$$S \rightarrow bS_1|\lambda|aB_2$$

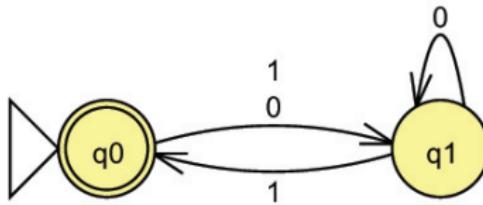
$$S_1 \rightarrow bS_1|\lambda$$

$$S_2 \rightarrow aB_2|\lambda$$

$$B_2 \rightarrow bS_2|b$$

Autômato Finito

- Nesta seção, introduzimos a classe básica de dispositivos de computação chamados de **autômatos finitos**. Estes dispositivos julgam a legalidade de cadeias sobre algum alfabeto finito. Vamos mostrar que a coleção de linguagens que podem ser aceitas por autômatos finitos é exatamente a de linguagens regulares.
- **Exemplo.** O autômato finito dado a seguir aceita a linguagem $((0 + 1)0^*1)^*$.



Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

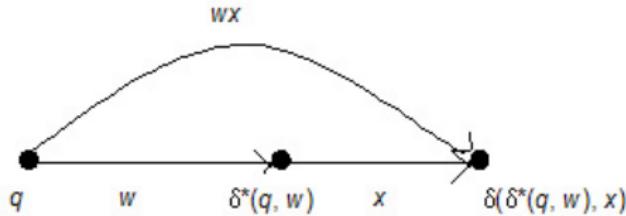
- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Autômato Finito Determinístico

- **Definição:** Um **autômato finito determinístico** (AFD) M é especificado por uma quíntupla $(Q, \Sigma, \delta, q_0, F)$ onde
 - Q é um conjunto finito de estados
 - Σ é o alfabeto terminal
 - $\delta: Q \times \Sigma \rightarrow Q$ é a função de transmissão de estado
 - $q_0 \in Q$ é o estado inicial
 - $F \subset Q$ é o conjunto de estados de aceitação.
- Então o **grafo de estado** de um AFD tem um nó para cada $q \in Q$, que é rotulado; tem uma seta livre apontando para o nó q_0 ; tem um círculo duplo para um nó q apenas no caso de $q \in F$; e tem um arco rotulado x levando de um nó q para um nó q' apenas no caso de $\delta(q, x) = q'$.

Autômato Finito Determinístico

- **Definição:** Dado um AFD M , define-se por indução a função $\delta^*: Q \times \Sigma^* \rightarrow Q$, tal que $\delta^*(q, w)$ é o estado para o qual M irá na cadeia de entrada w se começado no estado q_0 :
 - Passo Básico: $\delta^*(q, \lambda) = q$ para cada estado $q \in Q$. Se M começou no estado q , então quando ele receber a cadeia de entrada vazia ele deve ainda estar no mesmo estado q .
 - Passo de Indução: Para cada estado $q \in Q$, cada cadeia de entrada $w \in \Sigma^*$ e cada símbolo de entrada $x \in \Sigma$, estabelece-se que $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$.



Linguagem de Estados Finitos

- **Definição:** O conjunto $T(M)$ de cadeias aceitas pelo AFD $M = (Q, \Sigma, \delta, q_0, F)$ é

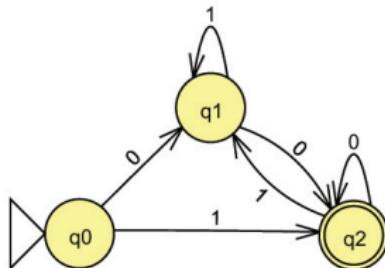
$$T(M) = \{w \mid w \in \Sigma^* \text{ e } \delta^*(q_0, w) \in F\}$$

compreendendo todas as cadeias terminais que enviam M do seu estado inicial q_0 para um estado de aceitação, isto é, um estado em F .

- Diz-se que um subconjunto L de Σ^* é uma **linguagem de estados finitos** (LEF) se for igual a $T(M)$ para algum AFD M .
- **Proposição:** Toda LEF é uma linguagem linear a direita. Por isso, as linguagens LEF são fechadas sob complemento.
- **Teorema:** Toda LEF é um conjunto regular.

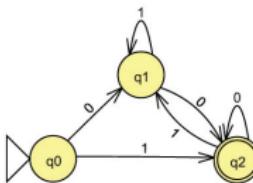
Expressão Regular Equivalente

- Seja o seguinte autômato:



- Para gerar a expressão regular equivalente a este autômato, é necessário estabelecer todos os caminhos que levam do estado inicial q_0 a um estado de aceitação (neste caso q_2). Os caminhos são:
 - De q_0 a q_2 : $1 \cdot 0^*$
 - De q_0 a q_1 a q_2 : $0 \cdot 1^* \cdot 0 \cdot 0^*$
 - De q_0 a q_1 a q_2 a q_1 a q_2 : $0 \cdot 1^* \cdot 0 \cdot 0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*$

Expressão Regular Equivalente



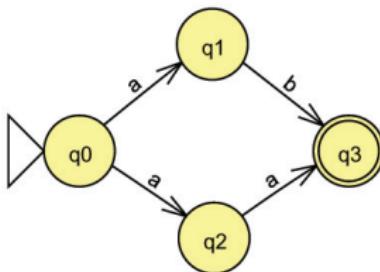
- Observe que há um ciclo se repetindo entre q_2 e q_1 , logo pode-se representar os caminhos que levam de q_0 a q_2 passando por q_1 da seguinte forma:
 - De q_0 a q_1 a q_2 a $(q_1 \text{ a } q_2)^*$: $0 \cdot 1^* \cdot 0 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$
 - Outro caminho é de q_0 a q_2 e realizando o ciclo $q_1 \Leftrightarrow q_2$:
 - De q_0 a q_2 a $(q_1 \text{ a } q_2)^*$: $1 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$, que inclui $1 \cdot 0^*$ inicial.
 - Logo a expressão total fica:
- $$1 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^* + 0 \cdot 1^* \cdot 0 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^* = \\ (1 \cdot 0^* + 0 \cdot 1^* \cdot 0 \cdot 0^*) \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$$

Não Determinismo

- Deseja-se provar que toda linguagem linear a direita é uma LEF. Considere a seguinte gramática linear a direita:

$$(\{a, b\}, \{q_0, q_1, q_2\}, q_0, \\ \{q_0 \rightarrow aq_1, q_1 \rightarrow b, q_0 \rightarrow aq_2, q_2 \rightarrow a\})$$

- Esta gramática leva à seguinte máquina:



- A estrutura obtida não é um AFD, porque dois arcos saindo de q_0 têm o mesmo rótulo, a . Isto viola a condição da regra de transição para um AFD ser uma função.

Autômato Finito Não Determinístico

- Quando transições múltiplas deste tipo estão presentes em M , diz-se que M é uma **máquina não-determinística**.
- **Definição:** Um **autômato finito não determinístico** (AFN) M é especificado por uma quíntupla $(Q, \Sigma, \delta, Q_0, F)$ onde
 - Q é um conjunto finito de estados
 - Σ é o alfabeto terminal
 - $\delta: Q \times \Sigma \rightarrow 2^Q$ atribui a cada par estado-entrada (q, x) um conjunto $\delta(q, x) \subset Q$ de próximos estados possíveis (2^Q é o conjunto potência de Q ou conjunto de todos os subconjuntos de Q)
 - $Q_0 \subset Q$ é o conjunto de estados iniciais
 - $F \subset Q$ é o conjunto de estados de aceitação.

Autômato Finito Não Determinístico

- Viu-se que um AFN é como um AFD exceto que existe um conjunto de estados iniciais e um conjunto de próximos estados possíveis.
- Estas complicações significam que uma cadeia pode induzir caminhos múltiplos através de um AFN, alguns terminando em estados de aceitação, outros terminando em estados de não aceitação.
- Lida-se com esta ambigüidade dizendo que $w \in \Sigma^*$ é aceita se existe pelo menos uma forma de chegar a um estado de aceitação.
- Vai-se definir agora $\delta^* : 2^Q \times \Sigma^* \rightarrow 2^Q$ para um AFN tal que $\delta^*(p, w)$ é o conjunto de estados alcançáveis a partir de algum estado em p pela aplicação da cadeia de entrada w .

Autômato Finito Não Determinístico

- **Definição:** Estenda $\delta : Q \times \Sigma \rightarrow 2^Q$ para um AFN para $\delta^* : 2^Q \times \Sigma^* \rightarrow 2^Q$ por:
 - Passo Básico: $\delta^*(p, \lambda) = p$ para cada $p \subset Q$.
 - Passo de Indução: $\delta^*(p, wx) = \bigcup\{\delta(q, x) | q \in \delta^*(p, w)\}$ para cada $w \in \Sigma^*$, $x \in \Sigma$ e $p \subset Q$.
- Estabelece-se que

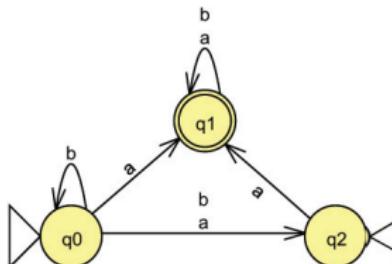
$$T(M) = \{w | w \in \Sigma^* \text{ e } \delta^*(Q_0, w) \cap F \neq \emptyset\}$$

é o conjunto de cadeias de entrada tal que no mínimo um caminho, rotulado com símbolos de w em ordem, através do grafo de estado leva M de um estado inicial para um estado de aceitação.

- **Proposição:** Um subconjunto de Σ^* é uma linguagem de estados finitos se e somente se ele for $T(M)$ para algum AFN M .

Conversão de AFN para AFD

- **Exemplo:** Considere o seguinte AFN.



- Os estados q_0 e q_2 são estados iniciais, enquanto que o estado q_1 é o único estado de aceitação.
- Existem $2^{|Q|}$ (neste caso, $2^3 = 8$) estados no AFD equivalente, mas na prática não é necessário considerar todos eles, porque muitos são inalcançáveis.

Conversão de AFN para AFD

- Tira-se vantagem desta observação começando do novo estado inicial q_{02} , correspondente à união de q_0 e q_2 , e construindo estados adicionais na medida em que eles são gerados. Começa-se com

δ	a	b
$\{q_0, q_2\}$	$\{q_2, q_1\}$	$\{q_0, q_2\}$

que descreve a ação de δ no estado inicial q_{02} de entradas a e b , respectivamente. De maneira análoga, chamar-se-á o estado $\{q_2, q_1\}$ de q_{21} :

δ	a	b
q_{02}	q_{21}	q_{02}

Conversão de AFN para AFD

- Apenas um novo estado foi gerado, assim produz-se sua linha:

δ	a	b
q_{21}	q_1	q_1

- Então tem-se:

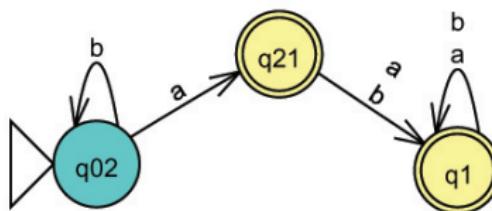
δ	a	b
q_1	q_1	q_1

- A tabela completa do δ fica:

δ	a	b
q_{02}	q_{21}	q_{02}
q_{21}	q_1	q_1
q_1	q_1	q_1

Conversão de AFN para AFD

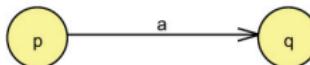
- Todos os estados resultantes que contêm estados finais do AFN original são estados de aceitação no AFD correspondente. O diagrama de estados do AFD equivalente é:



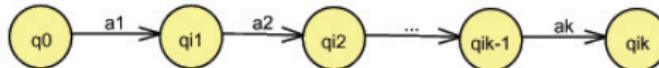
- Teorema:** Toda linguagem linear a direita L é uma LEF.
- Conclui-se portanto que as classes de linguagens lineares a direita, linguagens regulares e linguagens de estados finitos coincidem.

Lema do Bombeamento para Linguagens Regulares

- **Teorema:** (*O Lema do Bombeamento para Linguagens Regulares*). Seja M um AFD qualquer com n estados, e seja $z \in T(M)$, $|z| \geq n$. Então z pode ser escrito como uvw e a linguagem $uv^*w \subset T(M)$. Isto é, para todo $j \geq 0$, $uv^jw \in T(M)$.
- **PROVA:** Vai-se usar a seguinte notação

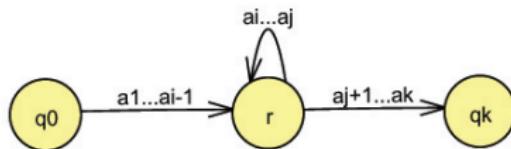


para indicar que $\delta(p, a) = q$ em M . Seja $z = a_1a_2\dots a_k$. Então pode-se tracejar o comportamento de M em z escrevendo



Lema do Bombeamento para Linguagens Regulares

Quando $|z| \geq n$, no mínimo $n + 1$ estados devem aparecer no tracejamento, e então algum estado, diga-se r , deve aparecer duas vezes.



Seja $z = uvw$ onde $v = a_i \dots a_j$. Então as cadeias uw , uvw , $uvvw$, etc., levam ao mesmo estado final de M , e portanto se $uvw \in T(M)$ então tem-se $uv^m w \in T(M)$ para todo $m \geq 0$.

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- **Arcos- λ**
- Autômato Mínimo

3 Autômatos Finitos com Saída

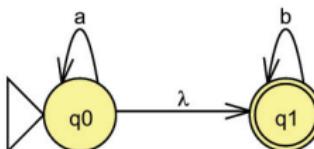
- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Autômatos Finitos com Arcos- λ

- Arcos- λ são arcos de transição em autômatos finitos não-determinísticos (AFN) que permitem a mudança de estado sem consumir símbolo da cadeia terminal.
- Uma das vantagens dos arcos- λ nos estudos das linguagens formais é o fato de facilitar algumas construções relacionadas com os autômatos [4].
- Vale lembrar que, assim como os autômatos não-determinísticos, a existência de arcos- λ não aumenta o poder computacional no reconhecimento de linguagens.
- Todo autômato finito com arcos- λ ($AF\lambda$) pode ser simulado por um autômato finito não-determinístico (AFN).

Autômatos Finitos com Arcos- λ

- Exemplo: Seja o seguinte autômato com arcos- λ (AF λ):



Este AF λ $M = (\{q_0, q_1\}, \{a, b\}, \delta, \{q_0\}, \{q_1\})$ processa a linguagem $L = \{w \in \{a, b\}^* \mid \text{os símbolos } a's \text{ antecedem os } b's\}$. O δ é especificado pela tabela

δ	a	b	λ
q_0	$\{q_0\}$	\emptyset	$\{q_1\}$
q_1	\emptyset	$\{q_1\}$	\emptyset

Conversão de AF λ para AFN

- O AFN equivalente a este AF λ é obtido, num processo análogo à obtenção do autômato determinístico, re-escrevendo a tabela do δ da seguinte forma:
 - elimina-se a coluna do λ , incluindo em cada estado que possui arco- λ o destino do mesmo:

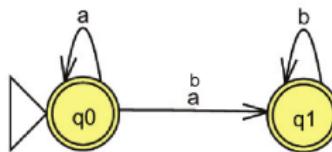
δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_1\}$

- incluir-se os demais estados, seguindo o mesmo procedimento:

δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	\emptyset	$\{q_1\}$

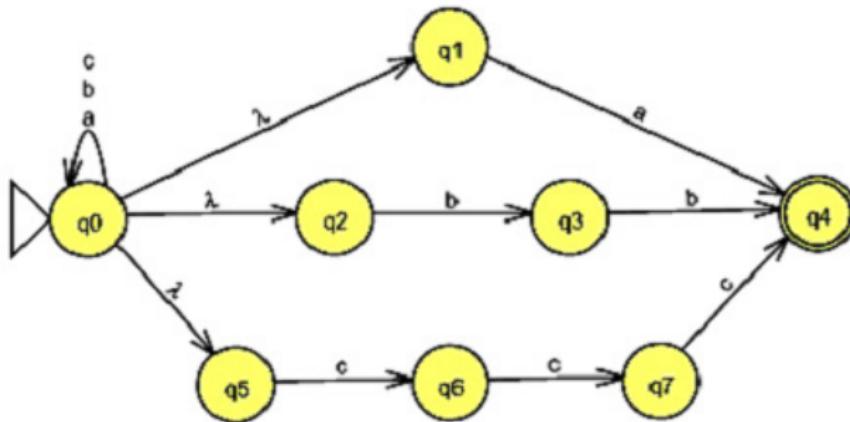
Conversão de AF λ para AFN

Além disso, o conjunto F de estados finais pode ser aumentado, incluindo os estados ligados aos estados finais originais pelo arco- λ . Logo o AFN equivalente fica:



Conversão de AF λ para AFN

- Exemplo: Seja o seguinte autômato com arcos- λ (AF λ):



Conversão de AF λ para AFN

- Este AF λ

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a, b, c\}, \delta, \{q_0\}, \{q_4\})$ processa a linguagem $L = \{w \in \{a, b, c\}^* \text{ e } w \text{ termina com } a, bb, \text{ ou } ccc\}$. O δ é especificado pela tabela

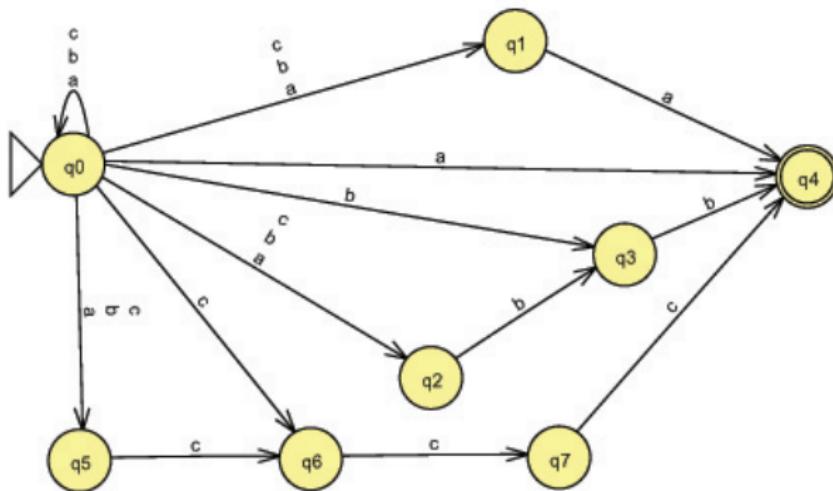
δ	a	b	c	λ
q_0	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$	$\{q_1, q_2, q_5\}$
q_1	$\{q_4\}$	\emptyset	\emptyset	\emptyset
q_2	\emptyset	$\{q_3\}$	\emptyset	\emptyset
q_3	\emptyset	$\{q_4\}$	\emptyset	\emptyset
q_4	\emptyset	\emptyset	\emptyset	\emptyset
q_5	\emptyset	\emptyset	$\{q_6\}$	\emptyset
q_6	\emptyset	\emptyset	$\{q_7\}$	\emptyset
q_7	\emptyset	\emptyset	$\{q_4\}$	\emptyset

Conversão de AF λ para AFN

- O AFN equivalente a este AF λ é obtido:

δ	a	b	c
q_0	$\{q_0, q_1, q_2, q_5, q_4\}$	$\{q_0, q_1, q_2, q_3, q_5\}$	$\{q_0, q_1, q_2, q_5, q_6\}$
q_1	$\{q_4\}$	\emptyset	\emptyset
q_2	\emptyset	$\{q_3\}$	\emptyset
q_3	\emptyset	$\{q_4\}$	\emptyset
q_4	\emptyset	\emptyset	\emptyset
q_5	\emptyset	\emptyset	$\{q_6\}$
q_6	\emptyset	\emptyset	$\{q_7\}$
q_7	\emptyset	\emptyset	$\{q_4\}$

Conversão de AF λ para AFN



Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Minimização de um Autômato Finito

- **Definição:** Para um dado conjunto A , $|A|$ denota o cardinal (número de elementos) de A .
- **Definição:** Um **autômato mínimo** de uma linguagem regular L é um autômato finito determinístico (AFD) $M = (Q, \Sigma, \delta, q_0, F)$ tal que $L = T(M)$ e que, para qualquer outro AFD $M' = (Q', \Sigma, \delta', q'_0, F')$ tal que $L = T(M')$, tem-se que $|Q'| \geq |Q|$.
- Um autômato finito para ser minimizado precisa satisfazer aos seguintes pré-requisitos:
 - 1 deve ser determinístico (AFD);
 - 2 não pode ter estados inacessíveis;
 - 3 δ deve ser total.

Minimização de um Autômato Finito

- Caso o autômato não satisfaça alguns dos pré-requisitos é necessário gerar um autômato equivalente:
 - ➊ gerando um autômato determinístico equivalente (de AF λ para AFN para AFD);
 - ➋ eliminando os estados inacessíveis e suas correspondentes transições;
 - ➌ transformando δ em total, introduzindo um estado *sumidouro* \otimes e incluindo as transições não previstas para este estado.

Algoritmo de Minimização

- **Definição: Algoritmo de Minimização.** Seja um AFD $M = (Q, \Sigma, \delta, q_0, F)$ que satisfaz os pré-requisitos de minimização. Para encontrar o autômato mínimo equivalente deve-se realizar a:
- 1 **construção da tabela de não-equivalências de estados [2].** Construir uma tabela relacionando estados diferentes, onde cada par de estados ocorre somente uma vez:

q_1					
q_2					
...					
q_n					
\otimes					
	q_0	q_1	...	q_{n-1}	q_n

Algoritmo de Minimização

2 marcação dos estados obviamente não equivalentes.

Marcar estados finais e não finais.

3 marcação dos estados não equivalentes.

Para cada par $\{q, q'\}$ e para cada símbolo $a \in \Sigma$, suponha que $\delta(q, a) = p$ e $\delta(q', a) = p'$ e:

- se $p = p'$, então q é equivalente a q' para a e não deve ser marcado;
- se $p \neq p'$ e o par $\{p, p'\}$ não está marcado, então $\{q, q'\}$ é incluído em uma lista para posterior análise;
- se $p \neq p'$ e o par $\{p, p'\}$ está marcado:
 - $\{q, q'\}$ não é equivalente e deve ser marcado;
 - se $\{q, q'\}$ encabeça uma lista de pares, então marcar todos os pares da lista.

Algoritmo de Minimização

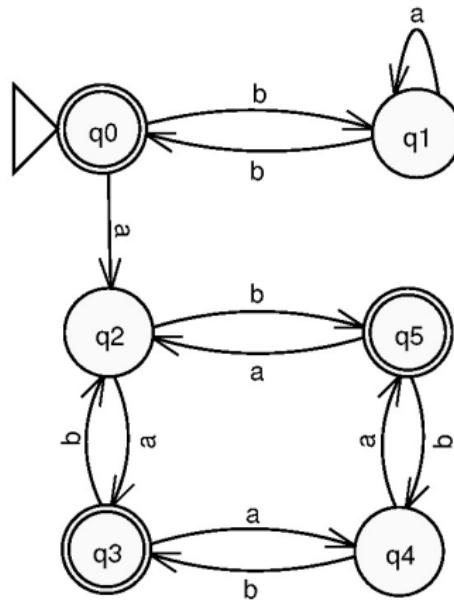
4 unificação dos estados equivalentes. Os estados dos pares não marcados são equivalentes e podem ser unificados como segue:

- a equivalência é transitiva;
- pares de estados não finais equivalentes podem ser unificados como um único estado não final;
- pares de estados finais equivalentes podem ser unificados como um único estado final;
- se algum dos estados equivalentes é inicial, então o correspondente estado unificado é inicial.

5 exclusão dos estados inúteis. Os estados inúteis devem ser excluídos. Um estado q é inútil se é não final e a partir de q não é possível atingir um estado final. Deve-se reparar que o estado \otimes (se incluído) sempre é inútil.

Exemplo: Minimização de um Autômato Finito

- **Exemplo:** Seja o seguinte AFD:



Exemplo: Minimização de um Autômato Finito

1 construção da tabela de não-equivalências de estados:

q_1					
q_2					
q_3					
q_4					
q_5					
	q_0	q_1	q_2	q_3	q_4

2 marcação dos estados finais e não finais:

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

Exemplo: Minimização de um Autômato Finito

3 análise dos pares não marcados (representados por -):

q_1	\times				
q_2	\times	-			
q_3	-	\times	\times		
q_4	\times	-	-	\times	
q_5	-	\times	\times	-	\times
	q_0	q_1	q_2	q_3	q_4

- 1. par $\{q_0, q_3\}$:

$$\begin{array}{ll} \delta(q_0, a) = q_2 & \delta(q_0, b) = q_1 \\ \delta(q_3, a) = q_4 & \delta(q_3, b) = q_2 \end{array}$$

como $\{q_1, q_2\}$ e $\{q_2, q_4\}$ são não marcados (-), $\{q_0, q_3\}$ é incluído nas listas encabeçadas por $\{q_1, q_2\}$ e $\{q_2, q_4\}$:

$$\begin{aligned} & [\{q_1, q_2\}, \{q_0, q_3\}] \\ & [\{q_2, q_4\}, \{q_0, q_3\}] \end{aligned}$$

Exemplo: Minimização de um Autômato Finito

- 2. par $\{q_0, q_5\}$:

$$\begin{array}{ll}\delta(q_0, a) = q_2 & \delta(q_0, b) = q_1 \\ \delta(q_5, a) = q_2 & \delta(q_5, b) = q_4\end{array}$$

como $\{q_1, q_4\}$ é não marcado (-) (e como $\{q_2, q_2\}$ é obviamente não equivalente), $\{q_0, q_5\}$ é incluído na lista encabeçada por $\{q_1, q_4\}$:

$$[\{\mathbf{q_1}, \mathbf{q_4}\}, \{q_0, q_5\}]$$

Exemplo: Minimização de um Autômato Finito

- 3. par $\{q_1, q_2\}$:

$$\begin{array}{ll}\delta(q_1, a) = q_1 & \delta(q_1, b) = q_0 \\ \delta(q_2, a) = q_3 & \delta(q_2, b) = q_5\end{array}$$

como $\{q_1, q_3\}$ é marcado (\times) então $\{q_1, q_2\}$ também é marcado (+). Como $\{q_1, q_2\}$ encabeça uma lista, $\{q_0, q_3\}$ também é marcado (+).

q_1	\times				
q_2	\times	+			
q_3	+	\times	\times		
q_4	\times	-	-	\times	
q_5	-	\times	\times	-	\times
	q_0	q_1	q_2	q_3	q_4

Exemplo: Minimização de um Autômato Finito

- 4. par $\{q_1, q_4\}$:

$$\begin{array}{ll} \delta(q_1, a) = q_1 & \delta(q_1, b) = q_0 \\ \delta(q_4, a) = q_5 & \delta(q_4, b) = q_3 \end{array}$$

como $\{q_1, q_5\}$ é marcado (\times) e $\{q_0, q_3\}$ também é marcado (+) então $\{q_1, q_4\}$ também é marcado (+). Como $\{q_1, q_4\}$ encabeça uma lista, $\{q_0, q_5\}$ também é marcado (+).

q_1	\times				
q_2	\times	+			
q_3	+	\times	\times		
q_4	\times	+	-	\times	
q_5	+	\times	\times	-	\times
	q_0	q_1	q_2	q_3	q_4

Exemplo: Minimização de um Autômato Finito

- 5. par $\{q_2, q_4\}$:

$$\begin{array}{ll}\delta(q_2, a) = q_3 & \delta(q_2, b) = q_5 \\ \delta(q_4, a) = q_5 & \delta(q_4, b) = q_3\end{array}$$

como $\{q_3, q_5\}$ é não marcado (-), $\{q_2, q_4\}$ é incluído na lista encabeçada por $\{q_3, q_5\}$:

$$[\{\textbf{q}_3, \textbf{q}_5\}, \{q_2, q_4\}]$$

- 6. par $\{q_3, q_5\}$:

$$\begin{array}{ll}\delta(q_3, a) = q_4 & \delta(q_3, b) = q_2 \\ \delta(q_5, a) = q_2 & \delta(q_5, b) = q_4\end{array}$$

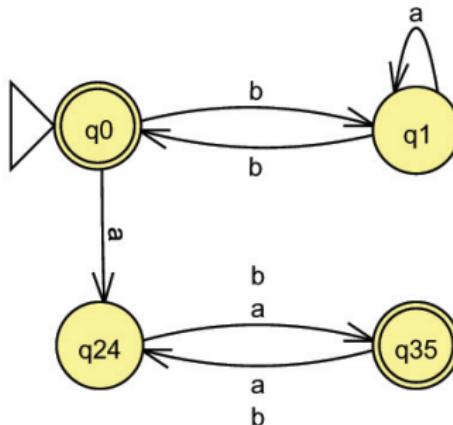
como $\{q_2, q_4\}$ é não marcado (-), $\{q_3, q_5\}$ é incluído na lista encabeçada por $\{q_2, q_4\}$:

$$[\{\textbf{q}_2, \textbf{q}_4\}, \{q_0, q_3\}, \{q_3, q_5\}]$$

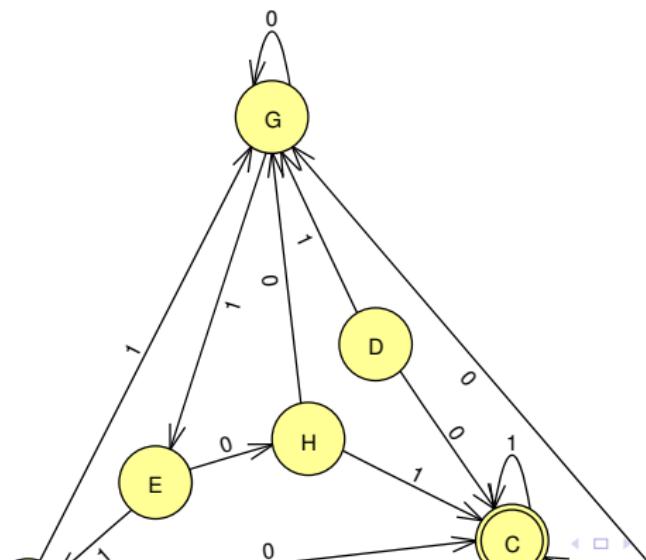
Exemplo: Minimização de um Autômato Finito

- 4 **unificação dos estados equivalentes:** como os estados $\{q_2, q_4\}$ e $\{q_3, q_5\}$ são não marcados, unifica-se:
 - q_2 e q_4 : unificação dos estados não finais: q_{24}
 - q_3 e q_5 : unificação dos estados finais: q_{35}
- 5 **exclusão dos estados inúteis.** Não há estados inúteis.

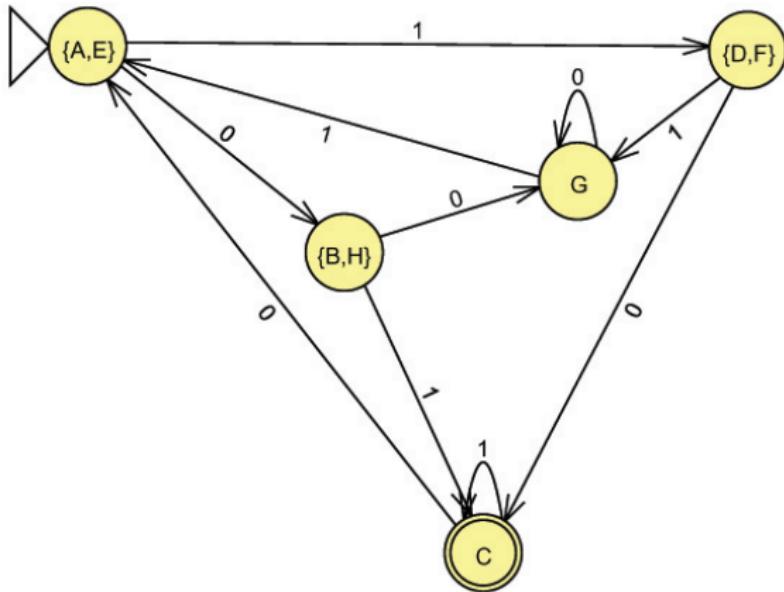
Logo, o **autômato mínimo** equivalente fica:



Outro Exemplo: AFD não mínimo



Outro Exemplo: AFD mínimo equivalente



Autômatos Finitos com Saída

Aplicações que incluem geração de cadeias de saída para os Autômatos Finitos associadas às transições (*Máquinas de Mealy*) ou aos estados (*Máquinas de Moore*) foram propostas [3, 6]. Em ambas, a saída não pode ser lida e é definida sobre um alfabeto de saída (Δ), que pode ser igual ao alfabeto de entrada (Σ). A saída é armazenada numa fita e o resultado do processamento é o seu estado final (aceita/rejeita) e a informação contida na fita de saída.

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Máquina de Mealy

- **Definição: Máquina de Mealy.** Uma Máquina de Mealy (MMe) M é um AFD com saídas associadas às transições. É uma sétupla $M = (Q, \Sigma, \delta, q_0, F, \Delta)$, onde

- Q é um conjunto finito de estados
- Σ é o alfabeto terminal
- $\delta : Q \times \Sigma \rightarrow Q \times \Delta^*$ é a função de transmissão de estado
- $q_0 \in Q$ é o estado inicial
- $F \subset Q$ é o conjunto de estados de aceitação
- Δ é o alfabeto de símbolos de saída.

O processamento de uma Máquina de Mealy para uma cadeia w ocorre da mesma forma que de um AFD, exceto pela fita de saída (cadeia de símbolos de Δ), que pode ser vazia, quando todas as transições geram saída vazia.

Neste caso, a MMMe é equivalente ao AFD.

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- **Máquinas de Moore**
- Exemplos

Máquina de Moore

- A Máquina de Moore possui uma segunda função que gera uma cadeia de saída (que pode ser vazia) para cada estado da máquina.
- **Definição. Máquina de Moore.** Uma Máquina de Moore (MMo) M é um AFD com saídas associadas aos estados. É uma séptupla $M = (Q, \Sigma, \delta, q_0, F, \Delta, \delta_S)$, onde
 - Q é um conjunto finito de estados
 - Σ é o alfabeto terminal
 - $\delta : Q \times \Sigma \rightarrow Q$ é a função parcial de transmissão de estado
 - $q_0 \in Q$ é o estado inicial
 - $F \subset Q$ é o conjunto de estados de aceitação
 - Δ é o alfabeto de símbolos de saída
 - $\delta_S : Q \rightarrow \Delta^*$ é a função total de saída.

Máquina de Moore

- Um analisador lexical usado em compiladores é um exemplo de aplicação de Máquina de Moore. Um analisador lexical é um autômato finito, em geral determinístico, que identifica os componentes básicos de uma linguagem de programação, como por exemplo, palavras reservadas, identificadores, números, etc. Para isso, a MMo teria:
 - Um estado final, associado a cada unidade lexical;
 - Cada estado final possui uma saída, que descreve a unidade identificada;
 - Para os demais estados (não finais) a saída é a cadeia vazia λ .

Sumário

1 Gramáticas e Linguagens

- A Primeira Linguagem
- Gramáticas e Linguagens
- Linguagens Regulares e de Estados Finitos

2 Autômatos de Estados Finitos

- Autômatos Finitos
- Arcos- λ
- Autômato Mínimo

3 Autômatos Finitos com Saída

- Máquinas de Mealy
- Máquinas de Moore
- Exemplos

Máquinas de Mealy e de Moore: Exemplos

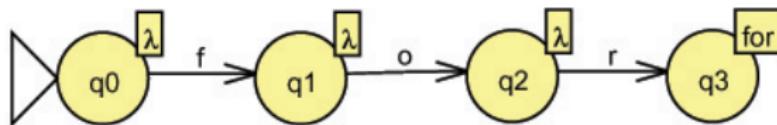
- Exemplo:** Uma Máquina de Moore para o reconhecimento da palavra **for**. Neste caso,
 $M = (\{q_0, q_1, q_2, q_3\}, \{f, o, r\}, \delta, q_0, \{q_3\}, \{f, o, r\}, \delta_S)$, onde

δ	f	o	r
q_0	q_1	-	-
q_1	-	q_2	-
q_2	-	-	q_3
q_3	-	-	-

δ_S	<i>cadeia de saída</i>
q_0	λ
q_1	λ
q_2	λ
q_3	for

Máquinas de Mealy e de Moore: Exemplos

- Veja que neste caso, para cada símbolo processado na cadeia de entrada, a cadeia vazia é saída, exceto para o estado final q_3 , que a cadeia de saída é **for**. Veja o diagrama de estados:



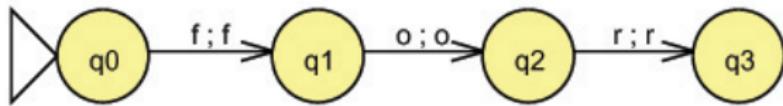
Máquinas de Mealy e de Moore: Exemplos

- **Exemplo:** Uma Máquina de Mealy para o mesmo reconhecimento do exemplo anterior (palavra **for**). Neste caso, $M = (\{q_0, q_1, q_2, q_3\}, \{f, o, r\}, \delta, q_0, \{q_3\}, \{f, o, r\})$, onde

δ	f	o	r
q_0	$\{q_1, f\}$	-	-
q_1	-	$\{q_2, o\}$	-
q_2	-	-	$\{q_3, r\}$
q_3	-	-	-

Máquinas de Mealy e de Moore: Exemplos

- Neste caso, ao contrário da MMo anterior, a cada transição um símbolo é saído: o mesmo símbolo da transição. Ao final do processamento, a cadeia **for** é formada na fita de saída. Diagrama de estados:



Referências I



[1] Hopcroft, J. E., Ullman, J. D.

Formal Languages and Their Relation to Automata.

Addison-Wesley Publishing Company, 1969.



[2] Hopcroft, J. E., Ullman, J. D. e Motwani, R.

Introdução à Teoria de Autômatos, Linguagens e Computação.

Tradução da segunda edição americana. Editora Campus, 2003.



[3] Mealy, G. H.

A method for synthesizing sequential circuits.

Bell Systems Technical Journal 34:5, pp. 1045-1079, 1955.

Referências II



[4] Menezes, P. B.

Linguagens Formais e Autômatos.

Série Livros Didáticos. 4a. Edição. Instituto de Informática da UFRGS. Editora Sagra Luzzatto, 1997.



[5] Moll, R. N., Arbib, M. A., and Kfoury, A. J.

An Introduction to Formal Language Theory.

Springer-Verlag, 1988.



[6] Moore, E. F.

Gedanken experiments on sequential machines.

in C. E. Shannon and J. McCarthy (Eds.), *Automata Studies*, Princeton University Press, pp. 129-153, 1956.

Referências III



[7] Rosa, J. L. G.

SCE-185: Teoria da Computação e Linguagens Formais.
Notas de Aula. Ciências de Computação. ICMC-USP, 2008.