Linguagens Regulares

Prof. Marcus Vinícius Midena Ramos

Universidade Federal do Vale do São Francisco

16 de novembro de 2023

marcus.ramos@univasf.edu.br
www.univasf.edu.br/~marcus.ramos

Bibliografia

1. *Linguagens Formais: Teoria, Modelagem e Implementação*
M.V.M. Ramos, J.J. Neto e I.S. Vega
Bookman, 2009

Roteiro

Gramáticas lineares à direita ou à esquerda

Gramáticas lineares à direita ou à esquerda são aquelas cujas regras $\alpha \to \beta$ atendem às seguintes condições:

- $\alpha \in N$;
- $\beta \in (\Sigma \cup \{\varepsilon\})(N \cup \{\varepsilon\})$ se linear à direita, ou
  $\beta \in (N \cup \{\varepsilon\})(\Sigma \cup \{\varepsilon\})$ se linear à esquerda.

Demonstra-se que as gramáticas lineares à esquerda ou à direita geram exatamente a mesma classe de linguagens. Portanto, é indiferente o emprego de uma ou outra dessas duas variantes de gramática, já que ambas possuem a mesma capacidade de representação de linguagens.

Gramáticas regulares

Por esse motivo, as gramáticas lineares à direita ou à esquerda são também denominadas **gramáticas regulares**. Este termo serve para designar ambos os tipos de gramática linear. As linguagens geradas por gramáticas regulares recebem o nome de **linguagens regulares**.

Gramáticas lineares não-unitárias

Alguns autores consideram as seguintes extensões na definição das regras $\alpha \to \beta$ de gramáticas lineares à direita e à esquerda:

- $\alpha \in N$;
- $\beta \in \Sigma^*(N \cup \{\varepsilon\})$ se linear à direita, ou
  $\beta \in (N \cup \{\varepsilon\})\Sigma^*$ se linear à esquerda.

Nessas extensões, admite-se uma quantidade qualquer de símbolos terminais no lado direito das produções gramaticais, e não no máximo um, como foi estabelecido na definição original. Tais extensões em nada alteram a classe de linguagens representáveis por esses tipos de gramáticas, constituindo o seu uso mera conveniência. O Teorema 1.1 traz a demonstração dessa equivalência.

Equivalência unitárias/não-unitárias

Teorema

Teorema 1.1 "Se G_1 é uma gramática composta apenas de produções do tipo $\alpha \to \beta, \alpha \in N, \beta \in \Sigma^*(N \cup \{\varepsilon\})$, então existe uma gramática equivalente G_2 composta apenas de produções do tipo $\alpha \to \beta, \alpha \in N, \beta \in (\Sigma \cup \{\varepsilon\})(N \cup \{\varepsilon\})$."
G_2 pode ser obtida a partir de G_1 pelo Algoritmo 1.1, o qual substitui as regras $\alpha \to \beta, \beta \in \Sigma^* N$, por um conjunto equivalente de novas regras $\alpha' \to \beta', \beta' \in (\Sigma \cup \{\varepsilon\})(N \cup \{\varepsilon\})$.[1]

[1] Tal tipo de gramática recebe, em alguns textos, a denominação de **gramática linear unitária à direita**.

Equivalência unitárias/não-unitárias

Algoritmo

Algoritmo 1.1 *"Mapeamento das produções de uma gramática linear à direita, na forma* $\alpha \to \beta$, $\beta \in \Sigma^*(N \cup \{\varepsilon\})$, *em conjuntos de produções equivalentes, na forma* $\alpha \to \beta$, $\beta \in (\Sigma \cup \{\varepsilon\})(N \cup \{\varepsilon\})$."

- *Entrada: uma gramática linear à direita* $G_1 = (V_1, \Sigma, P_1, S_1)$, *cujas produções são da forma* $\alpha \to \beta, \beta \in \Sigma^*(N \cup \{\varepsilon\})$;
- *Saída: uma gramática linear à direita* $G_2 = (V_2, \Sigma, P_2, S_2)$, *tal que* $L(G_2) = L(G_1)$ *e cujas produções são todas da forma* $\alpha \to \beta, \beta \in (\Sigma \cup \{\varepsilon\})(N \cup \{\varepsilon\})$;

Equivalência unitárias/não-unitárias

Algoritmo

Método:

1. $N_2 \leftarrow N_1$;
2. $P_2 \leftarrow \emptyset$;
3. *Para cada* $\alpha \rightarrow \beta \in P_1, \beta = \sigma_1 ... \sigma_n K$, *com* $K \in N \cup \{\varepsilon\}$ *e* $n \geqslant 0$, *faça:*
   - *Se* $\beta = \varepsilon$, $\beta \in \Sigma$, $\beta \in N$, *ou* $\beta \in \Sigma N$, *então* $P_2 \leftarrow P_2 \cup \{\alpha \rightarrow \beta\}$
   - *Se* $|\beta| \geqslant 2$ *e* $\beta \in \Sigma^*$, *ou seja, se* $\beta \in \Sigma\Sigma\Sigma^*$, *então:*
     - $N_2 \leftarrow N_2 \cup \{Y_1, Y_2, ..., Y_{n-1}\}$
     - $P_2 \leftarrow P_2 \cup \{\alpha \rightarrow \sigma_1 Y_1, Y_1 \rightarrow \sigma_2 Y_2 \ , ..., Y_{n-2} \rightarrow \sigma_{n-1}\ Y_{n-1}, Y_{n-1} \rightarrow \sigma_n\}$
   - *Se* $|\beta| \geqslant 3$ *e* $\beta \in \Sigma^* N$, *ou seja, se* $\beta \in \Sigma\Sigma\Sigma^* N$, *então:*
     - $N_2 \leftarrow N_2 \cup \{X_1, X_2, ..., X_{n-1}\}$
     - $P_2 \leftarrow P_2 \cup \{\alpha \rightarrow \sigma_1 X_1, X_1 \rightarrow \sigma_2 X_2, ..., X_{n-2} \rightarrow \sigma_{n-1} X_{n-1}, X_{n-1} \rightarrow \sigma_n K\}$

Equivalência unitárias/não-unitárias

Algoritmo semelhante pode ser facilmente desenvolvido para o caso das gramáticas lineares à esquerda, de forma a obter uma gramática equivalente cujas regras sejam do tipo:

$$\alpha \rightarrow \beta, \alpha \in N, \beta \in (N \cup \{\varepsilon\})(\Sigma \cup \{\varepsilon\})$$

também conhecida como **gramática linear unitária à esquerda**.

Equivalência unitárias/não-unitárias

Exemplo

Exemplo 1.1
Considere-se a gramática G_1:

$$\begin{aligned} S_1 &\rightarrow abcdP \\ P &\rightarrow efP \\ P &\rightarrow Q \\ Q &\rightarrow g \end{aligned}$$

Equivalência unitárias/não-unitárias

Exemplo

A aplicação do Algoritmo 1.1 resulta na gramática G_2:

$$
\begin{aligned}
S_2 &\rightarrow aP_1 \\
P_1 &\rightarrow bP_2 \\
P_2 &\rightarrow cP_3 \\
P_3 &\rightarrow dP \\
P &\rightarrow eP_4 \\
P_4 &\rightarrow fP \\
P &\rightarrow Q \\
Q &\rightarrow g
\end{aligned}
$$

Equivalência unitárias/não-unitárias

Exemplo

A título de ilustração, considerem-se as derivações da sentença $abcdefg$, respectivamente em G_1 e G_2:

- $S_1 \overset{G_1}{\Rightarrow} abcdP \overset{G_1}{\Rightarrow} abcdefP \overset{G_1}{\Rightarrow} abcdefQ \overset{G_1}{\Rightarrow} abcdefg$
- $S_2 \overset{G_2}{\Rightarrow} aP_1 \overset{G_2}{\Rightarrow} abP_2 \overset{G_2}{\Rightarrow} abcP_3 \overset{G_2}{\Rightarrow} abcdP \overset{G_2}{\Rightarrow} abcdeP_4 \overset{G_2}{\Rightarrow} abcdefP \overset{G_2}{\Rightarrow} abcdefQ \overset{G_2}{\Rightarrow} abcdefg$

Gramática e linguagem linear

Uma **gramática linear** (não necessariamente à esquerda ou à direita) é uma gramática que possui no máximo um único símbolo não-terminal do lado direito das suas regras. Assim, gramáticas lineares à esquerda ou à direita são casos particulares de gramáticas lineares: toda gramática linear à esquerda ou à direita é também uma gramática linear porém o inverso nem sempre é verdadeiro. A linguagem gerada por uma gramática linear é chamada de **linguagem linear**, e esta não é necessariamente uma linguagem regular.

Gramática linear e linguagem não-linear

Exemplo 1.2
A gramática linear apresentada a seguir baixo gera uma linguagem linear que é livre de contexto e não-regular, conforme será visto mais adiante:

$$\begin{aligned} S &\rightarrow aSa \\ S &\rightarrow b \end{aligned}$$

Equivalência direita/esquerda

Teorema

Teorema 1.2 "Se G_1 é uma gramática linear à direita unitária, então existe uma gramática linear à esquerda unitária G_2 tal que $L(G_1) = L(G_2)$, e vice-versa."
O Algoritmo 1.2 mostra como se pode mapear as regras de uma GLD G_1 nas regras de uma GLE G_2, de tal forma que $L(G_2) = L(G_1)$. O mesmo algoritmo pode ser usado, no sentido inverso, para mapear as regras de uma GLE G_2 nas regras de uma GLD G_1, de tal forma que $L(G_1) = L(G_2)$.

Equivalência direita/esquerda

Algoritmo

Algoritmo 1.2 *"Obtenção de uma GLE que gera a mesma linguagem que uma GLD"*

- *Entrada: uma GLD unitária* $G_1 = (V, \Sigma, P, S)$*; é importante que* S *não compareça do lado direito de nenhuma regra do conjunto* P *(veja observação v abaixo);*[2]
- *Saída: uma GLE unitária* $G_2 = (V, \Sigma, P', S)$ *tal que* $L(G_2) = L(G_1)$*;*

[2] Se este for o caso, pode-se criar um novo símbolo não-terminal Z, para ser usado como nova raiz da gramática, e adicionar a regra $Z \to S$ à mesma.

Equivalência direita/esquerda

Algoritmo

- *Método: o conjunto de regras de G_2 (P') é obtido a partir do conjunto de regras de G_1 (P), conforme o mapeamento da Tabela 1.*

Tabela 1: Mapeamento de GLD em GLE para L e vice-versa

GLD	*GLE*	*Observação*
$S \to \mu$	$S \to \mu$	$i)$
$S \to \mu A$	$A \to \mu$	$ii)$
$A \to \mu$	$S \to A\mu$	$iii)$
$A \to \mu B$	$B \to A\mu$	$iv)$

Equivalência direita/esquerda

Exemplo

Exemplo 1.3

Considere a gramática linear à direita G_1 definida a seguir:

$$
\begin{aligned}
S &\rightarrow aX && (1)\\
X &\rightarrow bX && (2)\\
X &\rightarrow bY && (3)\\
Y &\rightarrow cY && (4)\\
Y &\rightarrow \varepsilon && (5)
\end{aligned}
$$

A gramática G_1 gera uma linguagem sobre o alfabeto $\{a, b, c\}$ tal que as suas sentenças:

1. Iniciam com um único a;
2. Continuam com um ou mais b;
3. Terminam com zero ou mais c.

Equivalência direita/esquerda

Exemplo

Pelo Algoritmo 1.2, obtemos a gramática linear à esquerda G_2:

$$X \rightarrow a \tag{6}$$
$$X \rightarrow Xb \tag{7}$$
$$Y \rightarrow Xb \tag{8}$$
$$Y \rightarrow Yc \tag{9}$$
$$S \rightarrow Y \tag{10}$$

Equivalência direita/esquerda

Exemplo

Exemplo 1.4
Considere a gramática linear à esquerda G_1 definida a seguir:

$$S \rightarrow Y \tag{11}$$
$$Y \rightarrow Yc \tag{12}$$
$$Y \rightarrow X \tag{13}$$
$$X \rightarrow Xb \tag{14}$$
$$X \rightarrow ab \tag{15}$$

Equivalência direita/esquerda

Exemplo

Observe que a linguagem deste exemplo ($L(G_1)$) é a mesma do Exemplo 1.3. Pelo Algoritmo 1.2, obtemos a gramática linear à direita G_2:

$$Y \rightarrow \varepsilon \tag{16}$$
$$Y \rightarrow cY \tag{17}$$
$$X \rightarrow Y \tag{18}$$
$$X \rightarrow bX \tag{19}$$
$$S \rightarrow abX \tag{20}$$

Note que a regra é 3.16 obtida a partir da 3.11, a regra 3.17 é obtida a partir da 3.12 e assim por diante. Observe ainda que $L(G_2) = L(G_1)$.

Linguagem reversa GLD/GLE

Teorema

Teorema 1.3 "Se G' é uma gramática linear à direita, então existe uma gramática linear à esquerda G'' tal que $L(G'') = L^R(G')$."
A obtenção de uma GLE G'' a partir de uma GLD G' pode ser feita de acordo com o Algoritmo 1.3. Para isto, é suficiente reverter o lado direito das regras de G'.

Linguagem reversa GLD/GLE

Algoritmo

Algoritmo 1.3 *"Obtenção de uma GLE que gera L^R a partir de uma GLD que gera L."*

- *Entrada: uma gramática linear à direita* $G' = (V, \Sigma, P', S)$;
- *Saída: uma gramática linear à esquerda* $G'' = (V, \Sigma, P'', S)$, *tal que* $L(G'') = L^R(G')$;

Linguagem reversa GLD/GLE

Algoritmo

- *Método: o conjunto de regras de G'' (P'') é obtido a partir do conjunto de regras de G' (P'), conforme o mapeamento da Tabela 2.*

Tabela 2: Mapeamento de GLD em GLE para L^R

GLD	*GLE*
$S \to \mu$	$S \to \mu^R$
$S \to \mu A$	$S \to A\mu^R$
$A \to \mu$	$A \to \mu^R$
$A \to \mu B$	$A \to B\mu^R$

Com G'' assim construída, é possível demonstrar que $L(G'') = L^R(G')$. Note que G'' é, por construção, linear à esquerda. De maneira similar, é possível obter um algoritmo que constrói uma GLD G'' a partir de uma GLE G' tal que $L(G'') = L^R(G')$.

Linguagem reversa GLD/GLD

Teorema

Teorema 1.4 "Se G' é uma gramática linear à direita, então existe uma gramática linear à direita G'' tal que $L(G'') = L^R(G')$."
A obtenção de uma GLD G'' a partir de uma GLD G' é uma decorrência dos resultados anteriores, como mostra o Algoritmo 1.4.
Na prática, o Algoritmo 1.4 (e a Tabela 3) incorpora as seguintes ações que também podem ser executadas de forma independente:

1. Obtenção de uma GLE G_E a partir da GLD G' conforme o Algoritmo 1.2 (neste caso, $L(G_E) = L(G')$);
2. Obtenção de uma GLD G'' a partir da GLE G_E conforme o Algoritmo 1.3 (neste caso, $L(G'') = L^R(G_E) = L^R(G')$).

O mapeamento da GLD que gera L em outra GLD que gera L^R pode, portanto, ser feito de forma direta, num único passo, ou então em dois passos com a obtenção de uma GLE intermediária.

Linguagem reversa GLD/GLD

Algoritmo

Algoritmo 1.4 *"Obtenção de uma GLD que gera L^R a partir de uma GLD que gera L."*

- *Entrada: uma gramática linear à direita $G' = (V, \Sigma, P', S)$; é importante que S não compareça do lado direito das regras de G';*
- *Saída: uma gramática linear à direita $G'' = (V, \Sigma, P'', S)$, tal que $L(G'') = L^R(G')$;*

Linguagem reversa GLD/GLD

Algoritmo

- *Método: o conjunto de regras de G'' (P'') é obtido a partir do conjunto de regras de G' (P'), conforme o mapeamento da Tabela 3.*

Tabela 3: Mapeamento de GLD em GLD para L^R

GLD	GLD
$S \to \mu$	$S \to \mu^R$
$S \to \mu A$	$A \to \mu^R$
$A \to \mu$	$S \to \mu^R A$
$A \to \mu B$	$B \to \mu^R A$

Exemplo

Exemplo 1.5

Considere a gramática linear à direita G' definida a seguir:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow bS \\ S &\rightarrow P \\ P &\rightarrow cQ \\ Q &\rightarrow cR \\ R &\rightarrow dR \\ R &\rightarrow d \end{aligned}$$

$L(G')$ corresponde ao conjunto das cadeias w sobre $\{a,b,c,d\}$ tais que:

1. w começa com zero ou mais símbolos a ou b;
2. w continua com exatamente dois símbolos c;
3. w termina com um ou mais símbolos d.

Exemplo

Continuação

Inicialmente, é necessário incluir a regra $Z \rightarrow S$, onde Z é a nova raiz de G', a fim de eliminar a raiz do lado direito das regras:

$$
\begin{aligned}
Z &\rightarrow S \\
S &\rightarrow aS \\
S &\rightarrow bS \\
S &\rightarrow P \\
P &\rightarrow cQ \\
Q &\rightarrow cR \\
R &\rightarrow dR \\
R &\rightarrow d
\end{aligned}
$$

Exemplo

Continuação

Uma gramática linear à esquerda G_E, tal que $L(G_E) = L(G')$, pode ser obtida pela aplicação do Algoritmo 1.2:

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow Sa \\
S &\rightarrow Sb \\
P &\rightarrow S \\
Q &\rightarrow Pc \\
R &\rightarrow Qc \\
R &\rightarrow Rd \\
Z &\rightarrow Rd
\end{aligned}
$$

Exemplo

Continuação

A título de ilustração, considerem-se as derivações da sentença $abaccdd$, respectivamente em G' e G_E:

- $S \overset{G'}{\Rightarrow} aS \overset{G'}{\Rightarrow} abS \overset{G'}{\Rightarrow} abaS \overset{G'}{\Rightarrow} abaP \overset{G'}{\Rightarrow} abacQ \overset{G'}{\Rightarrow} abaccR \overset{G'}{\Rightarrow} abaccdR \overset{G'}{\Rightarrow} abaccdd$
- $Z \overset{G_E}{\Rightarrow} Rd \overset{G_E}{\Rightarrow} Rdd \overset{G_E}{\Rightarrow} Qcdd \overset{G_E}{\Rightarrow} Pccdd \overset{G_E}{\Rightarrow} Sccdd \overset{G_E}{\Rightarrow} Saccdd \overset{G_E}{\Rightarrow} Sbaccd \overset{G_E}{\Rightarrow} Sabaccdd \overset{G_E}{\Rightarrow} abaccdd$

Exemplo

Continuação

Uma gramática linear à direita G'', tal que $L(G'') = L^R(G_E)$, pode ser obtida pela aplicação do Algoritmo 1.3:

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow aS \\
S &\rightarrow bS \\
P &\rightarrow S \\
Q &\rightarrow cP \\
R &\rightarrow cQ \\
R &\rightarrow dR \\
Z &\rightarrow dR
\end{aligned}
$$

Exemplo

Continuação

Logo, $L(G'') = L^R(G')$ é tal que:

1. w começa com um ou mais símbolos d.
2. w continua com exatamente dois símbolos c;
3. w termina com zero ou mais símbolos a ou b;

Note que G' e G'' são ambas lineares à direita e, além disso, $L(G'') = L^R(G')$.

Exemplo

Exemplo 1.6
Considere a gramática linear à direita G' definida a seguir:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow X \\ X &\rightarrow bX \\ X &\rightarrow Y \\ Y &\rightarrow cY \\ Y &\rightarrow \varepsilon \end{aligned}$$

$L(G')$ corresponde ao conjunto das cadeias w sobre $\{a,b,c\}$ tais que:

1. w começa com zero ou mais símbolos a;
2. w continua com zero ou mais símbolos b;
3. w termina com zero ou mais símbolos c.

Exemplo

Continuação

Inicialmente, é necessário incluir a regra $Z \rightarrow S$, onde Z é a nova raiz de G', a fim de eliminar a raiz do lado direito das regras:

$$
\begin{aligned}
Z &\rightarrow S \\
S &\rightarrow aS \\
S &\rightarrow X \\
X &\rightarrow bX \\
X &\rightarrow Y \\
Y &\rightarrow cY \\
Y &\rightarrow \varepsilon
\end{aligned}
$$

Exemplo

Continuação

Uma gramática linear à esquerda G_E, tal que $L(G_E) = L(G')$, pode ser obtida pela aplicação do Algoritmo 1.2:

$$
\begin{array}{lcl}
S & \rightarrow & \varepsilon \\
S & \rightarrow & Sa \\
X & \rightarrow & S \\
X & \rightarrow & Xb \\
Y & \rightarrow & X \\
Y & \rightarrow & Yc \\
Z & \rightarrow & Y
\end{array}
$$

Exemplo

Continuação

Uma gramática linear à direita G'', tal que $L(G'') = L^R(G_E)$, pode ser obtida pela aplicação do Algoritmo 1.3:

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow aS \\
X &\rightarrow S \\
X &\rightarrow bX \\
Y &\rightarrow X \\
Y &\rightarrow cY \\
Z &\rightarrow Y
\end{aligned}
$$

Exemplo

Continuação

Logo, $L(G'') = L^R(G')$ é tal que:

1. w começa com zero ou mais símbolos c.
2. w continua com zero ou mais símbolos b;
3. w termina com zero ou mais símbolos a;

Note que G' e G'' são ambas lineares à direita e, além disso, $L(G'') = L^R(G')$. Note, ainda, que G'' pode ser obtida diretamente a partir de G' (com a nova raiz Z) pela aplicação do Algoritmo 1.4.

Exercício

1. Obter gramáticas lineares à direita que geram as linguagens cujas sentenças estão descritas a seguir;
2. Repita, obtendo gramáticas lineares à esquerda.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Exercício

1. Obter gramáticas lineares (à direita ou à esquerda) unitárias que geram as linguagens cujas sentenças estão descritas a seguir.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Linguagens regulares

Conjuntos e expressões regulares são notações alternativas utilizadas para representar a classe de linguagens mais simples que se conhece: a classe das linguagens regulares, a mais restrita dentro da Hierarquia de Chomsky.

Conjuntos regulares

Definição

Conjuntos regulares sobre um alfabeto finito Σ são linguagens definidas recursivamente da seguinte forma:

1. $\emptyset$ é um conjunto regular sobre Σ;
2. $\{\varepsilon\}$ é um conjunto regular sobre Σ;
3. $\{\sigma\}, \forall \sigma \in \Sigma$, é um conjunto regular sobre Σ.

Conjuntos regulares

Definição

Se X e Y são conjuntos regulares sobre Σ, então também são conjuntos regulares sobre Σ:

4. (X);
5. $X \cup Y$;
6. $X \cdot Y$, também denotado XY;
7. X^*.

Diz-se que um determinado subconjunto de Σ^* é um conjunto regular se ele puder ser formulado através do uso combinado dessas regras apenas.

Conjuntos regulares

Exemplo

Exemplo 2.1
Seja $L = \{0^m 1^n \mid m \geqslant 0, n \geqslant 0\}$ sobre $\Sigma = \{0,1\}$. A linguagem L é formada por sentenças em que a concatenação de um número arbitrário de símbolos "0" (incluindo nenhum) se concatena com a concatenação de um número também arbitrário de símbolos "1" (incluindo nenhum):

$$L = \{\varepsilon, 0, 1, 00, 01, 11, \ldots\}$$

Considerem-se as linguagens sobre Σ, abaixo discriminadas:

$$\begin{aligned} L_1 &= \{0\} \\ L_2 &= \{1\} \\ L_3 &= \{0^i \mid i \geqslant 0\} \\ L_4 &= \{1^i \mid i \geqslant 0\} \\ L_5 &= \{0^p 1^q \mid p \geqslant 0, q \geqslant 0\} \end{aligned}$$

Conjuntos regulares

Exemplo

Os conjuntos L_1 e L_2 são conjuntos regulares sobre Σ, por definição. L_3 e L_4 são obtidos a partir de L_1 e L_2, respectivamente, pela aplicação da operação fechamento reflexivo e transitivo, ou seja, $L_3 = L_1^*$ e $L_4 = L_2^*$. Por sua vez, o conjunto $L_5 = L$ pode ser expresso pela concatenação dos conjuntos L_3 e L_4, isto é, $L_5 = L_3 L_4$. Dessa maneira, demonstra-se que $L = \{0^m 1^n \mid m \geqslant 0, n \geqslant 0\}$ é um conjunto regular sobre $\{0,1\}$. Na notação dos conjuntos regulares, L pode ser denotado por $\{0\}^*\{1\}^*$.

Conjuntos regulares

Exemplo

Exemplo 2.2
A linguagem $\mathbb{N}$ formada pelos números naturais decimais é um conjunto regular sobre o alfabeto dos algarismos arábicos e pode ser representada através do seguinte conjunto regular:

$$\{0,1,2,3,4,5,6,7,8,9\}\{0,1,2,3,4,5,6,7,8,9\}^*$$

Se $D = \{0,1,2,3,4,5,6,7,8,9\}$, então $\mathbb{N} = DD^*$.
O conjunto R dos números reais decimais sem sinal é um conjunto regular sobre $D \cup \{.\}$, e pode ser representado por:

$$DD^*\{.\}D^* \cup D^*\{.\}DD^*$$

Conjuntos regulares

Exemplo

Observe-se que a definição acima inclui números iniciando ou terminando com o caractere "." (como, por exemplo, .315 ou 47.), porém exclui da linguagem a cadeia ".". O conjunto P dos números em ponto flutuante com expoente (denotado por "E") e sinal opcional ("+" ou "−") pode ser representado por:

$$\{+,-,\varepsilon\}(DD^*\{.\}D^* \cup D^*\{.\}DD^*)\{E\}\{+,-,\varepsilon\}DD^*$$

Assim, por exemplo, $27 \in \mathbb{N}, 915.4 \in R$ e $-211.56E+3 \in P$. Deve-se notar que $\mathbb{N} \subset R \subset P, P \neq R$ e $R \neq \mathbb{N}$.

Exercício

1. Obter conjuntos regulares que representam as linguagens cujas sentenças estão descritas a seguir.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Fechamentos

A definição de conjuntos regulares envolve a aplicação de três operações já estudadas para os conjuntos: união, concatenação e fechamento reflexivo e transitivo. No caso do fechamento, no entanto, cabem algumas observações adicionais válidas para o caso em que o seu operando seja não apenas um alfabeto, conforme anteriormente mencionado, mas eventualmente uma linguagem, como ocorre na definição acima.

Seja L uma linguagem qualquer, e considerem-se as novas linguagens L^* e L^+ obtidas pela aplicação, respectivamente, das operações de fechamento reflexivo e transitivo e do fechamento transitivo sobre L. Neste caso, deve-se observar que, diferentemente do que ocorre com os alfabetos, as seguintes identidades são verdadeiras:

- $L^+ = L^*$ se $\varepsilon \in L$
- $L^+ = L^* - \{\varepsilon\}$ se $\varepsilon \notin L$

Expressões regulares

Definição

Como alternativa para a representação dos conjuntos regulares, visando obter maior concisão e facilidade de manipulação, Kleene desenvolveu, na década 1950, a notação das **expressões regulares**. Da mesma forma como ocorre para os conjuntos regulares, as expressões regulares sobre um alfabeto Σ podem também ser definidas recursivamente como segue:

1. $\emptyset$ é uma expressão regular e denota o conjunto regular $\emptyset$;
2. ε é uma expressão regular e denota o conjunto regular $\{\varepsilon\}$;
3. Cada $\sigma, \sigma \in \Sigma$, é uma expressão regular e denota o conjunto regular $\{\sigma\}, \sigma \in \Sigma$;

Expressões regulares

Definição

Se x e y são expressões regulares sobre Σ que denotam, respectivamente, os conjuntos regulares X e Y, então:

4. (x)
5. $x \mid y$ ou $x + y$
6. $x \cdot y$ ou xy
7. x^*

também são expressões regulares e denotam, respectivamente, os conjuntos regulares $X, X \cup Y, XY$ e X^*.

Conjuntos regulares e expressões regulares

Note-se a eliminação, nas expressões regulares, do uso dos símbolos “{” e “}”, bem como a substituição do símbolo de união (“$\cup$”) por um símbolo “$+$” ou “$|$” (a critério de cada autor). Visando tornar ainda mais cômoda a utilização das expressões regulares, admite-se a eliminação dos pares de parênteses envolvendo sub-expressões que contenham seqüências exclusivas de operadores, de união ou de concatenação, uma vez que se trata de operações associativas. São designadas precedências distintas para as três operações, reduzindo ainda mais a necessidade de emprego de parênteses nas expressões regulares.

Precedências

Tabela 4: Precedência dos operadores nas expressões regulares

Precedência	**Operador**	**Representação**
Mais alta	Fechamento	x^*
Intermediária	Concatenação	$x \cdot y$ ou xy
Mais baixa	União	$x \mid y$ ou $x + y$

Os parênteses são empregados para modificar localmente a precedência ou a associatividade predefinida dos operadores, assim como ocorre nas expressões aritméticas tradicionais da matemática.

Exemplo

Exemplo 2.3
A expressão regular $(ab \mid c^*) = ((ab) \mid c^*) = ((ab) \mid (c^*))$ representa o conjunto $\{ab, \varepsilon, c, cc, ccc...\}$. A expressão regular $a(b \mid c)^*$ representa o conjunto $\{a, ab, ac, abc, abb, acc, ...\}$. Finalmente, $(ab \mid c)^*$ representa o conjunto $\{\varepsilon, ab, c, abc, cab, abab, cc, ...\}$.

Abreviação

Uma abreviação muito comum consiste na substituição da expressão regular xx^* por x^+, denotando com isso o conjunto regular correspondente ao fechamento transitivo de X (que é composto por todas as cadeias de comprimento maior ou igual a 1 que podem ser construídas sobre o conjunto X).

Exemplo

Exemplo 2.4
Considerem-se o alfabeto $\Sigma = \{a,b,c,d\}$ e os dois subconjuntos $A = \{a\}, B = \{b,c\}$. A seguir são apresentadas diferentes linguagens sobre Σ, definidas através da notação dos conjuntos e das expresões regulares:

- Sentenças que possuem no mínimo um símbolo a:
  $\Sigma^* A \Sigma^*$ ou $(a \mid b \mid c \mid d)^* a (a \mid b \mid c \mid d)^*$
- Sentenças que possuem exatamente dois símbolos a:
  $(\Sigma - A)^* A (\Sigma - A)^* A (\Sigma - A)^*$ ou $(b \mid c \mid d)^* a (b \mid c \mid d)^* a (b \mid c \mid d)^*$
- Sentenças que possuem um número par de símbolos a:
  $((\Sigma - A)^* A (\Sigma - A)^* A (\Sigma - A)^*)^*$ ou $((b \mid c \mid d)^* a (b \mid c \mid d)^* a (b \mid c \mid d)^*)^*$

Exemplo

- Sentenças que são iniciadas com o símbolo a e terminam com o símbolo b ou c:
  $A\Sigma^*B$ ou $a(a \mid b \mid c \mid d)^*(b \mid c)$
- Sentenças contendo apenas os símbolos a, b, c, com no mínimo um símbolo:
  $(A \cup B)^+$ ou $(a \mid b \mid c)^+$
- Sentenças formadas por símbolos do alfabeto $\{a, b, c, d\}$ contendo uma (e somente uma) subcadeia constituída por um símbolo do conjunto A e dois (e somente dois) do conjunto B, nesta ordem:
  $((\Sigma - A) - B)^*ABB((\Sigma - A) - B)^*$ ou $d^*a(b \mid c)(b \mid c)d^*$

Exemplo

Exemplo 2.5
Utilizando-se a notação das expressões regulares, a linguagem $L = \{0^m 1^n \mid m \geqslant 0, n \geqslant 0\}$ pode ser reescrita como $((0)^*(1)^*)$, ou, simplesmente, 0^*1^*. Para $m \geqslant 0$ e $n \geqslant 1$, a expressão correspondente seria 0^*11^*. Note-se que $0^*11^* = 0^*1^*1 = 0^*1^+$.

Exemplo

Exemplo 2.6
As expressões regulares $a \mid aaaa^*$ e $(aa)^*a(aaa)^*$, apesar de diferentes, representam a mesma linguagem: o conjunto das cadeias formadas pelo símbolo a cujo comprimento seja diferente de 0 e diferente de 2.

Exercício

1. Obter expressões regulares que representam as linguagens cujas sentenças estão descritas a seguir.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Leis algébricas

As principais leis algébricas das expressões regulares são apresentadas a seguir. Sejam α, β e γ três expressões regulares quaisquer. Então:

- Associatividade:
  - da união: $(\alpha \mid \beta) \mid \gamma = \alpha \mid (\beta \mid \gamma)$
  - da concatenação: $(\alpha\beta)\gamma = \alpha(\beta\gamma)$
- Comutatividade:
  - da união: $\alpha \mid \beta = \beta \mid \alpha$
  - da concatenação: Não se aplica.
- Elemento neutro:
  - da união: $\alpha \mid \emptyset = \emptyset \mid \alpha = \alpha$
  - da concatenação: $\alpha\varepsilon = \varepsilon\alpha = \alpha$
- Distributividade da concatenação sobre a união:
  - à esquerda: $\alpha(\beta \mid \gamma) = \alpha\beta \mid \alpha\gamma$
  - à direita: $(\beta \mid \gamma)\alpha = \beta\alpha \mid \gamma\alpha$

Relações de identidade

Algumas relações de identidade válidas para as expressões regulares, as quais podem ser demonstradas sem dificuldade, são relacionadas a seguir. Sejam x, y, z três expressões regulares quaisquer. Então...

Relações de identidade

- $x \mid y = y \mid x$
- $\emptyset^* = \varepsilon$
- $x \mid (y \mid z) = (x \mid y) \mid z$
- $x(yz) = (xy)z$
- $x(y \mid z) = xy \mid xz$
- $(x \mid y)z = xz \mid yz$
- $x\varepsilon = \varepsilon x = x$
- $x\emptyset = \emptyset x = \emptyset$
- $\varepsilon\emptyset = \emptyset\varepsilon = \emptyset$
- $x^* = x \mid x^*$
- $(x^*)^* = x^*$
- $x \mid x = x$
- $x \mid \emptyset = x$
- $(xy)^*x = x(yx)^*$

Concisão e uso

Por se tratar de uma notação concisa, que dispensa o uso da notação dos conjuntos e o emprego de símbolos não-terminais para a definição de linguagens, mas que, ao mesmo tempo, permite a plena representação dos conjuntos regulares, as expressões regulares são bastante utilizadas em áreas que abrangem desde a especificação de linguagens de programação e de comandos, entre outras, até a entrada de dados em editores de texto, programas de busca, análise de padrões etc.

Autômatos finitos e linguagens regulares

Da mesma forma como ocorre com as expressões regulares e com as gramáticas lineares à direita, os **autômatos finitos** também possibilitam a formalização das linguagens regulares, ou seja, das linguagens do tipo 3. No entanto, diferentemente daquelas notações, que constituem dispositivos de geração de sentenças, os autômatos finitos são dispositivos de aceitação de sentenças e constituem um caso particular do modelo geral de reconhecedores apresentado no Capítulo 2.

Particularidades

Os autômatos finitos correspondem à instância mais simples do modelo geral de reconhecedores apresentado anteriormente. As suas principais particularidades em relação ao modelo geral são:

1. Inexistência de memória auxiliar;
2. Utilização do cursor da fita de entrada apenas para leitura de símbolos, não havendo operações de escrita sobre a fita;
3. Movimentação do cursor de leitura em apenas um sentido, da esquerda para a direita;
4. A fita de entrada possui comprimento limitado, suficiente apenas para acomodar a cadeia a ser analisada.

Definição

Algebricamente, um autômato finito determinístico M pode ser definido como uma quíntupla:

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

- Q é um conjunto finito de estados;
- Σ é um alfabeto (finito e não-vazio) de entrada;
- δ é uma função de transição, $\delta : Q \times \Sigma \to Q$;
- q_0 é o estado inicial, $q_0 \in Q$;
- F é um conjunto de estados finais, $F \subseteq Q$.

Controle finito

A máquina de estados de um autômato finito, também denominada **controle finito**, é definida pelo conjunto de estados Q e pela função de transição δ, que associa pares ordenados do tipo *(estado corrente, entrada corrente)* com um novo estado a ser assumido pelo autômato quando da aplicação da transição.

Função de transição

A função de transição δ pode ser, no caso dos autômatos finitos determinísticos, uma função total, ou seja, uma função que é definida para todos os elementos de $Q \times \Sigma$, ou ainda uma função parcial. Se total, isso implica a especificação de transições com cada um dos possíveis símbolos de entrada $\sigma \in \Sigma$ para cada um dos possíveis estados $q \in Q$ do autômato finito. Assim, se $|\Sigma| = m$ e $|Q| = n$, então o autômato finito determinístico possuirá, exatamente, $m * n$ transições distintas.

Notação

As transições de um autômato finito podem ser denotadas através de expressões do tipo $(p, \sigma) \rightarrow q,$ com $p, q \in Q, \sigma \in \Sigma$. Alternativamente, pode-se explicitar a função δ, representando uma transição na forma $\delta(p, \sigma) = q$.

Autômato determinístico

A utilização do termo “determinístico” para designar esse tipo de autômato finito decorre do fato de que, enquanto houver símbolos na fita de entrada, será sempre possível determinar o estado seguinte a ser assumido pelo autômato, o qual será único em todas as situações.

Extensão da função de transição

Em certos casos, especialmente na demonstração de alguns teoremas, torna-se conveniente estender o domínio da função δ para Σ^*, em vez de apenas Σ, conforme indicado abaixo:

- $\delta(q, \varepsilon) = q$;
- $\delta(q, \sigma x) = \delta(\delta(q, \sigma), x), x \in \Sigma^*, \sigma \in \Sigma$.

Ao longo deste texto, a definição considerada para a função δ deverá variar conforme o contexto em que estiver sendo empregada.

Configuração

Conceito

A **configuração** de um autômato finito é definida pelo seu estado corrente e pela parte da cadeia de entrada ainda não analisada (incluindo o símbolo apontado pelo cursor). A **configuração inicial** de um autômato finito é aquela em que o estado corrente é q_0 (estado inicial) e o cursor de leitura se encontra posicionado sobre o símbolo mais à esquerda da cadeia de entrada. Uma **configuração final** é aquela em que o cursor aponta para a posição imediatamente além do último símbolo da cadeia (indicando com isso já ter ocorrido a leitura do último símbolo da cadeia de entrada), e o estado corrente pertence ao conjunto F de estados finais, especificado para o autômato. Note que ambas as condições devem ser simultaneamente verificadas para permitir a caracterização de uma configuração como sendo, respectivamente, inicial ou final.

Aceitação e rejeição

Quando ocorre o esgotamento da cadeia de entrada, deve-se analisar o tipo do estado corrente do autômato. Se for um estado final, diz-se que o autômato **reconheceu**, ou **aceitou**, a cadeia de entrada; se for um estado não-final, diz-se que a cadeia de entrada foi **rejeitada** pelo autômato — logo, a cadeia analisada não pertence à linguagem por ele definida.

Função de transição total

Quando definidos através de funções de transição totais, os correspondentes autômatos finitos determinísticos sempre percorrem integralmente toda e qualquer cadeia de entrada (sobre Σ) que lhes forem apresentadas para análise. Nesses casos, portanto, a configuração final definida para um autômato finito é sempre satisfeita no que se refere ao esgotamento da cadeia de entrada, restando apenas a análise do tipo do estado atingido em tal configuração (final ou não-final) para se determinar, respectivamente, a aceitação ou a rejeição da cadeia de entrada.

Configuração

Definição

Tais conceitos podem ser formalizados denotando-se a configuração de um autômato finito como um par $(q, y) \in Q \times \Sigma^*$, em que q representa o estado corrente e y a parte da cadeia de entrada ainda não analisada. A configuração inicial, com vistas ao reconhecimento de uma cadeia x, é representada como (q_0, x), e a configuração final como $(q_i, \varepsilon), q_i \in F$.

Movimentação

A movimentação de um autômato de uma configuração para a configuração seguinte é denotada através do símbolo "$\vdash$", que representa uma relação entre configurações sucessivas do autômato:

$$\vdash: Q \times \Sigma^* \rightarrow Q \times \Sigma^*$$

Portanto, a movimentação do autômato de uma configuração para a seguinte é denotada por:

$$(q_i, \sigma\beta) \vdash (q_j, \beta), \text{ com } q_i, q_j \in Q, \sigma \in \Sigma, \beta \in \Sigma^*, \delta(q_i, \sigma) = q_j$$

Linguagem reconhecida

A linguagem L definida por um autômato finito M é o conjunto de todas as cadeias w sobre o alfabeto Σ que levam M da sua configuração inicial para alguma configuração final através da aplicação sucessiva de transições definidas pela função δ. Denota-se como:

$$L(M) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q_F, \varepsilon), q_F \in F\}$$

Diagramas de transição de estados

Diagramas de transição de estados são grafos orientados não-ordenados, rotulados nos vértices com os nomes dos estados e nos arcos com os símbolos do alfabeto de entrada do autômato finito. Trata-se de uma representação gráfica equivalente à notação algébrica, oferecendo porém uma melhor visualização do autômato. Nessa representação, círculos representam os estados, e arcos as transições. O estado inicial é identificado por um arco cuja extremidade inicial não é ligada a nenhum outro estado. Os estados finais são representados por círculos duplos concêntricos.

Exemplo

Exemplo 3.1
Seja M um autômato finito determinístico, com função de transição total, definido abaixo. Sua representação algébrica é $M = (Q, \Sigma, \delta, q_0, F)$, onde:

$$
\begin{aligned}
Q &= \{q_0, q_1, q_2, q_3\} \\
\Sigma &= \{0, 1, 2\} \\
\delta &= \{(q_0, 0) \to q_0, (q_0, 1) \to q_1, (q_0, 2) \to q_3, \\
&\quad (q_1, 0) \to q_3, (q_1, 1) \to q_1, (q_1, 2) \to q_2, \\
&\quad (q_2, 0) \to q_3, (q_2, 1) \to q_3, (q_2, 2) \to q_2, \\
&\quad (q_3, 0) \to q_3, (q_3, 1) \to q_3, (q_3, 2) \to q_3\} \\
F &= \{q_1, q_2\}
\end{aligned}
$$

Exemplo

A Figura 1 mostra o diagrama de transição de estados para esse autômato.

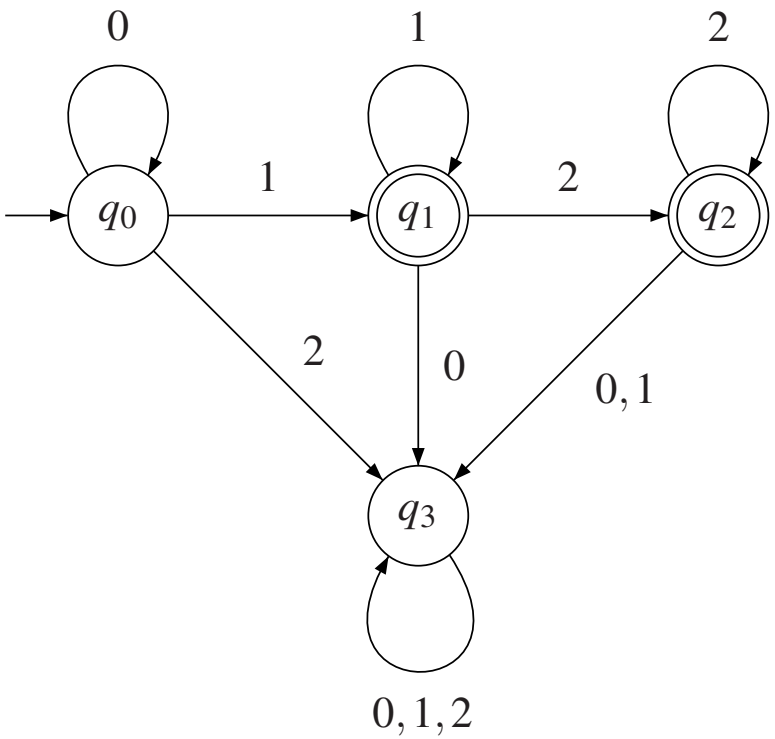


Figura 1: Autômato finito determinístico com função de transição total

Exemplo

Continuação

A linguagem aceita por esse autômato finito é formada pelo conjunto de sentenças x que o levam da configuração inicial (q_0, x) até a configuração final (q_1, ε) ou (q_2, ε). A inspeção cuidadosa desse autômato finito revela que as sentenças por ele aceitas contêm, nesta ordem, uma seqüência de símbolos "0" (incluindo nenhum), seguida de uma seqüência de símbolos "1" (no mínimo um) e, finalmente, de uma seqüência de símbolos "2" (incluindo nenhum). Na notação das expressões regulares, $L(M) = 0^*1^+2^*$.

Exemplo

Continuação

As seguintes identidades, por exemplo, são verdadeiras:

- $\delta(q_0, 00001) = q_1$
- $\delta(q_0, 0122) = q_2$
- $\delta(q_1, 12) = q_2$
- $\delta(q_2, 222) = q_2$

Exemplo

Continuação

Esquematicamente, a configuração inicial para o reconhecimento de uma cadeia de entrada, por exemplo, a cadeia 0011222, pode ser representada conforme a figura abaixo:

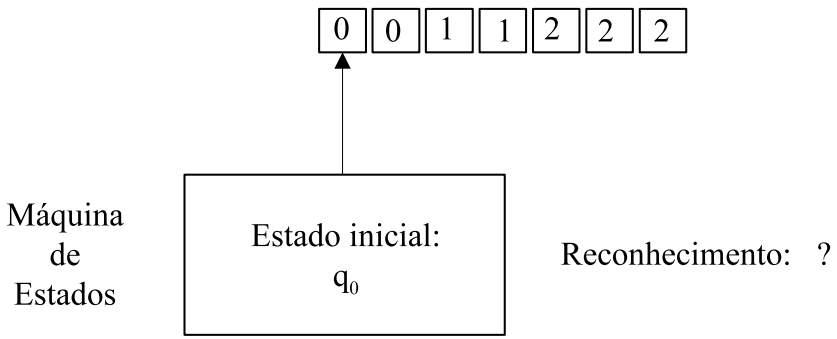


Figura 2: Configuração inicial

Exemplo

Continuação

A sucessão de movimentos efetuados pelo autômato finito com essa cadeia é apresentada a seguir:

- $(q_0, 0011222) \vdash (q_0, 011222) \vdash (q_0, 11222) \vdash (q_1, 1222) \vdash (q_1, 222) \vdash (q_2, 22) \vdash (q_2, 2) \vdash (q_2, \varepsilon)$

Portanto, $(q_0, 0011222) \vdash^* (q_2, \varepsilon)$, e $0011222 \in L(M)$. Esquematicamente, a configuração final do autômato após o reconhecimento da cadeia de entrada 0011222 pode ser representada conforme a figura a seguir:

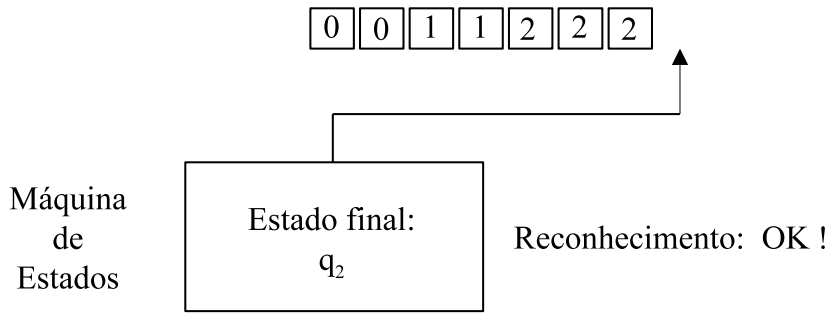


Figura 3: Configuração final

Exemplo

Continuação

Analisa-se a seguir o comportamento desse autômato em relação à cadeia de entrada 0022:

- $(q_0, 0022) \vdash (q_0, 022) \vdash (q_0, 22) \vdash (q_3, 2) \vdash (q_3, \varepsilon)$

Após a seqüência de movimentos acima, chega-se à configuração (q_3, ε), em que ocorre o esgotamento da cadeia de entrada. Como não se trata de uma configuração final, pois $q_3 \notin F$, conclui-se que a cadeia 0022 não pertence à linguagem aceita por M.

Exemplo

O autômato finito determinístico da Figura 4 possui função de transição parcial, uma vez que ela não é definida para os seguintes elementos de $Q \times \Sigma$: $(q_0, 2)$, $(q_1, 0)$, $(q_2, 1)$ e $(q_2, 0)$.

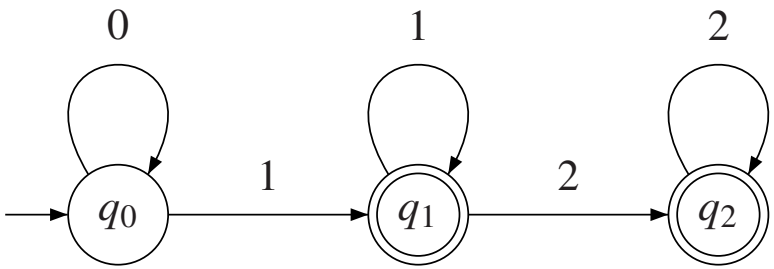


Figura 4: Autômato finito determinístico com função de transição parcial

Exemplo

Note-se que os autômatos das Figuras 1 e 4 definem a mesma linguagem. Os movimentos executados pelo autômato da Figura 4 com as cadeias 0011222 e 0022 são apresentados abaixo:

- $(q_0, 0011222) \vdash (q_0, 011222) \vdash (q_0, 11222) \vdash (q_1, 1222) \vdash (q_1, 222) \vdash (q_2, 22) \vdash (q_2, 2) \vdash (q_2, \varepsilon)$
- $(q_0, 0022) \vdash (q_0, 022) \vdash (q_0, 22)$

No primeiro caso, a cadeia 0011222 é completamente esgotada e o autômato pára em um estado final. Logo, 0011222 é aceita pelo autômato. No segundo caso, a cadeia de entrada é apenas parcialmente consumida, e o autômato pára no estado q_o, não-final. Logo, a cadeia 0022 é rejeitada.

Conceito de estado

Os estados de um autômato finito podem ser entendidos como memórias de informações passadas, referentes ao trecho da cadeia de entrada já analisada, as quais são relevantes, de alguma forma, para o reconhecimento final das cadeias que lhe são submetidas. O Exemplo 3.2 ilustra esse aspecto.

Exemplo

Exemplo 3.2
O autômato da Figura 5 aceita a linguagem formada por cadeias sobre o alfabeto $\{a, b, c\}$ em que as quantidades de símbolos a, b e c, consideradas isoladamente, são ímpares. São exemplos de sentenças dessa linguagem: abc, cba, $aaabc$, $ccabaac$, $abbbc$, $bcccbbabb$ etc.

Exemplo

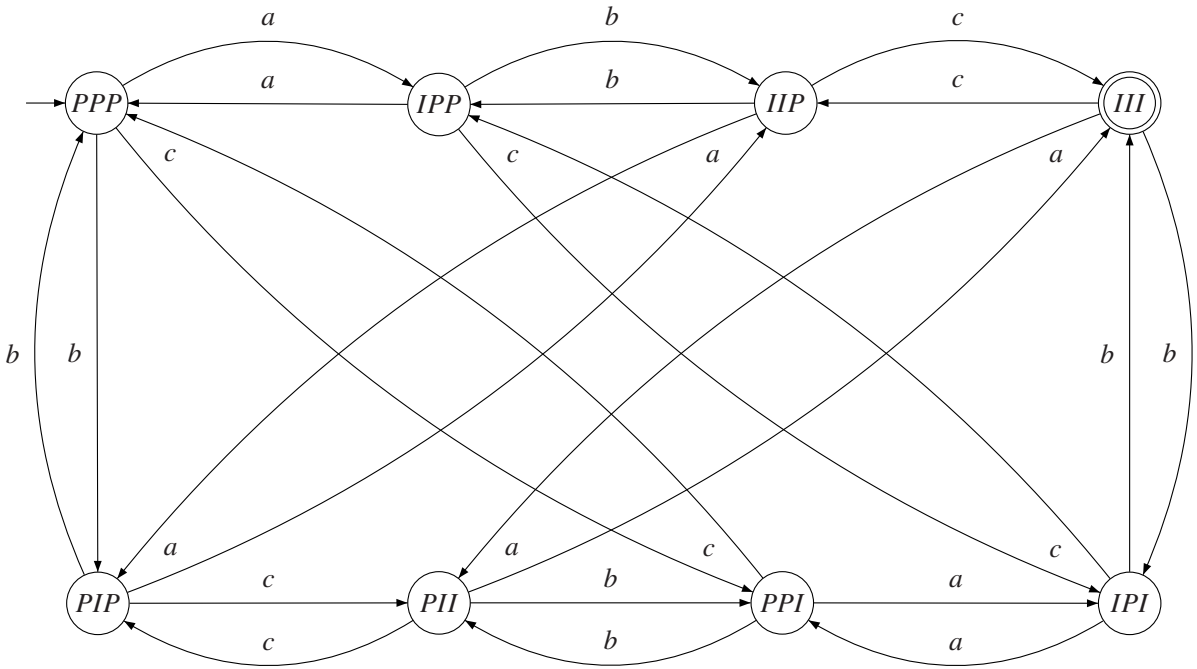


Figura 5: Autômato finito que aceita a linguagem $\{w \in \{a,b,c\}^* \mid$ as quantidades de a, b e c são ímpares$\}$

Exemplo

Continuação

De fato, uma análise cuidadosa dos estados *PPP*, *IPP*, *IIP*, *III*, *PIP*, *PII*, *PPI* e *IPI* revela que cada um deles mantém uma memória distinta sobre a quantidade de símbolos a, b e c consumidos até cada configuração. Por exemplo:

- O estado *PPP* memoriza o consumo prévio de uma quantidade par de símbolos a, par de símbolos b e par de símbolos c;
- O estado *IPP* memoriza o consumo prévio de uma quantidade ímpar de símbolos a, par de símbolos b e par de símbolos c;
- O estado *IIP* memoriza o consumo prévio de uma quantidade par de símbolos a, ímpar de símbolos b e par de símbolos c;
- O estado *III* memoriza o consumo prévio de uma quantidade ímpar de símbolos a, ímpar de símbolos b e ímpar de símbolos c.

Exercício

1. Obter autômatos finitos que reconhecem as linguagens cujas sentenças estão descritas a seguir.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Não-determinismo

Um **autômato finito não-determinístico**, sem transições em vazio, difere dos autômatos finitos determinísticos pelo fato de o co-domínio da função de transição δ ser 2^Q e não simplesmente Q:

$$\delta : Q \times \Sigma \to 2^Q$$

Não-determinismo

Como conseqüência, os autômatos finitos não-determinísticos generalizam o modelo dos autômatos finitos determinísticos através das seguintes extensões:

- Introduz-se o impasse em configurações não-finais;
  Como $\emptyset \in 2^Q$, é possível não especificar transições para certas combinações de estado corrente e próximo símbolo de entrada.
- Introduz-se o não-determinismo, no sentido literal da palavra.
  Nos casos em que $|(\delta(q, \sigma))| \geqslant 2$, haverá mais de uma possibilidade de movimentação para o autômato finito não-determinístico na configuração corrente.

Aceitação e rejeição

Diz-se que um autômato finito não-determinístico **aceita** uma cadeia de entrada quando houver alguma seqüência de movimentos que o leve da configuração inicial para uma configuração final. Diferentemente do autômato finito determinístico, em que essa seqüência, se existir, é única para cada cadeia de entrada, no caso do autômato finito não-determinístico é possível que exista mais de uma seqüência que satisfaça a essa condição para uma dada cadeia de entrada. Sempre que o autômato finito não-determinístico se deparar com mais de uma possibilidade de movimentação, é feita a escolha (arbitrária) de uma das alternativas; em caso de insucesso no reconhecimento, deve-se considerar sucessivamente cada uma das demais alternativas ainda não consideradas, até o seu esgotamento; persistindo o insucesso, e esgotadas as alternativas, diz-se que o autômato **rejeita** a cadeia. A Tabela 5 resume esses critérios.

Aceitação e rejeição

Tabela 5: Aceitação e rejeição de cadeias em autômatos finitos

	Dada uma cadeia de entrada, ele:	*Aceita a cadeia de entrada se:*	*Rejeita a cadeia de entrada se:*
Autômato finito determinístico	Executa uma única seqüência de movimentos.	Pára em uma configuração final.	Pára em uma configuração não-final.
Autômato finito não-determinístico	Pode executar várias seqüências distintas de movimentos.	Pára em uma configuração final em pelo menos uma seqüência.	Pára em configurações não-finais em todas as seqüências.

Exemplo

Exemplo 3.3
Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito não-determinístico:

$$\begin{aligned} Q &= \{q_0, q_1, q_2\} \\ \Sigma &= \{a, b, c\} \\ \delta &= \{(q_0, a) \to \{q_1, q_2\}, (q_1, b) \to \{q_1, q_2\}, (q_2, c) \to \{q_2\}\} \\ F &= \{q_1, q_2\} \end{aligned}$$

Exemplo

Continuação

O diagrama de transição de estados para esse autômato, que reconhece a linguagem $ab^* \mid ab^*bc^* \mid ac^*$, ou simplesmente ab^*c^*, é apresentado na Figura 6.

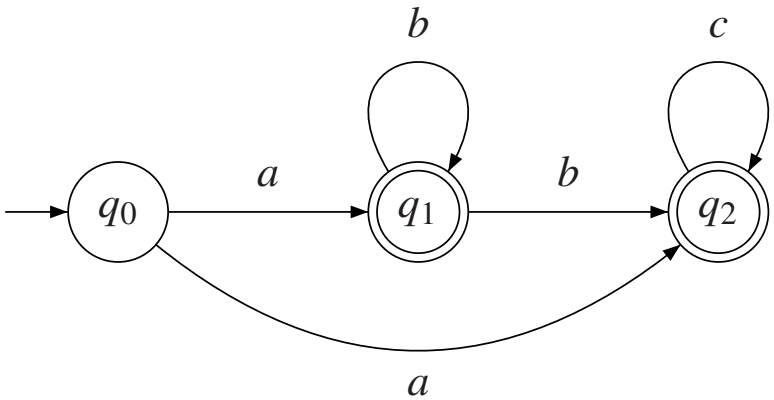


Figura 6: Autômato não-determinístico

Exemplo

Continuação

Considere-se a cadeia $abbccc$ e faça-se uma simulação da operação do autômato a partir de sua configuração inicial:

$$(q_0, abbccc) \vdash (q_2, bbccc)$$

Nesta seqüência, a escolha do ramo inferior, em resposta ao símbolo de entrada a, conduz o autômato a um impasse, pois não há possibilidade de movimentação em q_2 com o símbolo b. Deve-se, então, tentar a segunda alternativa de movimentação em q_0:

$$(q_0, abbccc) \vdash (q_1, bbccc) \vdash (q_2, bccc)$$

Apesar do avanço no reconhecimento, novo impasse é atingido no estado q_2 como conseqüência da escolha efetuada em q_1 para a transição usando o símbolo b. Como não restam outras alternativas em q_0, deve-se considerar a segunda alternativa de movimentação em q_1:

$$(q_0, abbccc) \vdash (q_1, bbccc) \vdash (q_1, bccc)$$

Exemplo

Continuação

Admitindo-se que a opção inicial de movimentação em q_1 em resposta ao símbolo b seja novamente q_2, a seguinte seqüência de movimentos conduz finalmente o autômato à sua configuração final (que, neste caso, é única):

$$(q_1, bccc) \vdash (q_2, ccc) \vdash (q_2, cc) \vdash (q_2, c) \vdash (q_2, \varepsilon)$$

Seja agora a cadeia aab. Como se pode perceber, não há nenhuma seqüência de movimentos, mesmo considerando-se as duas alternativas para a transição usando a em q_0, que conduza o autômato à sua configuração final. Em um caso, o impasse é atingido no estado q_1 em decorrência do símbolo a; no outro ele ocorre em q_2, também provocado por a. Como não há mais opções decorrentes de não-determinismos a serem consideradas, conclui-se que essa cadeia não pertence à linguagem definida pelo autômato.

Determinismo $\times$ não-determinismo

Apesar de não constituir regra geral, os autômatos finitos não-determinísticos, em certos casos, podem mostrar-se mais simples de serem analisados do que as correspondentes versões determinísticas. O Exemplo 3.4 ilustra uma situação desse tipo.

Exemplo

Exemplo 3.4
Os autômatos das Figuras 7 e 8 reconhecem a linguagem
$(a \mid b \mid c \mid d)^* abcd (a \mid b \mid c \mid d)^*$.

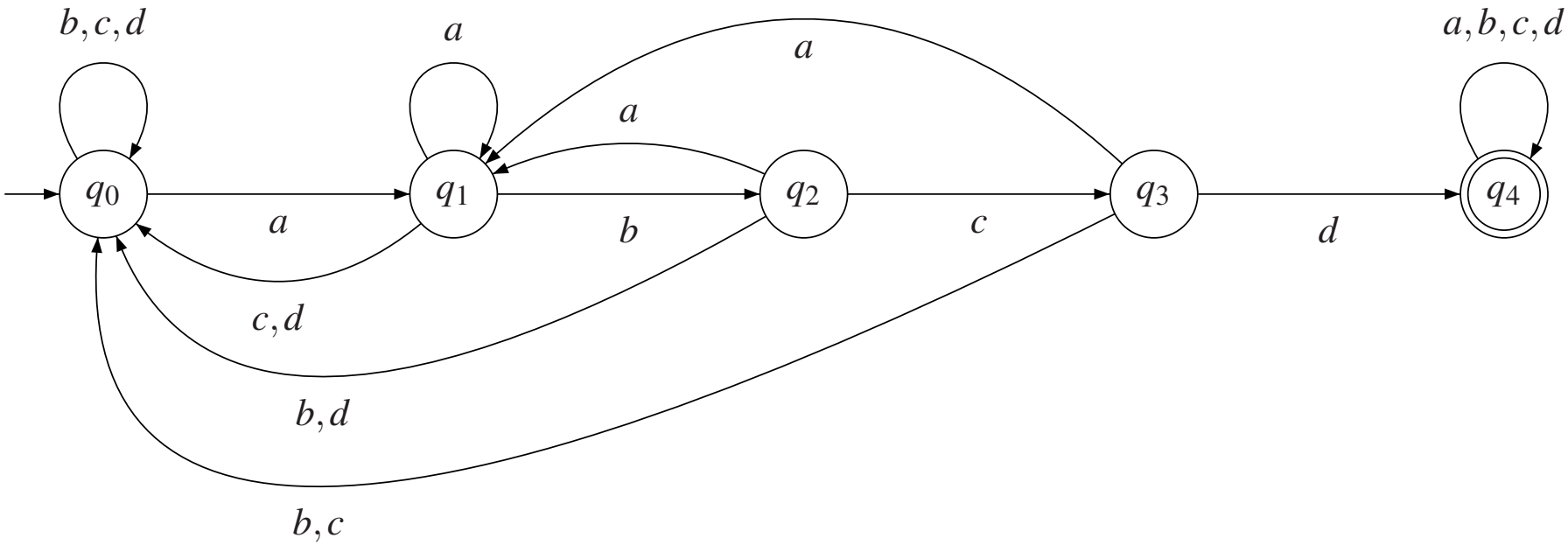


Figura 7: Autômato finito determinístico que reconhece a linguagem formada pelas cadeias que contêm a subcadeia $abcd$

Exemplo

Continuação

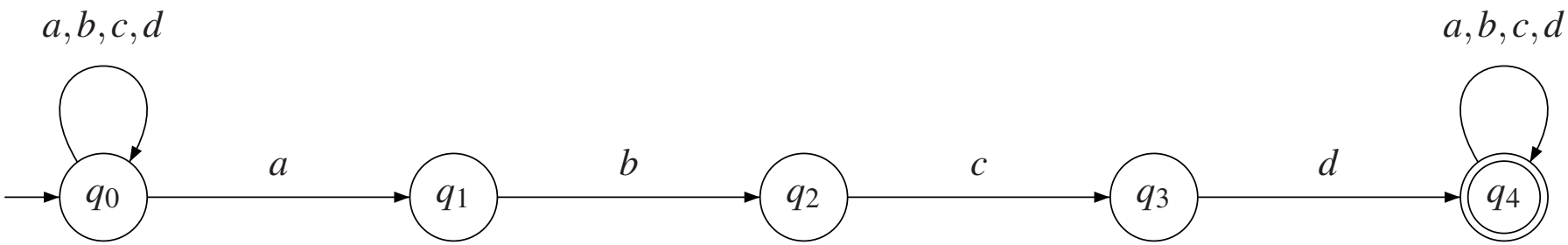


Figura 8: Autômato finito não-determinístico que reconhece a linguagem formada pelas cadeias que contêm a subcadeia $abcd$

Nesse caso é evidente a maior facilidade de interpretação da topologia exibida pela versão não-determinística (Figura 8), quando comparada com a da versão determinística equivalente (Figura 7).

Equivalência

A seguir é mostrada a equivalência entre os autômatos finitos não-determinísticos e os determinísticos, no que diz respeito à classe de linguagens que eles são capazes de reconhecer. A **equivalência**, ou **equipotência**, de tais classes de autômato, constitui um dos mais importantes resultados da teoria dos autômatos finitos, sem paralelo para a maioria dos demais modelos de reconhecedores anteriormente mencionados, uma vez que garante ser sempre possível a aceitação de toda e qualquer linguagem regular através de um autômato determinístico.

Notação tabular

Antes, porém, é preciso introduzir a **notação tabular** para a representação de autômatos finitos. De acordo com essa notação, cada linha da tabela representa um estado distinto q do autômato, e cada coluna é associada a um elemento distinto σ de seu alfabeto de entrada. As células correspondentes à intersecção de cada linha com cada coluna são preenchidas com o elemento (conjunto) de 2^Q determinado por $\delta(q, \sigma)$.

Exemplo

Exemplo 3.5
Considere-se novamente o autômato finito não-determinístico M do Exemplo 3.3 cujo diagrama de estados é apresentado na Figura 6. A representação tabular de M é apresentada na Tabela 6.

Tabela 6: Notação tabular para o autômato finito não-determinístico M da Figura 6

	δ	a	b	c
$\rightarrow$	q_0	$\{q_1, q_2\}$		
$\leftarrow$	q_1		$\{q_1, q_2\}$	
$\leftarrow$	q_2			$\{q_2\}$

Simbologia

Na notação tabular, como se pode perceber, o estado inicial é indicado através do símbolo "$\rightarrow$", ao passo que os estados finais são indicados por "$\leftarrow$". O símbolo "$\leftrightarrow$" indica um estado que seja simultaneamente inicial e final.

Equivalência

Teorema 3.1 "Seja L a linguagem aceita por um autômato finito não-determinístico sem transições em vazio. Então é possível definir um autômato finito determinístico equivalente que aceita L."

Equivalência

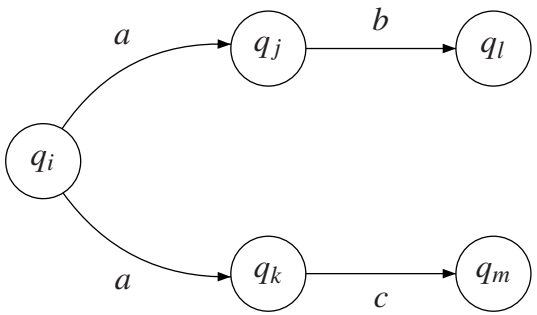


Figura 9: Situação não-determinística original

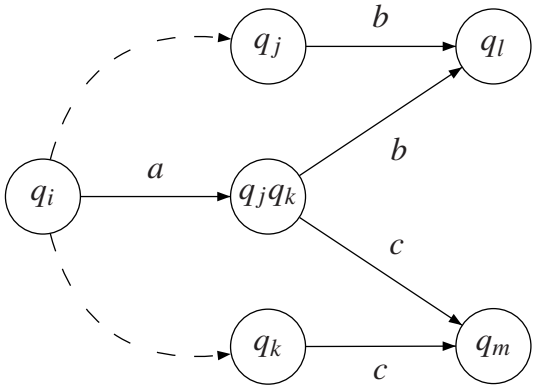


Figura 10: Situação determinística equivalente à da Figura 9

Equivalência

Fica claro, também, que se pelo menos um dos estados, q_j ou q_k, for um estado final, o mesmo deverá acontecer com o estado $q_j q_k$, já que em ambas as versões a cadeia a deve ser aceita pelo autômato. Por outro lado, caso haja coincidência entre os símbolos b e c, um novo não-determinismo será introduzido no estado $q_j q_k$. Daí a necessidade de se repetir o procedimento, removendo a cada iteração todos os novos não-determinismos que venham a ser introduzidos.

Equivalência

Seja $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ o autômato finito não-determinístico originalmente considerado e $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ o autômato finito determinístico correspondente que se deseja obter. A obtenção de M_2 a partir de M_1 pode ser efetuada através do Algoritmo 3.1.

Algoritmo

Algoritmo 3.1 *"Obtenção de um autômato finito determinístico M_2 a partir de um autômato finito não-determinístico M_1."*

- *Entrada: um autômato não-determinístico* $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, *com* $\delta_1 : Q_1 \times \Sigma \to 2^{Q_1}$;
- *Saída: um autômato determinístico* $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, *com* $\delta_2 : Q_2 \times \Sigma \to Q_2$, *tal que* $L(M_2) = L(M_1)$;

Algoritmo

Método:

1. $Q_2 \leftarrow \emptyset$;
2. $\forall i \geqslant 0$, *se* $q_{1i} \in Q_1$ *então* $Q_2 \leftarrow Q_2 \cup \{q_{2i}\}$;
3. $F_2 \leftarrow \emptyset$;
4. $\forall i \geqslant 0$, *se* $q_{1i} \in F_1$ *então* $F_2 \leftarrow F_2 \cup \{q_{2i}\}$;
5. $\delta_2 \leftarrow \emptyset$;
6. $\forall q_{1i} \in Q_1, \sigma \in \Sigma$, *se* $\delta_1(q_{1i}, \sigma) = \{q_{11}, ..., q_{1n}\}, n \geqslant 1$ *então* $\delta_2(q_{2i}, \sigma) = \{q_{21}, ..., q_{2n}\}$;
7. *Substituir todos os elementos* $\{q_{2i}\}$ *de* δ_2 *por* q_{2i};

Algoritmo

Enquanto houver transições não-determinísticas em δ_2, faça:

1. *Selecione uma transição não-determinística qualquer $\delta_2(q,\sigma)$, e considere $\delta_2(q,\sigma) = \{q_{21},...,q_{2i},...,q_{2n}\}, n \geqslant 2$;*
2. *Acrescente um novo estado $q_{21}...q_{2i}...q_{2n}$ à tabela de transição de estados (nesta notação, os estados do conjunto são concatenados formando uma cadeia, em que os índices dos estados estão organizados em ordem crescente, ou em qualquer outra ordem conveniente); se $q_{2i} = q_{2i1}...q_{2im}$, considerar a ordenação de todos os estados obtidos pela substituição de q_{2i} por $q_{2i1}...q_{2im}$ em $q_{21}...q_{2i}...q_{2n}$;*
3. *Substitua, na tabela, todas as referências a $\{q_{21},...,q_{2n}\}$ por $q_{21}...q_{2n}$;*
4. *Para cada $\sigma \in \Sigma$, faça:*
   1. $\delta_2(q_{21}...q_{2n},\sigma) \leftarrow \emptyset$;
   2. *Para cada estado $q_{2j} \in \{q_{21},...,q_{2n}\}$, faça:*
      1. $\delta_2\ (q_{21}...q_{2n},\sigma) \leftarrow \delta_2\ (q_{21}...q_{2n},\sigma) \cup \delta_2\ (q_{2j},\sigma)$;
      2. *Se $q_{2j} \in F_2$, então $F_2 \leftarrow F_2 \cup \{q_{21}...q_{2n}\}$.*

Exemplo

Exemplo 3.6
Considere-se uma vez mais o autômato não-determinístico M do Exemplo 3.3, representado na Figura 6 e na Tabela 6. A aplicação do Algoritmo 3.1 a M conduz à obtenção do autômato da Tabela 7.

Tabela 7: Autômato determinístico equivalente ao autômato M da Tabela 6

	δ'	a	b	c
$\rightarrow$	q_0	q_1q_2		
$\leftarrow$	q_1		q_1q_2	
$\leftarrow$	q_2			q_2
$\leftarrow$	q_1q_2		q_1q_2	q_2

Exemplo

Continuação

A Figura 11 mostra o diagrama de estados do autômato determinístico obtido neste exemplo. Deve-se observar que o estado q_1 tornou-se inacessível como resultado da aplicação do método de eliminação de não-determinismos.

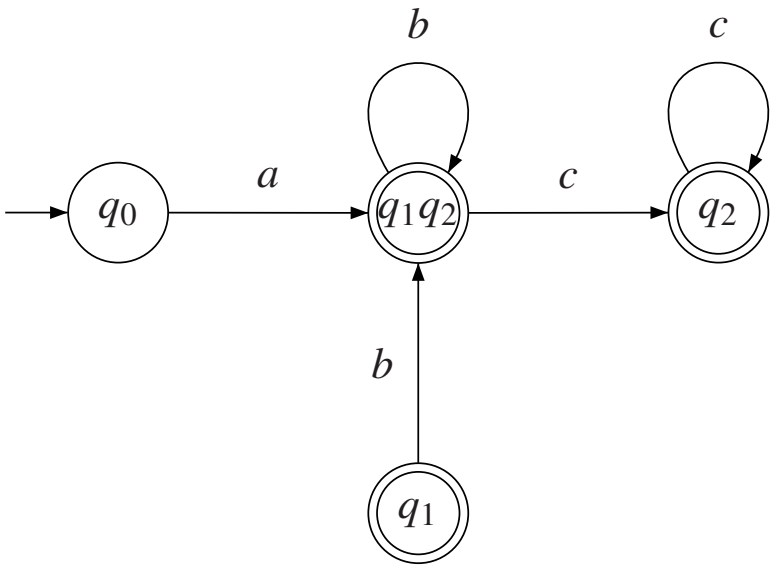


Figura 11: Autômato determinístico equivalente ao autômato M da Figura 6

Exemplo

Exemplo 3.7
Considere-se o autômato finito não-determinístico representado na Figura 12.

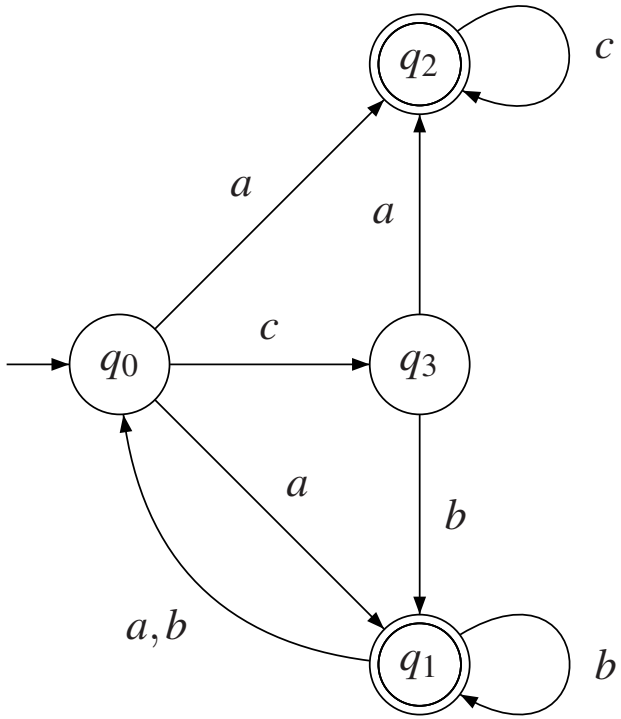


Figura 12: Autômato não-determinístico do Exemplo 3.7

Exemplo

Continuação

A Tabela 8 representa o autômato da Figura 12.

Tabela 8: Eliminação de não-determinismos, autômato inicial

	δ	a	b	c
$\rightarrow$	q_0	$\{q_1, q_2\}$		$\{q_3\}$
$\leftarrow$	q_1	$\{q_0\}$	$\{q_0, q_1\}$	
$\leftarrow$	q_2			$\{q_2\}$
	q_3	$\{q_2\}$	$\{q_1\}$	

Exemplo

Continuação

Os seguintes passos correspondem à aplicação do Algoritmo 3.1.

- Substituir $\{q_0\}$ por q_0, $\{q_1\}$ por q_1, $\{q_2\}$ por q_2 e $\{q_3\}$ por q_3:

Tabela 9: Eliminação de não-determinismos, passo 1

	δ	a	b	c
$\rightarrow$	q_0	$\{q_1, q_2\}$		q_3
$\leftarrow$	q_1	q_0	$\{q_0, q_1\}$	
$\leftarrow$	q_2			q_2
	q_3	q_2	q_1	

Exemplo

Continuação

- Criar um novo estado q_1q_2, substituindo $\{q_1, q_2\}$ na tabela por q_1q_2.

Tabela 10: Eliminação de não-determinismos, passo 2

	δ	a	b	c
$\rightarrow$	q_0	q_1q_2		q_3
$\leftarrow$	q_1	q_0	$\{q_0, q_1\}$	
$\leftarrow$	q_2			q_2
	q_3	q_2	q_1	
$\leftarrow$	q_1q_2	q_0	$\{q_0, q_1\}$	q_2

Exemplo

Continuação

- Criar um novo estado q_0q_1, substituindo $\{q_0, q_1\}$ na tabela por q_0q_1.

Tabela 11: Eliminação de não-determinismos, passo 3

	δ	a	b	c
$\rightarrow$	q_0	q_1q_2		q_3
$\leftarrow$	q_1	q_0	q_0q_1	
$\leftarrow$	q_2			q_2
	q_3	q_2	q_1	
$\leftarrow$	q_1q_2	q_0	q_0q_1	q_2
$\leftarrow$	q_0q_1	$\{q_1q_2, q_0\}$	q_0q_1	q_3

Exemplo

Continuação

- Criar um novo estado $q_0q_1q_2$, substituindo $\{q_1q_2, q_0\}$ na tabela por $q_0q_1q_2$.

Tabela 12: Eliminação de não-determinismos, passo 4

	δ	a	b	c
$\rightarrow$	q_0	q_1q_2		q_3
$\leftarrow$	q_1	q_0	q_0q_1	
$\leftarrow$	q_2			q_2
	q_3	q_2	q_1	
$\leftarrow$	q_1q_2	q_0	q_0q_1	q_2
$\leftarrow$	q_0q_1	$q_0q_1q_2$	q_0q_1	q_3
$\leftarrow$	$q_0q_1q_2$	$q_0q_1q_2$	q_0q_1	$\{q_2, q_3\}$

Exemplo

- Criar um novo estado q_2q_3, substituindo $\{q_2, q_3\}$ na tabela por q_2q_3.

Tabela 13: Eliminação de não-determinismos, autômato final

	δ	a	b	c
$\rightarrow$	q_0	q_1q_2		q_3
$\leftarrow$	q_1	q_0	q_0q_1	
$\leftarrow$	q_2			q_2
	q_3	q_2	q_1	
$\leftarrow$	q_1q_2	q_0	q_0q_1	q_2
$\leftarrow$	q_0q_1	$q_0q_1q_2$	q_0q_1	q_3
$\leftarrow$	$q_0q_1q_2$	$q_0q_1q_2$	q_0q_1	q_2q_3
$\leftarrow$	q_2q_3	q_2	q_1	q_2

Exemplo

Continuação

A Figura 13 apresenta o diagrama de estados do autômato determinístico obtido.

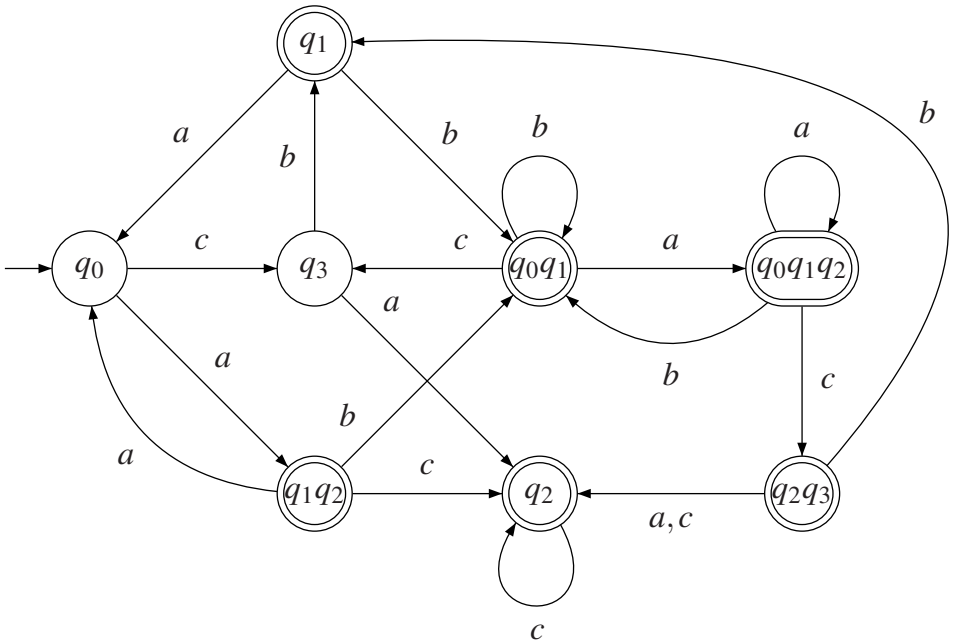


Figura 13: Autômato determinístico equivalente ao da Figura 12

Novos estados

Se, por exemplo, $\delta(q_i, \sigma) = \{q_j, q_k, q_m\}$, o estado $q_j q_k q_m$ do autômato determinístico atenderá ao propósito de permitir que o novo autômato se movimente, a partir deste estado, com transições similares às originalmente presentes em cada um dos estados q_j, q_k e q_m do autômato não-determinístico.

Novos estados

O surgimento de novos estados no autômato determinístico é limitado pela quantidade de combinações distintas que podem ser feitas entre os estados do autômato não-determinístico original. Se $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ é o autômato original não-determinístico, e $M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2)$ é o autômato determinístico equivalente, então $|Q_2| \leqslant 2^{|Q_1|} - 1$, uma vez que toda combinação de estados deverá conter pelo menos um estado do autômato original.

Exemplo

Exemplo 3.8
A eliminação dos não-determinismos do autômato finito representado na Tabela 14 resulta no autômato determinístico da Tabela 15.
Como se pode perceber, $Q_1 = \{q_0, q_1, q_2\}$ e
$Q_2 = \{q_0, q_1, q_2, q_0q_1, q_0q_2, q_1q_2, q_0q_1q_2\}$. Além disso, $|Q_1| = 3$ e $|Q_2| = 2^3 - 1 = 7$.
Trata-se, portanto, de um autômato determinístico no qual todas as combinações possíveis dos estados do autômato original não-determinístico estão consideradas.

Tabela 14: Autômato finito não-determinístico do Exemplo 3.8

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

Exemplo

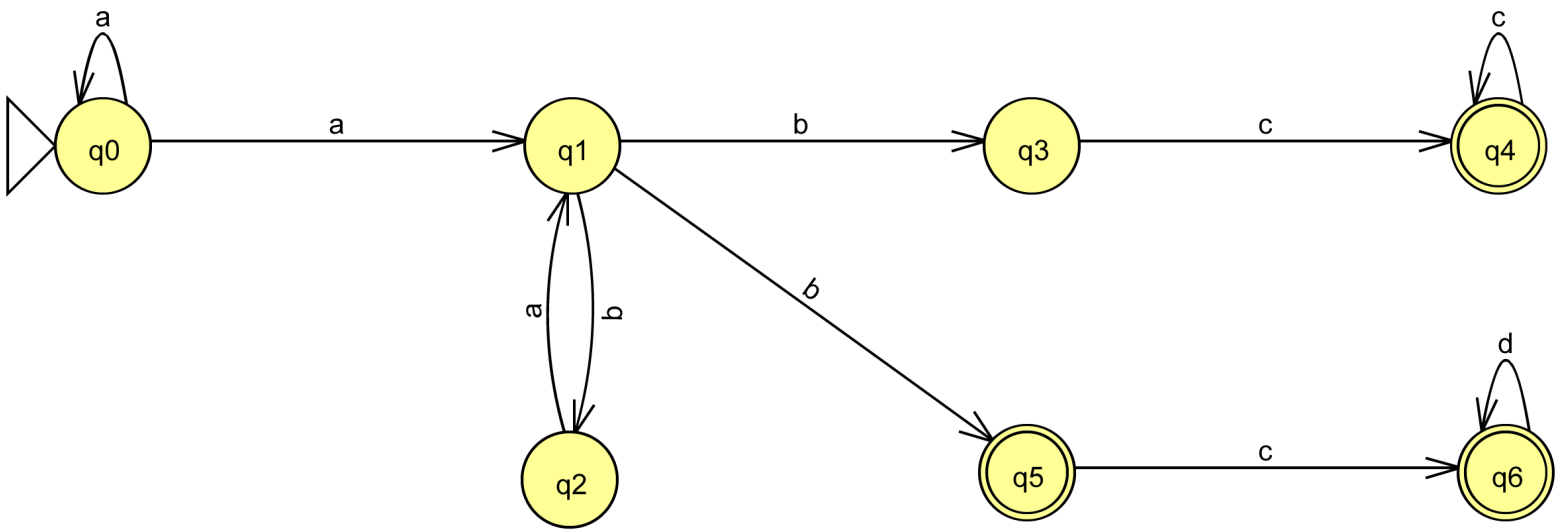
Continuação

Tabela 15: Autômato finito determinístico equivalente ao da Tabela 14

	δ	a	b
$\rightarrow$	q_0	q_1q_2	
$\leftarrow$	q_1		
	q_2	q_0q_2	q_0q_1
$\leftarrow$	q_1q_2	q_0q_2	q_0q_1
$\leftarrow$	q_0q_1	q_1q_2	
	q_0q_2	q_1q_2, q_0q_2	q_0q_1
$\leftarrow$	$q_0q_1q_2$	$q_0q_1q_2$	q_0q_1

Exercício

Obter um autômato finito determinístico que seja equivalente ao autômato:



Equivalência

Como conclusão da apresentação do Teorema 3.1, e com base no Algoritmo 3.1, deve-se acrescentar que é possível garantir, no caso geral, a existência de um autômato finito determinístico equivalente a qualquer autômato finito não-determinístico fornecido.
Dessa maneira, o fato de um autômato finito ser não-determinístico não o torna mais poderoso quanto à classe de linguagens que é capaz de reconhecer, quando comparado com os autômatos finitos determinísticos. Por se tratar, este último, de um modelo de reconhecimento que permite gerar implementações extremamente eficientes, conclui-se ser sempre possível a obtenção de modelos com tais características, independentemente da forma como o autômato se manifesta originalmente quanto ao seu determinismo.

Equivalência

Por outro lado, a existência de autômatos não-determinísticos que sejam equivalentes a autômatos determinísticos é imediata, pois a incorporação de não-determinismos pode ser feita trivialmente, sem alterar a linguagem aceita pelo autômato: basta, por exemplo, incorporar um caminho adicional alternativo que aceite qualquer seqüência de símbolos, a partir de qualquer estado, iniciada por um símbolo que já seja consumido a partir daquele estado, sem, no entanto, permitir que alguma configuração final seja atingida.

Exemplo

Exemplo 3.9
Considere-se M, o autômato determinístico da Figura 14.

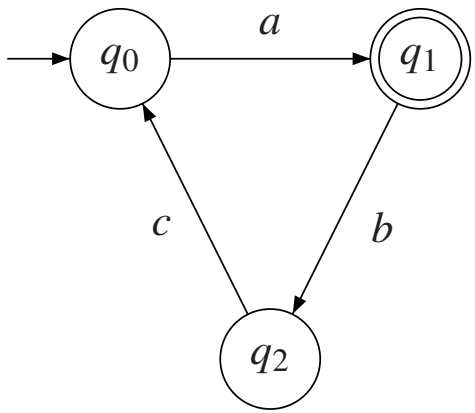


Figura 14: Autômato M determinístico que aceita $a(bca)^*$

Exemplo

O acréscimo de um único novo estado q_3 e da transição $\delta(q_2, c) = q_3$ já seria suficiente para tornar M não-determinístico, sem no entanto alterar a linguagem por ele aceita $(a(bca)^*)$. Naturalmente, inúmeros autômatos podem ser construídos dessa maneira. A Figura 15 apresenta um exemplo.

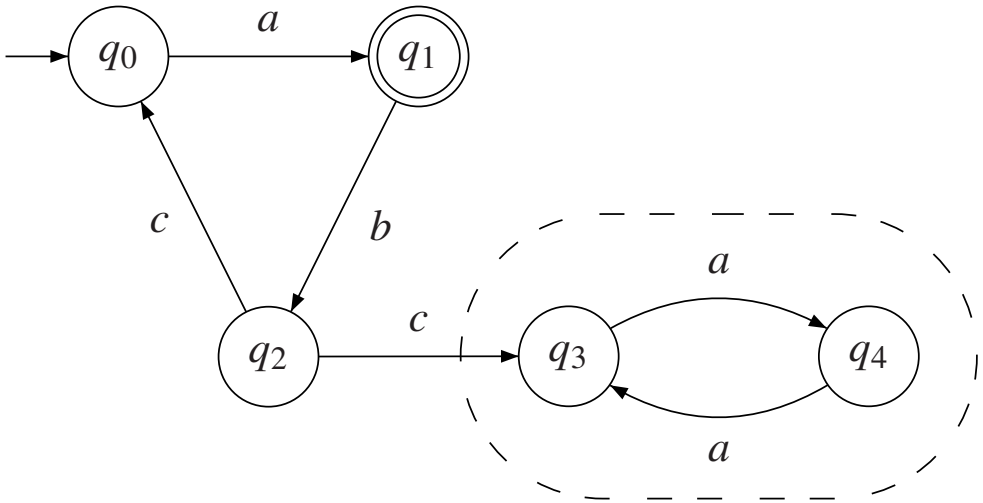


Figura 15: Autômato não-determinístico equivalente ao autômato da Figura 14

Autômatos Finitos Não-Determinísticos com Transições em Vazio

Autômatos finitos não-determinísticos que apresentam **transições em vazio** são aqueles que admitem transições de um estado para outro com ε, além das transições normais, que utilizam os símbolos do alfabeto de entrada. Transições em vazio podem ser executadas sem que seja necessário consultar o símbolo corrente da fita de entrada, e sua execução nem sequer causa o deslocamento do cursor de leitura. Com a introdução de transições em vazio, a função de transição para autômatos finitos não-determinísticos passa a ter seu domínio alterado para:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$$

Autômatos Finitos Não-Determinísticos com Transições em Vazio

Quando um autômato transita em vazio, isso significa que ele muda de estado sem consultar a cadeia de entrada. Sempre que ocorrer a coexistência entre alguma transição em vazio e outras transições (vazias ou não) com origem em um mesmo estado, isso acarreta a necessidade de uma escolha arbitrária da transição a ser aplicada na respectiva configuração, e isso, por sua vez, caracteriza a manifestação de um não-determinismo.

Exemplo

Exemplo 3.10
Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito com transições em vazio.

$$\begin{aligned} Q &= \{q_0, q_1\} \\ \Sigma &= \{a, b\} \\ \delta &= \{(q_0, a) \to \{q_0\}, (q_0, \varepsilon) \to \{q_1\}, (q_1, b) \to \{q_1\}\} \\ F &= \{q_1\} \end{aligned}$$

A linguagem aceita por esse autômato é a^*b^*, conforme pode ser deduzido a partir do diagrama de estados da Figura 16.

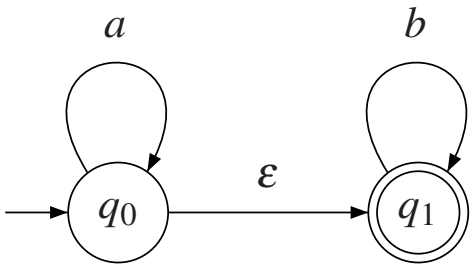


Figura 16: Autômato com transição em vazio

Exemplo

Tomando-se a cadeia de entrada ab como exemplo, as duas seqüências de movimentação possíveis a partir da configuração inicial seriam:

1. $(q_0, ab) \vdash (q_0, b) \vdash (q_1, b) \vdash (q_1, \varepsilon)$ (sucesso)
2. $(q_0, ab) \vdash (q_1, ab)$ (impasse)

No segundo caso, o impasse ocorre devido à aplicação da transição $(q_0, \varepsilon) \rightarrow q_1$ logo ao início do reconhecimento, antes de ser efetuada a leitura do símbolo a. No primeiro caso, ocorre a aceitação da sentença, pois a utilização da transição em vazio foi efetuada em um ponto favorável pelo autômato, ou seja, entre a utilização dos símbolos a e b da cadeia de entrada.

Uso de transições em vazio

Assim como ocorre no no caso dos autômatos determinísticos e não-determinísticos, alguns autômatos finitos com transições em vazio se mostram mais simples de serem analisados do que as correspondentes versões isentas de transições em vazio. O Exemplo 3.11 ilustra uma situação desse tipo.

Exemplo

Exemplo 3.11
Os autômatos das Figuras 17 e 18 reconhecem a linguagem $a^*b^*c^*a^*$.

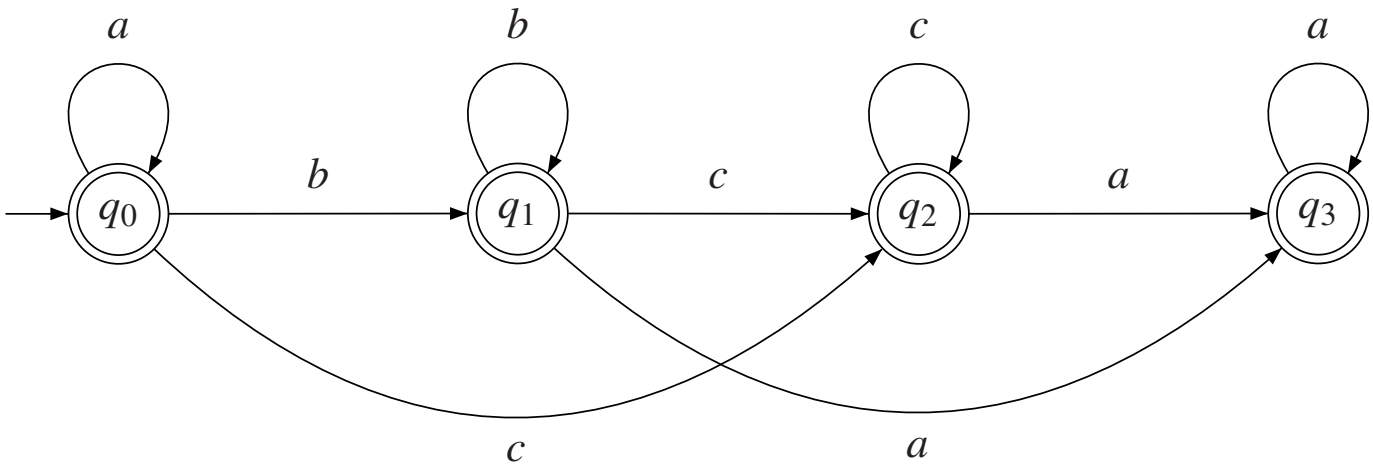


Figura 17: Autômato finito isento de transições em vazio que reconhece a linguagem $a^*b^*c^*a^*$

Exemplo

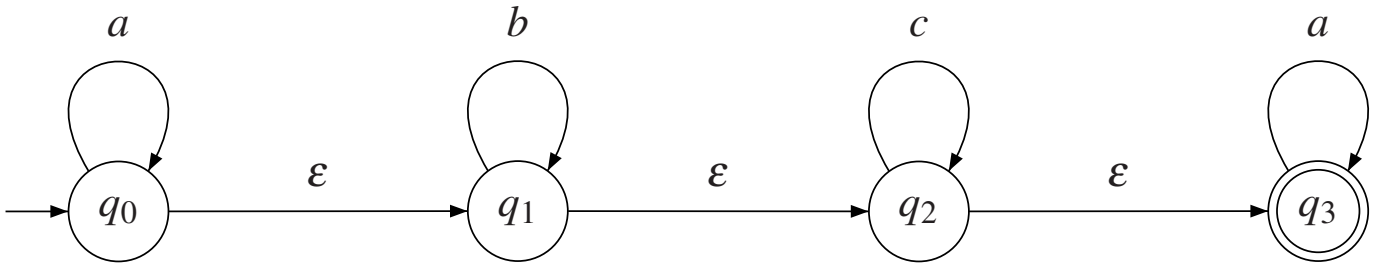


Figura 18: Autômato finito com transições em vazio que reconhece a linguagem $a^*b^*c^*a^*$

Eliminação de transições em vazio

Teorema 3.2 "Todo autômato com transições em vazio aceita uma linguagem que é aceita por algum autômato finito que não contém transições em vazio."

Sejam $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito contendo transições em vazio, representado em notação tabular, e $N = (Q, \Sigma, \delta', q_0, F')$ o autômato finito sem transições em vazio correspondente que a partir dele se deseja obter. A obtenção de N a partir de M pode ser efetuada através do Algoritmo 3.2.

Eliminação de transições em vazio

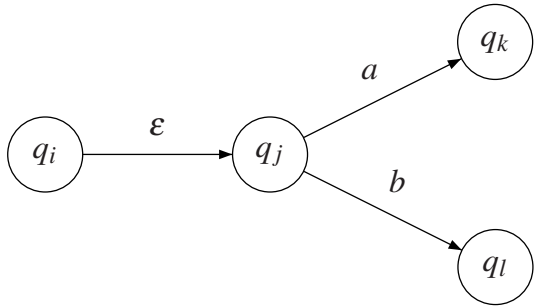


Figura 19: Situação com transição em vazio original

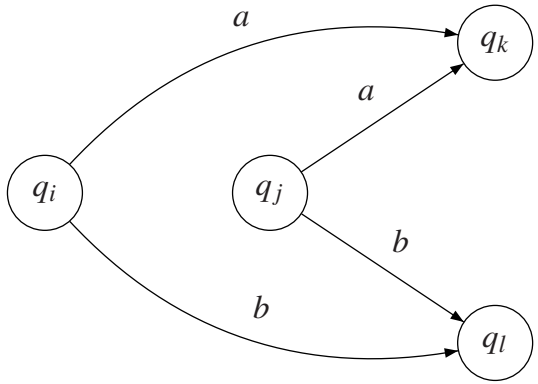


Figura 20: Situação sem transição em vazio equivalente à da Figura 19

Algoritmo para eliminação de transições em vazio

Algoritmo 3.2 *"Obtenção de um autômato finito N, sem transições em vazio, a partir de um autômato finito M, com transições em vazio."*

- *Entrada: um autômato finito com transições em vazio M;*
- *Saída: um autômato finito sem transições em vazio N, tal que* $L(N) = L(M)$*;*
- *Método:*

Algoritmo para eliminação de transições em vazio

1. *Eliminação das transições em vazio*
   Considere-se um estado qualquer $q_i \in Q$. Se houver uma transição em vazio de q_i para q_j, deve-se eliminá-la, copiando-se para a linha que representa o estado q_i todas as transições que partem dos estados q_j para os quais é feita a transição em vazio.
   Esse procedimento corresponde, em notação tabular, à realização de uma fusão ("merge") entre a linha do estado q_i que contém a transição em vazio para o estado-destino q_j e a própria linha do estado q_j, armazenando-se o resultado novamente na linha correspondente ao estado q_i.
   Havendo mais de uma transição em vazio indicadas, deve-se repetir cumulativamente o procedimento para todas elas.
   Se $\delta(q_i, \varepsilon) \in F$, então $F' \leftarrow F' \cup \{q_i\}$, sendo que inicialmente $F' \leftarrow F$.

Algoritmo para eliminação de transições em vazio

2. *Iteração*
   Repetir o passo anterior para os demais estados do autômato, até que todos eles tenham sido considerados (ou seja, até que a última linha tenha sido atingida).
   Nos casos em que houver transições em vazio para estados que por sua vez também transitam em vazio para outros estados, será necessário iterar o procedimento várias vezes sobre a tabela, até que todas as transições em vazio tenham sido eliminadas.

Exemplo

Exemplo 3.12
Considere-se o autômato finito M do Exemplo 3.10, representado na Tabela 16.

Tabela 16: Autômato original apresentando transições em vazio

	δ	a	b	ε
$\rightarrow$	q_0	q_0		q_1
$\leftarrow$	q_1		q_1	

Como há uma transição em vazio de q_0 para q_1, deve-se copiar as transições de q_1 para q_0 ($\delta(q_1, b)$ apenas, neste caso) e, além disso, considerar q_0 como estado final, uma vez que q_1 é estado final. A Tabela 17 representa a função de transição δ'.

Exemplo

Tabela 17: Autômato equivalente ao da Tabela 16, porém isento de transições em vazio

	δ'	a	b
$\leftrightarrow$	q_0	q_0	q_1
$\leftarrow$	q_1		q_1

O diagrama de estados do autômato finito correspondente, sem transições em vazio, é apresentado na Figura 21.

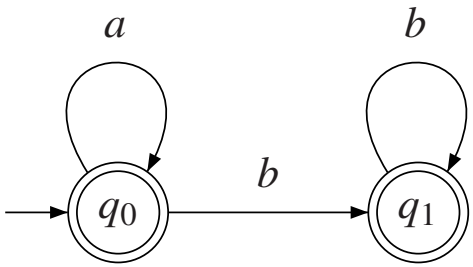
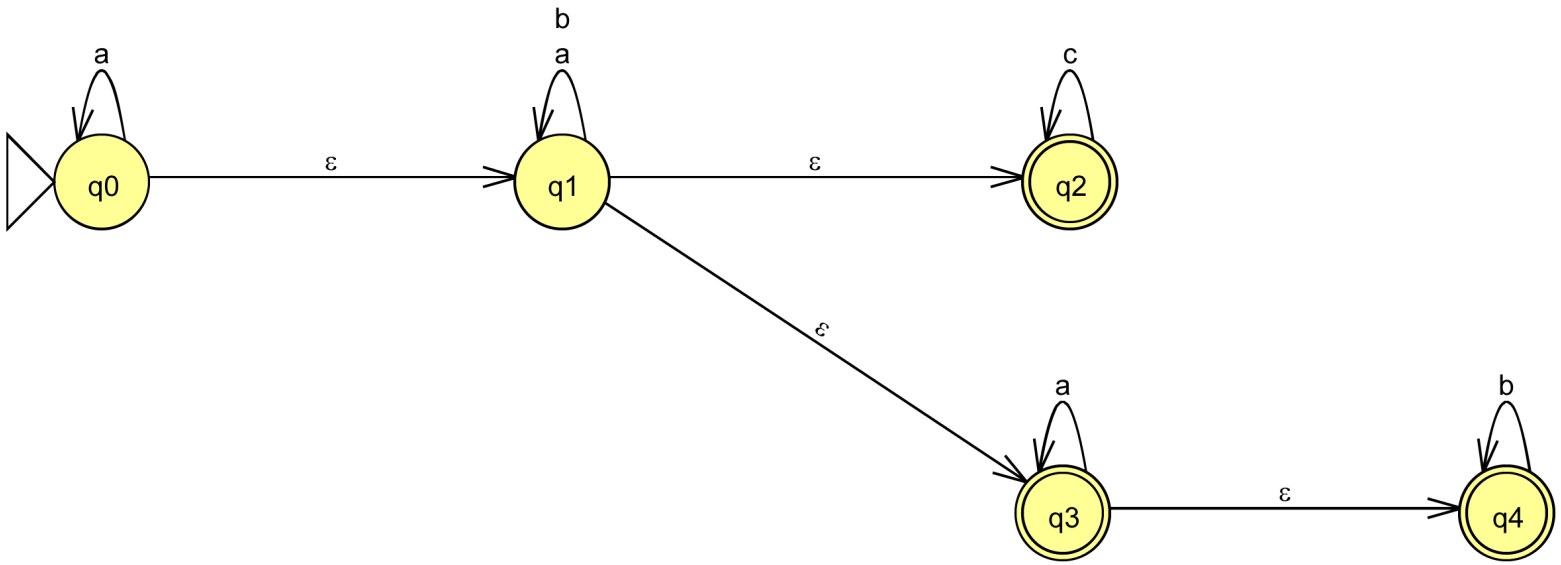


Figura 21: Autômato equivalente ao da Tabela 17, eliminadas as transições em vazio

Exercício

Obter um autômato finito sem transições em vazio que seja equivalente ao autômato:



Modelos de autômato finito

Os modelos de autômato finito considerados até o presente momento foram:

1. Determinístico sem transições em vazio, com $\delta : Q \times \Sigma \to Q$;
2. Não-determinístico sem transições em vazio, com $\delta : Q \times \Sigma \to 2^Q$;
3. Não-determinístico com transições em vazio, com $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$;

Para completar o quadro de possibilidades, define-se o modelo de autômato finito determinístico com transições em vazio:

4. Determinístico com transições em vazio, com $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to Q$.

Não-determinismo e transições em vazio

Considerados todos os casos, cumpre retomar a discussão sobre o conceito de “não-determinismo” e a relação do mesmo com a forma através da qual são definidas as funções de transição dos autômatos finitos.
Se, por um lado, os autômatos dos modelos (2) e (3) são ditos não-determinísticos, isso não significa que, em casos particulares, sua operação não possa ocorrer de forma determinística (ver Exemplo 3.13). Por outro lado, os autômatos do modelo (4), apesar de denominados determinísticos, podem perfeitamente exibir um comportamente não-determinístico durante sua operação (ver Exemplo 3.14). Os autômatos do modelo (1), por sua vez, exibem sempre um comportamento determinístico.

Exemplo

Exemplo 3.13
Seja $M_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ um autômato finito não-determinístico cuja função de transição δ é definida como:

$$\begin{aligned} \delta(q_0, a) &= \{q_0\} \\ \delta(q_0, b) &= \{q_1\} \\ \delta(q_1, \varepsilon) &= \{q_2\} \end{aligned}$$

Qualquer que seja a configuração corrente (q_i, α) considerada, existe sempre, no máximo, uma única transição de M_1 que pode ser aplicada e, portanto, no máximo, uma única próxima configuração possível. Logo, a operação de M_1 é sempre determinística.

Exemplo

Exemplo 3.14
Seja $M_2 = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$ um autômato finito não-determinístico cuja função de transição δ é definida como:

$$\begin{aligned} \delta(q_0, a) &= q_0 \\ \delta(q_0, b) &= q_0 \\ \delta(q_0, \varepsilon) &= q_1 \end{aligned}$$

Considere-se a cadeia de entrada a. As seguintes seqüências de movimentações são válidas em M_2:

- $(q_0, a) \vdash (q_0, \varepsilon)$
- $(q_0, a) \vdash (q_1, a)$

Logo, a operação de M_2 é, nesse caso, não-determinística.

Não-determinismo e transições em vazio

Como mostram os Exemplos 3.13 e 3.14, o que efetivamente determina o comportamento que um autômato exibe durante a sua operação (se determinístico ou não) não é o formato genérico da função de transição adotoda, mas as especificidades de sua própria definição.

Não-determinismo e transições em vazio

Um autômato não-determinístico do modelo (2) pode operar de forma determinística se:

- $\nexists q \in Q, \sigma \in \Sigma$, tal que $|\delta(q, \sigma)| \geqslant 2$;

Um autômato não-determinístico do modelo (3) pode operar de forma determinística se:

- $\nexists q \in Q, \sigma \in \Sigma$, tal que $|\delta(q, \sigma)| \geqslant 2$, e
- $\nexists q \in Q$, tal que $|\delta(q, \varepsilon)| \geqslant 2$, e
- $\nexists q \in Q, \sigma \in \Sigma$, tal que $|\delta(q, \sigma)| \geqslant 1$ e $|\delta(q, \varepsilon)| \geqslant 1$.

Um autômato determinístico do modelo (4), por sua vez, pode operar de forma não-determinística, se:

- $\exists q \in Q, \sigma \in \Sigma$ tal que $\delta(q, \sigma)$ e $\delta(q, \varepsilon)$ são definidas.

Estados inacessíveis e inúteis

Nem todos os estados de um autômato necessariamente contribuem para a definição da linguagem por ele aceita. Estados inacessíveis e estados inúteis são os mais importantes representantes desta categoria de estados, os primeiros porque não podem ser alcançados a partir do estado inicial do autômato, e os demais porque não levam a nenhum dos estados finais. Na Figura 22, q_1 é um estado inacessível e q_2 um estado inútil.

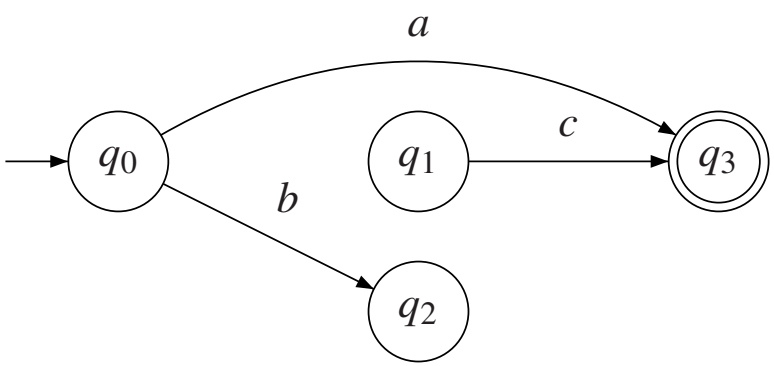


Figura 22: Ilustração dos conceitos de estado inacessível (q_1) e de estado inútil (q_2)

Estados inacessíveis

Seja $M = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$ um autômato finito qualquer. Formalmente, um estado $q_{1i} \in Q_1$ é dito "inacessível" quando não existir caminho, formado por transições válidas, que conduza o autômato do seu estado inicial q_{10} até o estado q_{1i}. Em outras palavras, não existe $\alpha \in \Sigma^*$ tal que $\delta(q_{10}, \alpha) = q_{1i}$. Caso contrário, o estado q_{1i} é dito "acessível".
Estados inacessíveis não contribuem para a definição da linguagem aceita por M, podendo ser sistematicamente identificados e eliminados do conjunto de estados, sem prejuízo para a linguagem aceita pelo autômato.

Eliminação de estados inacessíveis

Considere-se a função de transição δ representada na forma tabular. Acrescentem-se duas novas colunas à tabela: a primeira, denominada "acessível", com a função de marcar os estados acessíveis, e a segunda, denominada "considerado", cuja função é indicar que o correspondente estado já foi levado em conta pelo método. O Algoritmo 3.3 apresenta um método prático que sistematiza a identificação de estados inacessíveis.

Algoritmo

Algoritmo 3.3 *"Obtenção de um autômato sem estados inacessíveis equivalente a outro com estados inacessíveis."*

- *Entrada: um autômato finito* $M = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$*, representado na notação tabular;*
- *Saída: um autômato finito* $N = (Q_2, \Sigma, \delta_2, q_{20}, F_2)$*, isento de estados inacessíveis, e tal que* $L(N) = L(M)$*;*

Algoritmo

Método:

1. *Considere-se a linha correspondente ao estado inicial de M. Marque-se este estado como "acessível" na coluna apropriada.*
2. *Para cada célula, desta linha da tabela, que contiver o nome de algum estado, marcar tal estado, na linha correspondente, como "acessível". Por último, marque-se o estado corrente como "considerado".*
3. *Escolha-se arbitrariamente qualquer outro estado que tenha sido previamente marcado como "acessível", mas que ainda não esteja marcado como "considerado". Repitam-se os passos (2) e (3) até que não mais existam na tabela estados acessíveis, porém não considerados.*
4. *N é definido a partir de M eliminando-se todos os estados inacessíveis (ou seja, aqueles que não foram marcados como acessíveis pelo algoritmo), bem como todas as transições que deles partem ou a eles chegam.*

Exemplo

Exemplo 3.15
Seja $M = (Q, \Sigma, \delta, q_0, F)$, representado através da Tabela 18.

Tabela 18: Autômato original com estados inacessíveis

	δ	a	b	c	d	e	f	g
$\rightarrow$	q_0	q_0	q_4	q_3				
	q_1	q_4			q_1			
$\leftarrow$	q_2		q_4			q_1		
	q_3					q_4		
	q_4				q_3		q_5	
$\leftarrow$	q_5			q_0				q_5

Exemplo

Tabela 19: Autômato original com estados inacessíveis, q_0 acessível

	δ	a	b	c	d	e	f	g	Acessível	Considerado
$\rightarrow$	q_0	q_0	q_4	q_3					✓	
	q_1	q_4			q_1					
$\leftarrow$	q_2		q_4			q_1				
	q_3					q_4				
	q_4				q_3		q_5			
$\leftarrow$	q_5			q_0				q_5		

Exemplo

Tabela 20: Autômato original com estados inacessíveis, q_0 considerado

	δ	a	b	c	d	e	f	g	Acessível	Considerado
$\rightarrow$	q_0	q_0	q_4	q_3					✓	✓
	q_1	q_4			q_1					
$\leftarrow$	q_2		q_4			q_1				
	q_3					q_4			✓	
	q_4				q_3		q_5		✓	
$\leftarrow$	q_5			q_0				q_5		

Exemplo

Tabela 21: Autômato original com estados inacessíveis, q_3 considerado

	δ	a	b	c	d	e	f	g	Acessível	Considerado
$\rightarrow$	q_0	q_0	q_4	q_3					✓	✓
	q_1	q_4			q_1					
$\leftarrow$	q_2		q_4			q_1				
	q_3					q_4			✓	✓
	q_4				q_3		q_5		✓	
$\leftarrow$	q_5			q_0				q_5		

Exemplo

Tabela 22: Autômato original com estados inacessíveis, q_4 considerado

	δ	a	b	c	d	e	f	g	Acessível	Considerado
$\rightarrow$	q_0	q_0	q_4	q_3					✓	✓
	q_1	q_4			q_1					
$\leftarrow$	q_2		q_4			q_1				
	q_3					q_4			✓	✓
	q_4				q_3		q_5		✓	✓
$\leftarrow$	q_5			q_0				q_5	✓	

Exemplo

Tabela 23: Autômato original com estados inacessíveis, q_5 considerado

	δ	a	b	c	d	e	f	g	Acessível	Considerado
$\rightarrow$	q_0	q_0	q_4	q_3					✓	✓
	q_1	q_4			q_1					
$\leftarrow$	q_2		q_4			q_1				
	q_3					q_4			✓	✓
	q_4				q_3		q_5		✓	✓
$\leftarrow$	q_5			q_0				q_5	✓	✓

Exemplo

Tabela 24: Autômato equivalente ao da Tabela 18, eliminados os estados inacessíveis

	δ	a	b	c	d	e	f	g
$\rightarrow$	q_0	q_0	q_4	q_3				
	q_3					q_4		
	q_4				q_3		q_5	
$\leftarrow$	q_5			q_0				q_5

Estados inúteis

Estados inúteis são estados que, apesar de poderem ser alcançados a partir do estado inicial do autômato, não conduzem a nenhum de seus estados finais. Logo, eles em nada contribuem para a aceitação de sentenças da linguagem definida pelo autômato, podendo portanto ser removidos sem qualquer prejuízo para a linguagem reconhecida. A base de indução usada no algoritmo de eliminação de estados inúteis será o conjunto dos estados finais do autômato, que, por definição, são sempre úteis (todo estado final reconhece pelo menos uma cadeia, a cadeia vazia). Naturalmente, se não houver estados finais no autômato, todos os seus estados podem ser declarados inúteis, e a linguagem por ele definida será vazia.

Eliminação de estados inúteis

A aplicação deste algoritmo pode ser sistematizada de forma semelhante à que foi elaborada para o algoritmo de eliminação de estados inacessíveis. Basta representar a função de transição δ na forma tabular e acrescentar duas novas colunas à tabela: a primeira, denominada “útil”, e a segunda, denominada “considerado”, e executar os passos do Algoritmo 3.4.

Algoritmo

Algoritmo 3.4 *"Método prático para obtenção de um autômato sem estados inúteis equivalente a outro com estados inúteis, porém sem estados inacessíveis."*

- *Entrada: um autômato finito* $M = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$*, representado na notação tabular;*
- *Saída: um autômato finito* $N = (Q_2, \Sigma, \delta_2, q_{20}, F_2)$*, isento de estados inúteis, e tal que* $L(N) = L(M)$*;*

Algoritmo

Método:

1. *Considerem-se as linhas correspondentes aos estados finais de M. Marquem-se as mesmas como úteis, na coluna apropriada.*
2. *Selecione-se um estado qualquer marcado como "útil", porém ainda não marcado como "considerado". Inspecionem-se todos os demais estados do autômato, identificando quais deles permitem transitar para o estado selecionado. Marquem-se todos como úteis. Marque-se finalmente o estado selecionado como "considerado".*
3. *Repita-se o passo (2) até que não reste mais nenhum estado marcado como "útil", mas ainda não como "considerado".*
4. *O autômato N é definido a partir de M eliminando-se-lhe todos os estados inúteis, bem como todas as transições que deles partem ou a eles chegam.*

Exemplo

Exemplo 3.16
Considere-se o autômato da Figura 23, em que todos os estados são acessíveis mas nem todos são úteis.

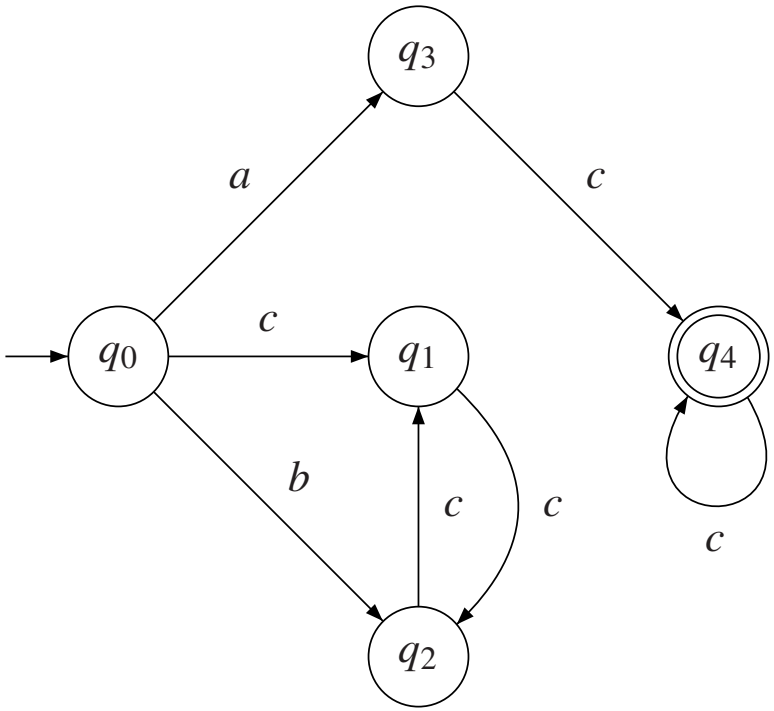


Figura 23: Autômato contendo estados inúteis

Exemplo

Tabela 25: Autômato da Figura 23, estado q_4 útil

	δ	a	b	c	Útil	Considerado
$\rightarrow$	q_0	q_3	q_2	q_1		
	q_1			q_2		
	q_2			q_1		
	q_3			q_4		
$\leftarrow$	q_4			q_4	✓	

Exemplo

Tabela 26: Autômato da Figura 23, estado q_4 considerado

	δ	a	b	c	Útil	Considerado
$\rightarrow$	q_0	q_3	q_2	q_1		
	q_1			q_2		
	q_2			q_1		
	q_3			q_4	✓	
$\leftarrow$	q_4			q_4	✓	✓

Exemplo

Tabela 27: Autômato da Figura 23, estado q_0 útil

	δ	a	b	c	Útil	Considerado
$\rightarrow$	q_0	q_3	q_2	q_1	✓	
	q_1			q_2		
	q_2			q_1		
	q_3			q_4	✓	✓
$\leftarrow$	q_4			q_4	✓	✓

Exemplo

Tabela 28: Autômato da Figura 23, estado q_0 considerado

	δ	a	b	c	Útil	Considerado
$\rightarrow$	q_0	q_3	q_2	q_1	✓	✓
	q_1			q_2		
	q_2			q_1		
	q_3			q_4	✓	✓
$\leftarrow$	q_4			q_4	✓	✓

Exemplo

Tabela 29: Autômato equivalente ao da Figura 23, estados inúteis eliminados

	δ	a	b	c
$\rightarrow$	q_0	q_3		
	q_3			q_4
$\leftarrow$	q_4			q_4

Perguntas

1. Dado um autômato finito qualquer, será possível obter uma versão determinística, isenta de transições em vazio e estados inacessíveis ou inúteis, aplicando-se uma só vez cada um dos algoritmos descritos?
2. Em caso afirmativo, em qual seqüência devem eles ser aplicados?

Fatos

Para responder a essas questões, é suficiente observar que:

- A eliminação de transições em vazio:
  - *Pode introduzir* não-determinismos;
  - *Pode fazer surgir* estados inacessíveis ou inúteis.
- A eliminação de não-determinismos:
  - *Pode fazer surgir* estados inacessíveis ou inúteis;
  - *Não introduz* transições em vazio.
- A eliminação de estados inacessíveis ou inúteis:
  - *Não faz surgir* não-determinismos;
  - *Não introduz* transições em vazio;
  - Pode ser feita em qualquer ordem.

Respostas

Logo, é fácil provar que a resposta para as questões inicialmente propostas é “sim”, bastando para isso aplicar os algoritmos na seguinte seqüência:

1. Eliminação de transições em vazio, se houver;
2. Eliminação dos não-determinismos restantes, caso haja algum;
3. Eliminação de estados inacessíveis e inúteis (em qualquer ordem), caso existam.

Exemplo

Exemplo 3.17
Considere-se um autômato finito M, conforme a Tabela 30, que apresenta transições em vazio e não-determinismo (conseqüência, no caso, das transições em vazio existentes).

Tabela 30: Autômato original com transições em vazio

	δ	a	b	c	d	e	ε
$\rightarrow$	q_0		q_1		q_1		q_2
	q_1	q_3			q_1		q_4
	q_2		q_2		q_3		
	q_3	q_4		q_3			
$\leftarrow$	q_4		q_2			q_4	q_0

Exemplo

Eliminação de transições em vazio

Tabela 31: Eliminação de $(q_0, \varepsilon) \rightarrow q_2$

	δ	a	b	c	d	e	ε
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$		
	q_1	q_3			q_1		q_4
	q_2		q_2		q_3		
	q_3	q_4		q_3			
$\leftarrow$	q_4		q_2			q_4	q_0

Exemplo

Eliminação de transições em vazio

Tabela 32: Eliminação de $(q_1, \varepsilon) \rightarrow q_4$

	δ	a	b	c	d	e	ε
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$		
$\leftarrow$	q_1	q_3	q_2		q_1	q_4	q_0
	q_2		q_2		q_3		
	q_3	q_4		q_3			
$\leftarrow$	q_4		q_2			q_4	q_0

Exemplo

Eliminação de transições em vazio

Tabela 33: Eliminação de $(q_1, \varepsilon) \rightarrow q_0$

	δ	a	b	c	d	e	ε
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$		
$\leftarrow$	q_1	q_3	$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4	
	q_2		q_2		q_3		
	q_3	q_4		q_3			
$\leftarrow$	q_4		q_2			q_4	q_0

Exemplo

Eliminação de transições em vazio

Tabela 34: Eliminação de $(q_4, \varepsilon) \rightarrow q_0$

	δ	a	b	c	d	e	ε
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$		
$\leftarrow$	q_1	q_3	$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4	
	q_2		q_2		q_3		
	q_3	q_4		q_3			
$\leftarrow$	q_4		$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4	

Exemplo

Eliminação de transições em vazio

Tabela 35: Autômato equivalente ao da Tabela 30, eliminadas as transições em vazio

	δ	a	b	c	d	e
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$	
$\leftarrow$	q_1	q_3	$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4
	q_2		q_2		q_3	
	q_3	q_4		q_3		
$\leftarrow$	q_4		$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4

Exemplo

Eliminação de não-determinismos

Tabela 36: Autômato isento de não-determinismos, versão inicial

	δ	a	b	c	d	e
$\rightarrow$	q_0		$\{q_1, q_2\}$		$\{q_1, q_3\}$	
$\leftarrow$	q_1	q_3	$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4
	q_2		q_2		q_3	
	q_3	q_4		q_3		
$\leftarrow$	q_4		$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4
$\leftarrow$	$\{q_1, q_2\}$	q_3	$\{q_1, q_2\}$		$\{q_1, q_3\}$	q_4
$\leftarrow$	$\{q_1, q_3\}$	$\{q_3, q_4\}$	$\{q_1, q_2\}$	q_3	$\{q_1, q_3\}$	q_4

Exemplo

Eliminação de não-determinismos

Tabela 37: Autômato isento de não-determinismos, versão final

	δ	a	b	c	d	e
$\rightarrow$	q_0		q_1q_2		q_1q_3	
$\leftarrow$	q_1	q_3	q_1q_2		q_1q_3	q_4
	q_2		q_2		q_3	
	q_3	q_4		q_3		
$\leftarrow$	q_4		q_1q_2		q_1q_3	q_4
$\leftarrow$	q_1q_2	q_3	q_1q_2		q_1q_3	q_4
$\leftarrow$	q_1q_3	q_3q_4	q_1q_2	q_3	q_1q_3	q_4
$\leftarrow$	q_3q_4	q_4	q_1q_2	q_3	q_1q_3	q_4

Exemplo

Renomeando os estados

Tabela 38: Autômato da Tabela 37, com estados renomeados

	δ	a	b	c	d	e
$\rightarrow$	q_0		q_5		q_6	
$\leftarrow$	q_1	q_3	q_5		q_6	q_4
	q_2		q_2		q_3	
	q_3	q_4		q_3		
$\leftarrow$	q_4		q_5		q_6	q_4
$\leftarrow$	q_5	q_3	q_5		q_6	q_4
$\leftarrow$	q_6	q_7	q_5	q_3	q_6	q_4
$\leftarrow$	q_7	q_4	q_5	q_3	q_6	q_4

Exemplo

Eliminação de estados inacessíveis

Tabela 39: Eliminação de estados inacessíveis

	δ	a	b	c	d	e	Acessível	Considerado
$\rightarrow$	q_0		q_5		q_6		✓	✓
$\leftarrow$	q_1	q_3	q_5		q_6	q_4		
	q_2		q_2		q_3			
	q_3	q_4		q_3			✓	✓
$\leftarrow$	q_4		q_5		q_6	q_4	✓	✓
$\leftarrow$	q_5	q_3	q_5		q_6	q_4	✓	✓
$\leftarrow$	q_6	q_7	q_5	q_3	q_6	q_4	✓	✓
$\leftarrow$	q_7	q_4	q_5	q_3	q_6	q_4	✓	✓

Exemplo

Resultado final

Tabela 40: Autômato final obtido pela eliminação de transições em vazio, não-determinismos e estados inacessíveis

	δ	a	b	c	d	e
$\rightarrow$	q_0		q_5		q_6	
	q_3	q_4		q_3		
$\leftarrow$	q_4		q_5		q_6	q_4
$\leftarrow$	q_5	q_3	q_5		q_6	q_4
$\leftarrow$	q_6	q_7	q_5	q_3	q_6	q_4
$\leftarrow$	q_7	q_4	q_5	q_3	q_6	q_4

Equivalência

Na presente seção é mostrada a equivalência entre as linguagens caracterizadas pelos conjuntos regulares e as geradas pelas gramáticas lineares à direita, ou seja, prova-se que toda e qualquer linguagem gerada por alguma gramática linear é um conjunto regular, e também, inversamente, que todo e qualquer conjunto regular pode ser expresso através de uma gramática linear.

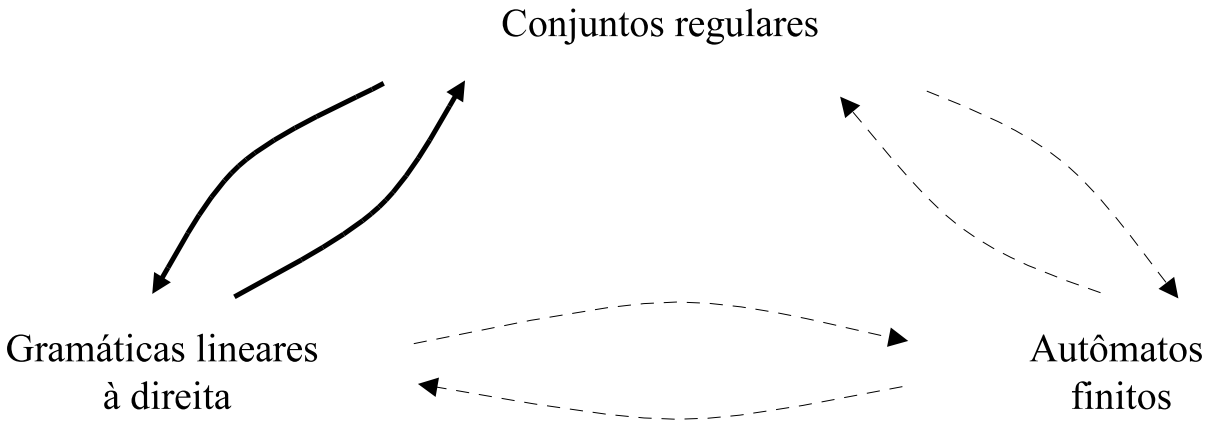


Figura 24: Equivalência dos formalismos — parte 1

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

Teorema 4.1 "Todo conjunto regular é gerado por uma gramática linear à direita."

Deseja-se demonstrar que todo e qualquer conjunto regular define uma linguagem que também pode ser gerada por uma gramática linear à direita.

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

Por definição, $\emptyset, \{\varepsilon\}, \{\sigma\}, \sigma \in \Sigma$, onde Σ é um alfabeto (conjunto finito e não-vazio), são conjuntos regulares. Da mesma forma, $X \cup Y, XY$ e X^*, com X e Y conjuntos regulares, também são conjuntos regulares. A equivalência de tais conjuntos regulares com as correspondentes gramáticas lineares à direita que os geram é mostrada a seguir:

- $\emptyset$ é uma linguagem linear à direita, pois:
  $G = (\{S\} \cup \Sigma, \Sigma, \emptyset, S)$ é tal que $L(G) = \emptyset$
- $\{\varepsilon\}$ é uma linguagem linear à direita, pois:
  $G = (\{S\} \cup \Sigma, \Sigma, \{S \to \varepsilon\}, S)$ é tal que $L(G) = \{\varepsilon\}$
- $\{\sigma\}$ é uma linguagem linear à direita, pois:
  $G = (\{\sigma, S\}, \{\sigma\}, \{S \to \sigma\}, S)$ é tal que $L(G) = \{\sigma\}$

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

Suponha-se agora que X e Y sejam dois conjuntos regulares, gerados por gramáticas lineares à direita. É válido, nesse caso, admitir que X e Y sejam gerados pelas gramáticas G_x e G_y:

$$\begin{aligned} X &= L(G_x), G_x = (\Sigma_x \cup N_x, \Sigma_x, P_x, S_x) \\ Y &= L(G_y), G_y = (\Sigma_y \cup N_y, \Sigma_y, P_y, S_y) \end{aligned}$$

Admita-se ainda, sem perda de generalidade, que $N_x \cap N_y = \emptyset$. Caso isso não seja verdadeiro, podem-se renomear os não-terminais, de modo que essa condição seja satisfeita. Nessa situação, a aplicação das operações de união, concatenação e fechamento reflexivo sobre X e Y geram novas linguagens lineares à direita.

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

- $Z = X \cup Y$ é uma linguagem linear à direita, pois:
  $G_z = (\Sigma_x \cup \Sigma_y \cup N_x \cup N_y \cup \{S_z\}, \Sigma_x \cup \Sigma_y, P_x \cup P_y \cup \{S_z \to S_x, S_z \to S_y\}, S_z)$
  é tal que $L(G_z) = X \cup Y$.

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

- $Z = XY$ é uma linguagem linear à direita, pois:
  $G_z = (\Sigma_x \cup \Sigma_y \cup N_x \cup N_y, \Sigma_x \cup \Sigma_y, P_y \cup P_z, S_x)$, sendo P_z obtido pela aplicação das regras:
  a) Se $A \to \sigma B \in P_x$, então $A \to \sigma B \in P_z$
  b) Se $A \to \sigma \in P_x$, então $A \to \sigma S_y \in P_z$
  c) Se $A \to \varepsilon \in P_x$, então $A \to S_y \in P_z$

  é tal que $L(G_z) = XY$.

Conjuntos regulares $\Rightarrow$ gramáticas lineares à direita

- $Z = X^*$ é uma linguagem linear à direita, pois:
  $G_z = (\Sigma_x \cup N_x \cup \{S_z\}, \Sigma_x, \{S_z \to S_x, S_z \to \varepsilon\} \cup P_z, S_z)$, sendo P_z obtido pela aplicação das regras:
  a) Se $A \to \sigma B \in P_x$, então $A \to \sigma B \in P_z$
  b) Se $A \to \sigma \in P_x$, então $A \to \sigma S_z \in P_z$
  c) Se $A \to \varepsilon \in P_x$, então $A \to S_z \in P_z$

  é tal que $L(G_z) = X^*$.

Exemplo

Exemplo 4.1
Considere-se o alfabeto $\Sigma = \{0,1,2\}$. Os conjuntos regulares $L_0 = \{0\}, L_1 = \{1\}, L_2 = \{2\}$ são definidos, respectivamente, pelas gramáticas:

- $G_0 = (\{0, S_0\}, \{0\}, \{S_0 \to 0\}, S_0)$
- $G_1 = (\{1, S_1\}, \{1\}, \{S_1 \to 1\}, S_1)$
- $G_2 = (\{2, S_2\}, \{2\}, \{S_2 \to 2\}, S_2)$

Continuação

Através da aplicação das regras correspondentes ao fechamento reflexivo e transitivo, obtêm-se, a partir de G_0 e G_1, novas gramáticas G_3 e G_4 representando, respectivamente, os conjuntos $\{0\}^*$ e $\{1\}^*$:

- $G_3 = (\{0, S_0, S_3\}, \{0\}, \{S_3 \to S_0, S_3 \to \varepsilon, S_0 \to 0S_3\}, S_3)$
- $G_4 = (\{1, S_1, S_4\}, \{1\}, \{S_4 \to S_1, S_4 \to \varepsilon, S_1 \to 1S_4\}, S_4)$

Continuação

Da concatenação dos conjuntos $\{0\}^*$ e $\{1\}^*$ resulta o conjunto regular $\{0\}^*\{1\}^*$, representado através da gramática linear à direita G_5:

- $G_5 = (\{0,1,S_0,S_1,S_3,S_4\},\{0,1\},P_5,S_3)$

  $P_5 = \{\underbrace{S_3 \to S_0, S_3 \to S_4, S_0 \to 0S_3}_{G_3}, \underbrace{S_4 \to S_1, S_4 \to \varepsilon, S_1 \to 1S_4}_{G_4}\}$

Continuação

Finalmente, a linguagem obtida pela união de $L(G_5)$ com $L(G_2)$ pode ser representada pela gramática linear G_6:

- $G_6 = (\{0,1,2,S_0,S_1,S_2,S_3,S_4,S_6\},\{0,1,2\},P_6,S_6)$

  $P_6 = \{S_6 \to S_2, S_6 \to S_3, \underbrace{S_2 \to 2}_{G_2},$

  $\underbrace{S_3 \to S_0, S_3 \to S_4, S_0 \to 0S_3, S_4 \to S_1, S_4 \to \varepsilon, S_1 \to 1S_4}_{G_5}\}$

Portanto, $L(G_6) = \{0\}^*\{1\}^* \cup \{2\}$.

Exemplo

Exemplo 4.2
A gramática G_6 do Exemplo 4.1 poderia ser simplificada, mantendo-se a linguagem original e reduzindo-se a quantidade de símbolos não-terminais e o número de produções. Uma possibilidade seria:

- $G_6 = (\{0,1,2,S,A,B\},\{0,1,2\},P_6,S)$
  $P_6 = \{\underbrace{S \to 2}_{\{2\}}, \underbrace{S \to A, A \to 0A}_{\{0\}^*}, \underbrace{A \to B}_{\{0\}^*\{1\}^*}, \underbrace{B \to 1B, B \to \varepsilon}_{\{1\}^*}\}$

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

Teorema 4.2 "Toda gramática linear à direita gera um conjunto regular."

Mostra-se agora a proposição inversa, ou seja, que toda e qualquer linguagem gerada por uma gramática linear à direita constitui um conjunto regular. Para tanto, deve-se lembrar que gramáticas lineares à direita se caracterizam por apresentarem apenas produções com os formatos seguintes:

- $X_i \rightarrow \sigma X_j$
- $X_i \rightarrow X_j$
- $X_i \rightarrow \sigma$
- $X_i \rightarrow \varepsilon$

com $\sigma \in \Sigma$ e $X_i, X_j \in N$.

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

O Algoritmo 4.1, a seguir apresentado, permite a obtenção, de forma sistemática, de uma expressão que representa o conjunto regular definido por uma gramática linear à direita fornecida como entrada. Inicialmente, a representação algébrica da gramática deve ser modificada, para permitir a representação explícita dos conjuntos denotados pelos seus símbolos não-terminais:

- $X_i \rightarrow \sigma X_j$ torna-se $X_i = \{\sigma\}X_j$
- $X_i \rightarrow X_j$ torna-se $X_i = \{\varepsilon\}X_j$
- $X_i \rightarrow \sigma$ torna-se $X_i = \{\sigma\}$
- $X_i \rightarrow \varepsilon$ torna-se $X_i = \{\varepsilon\}$

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

Como resultado, obtém-se um **sistema de equações regulares**, em que cada equação corresponde a uma diferente variável (não-terminal) da gramática original. A solução desse sistema de m variáveis e m equações fornece as expressões que definem os conjuntos regulares representados pelos diversos símbolos não-terminais. Em particular, o conjunto regular associado ao não-terminal raiz da gramática define a linguagem por ela descrita.

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

Nesse sistema, cada equação possui o seguinte formato geral:

$$X_i = A_{i1}X_1 \cup A_{i2}X_2 \cup \ldots \cup A_{im}X_m \cup B_{i1} \cup B_{i2} \cup \ldots B_{ik}$$

onde:

- $A_{ij} = \{\sigma_{ij}\}$ se $X_i \to \sigma_{ij}X_j \in P, \sigma_{ij} \in (\Sigma \cup \{\varepsilon\})$, ou $A_{ij} = \emptyset$ em caso contrário;
- $B_{ij} = \{\sigma_{ij}\}$ se $X_i \to \sigma_{ij} \in P, \sigma_{ij} \in (\Sigma \cup \{\varepsilon\})$.

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

O sistema de equações propriamente dito assume então a forma genérica:

$$\begin{aligned} X_1 &= A_{11}X_1 \cup A_{12}X_2 \cup \ldots \cup A_{1m}X_m \cup B_{11} \cup B_{12} \cup \ldots B_{1p} \\ X_2 &= A_{21}X_1 \cup A_{22}X_2 \cup \ldots \cup A_{2m}X_m \cup B_{21} \cup B_{22} \cup \ldots B_{2q} \\ \ldots &\quad \ldots \\ X_m &= A_{m1}X_1 \cup A_{m2}X_2 \cup \ldots \cup A_{mm}X_m \cup B_{m1} \cup B_{m2} \cup \ldots B_{mr} \end{aligned}$$

Observe-se que cada equação assim obtida pode conter referências a todos os símbolos não-terminais da gramática (X_1 a X_m) bem como a termos constantes (B_{ij}), em quantidade variável conforme a equação.

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

A seguir, coloca-se em evidência o símbolo não-terminal X_i definido em cada equação. Isso, e o agrupamento de todos os termos que não dependem de X_i, pela aplicação da propriedade associativa da união dos conjuntos, altera a representação da equação para:

$$X_i = A_{ii}X_i \cup (A_i \cup B_i)$$

onde:

- $A_i = \cup A_{ij}X_j, 1 \leqslant j \leqslant m, j \neq i$
- $B_i = B_{i1} \cup B_{i2} \cup \ldots B_{ik}$

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

Fazendo-se $C_i = A_i \cup B_i$, são obtidas expressões do tipo:

$$X_i = A_{ii} X_i \cup C_i$$

onde C_i representa todos os termos da i-ésima equação que não contêm referências diretas ao símbolo não-terminal X_i, e A_{ii} representa o conjunto de todas as cadeias sobre Σ que prefixam X_i na equação em que este não-terminal é definido.

Conjuntos regulares $\Leftarrow$ gramáticas lineares à direita

É possível demonstrar, por indução, que equações com o formato genérico $X_i = A_{ii}X_i \cup C_i$ possuem, como solução geral, o conjunto:

$$X_i = A_{ii}^* C_i$$

Nesta expressão, não existem ocorrências do símbolo X_i. De fato, substituindo-se X_i por $A_{ii}^* C_i$ na segunda ocorrência deste símbolo em $X_i = A_{ii}X_i \cup C_i$, obtém-se a identidade:

$$X_i = A_{ii}X_i \cup C_i = A_{ii}(A_{ii}^* C_i) \cup C_i = A_{ii}^+ C_i \cup C_i = A_{ii}^* C_i$$

Solução de $X = AX \cup C$

Teorema 4.3 "A equação regular $X = AX \cup C$ tem como solução geral o conjunto regular $X = A^*C$."

Partindo-se de $X = AX \cup C$, pode-se levantar o conjunto de valores da variável X que satisfazem a essa igualdade. Há dois caminhos possíveis:

1. $X = C$, que é uma solução é trivial, obtida pela simples inspeção de $X = AX \cup C$;

Solução de $X = AX \cup C$

2. $X = AX$, que deve ser assim interpretada: "para obter um novo valor de X, utiliza-se algum valor já conhecido, e concatene-se-lhe um A à esquerda". Aplicando-se essa interpretação à solução obtida em (1), tem-se: $X = AC$, que passa a ser uma nova solução conhecida. Aplicando-se outra vez essa interpretação à solução $X = AC$, obtém-se $X = AAC$, e assim por diante. Note-se que dessa forma foi obtida uma seqüência de soluções progressivamente mais longas, partindo-se da solução trivial, através da concatenação de elementos A à esquerda. Generalizando-se, tem-se a forma $X = A^*C$ como solução geral da equação $X = AX \cup C$.

Resolução de um sistema de equações regulares

Algoritmo 4.1 *"Resolução de um sistema de equações regulares."*

- *Entrada: Uma série de equações regulares sobre variáveis* $X_i, 1 \leqslant i \leqslant m$, *cujos coeficientes* A_{ij} *são conjuntos regulares sobre um alfabeto* Σ.
- *Saída: Uma série de conjuntos regulares* γ_i *sobre* Σ, *de tal forma que* $X_i = \gamma_i$.
- *Método:*
  1. $i \leftarrow 1$;
  2. *Se* $i = m$, *então desviar para (4). Caso contrário, considerar a i-ésima equação na forma* $X_i = A_{ii}X_i \cup C_i$ *e substituir todas as ocorrências de* X_i *por* $A_{ii}^* C_i$ *nas equações referentes às variáveis* $X_j, (i+1) \leqslant j \leqslant m$;
  3. $i \leftarrow i+1$ *e desviar para (2);*
  4. *Se* $i = 0$, *então FIM. Caso contrário,* $X_i \leftarrow \gamma_i = A_{ii}^* C_i$ *e substituir todas as ocorrências de* X_i *pelo conjunto* γ_i *nas equações referentes às variáveis* $X_j, 1 \leqslant j \leqslant (i-1)$.
  5. $i \leftarrow i-1$ *e desviar para (4).*

Exemplo

Exemplo 4.3
Considere-se a gramática linear à direita G_0:

$$\begin{aligned} G_0 &= (\{a,b,c,d,X_0,X_1,X_2\},\{a,b,c,d\},P_0,X_0) \\ P_0 &= \{X_0 \to aX_0, X_0 \to aX_1, X_0 \to b, \\ &\quad X_1 \to bX_1, X_1 \to cX_1, X_1 \to cX_2, X_1 \to d, \\ &\quad X_2 \to aX_0, X_2 \to bX_1, X_2 \to cX_2, X_2 \to c, X_2 \to d\} \end{aligned}$$

Exemplo

Continuação

Convertendo-se as produções algébricas para a notação de conjuntos, obtém-se:

- $X_0 = \{a\}X_0, X_0 = \{a\}X_1, X_0 = \{b\},$
- $X_1 = \{b\}X_1, X_1 = \{c\}X_1, X_1 = \{c\}X_2, X_1 = \{d\},$
- $X_2 = \{a\}X_0, X_2 = \{b\}X_1, X_2 = \{c\}X_2, X_2 = \{c\}, X_2 = \{d\}$

Exemplo

Continuação

Do agrupamento das alternativas de substituição de cada símbolo não-terminal resulta o sistema de equações seguinte:

- $X_0 = \{a\}X_0 \cup \{a\}X_1 \cup \{b\}$
- $X_1 = \{b\}X_1 \cup \{c\}X_1 \cup \{c\}X_2 \cup \{d\}$
- $X_2 = \{a\}X_0 \cup \{b\}X_1 \cup \{c\}X_2 \cup \{c\} \cup \{d\}$

Exemplo

Continuação

Colocando-se em evidência os símbolos não-terminais definidos em cada equação, o novo sistema toma o seguinte aspecto:

- $X_0 = \{a\}X_0 \cup (\{a\}X_1 \cup \{b\}) = A_{00}X_0 \cup C_0$
- $X_1 = \{b,c\}X_1 \cup (\{c\}X_2 \cup \{d\}) = A_{11}X_1 \cup C_1$
- $X_2 = \{c\}X_2 \cup (\{a\}X_0 \cup \{b\}X_1 \cup \{c,d\}) = A_{22}X_2 \cup C_2$

Exemplo

Continuação

com:

- $A_{00} = \{a\}, A_0 = \{a\}X_1, B_0 = \{b\}$
- $A_{11} = \{b,c\}, A_1 = \{c\}X_2, B_1 = \{d\}$
- $A_{22} = \{c\}, A_2 = \{a\}X_0 \cup \{b\}X_1, B_2 = \{c,d\}$

e:

- $C_0 = A_0 \cup B_0 = \{a\}X_1 \cup \{b\}$
- $C_1 = A_1 \cup B_1 = \{c\}X_2 \cup \{d\}$
- $C_2 = A_2 \cup B_2 = (\{a\}X_0 \cup \{b\}X_1) \cup \{c,d\}$

Exemplo

Continuação

Eliminando-se as referências aos símbolos não-terminais $X_i, 0 \leqslant i \leqslant 2$, nas equações em que os mesmos são definidos, resulta:

- $X_0 = \{a\}^*(\{a\}X_1 \cup \{b\})$
- $X_1 = \{b,c\}^*(\{c\}X_2 \cup \{d\})$
- $X_2 = \{c\}^*(\{a\}X_0 \cup \{b\}X_1 \cup \{c,d\})$

Exemplo

Continuação

- Passos 1 e 2 do algoritmo, primeira passagem:

$$
\begin{aligned}
X_0 &= \{a\}^*(\{a\}X_1 \cup \{b\}) \text{ e substituição nas equações de } X_1 \text{ e } X_2 : \\
X_0 &= \{a\}^*(\{a\}X_1 \cup \{b\}) \\
X_1 &= \{b,c\}X_1 \cup (\{c\}X_2 \cup \{d\}) \\
X_2 &= \{c\}X_2 \cup (\{a\}(\{a\}^*(\{a\}X_1 \cup \{b\})) \cup \{b\}X_1 \cup \{c,d\}) \\
&= \{c\}X_2 \cup (\{a\}(\{a\}^*\{a\}X_1 \cup \{a\}^*\{b\}) \cup \{b\}X_1 \cup \{c,d\}) \\
&= \{c\}X_2 \cup (\{a\}\{a\}^*\{a\}X_1 \cup \{a\}\{a\}*\{b\} \cup \{b\}X_1 \cup \{c,d\}) \\
&= \{c\}X_2 \cup \{a\}\{a\}^*\{a\}X_1 \cup \{a\}\{a\}^*\{b\} \cup \{b\}X_1 \cup \{c,d\} \\
&= \{c\}X_2 \cup \{a\}\{a\}^+X_1 \cup \{a\}^+\{b\} \cup \{b\}X_1 \cup \{c,d\} \\
&= \{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})X_1 \cup (\{a\}^+\{b\} \cup \{c,d\})
\end{aligned}
$$

Exemplo

Continuação

- Passos 1 e 2 do algoritmo, segunda passagem:

$$
\begin{aligned}
X_1 &= \{b,c\}^*(\{c\}X_2 \cup \{d\}) \text{ e substituição na equação de } X_2 : \\
X_0 &= \{a\}^*(\{a\}X_1 \cup \{b\}) \\
X_1 &= \{b,c\}^*(\{c\}X_2 \cup \{d\}) \\
X_2 &= \{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})(\{b,c\}^*(\{c\}X_2 \cup \{d\})) \cup \{a\}^+\{b\} \cup \{c,d\}) \\
&= \{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})(\{b,c\}^*\{c\}X_2 \cup \{b,c\}^*\{d\}) \cup \{a\}^+\{b\} \cup \{c,d\} \\
&= \{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})\{b,c\}^*\{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})\{b,c\}^*\{d\} \cup \\
&\quad \{a\}^+\{b\} \cup \{c,d\} \\
&= \{c\}X_2 \cup (\{a\}\{a\}^+ \cup \{b\})\{b,c\}^*\{c\}X_2 \cup \{a\}\{a\}^+\{b,c\}^*\{d\} \cup \\
&\quad \{b\}\{b,c\}^*\{d\} \cup \{a\}^+\{b\} \cup \{c,d\} \\
&= (\{c\} \cup (\{a\}\{a\}^+ \cup \{b\})\{b,c\}^*\{c\})X_2 \cup \{a\}\{a\}^+\{b,c\}^*\{d\} \cup \\
&\quad \{b\}\{b,c\}^*\{d\} \cup \{a\}^+\{b\} \cup \{c,d\}
\end{aligned}
$$

Exemplo

Continuação

- Passos 1 e 2 do algoritmo, terceira passagem:

$$
\begin{aligned}
X_2 &= A_{22}^{*} C_2, \text{ com :} \\
A_{22} &= (\{c\} \cup (\{a\}\{a\}^{+} \cup \{b\})\{b,c\}^{*}\{c\}) \\
C_2 &= \{a\}\{a\}^{+}\{b,c\}^{*}\{d\} \cup \{b\}\{b,c\}^{*}\{d\} \cup \{a\}^{+}\{b\} \cup \{c,d\}
\end{aligned}
$$

Exemplo

Continuação

- Passos 4 e 5 do algoritmo, primeira passagem:

  Substituição de X_2 por $A_{22}^* C_2$ nas equações de X_0 e X_1:

$$\begin{aligned} X_0 &= \{a\}^*(\{a\}X_1 \cup \{b\}) \\ X_1 &= \{b,c\}^*(\{c\}(A_{22}^* C_2) \cup \{d\}) = \{b,c\}^*\{c\}A_{22}^* C_2 \cup \{b,c\}^*\{d\} \\ X_2 &= A_{22}^* C_2 \end{aligned}$$

Exemplo

Continuação

- Passos 4 e 5 do algoritmo, segunda passagem:

  Substituição de X_1 por $\{b,c\}^*\{c\}A_{22}^*C_2 \cup \{b,c\}^*\{d\}$ na equação de X_0:

$$
\begin{aligned}
X_0 &= \{a\}^*(\{a\}(\{b,c\}^*\{c\}A_{22}^*C_2 \cup \{b,c\}^*\{d\}) \cup \{b\}) \\
&= \{a\}^*(\{a\}\{b,c\}^*\{c\}A_{22}^*C_2 \cup \{a\}\{b,c\}^*\{d\} \cup \{b\}) \\
&= \{a\}^*\{a\}\{b,c\}^*\{c\}A_{22}^*C_2 \cup \{a\}^*\{a\}\{b,c\}^*\{d\} \cup \{a\}^*\{b\} \\
X_1 &= \{b,c\}^*\{c\}A_{22}^*C_2 \cup \{b,c\}^*\{d\} \\
X_2 &= A_{22}^*C_2
\end{aligned}
$$

Exemplo

Continuação

Portanto, o conjunto regular definido por G_0 é $\Omega_0\Omega_1 \cup \Omega_3 \cup \Omega_4$, com:

$$
\begin{aligned}
\Omega_0 &= \{a\}^*\{a\}\{b,c\}^*\{c\}(\{c\} \cup (\{a\}\{a\}^+ \cup \{b\})\{b,c\}^*\{c\})^* \\
\Omega_1 &= \{a\}\{a\}^+\{b,c\}^*\{d\} \cup \{b\}\{b,c\}^*\{d\} \cup \{a\}^+\{b\} \cup \{c,d\} \\
\Omega_3 &= \{a\}^*\{a\}\{b,c\}^*\{d\} \\
\Omega_4 &= \{a\}^*\{b\}
\end{aligned}
$$

Exemplo

Exemplo 4.4
Seja G_1 uma gramática linear à direita:

$$\begin{aligned} G_1 &= (\{0,1,2,X_1,X_2,X_3\},\{0,1,2\},P_1,X_1) \\ P_1 &= \{X_1 \to 0X_1, X_1 \to X_2, \\ &\quad X_2 \to 1X_2, X_2 \to 1X_3, \\ &\quad X_3 \to 2X_3, X_3 \to 22\} \end{aligned}$$

Exemplo

Continuação

Efetuando-se a mudança de notação e agrupando-se as alternativas, obtemos o sistema de equações:

$$\begin{aligned} X_1 &= \{0\}X_1 \cup \{\varepsilon\}X_2 \\ X_2 &= \{1\}X_2 \cup \{1\}X_3 \\ X_3 &= \{2\}X_3 \cup \{22\} \end{aligned}$$

Pelo fato de esta gramática não conter referências à variável (não-terminal) X_1 nas equações de X_2 e X_3, nem tampouco referências à variável X_2 na equação da variável X_3, torna-se prático em primeiro lugar resolver diretamente a equação referente à variável X_3:

$$X_3 = \{2\}^*\{22\}$$

Exemplo

Continuação

A seguir, substituem-se pela expressão acima todas as ocorrências da variável X_3 nas equações anteriores. O sistema de equações torna-se então:

$$\begin{aligned} X_1 &= \{0\}X_1 \cup \{\varepsilon\}X_2 \\ X_2 &= \{1\}X_2 \cup \{1\}\{2\}^*\{22\} \\ X_3 &= \{2\}^*\{22\} \end{aligned}$$

Exemplo

Continuação

De forma análoga, obtém-se $X_2 = \{1\}^*\{1\}\{2\}^*\{22\}$ e efetua-se a substituição desta expressão em todas as ocorrências da variável X_2 nas equações anteriores, que então se tornam:

$$\begin{aligned} X_1 &= \{0\}X_1 \cup \{\varepsilon\}\{1\}^*\{1\}\{2\}^*\{22\} \\ X_2 &= \{1\}^*\{1\}\{2\}^*\{22\} \\ X_3 &= \{2\}^*\{22\} \end{aligned}$$

Exemplo

Continuação

Finalmente, obtém-se $X_1 = \{0\}^*\{1\}^*\{1\}\{2\}^*\{22\}$, que representa também o conjunto regular definido pela gramática linear à direita G_1. Resulta:

$$\begin{aligned} X_1 &= \{0\}^*\{1\}^*\{1\}\{2\}^*\{22\} \\ X_2 &= \{1\}^*\{1\}\{2\}^*\{22\} \\ X_3 &= \{2\}^*\{22\} \end{aligned}$$

Equivalência

É mostrada a seguir a equivalência entre as gramáticas lineares à direita e os autômatos finitos, no que diz respeito à classe de linguagens que tais dispositivos são capazes de definir.
As equivalências discutidas nesta seção estão representadas com destaque na Figura 25.

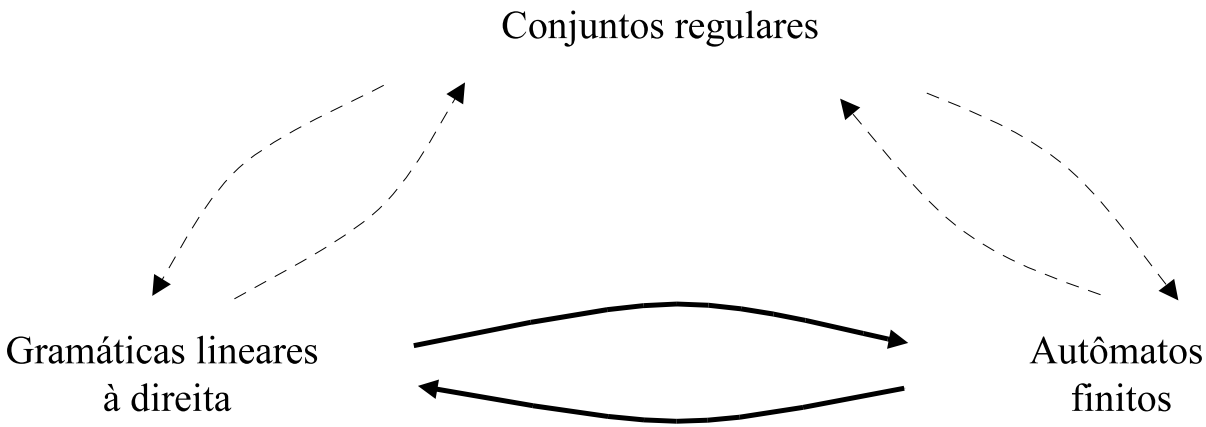


Figura 25: Equivalência dos formalismos — parte 2

Gramáticas lineares à direita $\Rightarrow$ autômatos finitos

Teorema 5.1 "Seja G uma gramática linear à direita. Então é possível definir um autômato finito M de tal modo que $L(G) = L(M)$."

Considere-se $G = (V, \Sigma, P, S)$. As produções de P são do tipo:

1. $X \to aY$
2. $X \to Y$
3. $X \to a$
4. $X \to \varepsilon$

com $X, Y \in N$, $a \in \Sigma$.

Gramáticas lineares à direita $\Rightarrow$ autômatos finitos

Algoritmo 5.1 *"Construção de um autômato finito a partir de uma gramática linear à direita."*

- *Entrada: uma gramática linear à direita G;*
- *Saída: um autômato finito M (eventualmente não-determinístico e com transições em vazio) tal que $L(M) = L(G)$;*
- *Método:*
  1. *Conjunto de estados:*
     Cada estado de M corresponde a um dos símbolos não-terminais de G. A esse conjunto acrescenta-se um novo símbolo (estado) $Z \notin N$, ou seja, $Q = N \cup \{Z\}$. O estado inicial de M é S, a raiz da gramática. O estado final de M é Z, o novo estado acrescentado.
  2. *Alfabeto de entrada:*
     O alfabeto de entrada Σ de M é o mesmo alfabeto Σ de G.

Gramáticas lineares à direita $\Rightarrow$ autômatos finitos

3. *Função de transição:*
   $\delta \leftarrow \emptyset$;
   Para cada regra de produção em P da gramática G, e conforme seu tipo:
   1. *Se* $X \to aY$ *então* $\delta \leftarrow \delta \cup \{(X,a) \to Y\}$;
   2. *Se* $X \to Y$ *então* $\delta \leftarrow \delta \cup \{(X,\varepsilon) \to Y\}$;
   3. *Se* $X \to a$ *então* $\delta \leftarrow \delta \cup \{(X,a) \to Z\}$;
   4. *Se* $X \to \varepsilon$ *então* $\delta \leftarrow \delta \cup \{(X,\varepsilon) \to Z\}$.

Exemplo

Exemplo 5.1
Seja G uma gramática linear à direita:

$$\begin{aligned} G &= (V,\Sigma,P,S) \\ V &= \{a,b,c,S,K,L\} \\ \Sigma &= \{a,b,c\} \\ P &= \{S \to a, S \to aK, K \to bK, K \to L, L \to cL, L \to \varepsilon\} \end{aligned}$$

Aplicando-se o algoritmo de conversão a G, obtém-se o autômato da Tabela 41.

Exemplo

Continuação

Tabela 41: Mapeamento das produções da gramática G para as regras de transição do autômato M

P	δ
$S \to a$	$\delta(S,a) = Z$
$S \to aK$	$\delta(S,a) = K$
$K \to bK$	$\delta(K,b) = K$
$K \to L$	$\delta(K,\varepsilon) = L$
$L \to cL$	$\delta(L,c) = L$
$L \to \varepsilon$	$\delta(L,\varepsilon) = Z$

Exemplo

Continuação

O autômato finito completo M assim especificado é apresentado na Figura 26.
Note-se que $L(G) = L(M) = ab^*c^*$.

$$
\begin{aligned}
M &= (Q, \Sigma, \delta, S, F) \\
Q &= \{S, K, L, Z\} \\
\Sigma &= \{a, b, c\} \\
\delta &= \{(S,a) \to Z, (S,a) \to K, (K,b) \to K, (K,\varepsilon) \to L, (L,c) \to L, (L,\varepsilon) \to Z\} \\
F &= \{Z\}
\end{aligned}
$$

Exemplo

Continuação

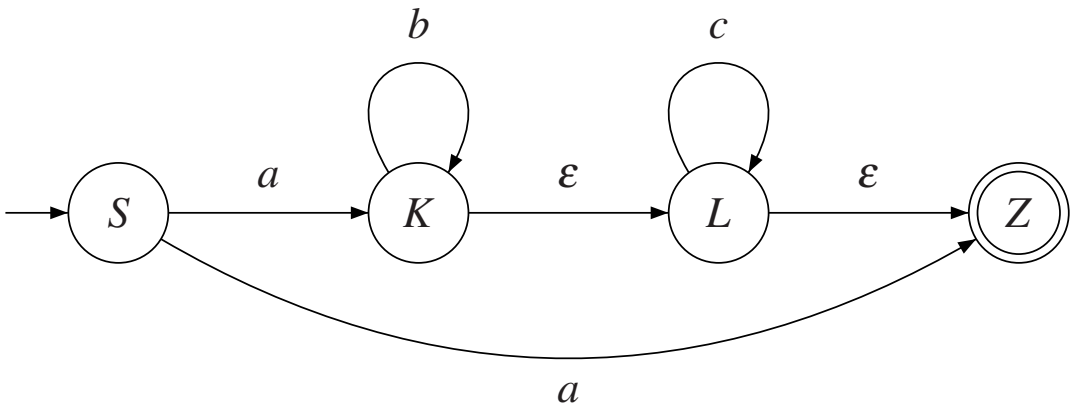


Figura 26: Autômato resultante do mapeamento mostrado na Tabela 41

Exemplo

Continuação

Considere-se a sentença $abbcc$, e as correspondentes seqüências de derivações em G e de movimentações em M:

- $S \Rightarrow aK \Rightarrow abK \Rightarrow abbK \Rightarrow abbL \Rightarrow abbcL \Rightarrow abbccL \Rightarrow abbcc$
- $(S, abbcc) \vdash (K, bbcc) \vdash (K, bcc) \vdash (K, cc) \vdash (L, cc) \vdash (L, c) \vdash (L, \varepsilon) \vdash (Z, \varepsilon)$

Comparando-se as formas sentenciais geradas por G com as configurações assumidas por M, é fácil perceber que há uma relação direta entre elas, expressa na idéia anteriormente exposta de que a linguagem gerada por um certo símbolo não-terminal de G corresponde à linguagem reconhecida pelo respectivo estado de M. Note-se, em particular, que o número de formas sentenciais geradas por G é igual ao número de configurações assumidas por M no reconhecimento da mesma cadeia de entrada (no caso, oito formas sentenciais e oito configurações).

Gramáticas lineares à direita $\Leftarrow$ autômatos finitos

Teorema 5.2 "Seja M um autômato finito qualquer. Então é possível definir uma gramática linear à direita G, de tal modo que $L(M) = L(G)$."

Considere-se $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito genérico, não-determinístico e com transições em vazio. O Algoritmo 5.2 mostra como construir uma gramática linear à direita $G = (V, \Sigma, P, S)$ a partir de M.

Gramáticas lineares à direita $\Leftarrow$ autômatos finitos

Algoritmo 5.2 *"Construção de uma gramática linear à direita a partir de um autômato finito."*

- *Entrada: um autômato finito M;*
- *Saída: uma gramática linear à direita G tal que $L(G) = L(M)$;*
- *Método:*
  1. *Definição do conjunto de símbolos não-terminais:*
     Os símbolos não-terminais de G correspondem aos estados de M. A raiz da gramática é q_0.
  2. *Alfabeto de entrada:*
     O alfabeto Σ de G é o próprio alfabeto de entrada Σ de M.

Gramáticas lineares à direita $\Leftarrow$ autômatos finitos

3. *Produções:*
   $P \leftarrow \emptyset$;
   Para cada elemento de δ *da máquina* M*, e conforme o tipo das transições de* M*:*
   1. *Se* $\delta(q_i, a) = q_j$*, então* $P \leftarrow P \cup \{q_i \rightarrow aq_j\}$;
   2. *Se* $\delta(q_i, \varepsilon) = q_j$*, então* $P \leftarrow P \cup \{q_i \rightarrow q_j\}$.

   Para cada elemento de Q *da máquina* M*:*
   1. *Se* $q_i \in F$*, então* $P \leftarrow \{q_i \rightarrow \varepsilon\}$.

Exemplo

Exemplo 5.2
Seja M representado na Figura 27.

$$
\begin{aligned}
M &= (Q, \Sigma, \delta, q_0, F) \\
Q &= \{q_0, q_1, q_2\} \\
\Sigma &= \{a, b, c\} \\
\delta &= \{(q_0, a) \to q_1, (q_1, b) \to q_1, (q_1, c) \to q_2, (q_1, \varepsilon) \to q_2, (q_2, c) \to q_2\} \\
F &= \{q_2\}
\end{aligned}
$$

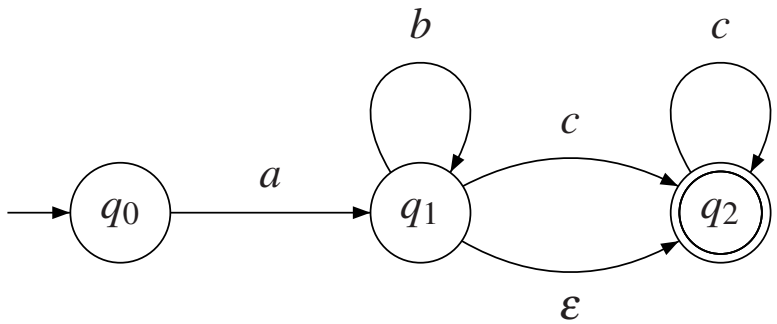


Figura 27: Autômato original M do Exemplo 5.2

Exemplo

Continuação

Aplicando-se o algoritmo de conversão à máquina M, obtém-se a gramática linear à direita G apresentada na Tabela 42, cujo conjunto de produções P corresponde à segunda coluna da mesma. Note que $L(M) = L(G) = ab^*c^*$.

$$
\begin{aligned}
G &= (V, \Sigma, P, q_0) \\
V &= \{a, b, c, q_0, q_1, q_2\} \\
\Sigma &= \{a, b, c\} \\
P &= \{q_0 \to aq_1, \\
&\quad q_1 \to bq_1, \\
&\quad q_1 \to cq_2, \\
&\quad q_1 \to q_2, \\
&\quad q_2 \to cq_2, \\
&\quad q_2 \to \varepsilon\}
\end{aligned}
$$

Exemplo

Continuação

Tabela 42: Gramática G equivalente ao autômato M da Figura 27

δ	P
$\delta(q_0, a) = q_1$	$q_0 \to aq_1$
$\delta(q_1, b) = q_1$	$q_1 \to bq_1$
$\delta(q_1, c) = q_2$	$q_1 \to cq_2$
$\delta(q_1, \varepsilon) = q_2$	$q_1 \to q_2$
$\delta(q_2, c) = q_2$	$q_2 \to cq_2$
Q	P
$q_2 \in F$	$q_2 \to \varepsilon$

Exemplo

Continuação

Considere-se a cadeia $abbbc$, e as correspondentes seqüências de movimentações em M e de derivações em G:

- $(q_0, abbbc) \vdash (q_1, bbbc) \vdash (q_1, bbc) \vdash (q_1, bc) \vdash (q_1, c) \vdash (q_2, \varepsilon)$
- $q_0 \Rightarrow aq_1 \Rightarrow abq_1 \Rightarrow abbq_1 \Rightarrow abbbq_1 \Rightarrow abbbcq_2 \Rightarrow abbbc$

Exemplo

Continuação

Em particular, a cadeia *abbbc* possui mais de uma seqüência de movimentações que conduzem à sua aceitação em M. Tal fato implica a existência de uma outra seqüência de derivações que é capaz de gerar essa cadeia em G, como pode ser percebido abaixo:

- $(q_0, abbbc) \vdash (q_1, bbbc) \vdash (q_1, bbc) \vdash (q_1, bc) \vdash (q_1, c) \vdash (q_2, c) \vdash (q_2, \varepsilon)$
- $q_0 \Rightarrow aq_1 \Rightarrow abq_1 \Rightarrow abbq_1 \Rightarrow abbbq_1 \Rightarrow abbbq_2 \Rightarrow abbbcq_2 \Rightarrow abbbc$

Equivalência

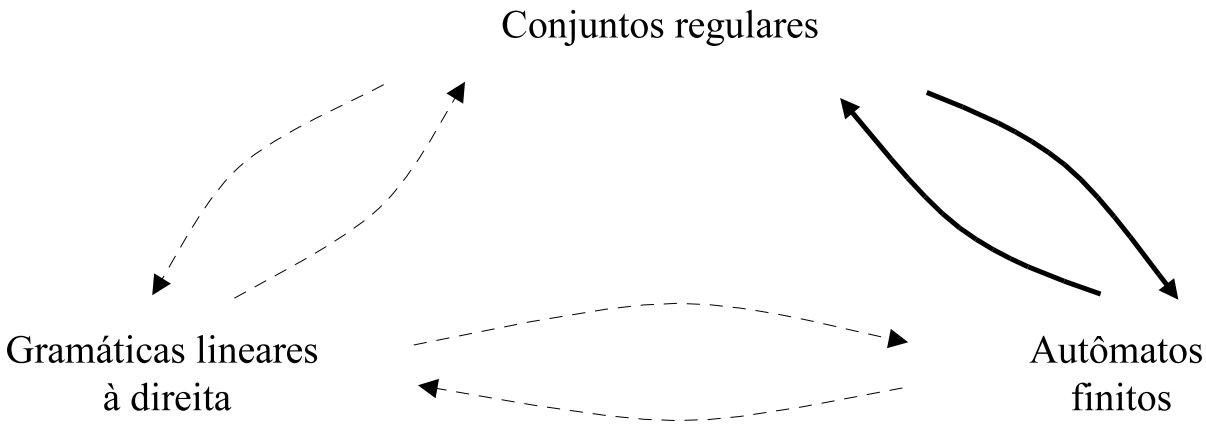


Figura 28: Equivalência dos formalismos — parte 3

Expressão regular $\Rightarrow$ autômato finito

Teorema 6.1 "Seja r uma expressão regular sobre o alfabeto Σ. Então existe um autômato finito M que aceita a linguagem definida por r."

O autômato finito não-determinístico que aceita a linguagem definida por r pode ser obtido através da aplicação do Algoritmo 6.1, que especifica as regras de mapeamento parciais que abrangem casos triviais de sentenças (itens 1, 2 e 3) e cada um dos operadores de união (4), concatenação (5) e fechamento (6), conforme a própria definição das expressões regulares.

Expressão regular $\Rightarrow$ autômato finito

Algoritmo 6.1 *"Obtenção de um autômato finito a partir de uma expressão regular."*

- *Entrada: uma expressão regular r sobre um alfabeto Σ;*
- *Saída: um autômato finito M tal que $L(M) = r$;*
- *Método:*

Expressão regular $\Rightarrow$ autômato finito

$r = \varepsilon$

r é aceita por M representado na Figura 29.

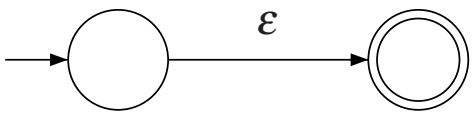


Figura 29: Autômato M que aceita ε

Expressão regular $\Rightarrow$ autômato finito

$r = \emptyset$

r é aceita por M representado na Figura 30.

Figura 30: Autômato M que aceita $\emptyset$

Expressão regular $\Rightarrow$ autômato finito

$r = \sigma, \sigma \in \Sigma$

r é aceita por M representado na Figura 31.

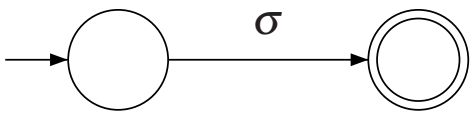


Figura 31: Autômato M que aceita σ

Expressão regular $\Rightarrow$ autômato finito

$r = r_1 \mid r_2$

r é aceita por M representado na Figura 32.

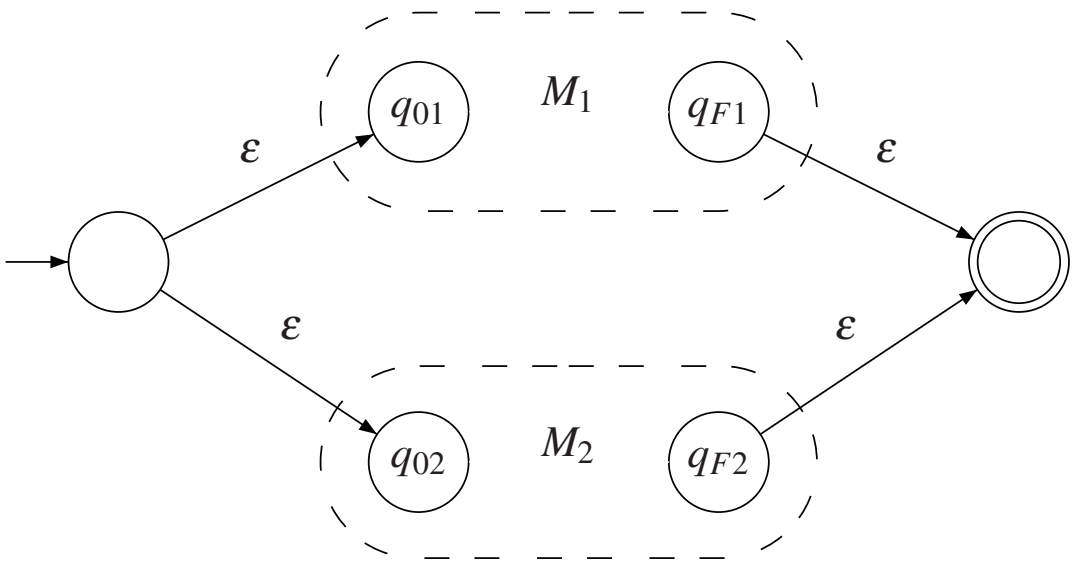


Figura 32: Autômato M que aceita $r_1 \mid r_2$

Expressão regular $\Rightarrow$ autômato finito

$r = r_1 r_2$

r é aceita por M representado na Figura 33.

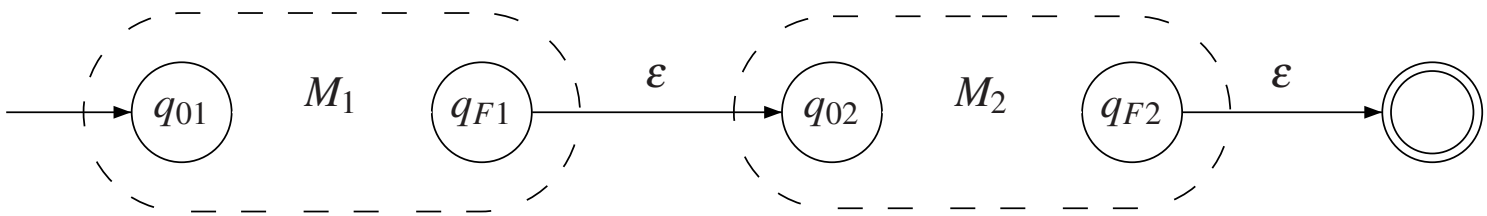


Figura 33: Autômato M que aceita $r_1 r_2$

Expressão regular $\Rightarrow$ autômato finito

$r = r_1^*$

r é aceita por M representado na Figura 34.

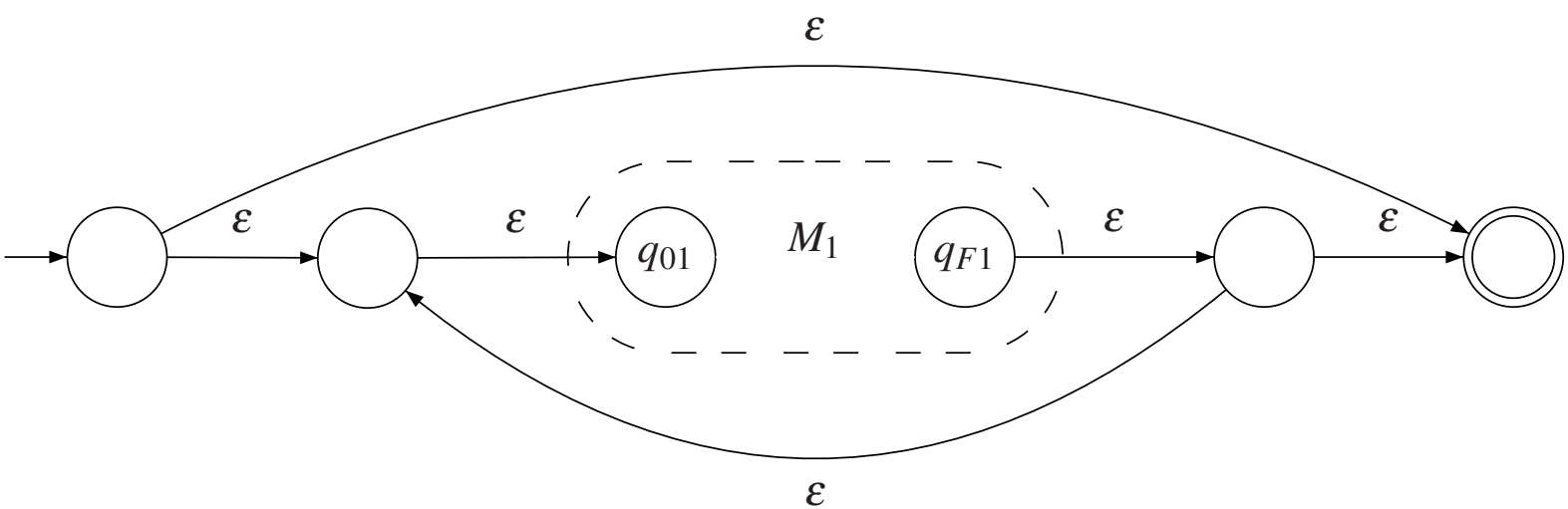


Figura 34: Autômato M que aceita r_1^*

Exemplo

Exemplo 6.1
Considere-se a expressão regular $ab^* \mid c$. É possível identificar, nessa expressão, as seguintes linguagens triviais: $L_1 = a, L_2 = b, L_3 = c$. Portanto, $L_1 = L_1(M_1)$, com M_1 representado na Figura 35.

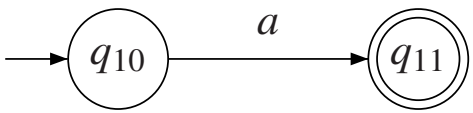


Figura 35: Autômato que aceita a

Exemplo

$L_2 = L_2(M_2)$, com M_2 representado na Figura 36.

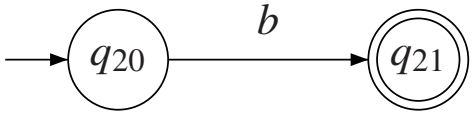


Figura 36: Autômato que aceita b

$L_3 = L_3(M_3)$, com M_3 representado na Figura 37.

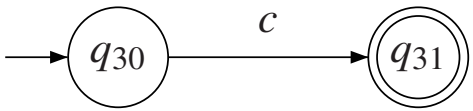


Figura 37: Autômato que aceita c

Exemplo

Seja $L_4 = b^* = L_2^*$. Então, $L_4 = L_4(M_4)$, com M_4 representado na Figura 38.

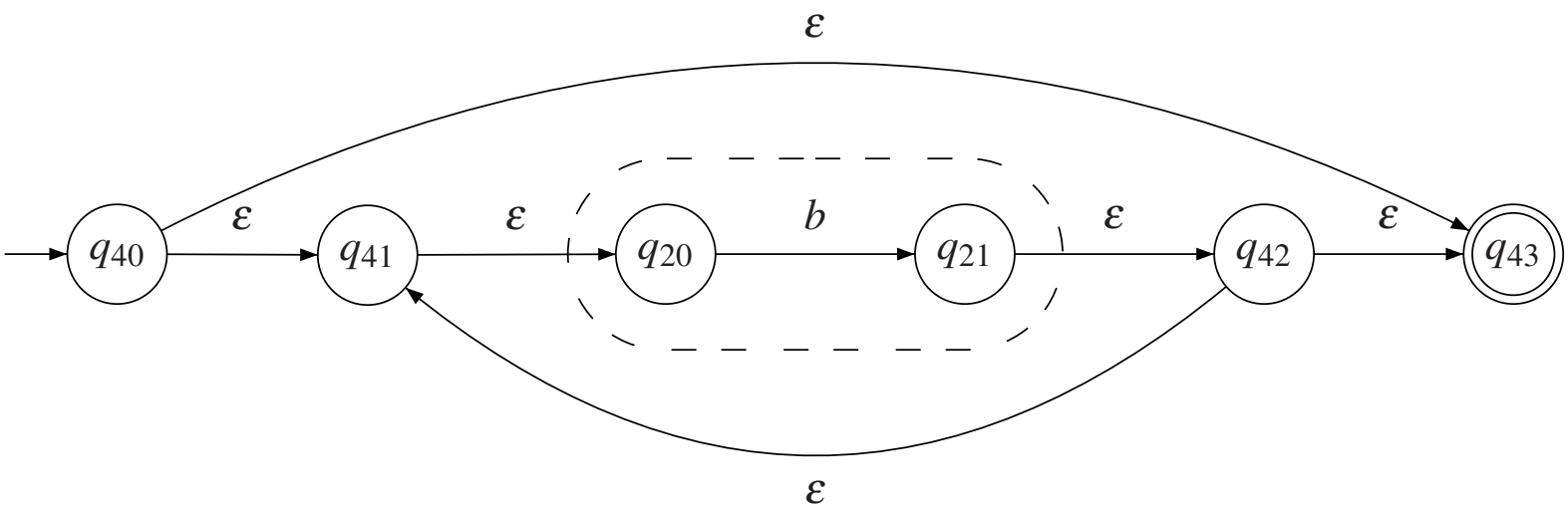


Figura 38: Autômato que aceita b^*

Exemplo

$L_5 = ab^* = L_1 L_4$. Então, $L_5 = L_5(M_5)$, com M_5 representado na Figura 39.

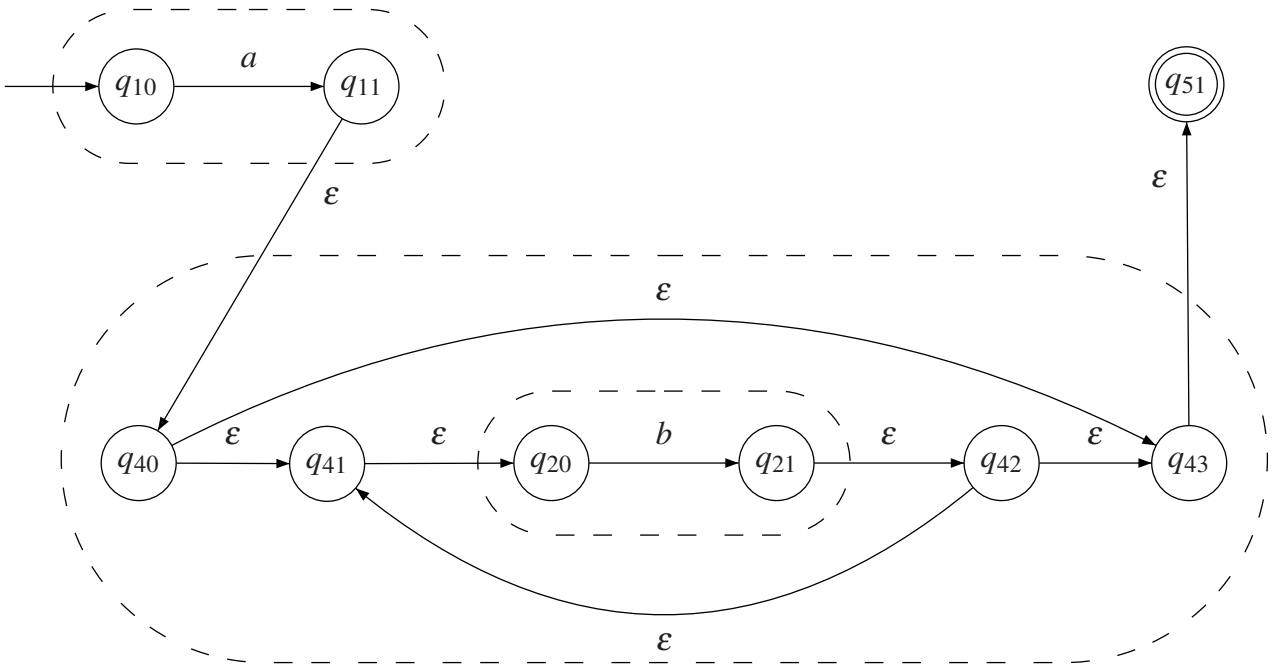


Figura 39: Autômato que aceita ab^*

Exemplo

Finalmente, $L_6 = ab^* \mid c = L_5 \cup L_3 = L_6(M_6)$, com M_6 representado na Figura 40.

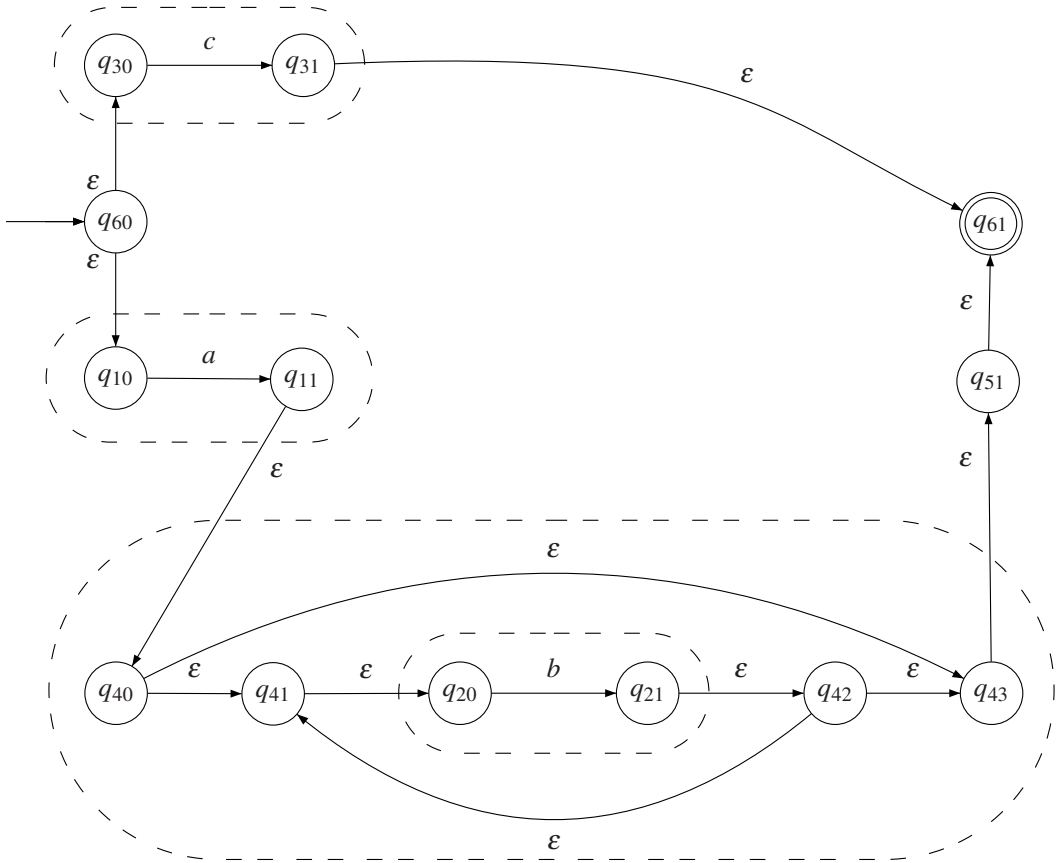


Figura 40: Autômato que aceita $ab^* \mid c$

Exemplo

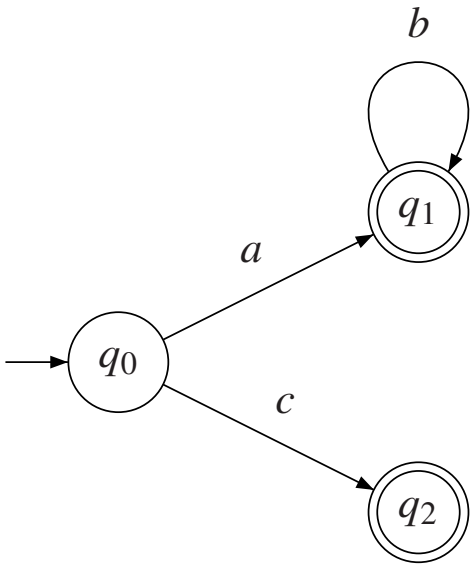


Figura 41: Outro autômato que aceita $ab^* \mid c$

Grafo de expressões

Um **grafo de expressões** é uma máquina de estados na qual as transições entre os estados são rotuladas por expressões regulares construídas sobre um mesmo alfabeto Σ. Formalmente:

$$N = (Q, \Sigma, \delta', q_0, F)$$

em que:

- Q, Σ, q_0 e F são definidos como no caso dos autômatos finitos;
- $\delta' : Q \times ER_\Sigma \to 2^Q$, onde ER_Σ representa o conjunto de todas as expressões regulares que podem ser definidas sobre o alfabeto Σ.

Note-se que todo autômato finito é, por definição, um caso particular de um grafo de expressões em que as expressões regulares entre os estados são reduzidas aos casos elementares σ, ε e $\emptyset$.

Grafo de expressões

De forma similar ao caso do autômato finito, define-se a linguagem aceita por um grafo de expressões como sendo aquela que é gerada pela união de todas as expressões regulares que podem ser obtidas pela concatenação das expressões regulares que rotulam algum caminho entre o estado inicial e algum estado final do grafo.
A obtenção sistemática de uma expressão regular que gera linguagem aceita por um autômato finito pode ser feita através de um método que elimina, um a um, todos os estados do autômato que não sejam o inicial ou o final. Como decorrência da eliminação desses estados, as transições originais são substituídas por expressões regulares que preservam a linguagem aceita pelo dispositivo, e o correspondente autômato torna-se, finalmente, um grafo de expressões com apenas dois estados.

Autômato finito $\Rightarrow$ expressão regular

Teorema 6.2 "Seja M um autômato finito. Então existe uma expressão regular r que gera a linguagem aceita por M."

Autômato finito ⇒ expressão regular

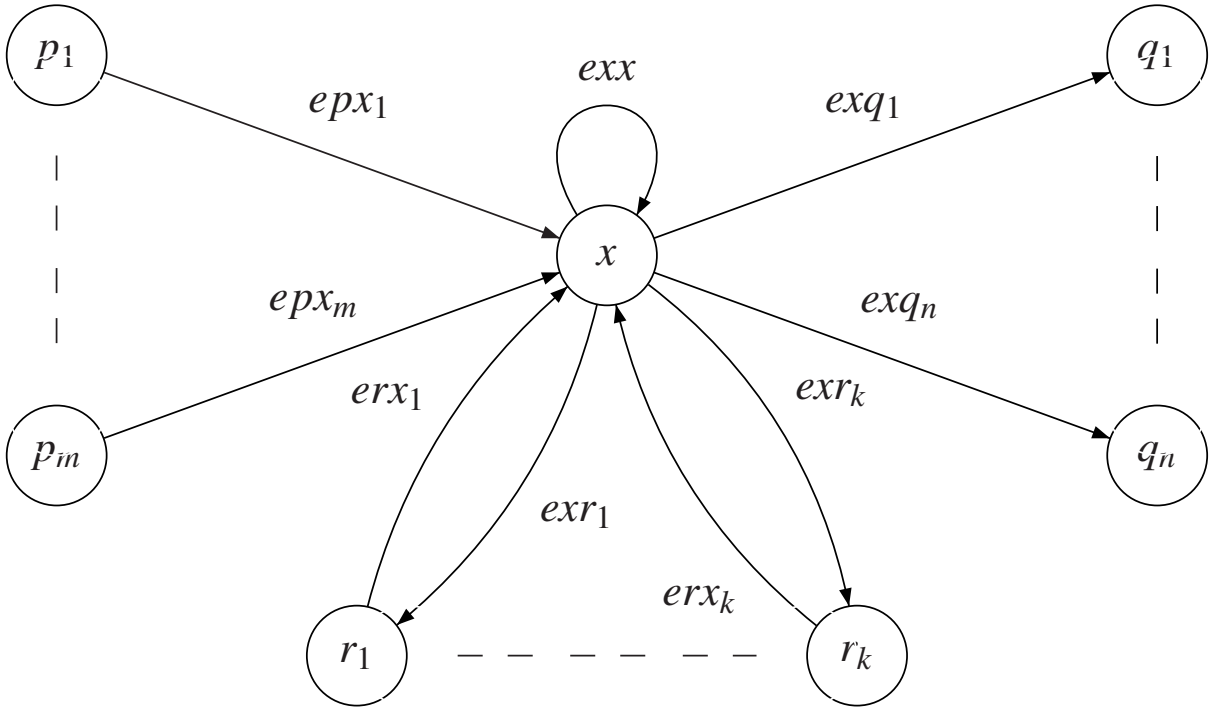


Figura 42: Contextualização de um x estado a ser eliminado

Autômato finito $\Rightarrow$ expressão regular

Considera-se, genericamente, que o estado x possui m estados que são apenas predecessores (identificados por p_1 até p_m), n estados que são apenas sucessores (q_1 até q_n) e k estados que são simultaneamente predecessores e sucessores. O estado x possui, portanto:

- $m+k$ estados predecessores;
- $n+k$ estados sucessores.

Autômato finito $\Rightarrow$ expressão regular

Algoritmo 6.2 *"Obtenção de uma expressão regular a partir de um autômato finito."*

- *Entrada: um autômato finito* $M = (Q, \Sigma, \delta, q_0, F)$;
- *Saída: uma expressão regular* r *tal que* $r = L(M)$;
- *Método:*

Autômato finito $\Rightarrow$ expressão regular

1. *Se* $|F| > 1$, *obter* $M' = (Q', \Sigma, \delta', q_0, F')$ *tal que* $L(M') = L(M)$ *e* $|F'| = 1$:
   1. $Q' \leftarrow Q \cup \{q_F\}$;
   2. $\delta' \leftarrow \delta \cup \{(q_i, \varepsilon) \rightarrow q_F) \mid q_i \in F\}$;
   3. $F' \leftarrow \{q_F\}$;

   Caso contrário, renomear o estado final como q_F *e considerar* $M' = M$.

Autômato finito $\Rightarrow$ expressão regular

Para cada um dos estados $x \in (Q' - \{q_0, q_F\})$, *fazer:*

1. *Identificar os estados predecessores e sucessores associados ao estado* x, *bem como as respectivas transições associadas, da forma como foi apresentado anteriormente, na Figura 42.*
2. *Para* $1 \leqslant i \leqslant m$, *fazer:*
   1. *Para* $1 \leqslant j \leqslant n$, *fazer* $\delta' \leftarrow \delta' \cup \{(p_i, epx_iexx^*exq_j) \rightarrow q_j\}$;
   2. *Para* $1 \leqslant j \leqslant k$, *fazer* $\delta' \leftarrow \delta' \cup \{(p_i, epx_iexx^*exr_j) \rightarrow r_j\}$;
3. *Para* $1 \leqslant i \leqslant k$, *fazer:*
   1. *Para* $1 \leqslant j \leqslant n$, *fazer* $\delta' \leftarrow \delta' \cup \{(r_i, erx_iexx^*exq_j) \rightarrow q_j\}$;
   2. *Para* $1 \leqslant j \leqslant k$, *fazer* $\delta' \leftarrow \delta' \cup \{(r_i, erx_iexx^*exr_j) \rightarrow r_j\}$;
4. $\forall q \in Q'$ *e* $r \in$ *conjunto das expressões regulares sobre* Σ*:*
   1. $\delta' \leftarrow \delta' - \{(q, r) \rightarrow x\}$;
   2. $\delta' \leftarrow \delta' - \{(x, r) \rightarrow q\}$;
   3. $Q' \leftarrow Q' - \{x\}$.

Autômato finito $\Rightarrow$ expressão regular

O processo de eliminação de estados prossegue até que todos os estados, exceto o inicial (q_0) e o final (q_F), tenham sido eliminados. Nessas condições, o autômato resultante terá apenas dois estados e no máximo quatro transições, conforme a figura 43.

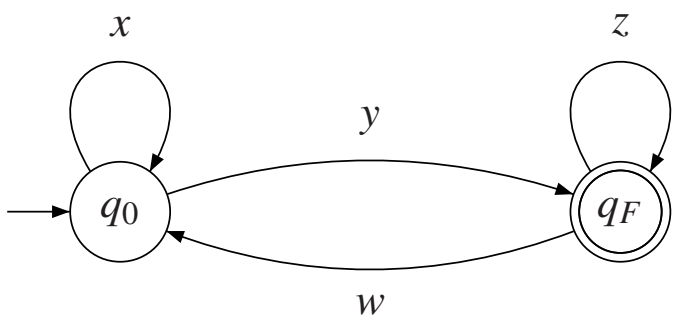


Figura 43: Autômato com dois estados que aceita a linguagem $x^*yz^*(wx^*yz^*)^*$

Exemplo

Exemplo 6.2
O autômato da Figura 44 reconhece a linguagem:

$$L = \{w \in \{a,b,c\}^* \mid w \text{ não contém a subcadeia } abc\}$$

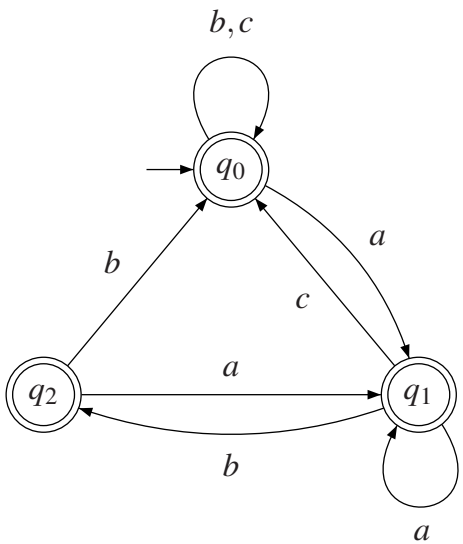


Figura 44: Autômato com múltiplos estados finais

Exemplo

A construção de uma expressão regular que gera a linguagem aceita por M inicia-se com a obtenção de um autômato equivalente, com um único estado final, conforme o Algoritmo 6.2. O resultado é apresentado na Figura 45.

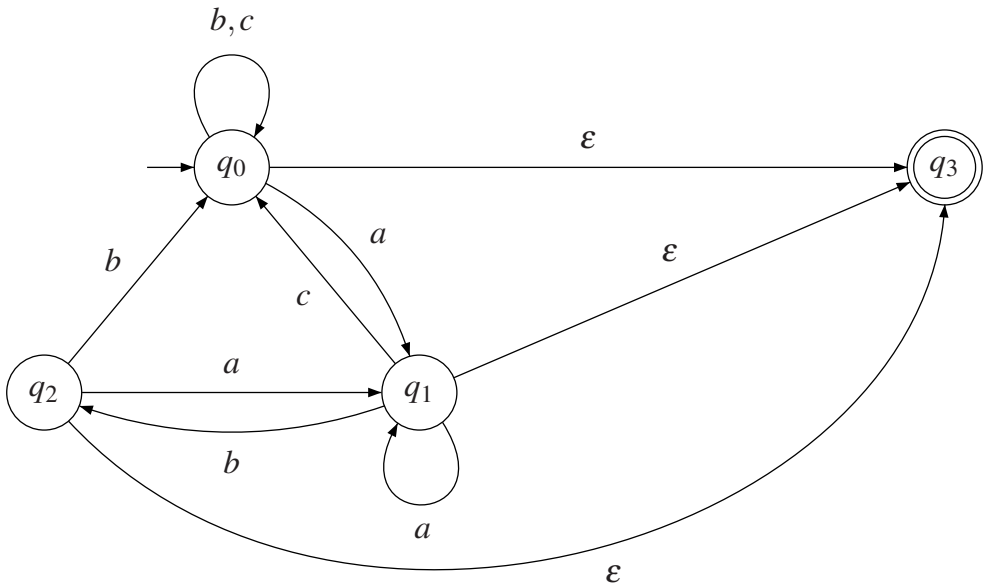


Figura 45: Autômato equivalente com um único estado final

Exemplo

A eliminação do estado q_2 resulta no grafo de expressões da Figura 46.

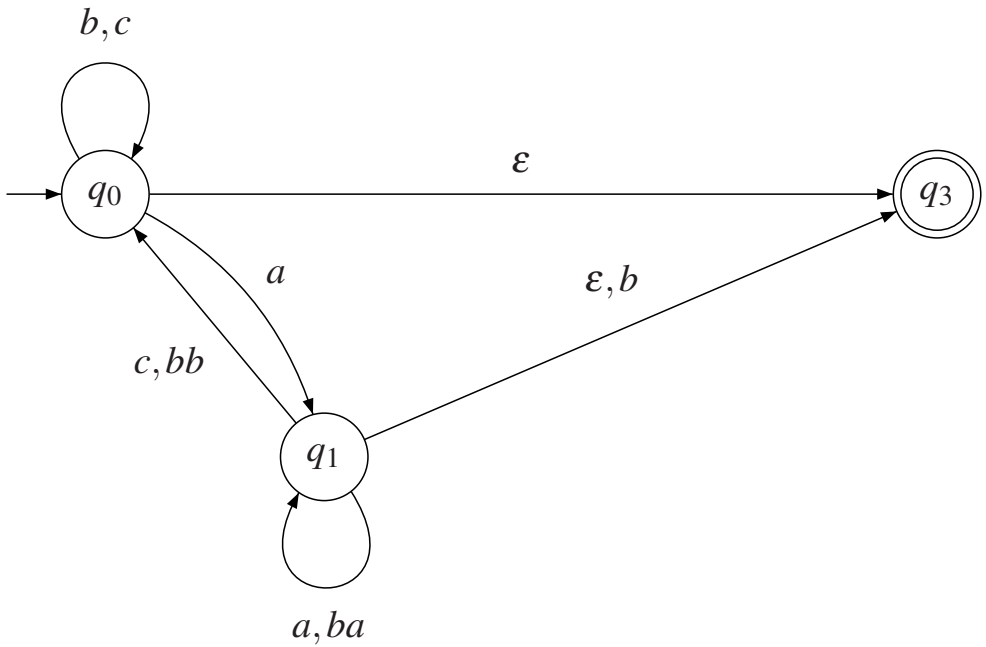


Figura 46: Grafo de expressões equivalente, após a eliminação do estado q_2

Exemplo

A eliminação do estado q_1 resulta no grafo de expressões da Figura 47.

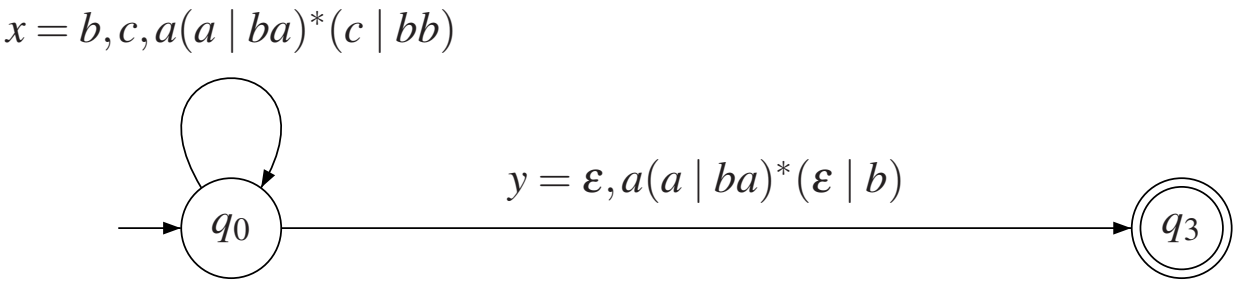


Figura 47: Grafo de expressões equivalente, após a eliminação dos estados q_2 e q_1

Exemplo

A análise da Figura 47 permite, finalmente, inferir a expressão regular que representa a linguagem aceita pelo autômato da Figura 44:

$$\Big(\ \underbrace{b \mid c \mid a(a \mid ba)^*(c \mid bb)}_{\text{Referente à transição } \delta(q_0,x)=q_0}\ \Big)^*\Big(\ \underbrace{\varepsilon \mid a(a \mid ba)^*(\varepsilon \mid b)}_{\text{Referente à transição } \delta(q_0,y)=q_3}\ \Big)$$

Exemplo

As Figuras 48 e 49 representam, respectivamente, os grafos de expressão intermediários correspondentes caso a ordem de eliminação dos estados seja invertida (isto é, primeiro o estado q_1 e depois o estado q_2).

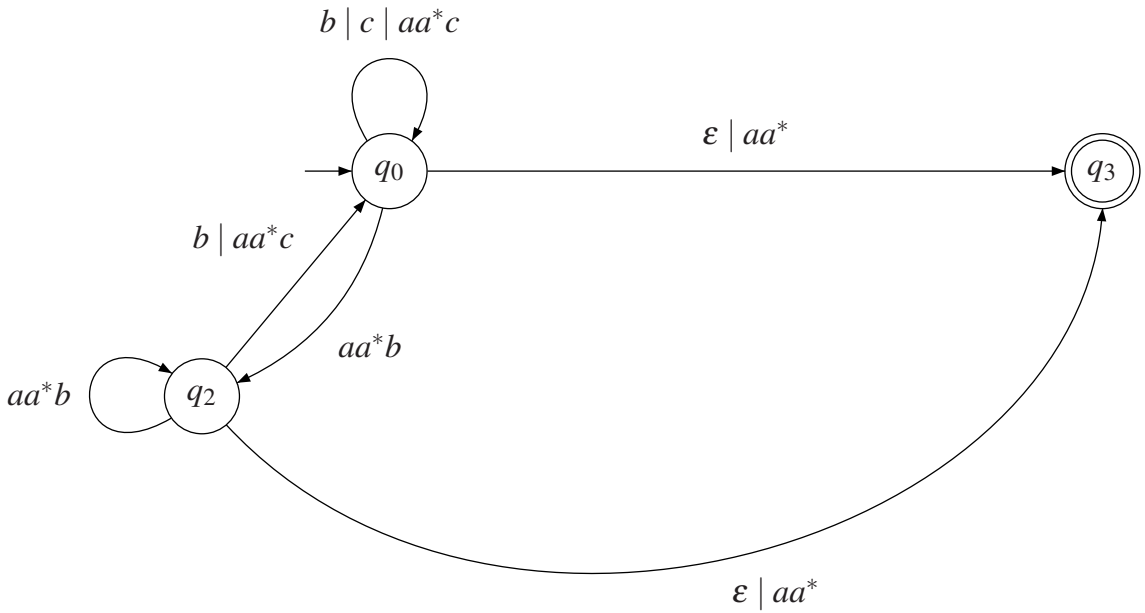


Figura 48: Grafo de expressões equivalente, após a eliminação do estado q_1

Exemplo

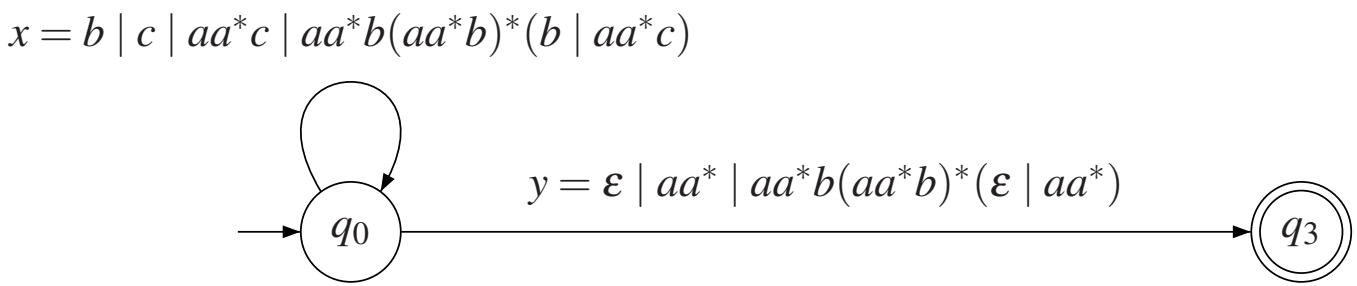


Figura 49: Grafo de expressões equivalente, após a eliminação dos estados q_1 e q_2

O resultado obtido é uma expressão regular equivalente porém diferente da anterior:

$$\underbrace{(b \mid c \mid aa^*c \mid aa^*b(aa^*b)^*(b \mid aa^*c))}_{\text{Referente à transição } \delta(q_0,x)=q_0}{}^*\,(\,\underbrace{\varepsilon \mid aa^* \mid aa^*b(aa^*b)^*(\varepsilon \mid aa^*)}_{\text{Referente à transição } \delta(q_0,y)=q_3}\,)$$

Exemplo

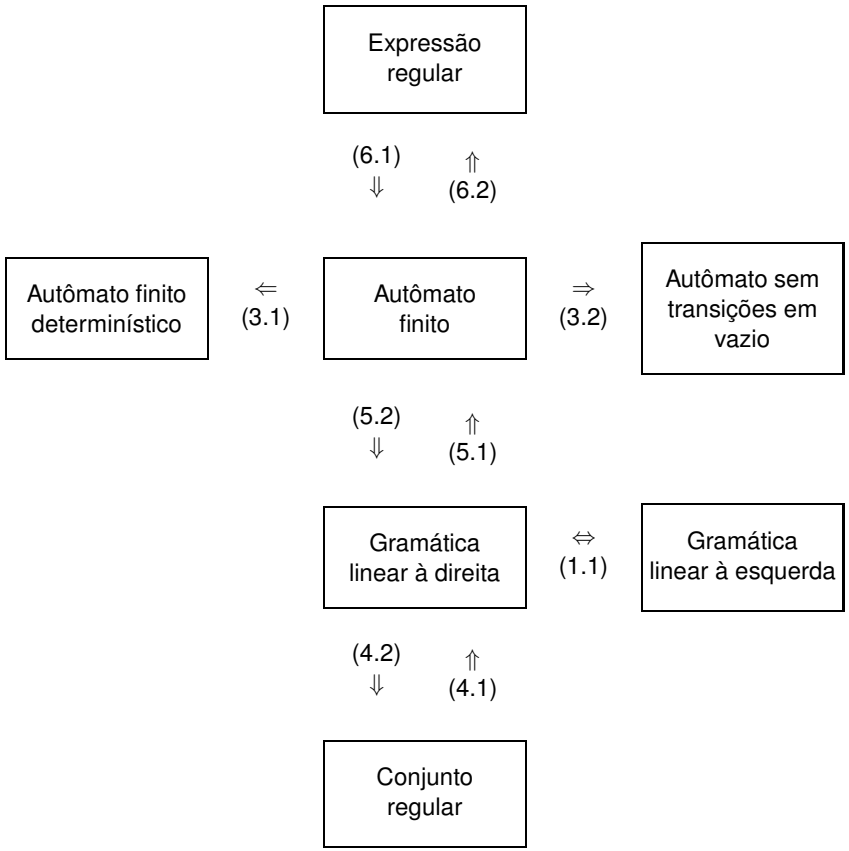


Figura 50: Visão geral da equivalência dos formalismos e respectivos teoremas

Fatos

- Cada conjunto regular é reconhecido por um autômato finito mínimo e único;
- O termo **mínimo** é empregado para designar um autômato finito que tenha o número mínimo possível de estados;
- Esse resultado é válido apenas para a classe das linguagens definidas por autômatos finitos;
- Importância prática;
- Existe um algoritmo que é capaz de transformar qualquer autômato finito em uma versão equivalente mínima;
- O autômato finito mínimo é **único** para cada linguagem regular.

Estados equivalentes

Dois estados A e B de um autômato finito são ditos equivalentes se o conjunto de cadeias aceitas em cada um deles for o mesmo. Considere-se a **linguagem aceita a partir de um estado** X como sendo definida da seguinte forma:

$$L(X) = \{w \in \Sigma^* \mid (X, w) \vdash^* (q_F, \varepsilon), q_F \in F\}$$

Logo, é fácil perceber que $A \equiv B$ se e somente se $L(A) = L(B)$. Esse resultado pode simplificar a verificação da equivalência de estados para os quais a determinação da linguagem aceita em cada um deles seja uma tarefa simples de ser feita (por inspeção visual ou pela aplicação de um método qualquer).

Estados equivalentes

Exemplo 7.1
Considere-se o autômato da Figura 51.

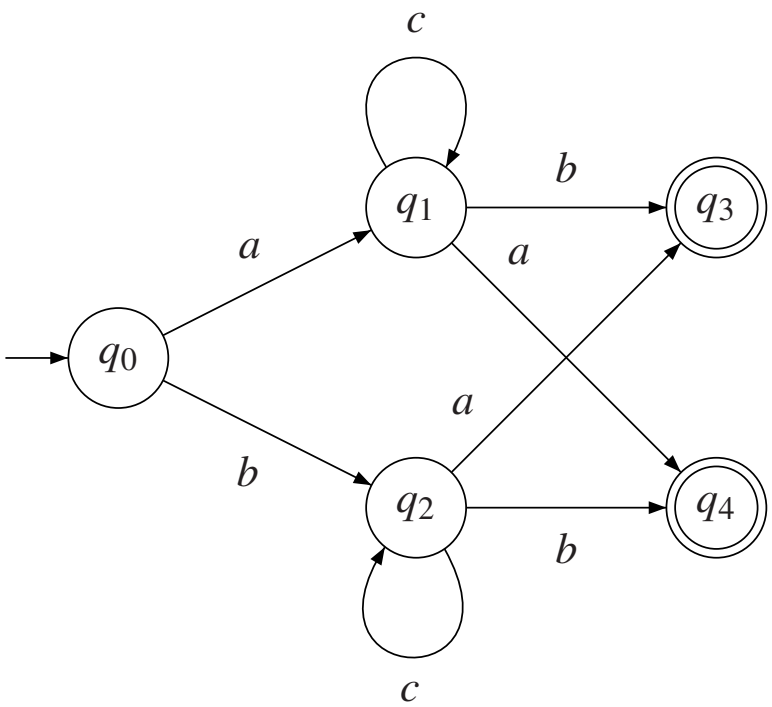


Figura 51: Autômato original do Exemplo 7.1

Exemplo

Uma rápida inspeção visual permite concluir que:

- $L(q_0) = (a \mid b)c^*(a \mid b)$
- $L(q_1) = c^*(a \mid b)$
- $L(q_2) = c^*(a \mid b)$
- $L(q_3) = \varepsilon$
- $L(q_4) = \varepsilon$

Exemplo

Portanto, como $L(q_1) = L(q_2)$ e $L(q_3) = L(q_4)$, então $q_1 \equiv q_2$ e $q_3 \equiv q_4$, e a versão mínima corresponde à apresentada na Figura 52.

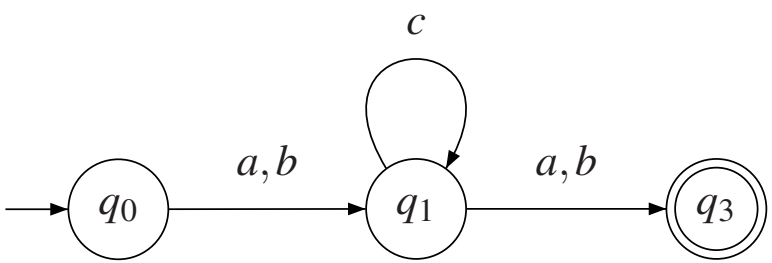


Figura 52: Autômato mínimo equivalente ao da Figura 51

Método

O método apresentado a seguir parte da hipótese de que o autômato a ser minimizado é determinístico, isento de transições em vazio, isento de estados inacessíveis e possui função de transição total. Dado um autômato finito qualquer, a aplicação dos algoritmos apresentados anteriormente permite a sua transformação em autômatos equivalentes com essas características. A minimização do número de estados de um autômato finito é feita em duas etapas:

1. Eliminação de não-determinismos, transições em vazio e estados inacessíveis. Tornar a função de transição total;
2. Agrupamento e fusão de estados equivalentes.

Método

O presente método opera em dois passos. No primeiro, eliminam-se do autômato as transições em vazio, os não-determinismos e os estados inacessíveis. A função de transição é tornada total. No segundo passo, criam-se classes de equivalência com base no critério da coincidência do conjunto de entradas aceitas pelos possíveis pares de estados considerados.

Método

O método descrito pode ser enunciado na forma do Algoritmo 7.1, uma alternativa simples e prática para a minimização de autômatos finitos. Como o algoritmo é baseado na análise exaustiva de todos os possíveis pares de estados de um autômato M, torna-se conveniente representar tais pares na forma de uma matriz, considerada apenas da diagonal principal (inclusive) para cima, uma vez que, para efeito de análise da equivalência de estados, o par (q_i, q_j) e o par (q_j, q_i), com $i \neq j$, são idênticos.
Seja, portanto, M um autômato finito com $n+1$ estados. A Tabela 43 mostra uma forma de representar todos os possíveis pares de estados de M, sem repetição de pares e sem repetição de estados dentro de um mesmo par.

Representação

Tabela 43: Representação dos pares de estados de um autômato M com $n+1$ estados

	q_1	q_2	...	q_{n-1}	q_n
q_0	(q_0, q_1)	(q_0, q_2)	...	(q_0, q_{n-1})	(q_0, q_n)
q_1		(q_1, q_2)	...	(q_1, q_{n-1})	(q_1, q_n)
...				...	...
q_{n-2}				(q_{n-2}, q_{n-1})	(q_{n-2}, q_n)
q_{n-1}					(q_{n-1}, q_n)

Notação

A notação $(q_i, q_j) \xrightarrow{\sigma} (q_m, q_n)$ é usada para indicar que as duas seguintes condições são simultaneamente verificadas por M:

1. $\delta(q_i, \sigma) = q_m$, e
2. $\delta(q_j, \sigma) = q_n$.

Algoritmo

Algoritmo 7.1 *"Método prático para a minimização do número de estados de um autômato finito."*

- *Entrada: Um autômato finito M sem transições em vazio, determinístico, com função de transição total, isento de estados inacessíveis e cujos pares de estados estão representados conforme a Tabela 43.*
- *Saída: Uma partição do conjunto de estados Q de M, correspondente às maiores classes de equivalência encontradas em M;*

Algoritmo

- *Marcar, na tabela, todos os pares do tipo* $(q_a, q_b), q_a \in F, q_b \in (Q - F)$, *ou vice-versa, como não-equivalentes ("*$\not\equiv$*");*
- *Para cada um dos pares de estados restantes* (q_a, q_b) *(escolhidos arbitrariamente), fazer:*
  - *Se para toda entrada* σ *aceita por* q_a *e* q_b*:*
    - *Se* $\delta(q_a, \sigma) = \delta(q_b, \sigma)$*, ou*
    - *Se* $\delta(q_a, \sigma) \neq \delta(q_b, \sigma)$*, mas* $\delta(q_a, \sigma)$ *e* $\delta(q_b, \sigma)$ *forem equivalentes.*

  Então marcar o par (q_a, q_b)*, na tabela, como equivalente ("*$\equiv$*"); caso contrário, marcar o par como não-equivalente ("*$\not\equiv$*"); em seguida, deve-se verificar se existem pares cuja relação de equivalência esteja na dependência do resultado obtido e, em caso afirmativo, marcar os respectivos pares na tabela de forma correspondente;*
  Caso não seja possível concluir pela equivalência (ou não) de um par de estados, prosseguir com a análise de outros pares, deixando o par corrente na dependência dos resultados que forem obtidos para os demais pares;

Algoritmo

- *Marcar os pares restantes, se houver, como equivalentes ("$\equiv$");*
- *A inspeção dos pares marcados indica as classes de equivalência obtidas.*

Exemplo

Exemplo 7.2
Considere-se o autômato finito determinístico, sem transições em vazio, sem estados inacessíveis e com a função de transição total da Tabela 44.

Tabela 44: Autômato original do Exemplo 7.2

	δ	a	b
$\rightarrow$	q_0	q_1	q_6
	q_1	q_2	q_3
$\leftarrow$	q_2	q_2	q_3
	q_3	q_4	q_2
$\leftarrow$	q_4	q_2	q_3
	q_6	q_4	q_4

Exemplo

A Tabela 45 representa todos os possíveis pares de estados desse autômato, e também indica a partição inicial de seu conjunto de estados (finais x não-finais).

Tabela 45: Partição inicial dos estados do autômato da Tabela 44

	q_1	q_2	q_3	q_4	q_6
q_0		$\not\equiv$		$\not\equiv$	
q_1	-	$\not\equiv$		$\not\equiv$	
q_2	-	-	$\not\equiv$		$\not\equiv$
q_3	-	-	-	$\not\equiv$	
q_4	-	-	-	-	$\not\equiv$

Exemplo

Passa-se, então, a considerar cada um dos pares não marcados dessa tabela (escolhidos arbitrariamente).

- $(q_0, q_1) \xrightarrow{a} (q_1, q_2) \not\equiv$
  Como q_1 e q_2 não são equivalentes (ver Tabela 45), marca-se o par (q_0, q_1) como "$\not\equiv$" e torna-se desnecessária a análise das transições desses estados com a entrada b.
- $(q_0, q_3) \xrightarrow{a} (q_1, q_4) \not\equiv$
  Similar ao item acima. O par (q_0, q_3) é marcado como "$\not\equiv$".
- $(q_1, q_3) \xrightarrow{a} (q_2, q_4)$?
  $(q_1, q_3) \xrightarrow{b} (q_3, q_2) \not\equiv$
  Apesar de ainda não se dispor de nenhuma informação sobre o par (q_2, q_4), o par (q_3, q_2) já foi determinado como sendo não-equivalente (ver tabela 45). Logo, marca-se o par (q_1, q_3) como "$\not\equiv$".

Exemplo

- $(q_0, q_6) \xrightarrow{a} (q_1, q_4) \not\equiv$
  Como q_1 e q_4 não são equivalentes (ver tabela 45), marca-se o par (q_0, q_6) como "$\not\equiv$" e torna-se desnecessária a análise das transições desses estados com a entrada b.
- $(q_1, q_6) \xrightarrow{a} (q_2, q_4)$?
  $(q_1, q_6) \xrightarrow{b} (q_3, q_4) \not\equiv$
  Apesar de ainda não se dispor de nenhuma informação sobre o par (q_2, q_4), o par (q_3, q_4) já foi determinado como sendo não-equivalente (ver tabela 45). Logo, marca-se o par (q_1, q_6) como "$\not\equiv$".
- $(q_3, q_6) \xrightarrow{a} (q_4, q_4) \equiv$
  $(q_3, q_6) \xrightarrow{b} (q_2, q_4)$?
  Neste caso, q_3 e q_6 transitam para o mesmo estado q_4 com a entrada a. Por outro lado, ainda não se dispõe de nenhuma informação sobre o par (q_2, q_4). Assim, a equivalência do par (q_3, q_6) fica condicionada à verificação da equivalência do par (q_2, q_4). O par (q_3, q_6) não recebe nenhuma marcação neste momento.

Exemplo

- $(q_2, q_4) \xrightarrow{a} (q_2, q_2) \equiv$
  $(q_2, q_4) \xrightarrow{b} (q_3, q_3) \equiv$
  Os estados q_2 e q_4 transitam com as mesmas entradas para estados idênticos (com a entrada a para q_2 e com a entrada b para q_3). Logo, esses estados são equivalentes e o par recebe a marcação "$\equiv$" na tabela. Além disso, conclui-se que o par (q_3, q_6) (ver item acima) é equivalente, e o mesmo deve ser marcado como "$\equiv$".

Exemplo

Ao término do algoritmo, a Tabela 46 resume o resultado da análise.

Tabela 46: Resultado final da análise da equivalência de estados para o autômato da Tabela 44

	q_1	q_2	q_3	q_4	q_6
q_0	$\not\equiv$	$\not\equiv$	$\not\equiv$	$\not\equiv$	$\not\equiv$
q_1	-	$\not\equiv$	$\not\equiv$	$\not\equiv$	$\not\equiv$
q_2	-	-	$\not\equiv$	$\equiv$	$\not\equiv$
q_3	-	-	-	$\not\equiv$	$\equiv$
q_4	-	-	-	-	$\not\equiv$

Exemplo

As classes de equivalência desse autômato são, portanto, $\{q_0\}, \{q_1\}, \{q_2, q_4\}$ e $\{q_3, q_6\}$. O autômato resultante (ver Tabela 47) possui quatro estados, denominados respectivamente $[q_0]$, $[q_1]$, $[q_2, q_4]$, e $[q_3, q_6]$, e corresponde à versão mínima do autômato da Tabela 44.

Tabela 47: Autômato mínimo equivalente ao da Tabela 44

	δ'	a	b
$\rightarrow$	$[q_0]$	$[q_1]$	$[q_3, q_6]$
	$[q_1]$	$[q_2, q_4]$	$[q_3, q_6]$
$\leftarrow$	$[q_2, q_4]$	$[q_2, q_4]$	$[q_3, q_6]$
	$[q_3, q_6]$	$[q_2, q_4]$	$[q_2, q_4]$

Função total

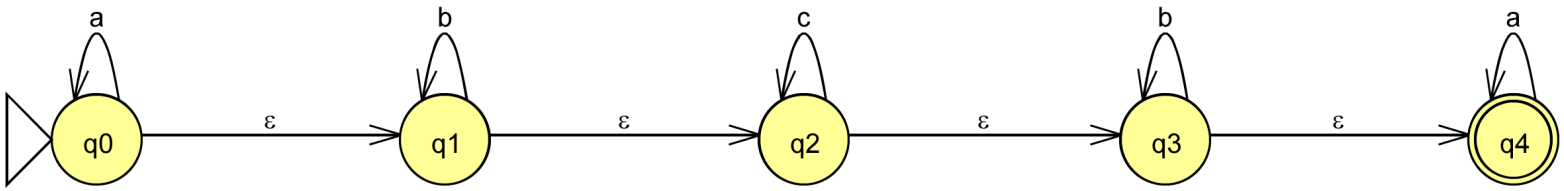
Note-se, no Algoritmo 7.1, a exigência de que o autômato a ser minimizado possua função de transição δ total. Tal exigência visa unicamente a garantir que todos os pares de estados, quaisquer que sejam os estados considerados, possam sempre ser comparados em relação a todas as entradas.
No entanto, ao tornar total a função de transição de algum autômato cuja função de transição seja parcial, isso normalmente implica a incorporação de um estado inútil ao autômato, o qual acaba sendo agrupado com outros estados inúteis eventualmente existentes no autômato original e preservado na versão mínima correspondente.

Função total

Assim, o autômato mínimo obtido é tal que a função de transição é total e, além disso, é isento de estados equivalentes. Eventualmente, a eliminação de estados inúteis, em um autômato minimizado de acordo com o Algoritmo 7.1, poderá resultar em um autômato com um número ainda menor de estados, porém cuja função de transição seja parcial. Isso não significa que o autômato inicialmente obtido não seja o mínimo, mas apenas que ele é o autômato mínimo com função de transição total e sem estados equivalentes.
A diferença, portanto, entre um autômato mínimo obtido pela aplicação do Algoritmo 7.1 e um autômato mínimo equivalente, isento de estados inúteis, é de no máximo um estado.

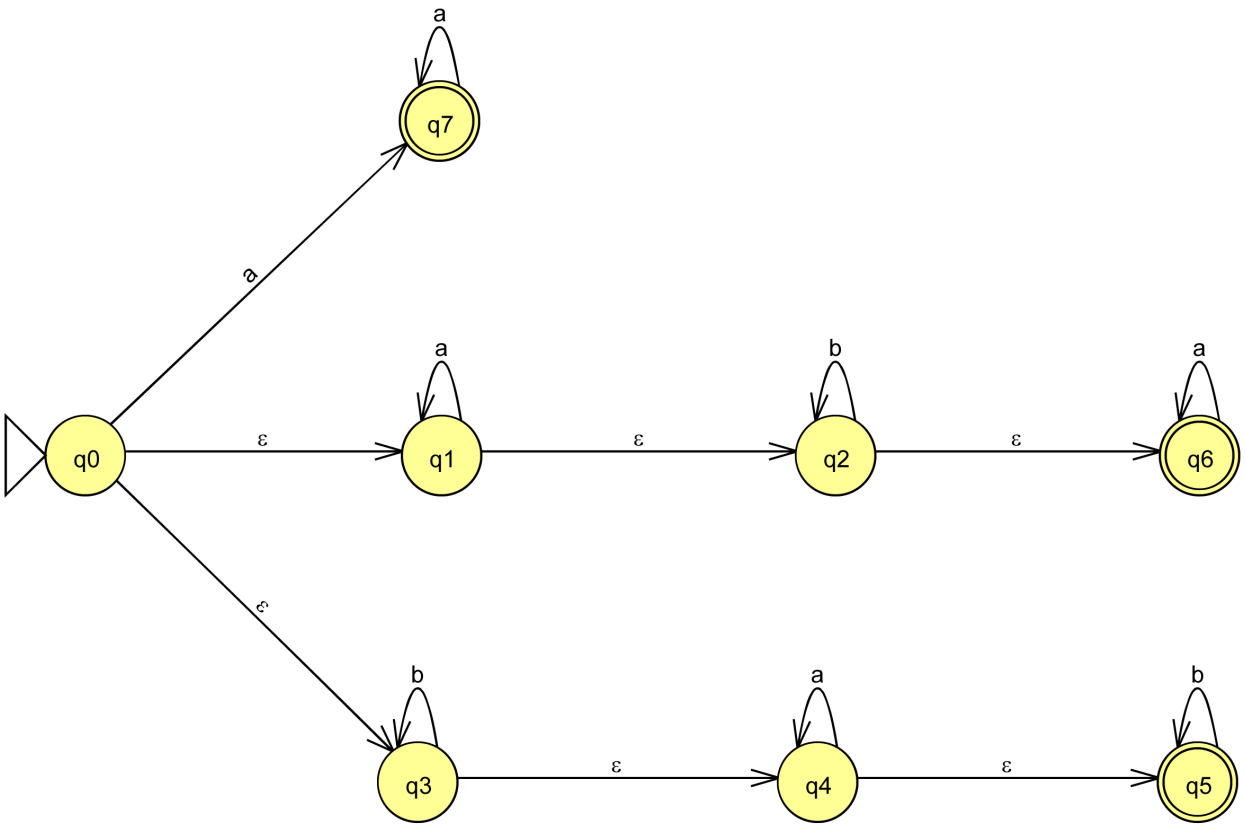
Exercício 1

Obter um autômato finito mínimo que seja equivalente ao autômato:



Exercício 2

Obter um autômato finito mínimo que seja equivalente ao autômato:



Conclusões

Dada uma linguagem regular L qualquer, então:

1. Existe um autômato finito mínimo que aceita L. Em outras palavras, não existe nenhum outro autômato, com um número inferior de estados, que aceite L;
2. O autômato finito mínimo que aceita L é único. Isso significa que não existem dois autômatos finitos com o mesmo número de estados, porém com funções de transição distintas, que aceitam a linguagem L.

Conclusões

A existência e unicidade de um autômato finito mínimo para toda e qualquer linguagem regular L permite, entre outros resultados, estabelecer critérios para determinar se um conjunto de linguagens regulares representa a mesma linguagem: basta obter as versões mínimas dos autômatos finitos que reconhecem cada uma dessas linguagens, e verificar se são todos iguais. Em caso afirmativo, as linguagens são todas idênticas. Caso contrário, não são todas idênticas.

Conceito

- Extensões que associam, a cada sentença de entrada, uma correspondente cadeia de saída sobre um segundo alfabeto, eventualmente distinto do alfabeto de entrada;
- A associação de símbolos de um alfabeto de saída com a movimentação do autômato finito no reconhecimento de uma sentença pode ser feita de duas formas distintas: a partir da seqüência de estados percorridos ou das transições de que se compõe o autômato finito no qual se baseia o transdutor em questão;
- O primeiro caso caracteriza as denominadas Máquinas de Moore, e o segundo as chamadas Máquinas de Mealy.

Máquina de Moore

Formalização

Um transdutor finito do tipo **Máquina de Moore** é definido como sendo uma sétupla $T_{Moore} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ sobre um autômato finito $M = (Q, \Sigma, \delta, q_0, F)$, em que Δ é o **alfabeto de saída** do transdutor e $\lambda : Q \rightarrow \Delta$ é a **função de transdução** de T.

Máquina de Moore

Exemplo

Exemplo 8.1
Seja T um transdutor finito do tipo Máquina de Moore, com $\lambda : Q \to \Delta^*$:

$$
\begin{aligned}
T &= (Q, \Sigma, \Delta, \delta, \lambda, q_0, F) \\
Q &= \{q_0, q_1\} \\
\Sigma &= \{a, b, c\} \\
\Delta &= \{1\} \\
\delta &= \{(q_0, a) \to q_1, (q_1, b) \to q_1, (q_1, c) \to q_0\} \\
\lambda &= \{q_0 \to 1, q_1 \to \varepsilon\} \\
F &= \{q_1\}
\end{aligned}
$$

Máquina de Moore

Exemplo

O grafo correspondente a esse transdutor é apresentado na Figura 53.

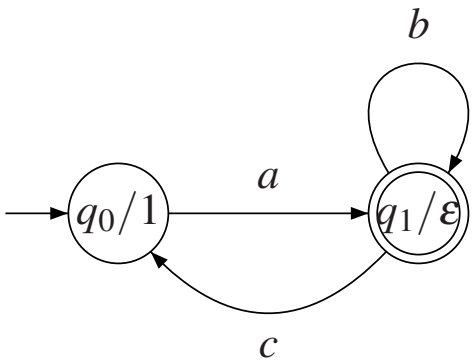


Figura 53: Transdutor do tipo Máquina de Moore

Máquina de Moore

Exemplo

A linguagem aceita por esse transdutor é $ab^*(cab^*)^*$, ou seja, uma seqüência de uma ou mais cadeias ab^* separadas pelo símbolo c. A função de transição λ, neste caso, faz com que o transdutor emita o símbolo "1" toda vez que estiver iniciando o reconhecimento de uma nova cadeia com o formato ab^*. Assim, T funciona como um contador do número de subcadeias ab^* presentes na cadeia de entrada. Como exemplo, a Tabela 48 apresenta um conjunto de cadeias que são respectivamente aceitas e geradas por T.

Máquina de Moore

Exemplo

Tabela 48: Sentenças aceitas e cadeias geradas pelo transdutor do tipo Máquina de Moore T

Sentença aceita	Cadeia Gerada
abbcabbbcab	111
abbbcab	11
acacaca	1111
a	1

Máquina de Mealy

Formalização

Um transdutor finito do tipo **Máquina de Mealy**, por sua vez, é definido como sendo uma sétupla $T_{Mealy} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ sobre um autômato finito $M = (Q, \Sigma, \delta, q_0, F)$, em que Δ é o **alfabeto de saída** do transdutor e $\lambda : Q \times \Sigma \rightarrow \Delta$ é a **função de transdução** de T.
No caso das Máquinas de Mealy, associam-se os símbolos do alfabeto de saída às transições, e não aos estados, como ocorre com as Máquinas de Moore (o domínio da função λ se altera para $Q \times \Sigma$).

Máquina de Mealy

Exemplo

Exemplo 8.2
Seja T um transdutor finito do tipo Máquina de Mealy:

$$
\begin{aligned}
T &= (Q, \Sigma, \Delta, \delta, \lambda, q_0, F) \\
Q &= \{q_0, q_1\} \\
\Sigma &= \{a, b, c\} \\
\Delta &= \{a, b, c\} \\
\delta &= \{(q_0, a) \to q_1, (q_1, b) \to q_1, (q_1, c) \to q_0\} \\
\lambda &= \{(q_0, a) \to ab, (q_1, b) \to \varepsilon, (q_1, c) \to c\} \\
F &= \{q_1\}
\end{aligned}
$$

Máquina de Mealy

Exemplo

O grafo correspondente é apresentado na Figura 54.

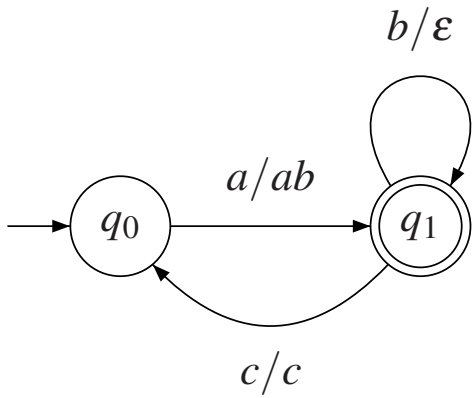


Figura 54: Transdutor do tipo Máquina de Mealy

Como se pode notar, o autômato finito que serve de base para esse transdutor é o mesmo do exemplo anterior. Assim, a linguagem aceita por ambos os transdutores é a mesma. A Tabela 49 apresenta alguns exemplos de cadeias respectivamente aceitas e geradas por T.

Máquina de Mealy

Exemplo

Tabela 49: Sentenças aceitas e cadeias geradas pelo transdutor do tipo Máquina de Mealy T

Sentença aceita	Cadeia Gerada
$abbcabbbcab$	$abcabcab$
$abbbcab$	$abcab$
$acacaca$	$abcabcabcab$
a	ab

Equivalência

Apesar de se tratar de dois modelos distintos de transdutores finitos, pode-se demonstrar a plena equivalência de ambos: toda e qualquer Máquina de Moore pode ser simulada por uma Máquina de Mealy e vice-versa. Dessa maneira, portanto, a opção por um ou outro tipo de máquina pode ser feita levando-se em conta exclusivamente a conveniência de manipulação e a facilidade de representação obtidas conforme o caso em questão.

Equivalência

Exemplo

Exemplo 8.3
Considere-se a linguagem $L_1 = xx^*(-xx^*)^*$, definida sobre o alfabeto $\{x, -\}$. Considere-se também a linguagem L_2, definida sobre o alfabeto de saída $\{x, y, \#\}$, de tal forma que as cadeias de L_2 reproduzem na saída as cadeias de L_1, com as seguintes modificações:

- As subcadeias de entrada xx^* que contiverem três ou menos símbolos x devem ser reproduzidas de forma idêntica na saída (com um, dois ou três símbolos x);
- As subcadeias de entrada xx^* que contiverem quatro ou mais símbolos x devem ser reproduzidas na saída como $xxxy$;
- Todos os símbolos "-" da cadeia entrada devem ser substituídos pelo símbolo "#" na cadeia de saída.

Equivalência

Exemplo

A Tabela 50 apresenta exemplos de cadeias de entrada e correspondentes cadeias de saída.

Tabela 50: Sentenças aceitas e cadeias geradas pelos transdutores do Exemplo 8.3

Sentença aceita	Cadeia Gerada
$x - x$	$x\#x$
$xxx - xxxx$	$xxx\#xxxy$
$xxxxxx - xxx - xx$	$xxxy\#xxx\#xx$
$x - xx - xxx - xxxx - xxxxx$	$x\#xx\#xxx\#xxxy\#xxxy$

Equivalência

Exemplo

Os transdutores finitos das Figuras 55 e 56 — respectivamente Máquina de Mealy e Máquina de Moore — são equivalentes, pois possuem autômatos subjacentes que reconhecem a mesma linguagem L_1 (apesar de serem diferentes) e geram a mesma linguagem L_2, conforme as especificações acima.

Equivalência

Exemplo

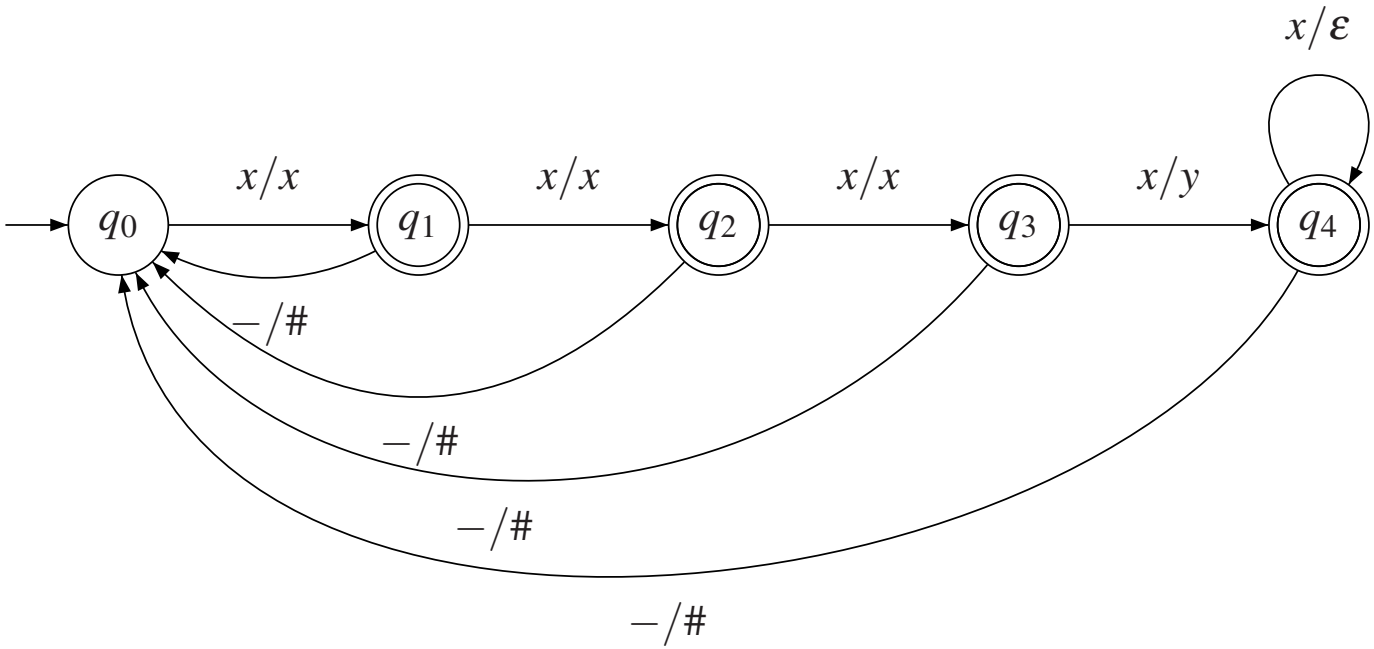


Figura 55: Transdutor do tipo Máquina de Mealy do Exemplo 8.3

Equivalência

Exemplo

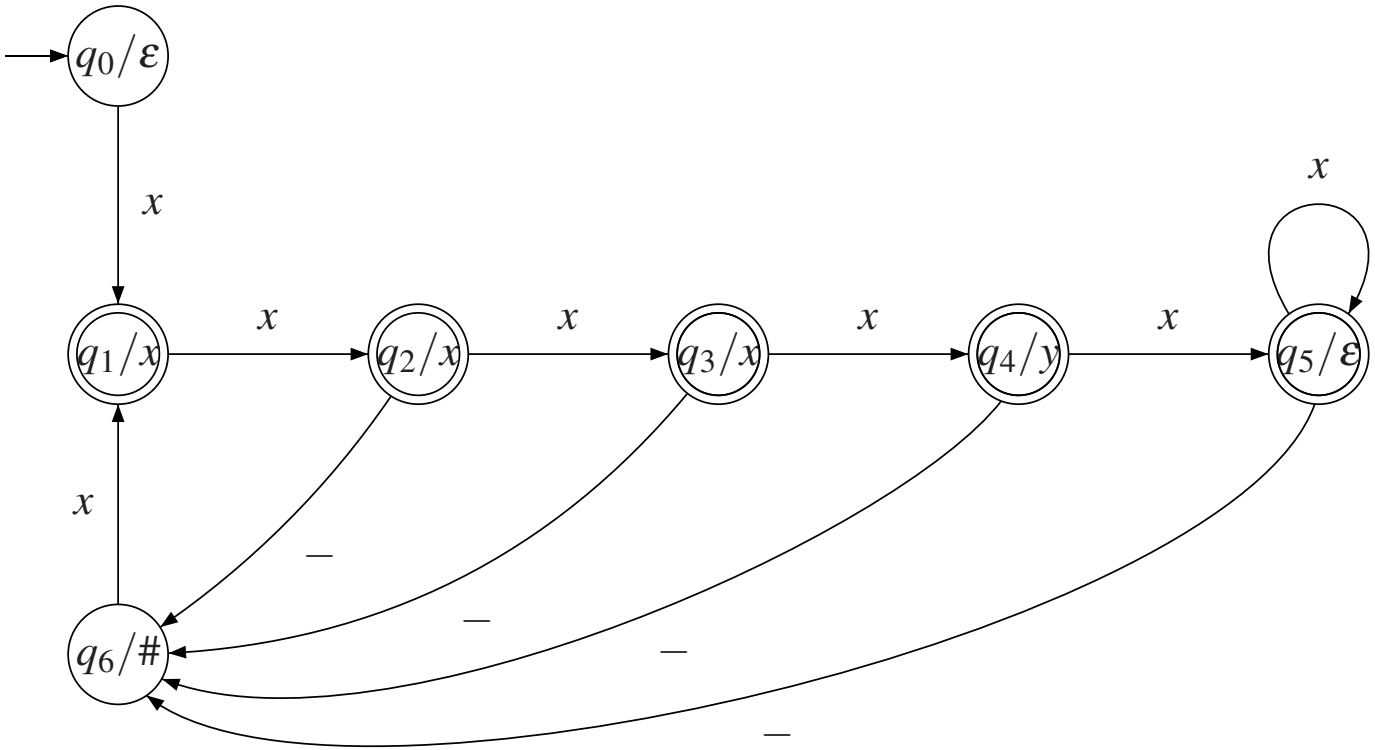


Figura 56: Transdutor do tipo Máquina de Moore equivalente ao transdutor do tipo Máquina de Mealy da Figura 55

Linguagem de saída

Além de definirem a linguagem de entrada associada ao autômato finito subjacente, os transdutores, sejam eles de um tipo ou de outro, definem uma segunda linguagem, denominada **linguagem de saída**, denotada por $L(T)$, correspondente ao conjunto das sentenças sobre Δ que são geradas quando do reconhecimento de sentenças pertencentes a $L(M)$, onde M é o autômato finito em que o transdutor é baseado. Demonstra-se que a classe de linguagens que pode ser gerada por um transdutor finito corresponde exatamente à classe de linguagens que pode ser reconhecida pelo autômato finito em que ele se baseia: a classe das linguagens regulares.

Exercício

Considere a linguagem $(x \mid X \mid \sqcup)^{+}\mathbf{.}$, definida sobre o alfabeto $\{x, X, \sqcup, \mathbf{.}\}$. Alguns exemplos de sentenças pertencentes a essa linguagem são:

- $x\mathbf{.}$
- $\sqcup\mathbf{.}$
- $X\sqcup\sqcup\sqcup\mathbf{.}$
- $\sqcup\sqcup x\sqcup\sqcup\mathbf{.}$
- $\sqcup\sqcup xXX\sqcup XxxXx\mathbf{.}$
- $xXxx\sqcup xx\sqcup\sqcup x\sqcup xXX\sqcup\sqcup\sqcup\mathbf{.}$

Exercício

Essa linguagem representa frases em que as palavras são elementos de $(x \mid X \mid \sqcup)^+$, letras minúsculas e maiúsculas são representadas, respectivamente, pelos símbolos "x" e "X", cada espaço em branco, usado para separar as palavras, é representado pelo símbolo "$\sqcup$", e o ponto, ao final da frase, é representado pelo símbolo "**.**". Pede-se para construir uma seqüência de transdutores (Mealy ou Moore) que, além de aceitarem essa linguagem de entrada, incorporem, cada qual, uma das transduções a seguir especificadas, de forma cumulativa.

Exercício

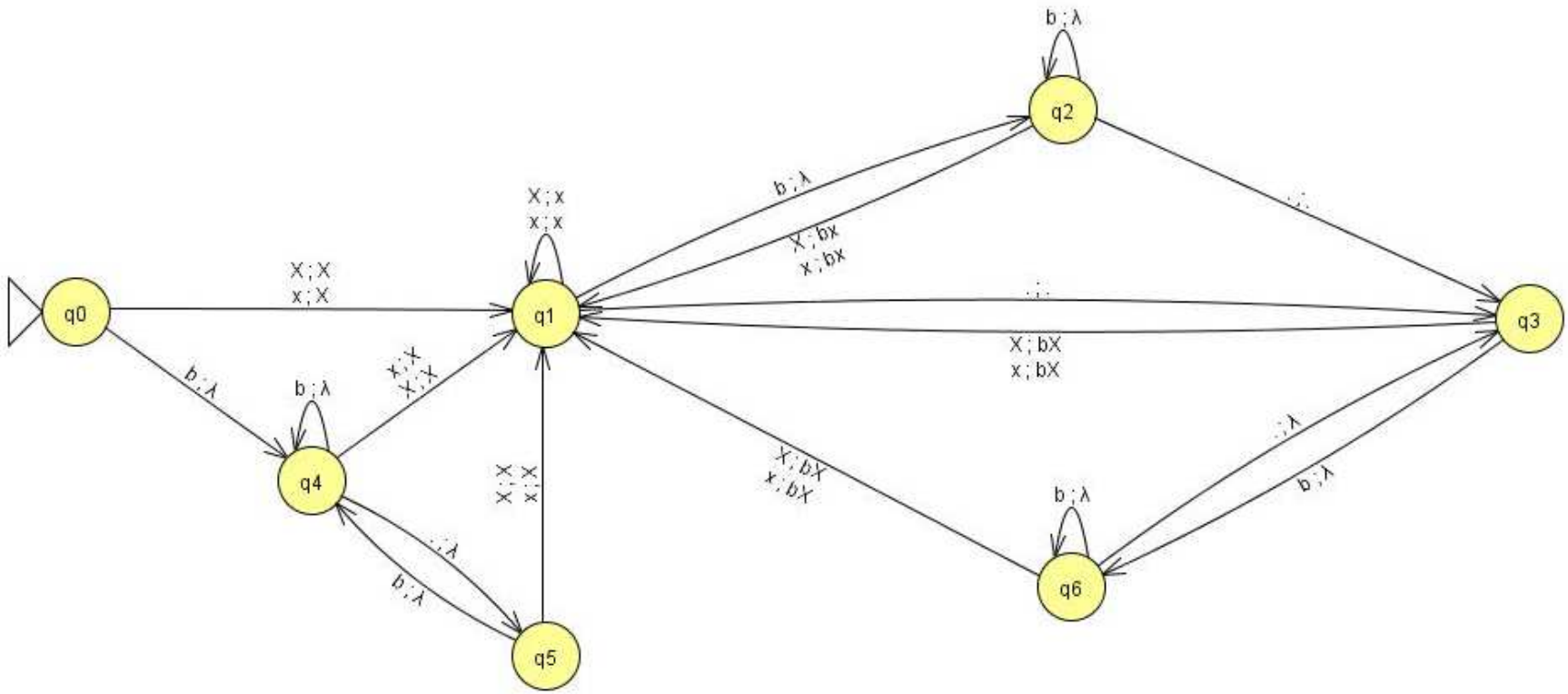
1. Remoção do excesso de brancos entre palavras consecutivas, preservando um único espaço entre elas (exemplo: $x \sqcup \sqcup \sqcup xxx. \Rightarrow x \sqcup xxx.$);
2. Remoção de todos os brancos no início da frase, antes da primeira palavra (exemplo: $\sqcup \sqcup \sqcup xxx. \Rightarrow xxx.$);
3. Remoção de todos os brancos no final da frase, depois da última palavra e antes do ponto final (exemplo: $x \sqcup xxxx \sqcup \sqcup. \Rightarrow x \sqcup xxx.$);

Exercício

4. Substituição da primeira letra da primeira palavra da frase por uma letra maiúscula, convertendo todas as demais para minúscula (exemplo: $xXX \sqcup XxXx\mathbf{.} \Rightarrow Xxx \sqcup xxxx\mathbf{.}$);
5. Reconhecimento e transdução de uma seqüência de frases, ou seja, da linguagem $((x \mid X \mid \sqcup)^+\mathbf{.})^+$, garantindo a existência de exatamente um espaço em branco entre duas frases consecutivas, logo depois do ponto ao final da primeira e imediatamente antes da primeira palavra da frase seguinte (exemplo: $Xx \sqcup xx\mathbf{.}Xxx \sqcup x\mathbf{.} \Rightarrow Xx \sqcup xx\mathbf{.} \sqcup Xxx \sqcup x\mathbf{.}$);
6. Remoção de frases vazias, ou seja, constituídas apenas por um espaço em branco e um ponto final (exemplo: $X\mathbf{.} \sqcup \mathbf{.} \sqcup Xx \sqcup x\mathbf{.} \Rightarrow X\mathbf{.} \sqcup Xx \sqcup x\mathbf{.}$).

Exercício

Solução:



Nota: q_3 e q_5 são finais.

“Pumping Lemma”

- O “Pumping Lemma” estabelece uma propriedade que é sempre verdadeira para toda e qualquer linguagem regular. Caso a linguagem considerada não exiba tal propriedade, pode-se concluir imediatamente que a mesma não é regular;
- Ele não pode ser usado para provar que uma linguagem é regular, uma vez que ele é satisfeito por todas as linguagens regulares e mais outras que não são regulares. Assim, ele só pode ser usado para provar que uma linguagem não é regular;
- Para provar que uma linguagem é regular, deve-se apresentar uma gramática regular, uma expressão regular ou um autômato finito que a represente.

“Pumping Lemma”

Teorema 9.1 “Seja L um conjunto regular qualquer. Então existe uma constante n, dependente apenas de L, tal que, para quaisquer sentenças $w \in L$, com $|w| \geqslant n$, w pode ser subdividida em três subcadeias x, y e z, de tal forma que $w = xyz, 1 \leqslant |y|, |xy| \leqslant n$, ou seja, $1 \leqslant |y| \leqslant n$, e, além disso, $xy^i z \in L, \forall i \geqslant 0$.”

“Pumping Lemma”

O reconhecimento de qualquer cadeia $w \in L$, com $|w| \geqslant n$, sendo L aceita por um autômato finito (sem transições em vazio e determinístico) M com n estados, ocorre percorrendo-se pelo menos dois estados idênticos entre as $n+1$ configurações assumidas por M durante o reconhecimento dos primeiros n símbolos de w.
Seja $w = a_1 a_2 ... a_m, |w| = m, m \geqslant n$. A seqüência abaixo ilustra a evolução da configuração do autômato M no reconhecimento de w:

$$q_0 \overset{a_1}{\rightarrow} q_1 \overset{a_2}{\rightarrow} q_2 \overset{a_3}{\rightarrow} q_3 \ldots \overset{a_{n-1}}{\rightarrow} q_{n-1} \overset{a_n}{\rightarrow} q_n \overset{a_{n+1}}{\rightarrow} q_{n+1} \ldots \overset{a_{m-1}}{\rightarrow} q_{m-1} \overset{a_m}{\rightarrow} q_m$$

onde $q_0 ... q_m$ são os estados sucessivamente percorridos por M (não necessariamente distintos entre si).

“Pumping Lemma”

Considerando-se os $n+1$ estados inicialmente percorridos por M $(q_0, q_1 \ldots q_n)$, é fato que pelo menos dois desses estados devem ser idênticos. Existem então duas possibilidades extremas a serem consideradas, no que diz respeito à localização desses estados idênticos na seqüência:

1. A distância entre eles é a menor possível:
   $(q_i, a_k \ldots a_m) \vdash (q_j, a_{k+1} \ldots a_m), q_i = q_j, j \leqslant n$;
2. A distância entre eles é a maior possível:
   $(q_0, a_1 \ldots a_m) \vdash^* (q_n, a_{n+1} \ldots a_m), q_0 = q_n$.

“Pumping Lemma”

Reescrevendo-se w como xyz, em que x corresponde à parte da cadeia de entrada que leva M à primeira ocorrência de um estado repetido na seqüência, e y corresponde à parte da cadeia que leva M à sua segunda ocorrência, tem-se que:

- $|y| \geqslant 1$;
- $|xy| \leqslant n$;
- Portanto, $1 \leqslant |y| \leqslant n$, pois $|y| \leqslant |xy|$.

“Pumping Lemma”

Como se pode perceber, o fato de a subcadeia y levar o autômato de um estado q_i, anterior ao seu reconhecimento, para o mesmo estado $q_j = q_i$, posterior ao seu reconhecimento, caracteriza como um ciclo o caminho percorrido pelos estados de M, com os símbolos de y. Pelo fato de se tratar de um ciclo, repetições arbitrárias do mesmo conduzem ao reconhecimento de sentenças também pertencentes à linguagem definida pelo autômato. Dessa forma, todas as sentenças do tipo xy^iz, com $i \geqslant 0$, pertencem necessariamente a $L(M)$.

A constante n

O “Pumping Lemma” das linguagens regulares estabelece a propriedade de que, dada uma sentença de comprimento mínimo n pertencente a esta linguagem, é sempre possível identificar, na subcadeia formada pelos seus n primeiros símbolos, uma nova subcadeia cujo comprimento está entre 1 e n, de tal modo que repetições arbitrárias da mesma geram sentenças que também pertencem à linguagem definida.
Assim, a constante n corresponde ao número de estados do autômato finito utilizado para definir a linguagem regular. No entanto, como é sabido, uma mesma linguagem regular pode ser definida por autômatos finitos distintos, os quais podem possuir, eventualmente, um número de estados também distintos.

A constante n

Por outro lado, é natural que se questione a existência de um valor para a constante n que independa do autômato analisado, e que possa, portanto, ser considerado como inerente à linguagem.
Considerando-se a existência de um autômato finito mínimo que reconhece uma dada linguagem regular L, é natural que se considere o número de estados do correspondente autômato finito como o valor n inerente à linguagem L.
Observe-se que, embora o teorema prove a existência da constante n, a sua aplicação em casos práticos não exige que se determine o valor dessa constante.

Exemplo

Exemplo 9.1
Considere-se um autômato finito M com cinco estados distintos, e suponha-se que M efetue a análise de uma cadeia $p \in L(M), |p| = 5$. Claramente, M deverá percorrer seis estados durante o reconhecimento da cadeia. Não obstante, como M apresenta apenas cinco estados distintos, é evidente que pelo menos dois (eventualmente mais) dos estados assumidos por M durante o reconhecimento de p são idênticos.
Considere-se agora uma cadeia $q \in L(M), |q| = 20$. Da mesma forma, analisando-se os seis primeiros estados percorridos por M, constata-se que obrigatoriamente haverá pelo menos dois estados repetidos entre eles, correspondentes ao reconhecimento dos cinco primeiros símbolos de q.

Exemplo

Exemplo 9.2
Seja $M = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, F)$ um autômato finito sem transições em vazio. Se $ab \in L(M)$, então a seqüência de configurações assumidas por M durante a análise dessa cadeia deve, necessariamente, corresponder a alguma das seguintes possibilidades:

1. $(q_0, ab) \vdash (q_0, b) \vdash (q_0, \varepsilon)$
2. $(q_0, ab) \vdash (q_0, b) \vdash (q_1, \varepsilon)$
3. $(q_0, ab) \vdash (q_1, b) \vdash (q_0, \varepsilon)$
4. $(q_0, ab) \vdash (q_1, b) \vdash (q_1, \varepsilon)$

Exemplo

Considerando-se os demais elementos de M desconhecidos, pode-se apenas especular sobre a real seqüência que corresponde à aceitação da cadeia ab por M. De qualquer forma, as seguintes conclusões são válidas:

1. Se $(q_0, ab) \vdash (q_0, b) \vdash (q_0, \varepsilon)$, então as três possibilidades seguintes são verdadeiras:
   1. $x = \varepsilon, y = ab, z = \varepsilon$;
   2. $x = a, y = b, z = \varepsilon$;
   3. $x = \varepsilon, y = a, z = b$.
2. Se $(q_0, ab) \vdash (q_0, b) \vdash (q_1, \varepsilon)$, então:
   1. $x = \varepsilon, y = a, z = b$.
3. $(q_0, ab) \vdash (q_1, b) \vdash (q_0, \varepsilon)$
   1. $x = \varepsilon, y = ab, z = \varepsilon$.
4. $(q_0, ab) \vdash (q_1, b) \vdash (q_1, \varepsilon)$
   1. $x = a, y = b, z = \varepsilon$.

Exemplo

Portanto, qualquer que seja o caso, é sempre possível identificar, na cadeia ab, cujo comprimento coincide com o número de estados do autômato que a aceita, uma subcadeia y, de comprimento maior ou igual a 1 e menor ou igual a 2, que provoca um ciclo na seqüência de movimentações executada pelo autômato.

Exemplo

Exemplo 9.3
Considere-se o autômato da Figura 57.

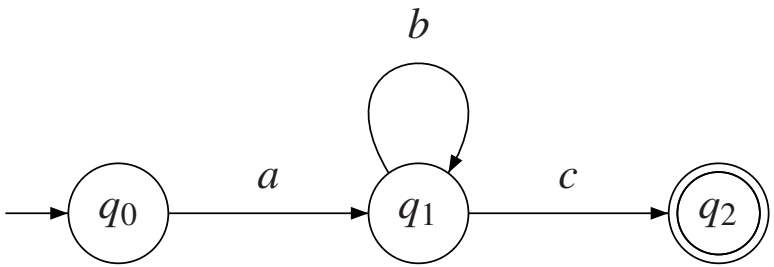


Figura 57: Aplicação do “Pumping Lemma” ao autômato finito que aceita ab^*c

Exemplo

A aplicação das propriedades enunciadas através do "Pumping Lemma" a este autômato podem ser verificadas através do uso de cadeias de comprimento maior ou igual a 3, uma vez que ele possui três estados:

- Considere-se a cadeia $w = abc, |w| = 3$. Então, w pode ser reescrito como $xyz, |xy| \leqslant 3, 1 \leqslant |y| \leqslant 3$ e, finalmente, $xy^iz \in L, \forall i \geqslant 0$. Nesse caso, deve-se escolher $x = a, y = b, z = c$. Assim, $xz = ac, xyyz = abbc, xyyyz = abbbc$ etc. são todas cadeias que pertencem a L.

Exemplo

- Considere-se a cadeia $w = abbbc, |w| = 5$. Então, w pode ser reescrito como $xyz, |xy| \leqslant 3, 1 \leqslant |y| \leqslant 3$ e, finalmente, $xy^i z \in L, \forall i \geqslant 0$. Nesse caso podem-se fazer três escolhas distintas de subdivisão da cadeia w, todas em conformidade com os critérios do "Pumping Lemma":
  - $x = a, y = b, z = bbc$. As cadeias $(a)(b)^*(bbc)$ estão contidas em L.
  - $x = a, y = bb, z = bc$. As cadeias $(a)(bb)^*(bc)$ estão contidas em L.
  - $x = ab, y = b, z = bc$. As cadeias $(ab)(b)^*(bc)$ estão contidas em L.

  Nem todas as subdivisões de uma cadeia w geram cadeias que produzem cadeias que pertencem à linguagem. Note-se, em particular, no exemplo acima, que seria possível relacionar, entre as subdivisões possíveis da cadeia de comprimento 5, as seguintes alternativas:

  i) $x = \varepsilon, y = a, z = bbbc$;
  ii) $x = \varepsilon, y = ab, z = bbc$;
  iii) $x = \varepsilon, y = abb, z = bc$.

Exemplo

Em todos esses casos, xy^iz gera cadeias que não pertencem a L. Qualquer que seja a cadeia escolhida, o "Pumping Lemma" garante apenas que, se ela possuir comprimento mínimo, então ao menos uma subdivisão xyz da mesma será possível de ser feita, de modo que todas as cadeias xy^iz também pertençam à linguagem.

Aplicações do “Pumping Lemma”

A principal aplicação do “Pumping Lemma” consiste na demonstração da existência de linguagens não-regulares. Outras aplicações importantes podem ser encontradas na demonstração de certas questões decidíveis da classe das linguagens regulares.
A demonstração de que uma dada linguagem não é regular pode ser feita por contradição, da seguinte forma:

1. Admite-se inicialmente, por hipótese, que a linguagem sob análise seja regular;
2. Através de manipulações, demonstra-se que a linguagem não exibe as propriedades descritas pelo “Pumping Lemma”;
3. Conclui-se, por contradição, que a hipótese não é verdadeira, e portanto que a linguagem não é regular.

Exemplo

Exemplo 9.4
Seja $L = \{a^k b^k \mid k \geqslant 0\}$. Supondo que L seja uma linguagem regular, tome-se a sentença $a^n b^n$, onde n é a constante definida pelo "Pumping Lemma". Essa sentença pertence a L e possui comprimento $2n$, portanto maior ou igual a n. De acordo com o "Lemma", essa sentença pode ser decomposta em três subcadeias x, y e z, tais que $xyz = a^n b^n, |xy| \leqslant n, |y| \geqslant 1$.
Logo, $y = a^i, 1 \leqslant i \leqslant n$, e xyz pode ser reescrito como $a^{n-i} a^i b^n$. No entanto, nenhuma das seguintes cadeias pertence a L:

1. $xy^0 z = a^{n-i} b^n$
2. $xyyz = a^{n-i} a^i a^i b^n = a^{n+i} b^n$

uma vez que as ocorrências do símbolo a estão desbalanceadas em relação às ocorrências dos símbolos b. Logo, L não é regular.

Exemplo

Exemplo 9.5
Seja $L = \{0^k 10^k \mid k \geqslant 1\}$ e considere-se uma sentença w de comprimento suficientemente longo pertencente a esta linguagem, $w = 0...010...0$. Admitindo-se que seja possível escrever w como xyz, tem-se que $1 \leqslant y \leqslant n$, onde n é a constante de L, e y pode assumir uma das cinco formas seguintes:

1. $y = 1$
2. $y \in 0^+$
3. $y \in 0^+1$
4. $y \in 10^+$
5. $y \in 0^+10^+$

Exemplo

Como é fácil perceber, se $y = 1$, então $xy^0z \notin L$, pois faltará o símbolo "1", obrigatório em todas as sentenças de L.
Se $y \in 0^+$, então $xyyz \notin L$, pois haverá quantidades diferentes do símbolo "0" antes e após o símbolo "1" na sentença.
Se $y \in 0^+1, y \in 10^+$ ou, ainda, $y \in 0^+10^+$, então $xyyz \notin L$, uma vez que $xyyz$ terá mais que um único símbolo "1". Fica assim demonstrado, por contradição, que L não é uma linguagem regular, visto que não atende ao "Pumping Lemma".

Exemplo

Exemplo 9.6
Considere-se a linguagem $L = \{a^{k*k} \mid k \in \mathbb{Z}_+\}$. De acordo com essa definição, as sentenças de L são seqüências formadas por símbolos a de comprimento 1, 4, 9, 16 etc. Seja n a constante de L e considere-se a sentença a^{n*n}.
Essa cadeia pode ser reescrita como xyz, em que $1 \leqslant |y| \leqslant n$. Pelo "Pumping Lemma", se $xyz \in L$, então $xyyz \in L$. Considerando a sentença $xyyz$, tem-se que $n^2 < |xyyz| \leqslant n^2 + n$. Por outro lado, $n^2 + n < (n+1)^2$, portanto, $n^2 < |xyyz| < (n+1)^2$. Ora, isso contraria a hipótese de que o comprimento de todas as sentenças dessa linguagem correspondem ao quadrado de algum número inteiro positivo, uma vez que não existe $i \in \mathbb{Z}_+$ tal que $n^2 < i^2 < (n+1)^2, \forall n \in \mathbb{Z}_+$. Fica assim demonstrado, por contradição, que L não é uma linguagem regular.

Exemplo

Exemplo 9.7
Seja $L = \{a^k b^k c^k \mid k \geqslant 1\}$. Supondo que L seja uma linguagem regular, tome-se a sentença $a^n b^n c^n$, onde n é a constante definida pelo "Pumping Lemma". Claramente essa sentença pertence a L. Mas, de acordo com o "Lemma", essa sentença pode ser decomposta em três subcadeias x, y e z, tais que $xyz = a^n b^n c^n, |xy| \leqslant n, |y| \geqslant 1$. Logo, $y = a^i, 1 \leqslant i \leqslant n$, e xyz pode ser reescrito como $a^{n-i} a^i b^n c^n$. No entanto, nenhuma das seguintes cadeias pertence a L:

1. $xy^0 z = a^{n-i} b^n c^n$
2. $xyyz = a^{n-i} a^i a^i b^n c^n = a^{n+i} b^n c^n$

uma vez que as ocorrências do símbolo a estão desbalanceadas em relação às dos símbolos b e c. Logo, L não é regular. Observe-se a semelhança da presente demonstração com a que foi efetuada para a linguagem $a^k b^k$ no Exemplo 9.4.

Exemplo

Exemplo 9.8
Considere-se $L = \{a^k \mid k \geqslant 1$ é um número primo$\}$. Admitindo-se que L seja uma linguagem regular, tome-se a sentença a^m, onde m é o primeiro número primo superior à constante n definida pelo "Pumping Lemma". Logo, $m > n$. De acordo com o "Lemma", como $|a^m| = m \geqslant n$, essa sentença pode ser decomposta em três subcadeias x, y e z, com $xyz = a^m, |xy| \leqslant n, |y| \geqslant 1$.
Além disso, $xy^iz \in L, \forall i \geqslant 0$. Em particular, pode-se fazer $i = m+1$. Logo, de acordo com o "Lemma", a cadeia $xy^{m+1}z$ deveria pertencer a L. No entanto,
$|xy^{m+1}z| = |xyzy^m| = |xyz| + |y^m|$. Como $|xyz| = m$ e $|y^m| = m * |y|$, então
$|xy^{m+1}z| = m + m * |y| = m * (1 + |y|)$.

Exemplo

Esse resultado mostra que o comprimento de $xy^{m+1}z$, ou seja, $m * (1 + |y|)$, não é um número primo, uma vez que:

- Ele é divisível por m, pois $\frac{m * (1 + |y|)}{m} = (1 + |y|)$;
- $m \neq 1$, pois $n \geqslant 1$ e $m > n$;
- $m \neq m * (1 + |y|)$, pois, de acordo com o "Pumping Lemma", $|y| \geqslant 1$.

Logo, L não é regular.

Exercício

Prove que a linguagem:

$$\{ww|w \in \{a,b\}^+\}$$

não é regular.

Conceito

- Uma determinada classe de linguagens é fechada em relação a uma operação se da aplicação da operação a quaisquer linguagens dessa classe resultar sempre uma linguagem que também pertença à classe em questão.
- O estudo de uma classe de linguagens do ponto de vista das operações em relação às quais ela é fechada é muito importante, uma vez que auxilia, na prática, na determinação da classe de linguagens a que uma certa linguagem possa ou não pertencer.

União, concatenação e fechamento

Teorema 10.1 "A classe das linguagens regulares é fechada em relação às operações de união, concatenação e fechamento reflexivo e transitivo."

Imediata, a partir da definição dos conjuntos regulares.

Complementação

Teorema 10.2 "A classe das linguagens regulares é fechada em relação à operação de complementação."

Seja $L(M)$ a linguagem aceita por um autômato finito determinístico M=$(Q, \Delta, \delta, q_0, F)$, sendo δ uma função total, e considere-se $\Delta \subseteq \Sigma$. Como se pode perceber pela Figura 58,
$\Sigma^* - L = (\Sigma^* - \Delta^*) \cup (\Delta^* - L(M))$.

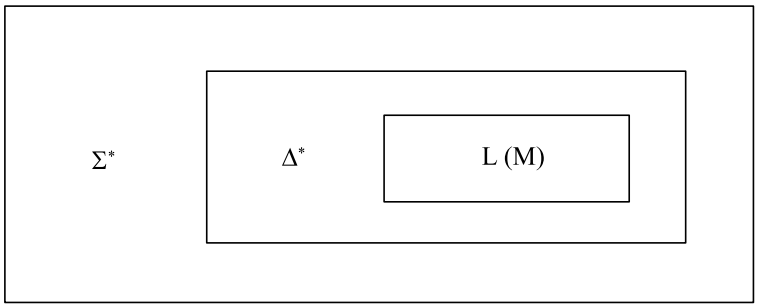


Figura 58: Representação de Σ^*, Δ^* e $L(M)$

Complementação

- $\Sigma^* - \Delta^*$ contém as sentenças que possuem pelo menos um elemento de $\Sigma - \Delta$, e $\Delta^* - L(M)$ as sentenças sobre Δ rejeitadas por M.
- A linguagem $\Delta^* - L(M)$ é aceita pelo autômato $M' = (Q, \Delta, \delta, q_0, Q - F)$, em que os estados finais de M tornam-se não-finais em M' e vice-versa. Assim, se $x \in L(M)$, ou seja, se $\delta(q_0, x) \in F$, então $x \notin L(M')$, uma vez que $\delta(q_0, x) \notin (Q - F)$.
- Logo, conclui-se que, se L for uma linguagem regular, então $\Delta^* - L(M)$ será também uma linguagem regular, uma vez que ela é aceita pelo autômato finito M'.

Complementação

Por outro lado, a linguagem $\Sigma^* - \Delta^*$, de acordo com a sua interpretação (conjunto de "sentenças que possuem pelo menos um elemento de $\Sigma - \Delta$"), pode ser reescrita como:

$$\Sigma^* - \Delta^* = \Sigma^*(\Sigma - \Delta)\Sigma^*$$

Portanto, como decorrência do fechamento das linguagens regulares sobre as operações de fechamento reflexivo e transitivo e de concatenação, é possível afirmar que $\Sigma^* - \Delta^*$ é regular.
Finalmente, $(\Sigma^* - \Delta^*) \cup (\Delta^* - L(M))$ é também uma linguagem regular, uma vez que $(\Sigma^* - \Delta^*)$ e $(\Delta^* - L(M))$ são fechadas em relação à operação de união.

Exemplo

Exemplo 10.1
Considere-se M o autômato finito determinístico da Figura 59, $L(M) = (ab \mid c)d^*e^*$.

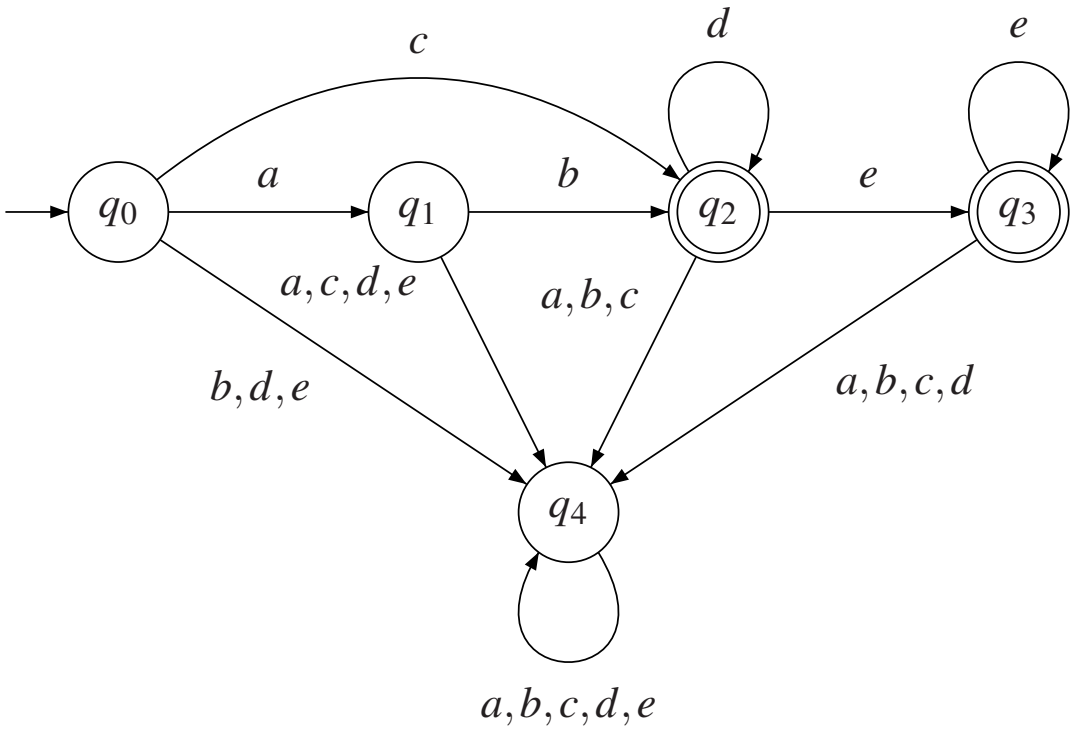


Figura 59: Autômato finito que aceita L

Exemplo

Através do método apresentado na demonstração do Teorema 10.2 obtém-se M', representado na Figura 60, de modo que $L(M') = \overline{L(M)}$, com $\Sigma = \Delta = \{a, b, c, d, e\}$.

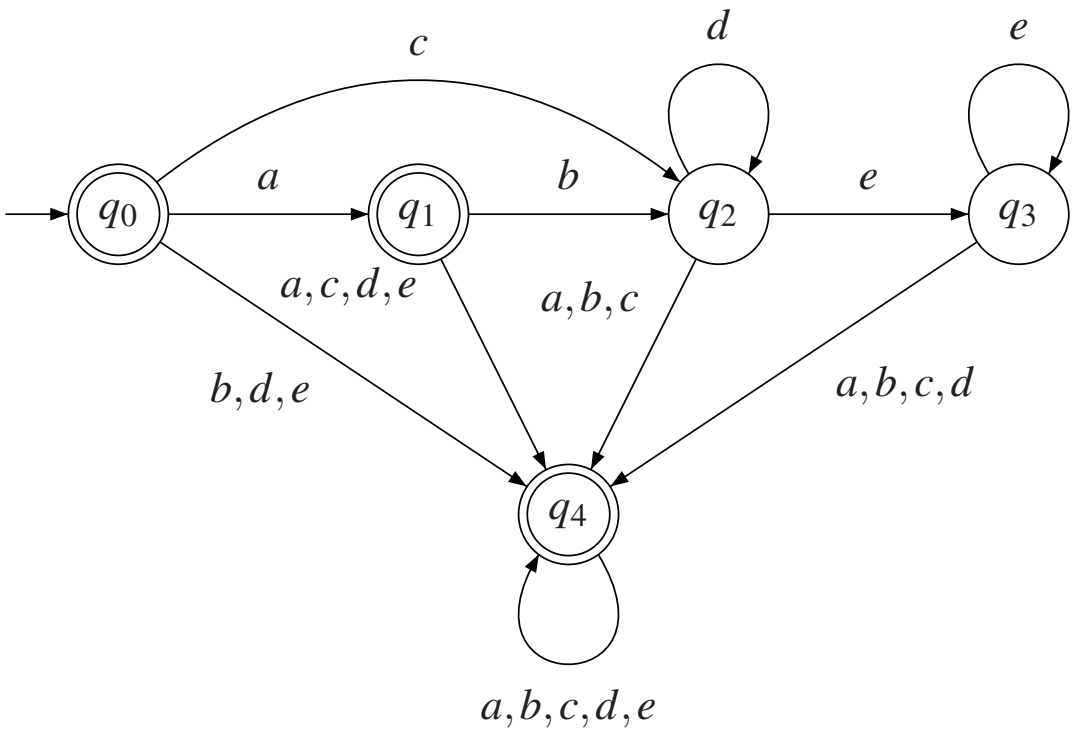


Figura 60: Autômato finito que aceita $\overline{L}$ (ver Figura 59)

Intersecção

Teorema 10.3 "A classe das linguagens regulares é fechada em relação à operação de intersecção."

Considere-se a linguagem L_1 sobre Σ_1, e L_2 sobre Σ_2, sendo $\Sigma_1, \Sigma_2 \subseteq \Sigma$. Então, considerando-se as complementações em relação a Σ, a seguinte relação é verdadeira (Lei de De Morgan):

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

Portanto, a regularidade da linguagem resultante da intersecção de duas outras linguagens regulares depende da preservação da regularidade pelas operações de união e complemento. Como esse fato já foi constatado nos Teoremas 10.1 e 10.2, é possível afirmar, com base no presente teorema, que $L_1 \cap L_2$ será necessariamente uma linguagem regular.

Reversão

Teorema 10.4 "A classe das linguagens regulares é fechada em relação à operação de reversão de suas sentenças (linguagem reversa)."

Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito que aceita L. O Algoritmo 10.1 mostra como construir $M' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F')$, eventualmente não-determinístico, de tal modo que $L(M') = L^R$.

Algoritmo

Algoritmo 10.1 *"Construção do autômato finito que aceita L^R a partir do autômato finito que aceita L."*

- *Entrada: um autômato finito $M = (Q, \Sigma, \delta, q_0, F)$;*
- *Saída: um autômato finito $M' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F')$, tal que $L(M') = L(M)^R$;*
- *Método:*

Algoritmo

1. *Construção de* F'*:*
   a) *Se* $\varepsilon \notin L$*, então* $F' = \{q_0\}$*;*
   b) *Se* $\varepsilon \in L$*, então* $F' = \{q_0, q_0'\}$.
2. *Construção de* δ'*:*
   c) *Se* $\delta(q, \sigma) \in F$*, então* $\delta'(q_0', \sigma) = q$*;*
   d) *Se* $\delta(q_a, \sigma) = q_b$*, então* $\delta'(q_b, \sigma) = q_a$.

Exemplo

Exemplo 10.2
Considere a linguagem a^*bc^* aceita por M, conforme a Figura 61.

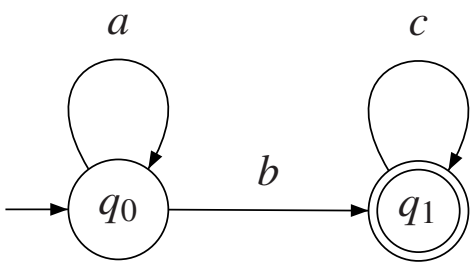


Figura 61: Autômato finito que aceita $L = a^*bc^*$

Exemplo

A aplicação do algoritmo descrito conduz à obtenção do autômato M' da Figura 62.

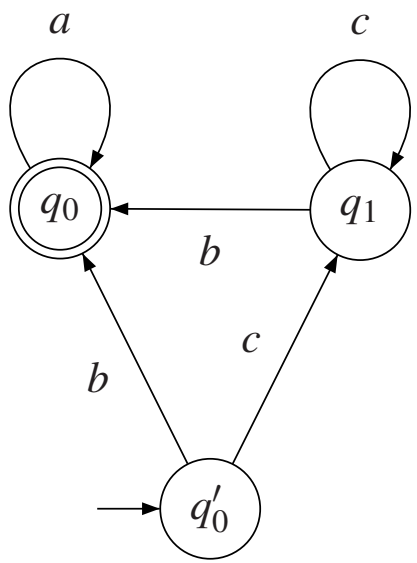


Figura 62: Autômato finito que aceita $L^R = (a^*bc^*)^R = c^*ba^*$

Como se pode observar, $L(M') = ba^* \mid cc^*ba^* = c^*ba^* = L(M)^R$.

Aplicações

Uma das principais aplicações do estudo do fechamento de uma classe de linguagens em relação a um determinado conjunto de operações consiste na possibilidade de se determinar a classe de uma linguagem a partir da decomposição da mesma em linguagens mais simples, de classe conhecida, e que, combinadas por intermédio de operadores que preservam a classe dessas linguagens mais simples, nos permitem inferir diretamente a classe das linguagens resultantes.

Exemplo

Exemplo 10.3
Considere-se a linguagem $L = \{a^*bc^* \mid$ o comprimento das sentenças é maior ou igual a $3\}$. L é regular?
L pode ser representada como $L_1 \cap L_2$, onde $L_1 = \{a^*bc^*\}$ e $L_2 = (a \mid b \mid c)(a \mid b \mid c)(a \mid b \mid c)(a \mid b \mid c)^*$. Como L_1 e L_2 são regulares (pois estão expressas através de expressões regulares), e a classe das linguagens regulares é fechada em relação à operação de intersecção, então L também é regular. De fato, não é difícil perceber que $L = a^*(aab \mid abc \mid bcc)c^*$.

Exemplo

Exemplo 10.4
Seja a linguagem $L = \{w \in \{a,b,c,d\}^* \mid w$ contém a subcadeia "bb" e w não contém a subcadeia "dd"$\}$. L é regular?
$L = L_1 \cap \overline{L_2}$, onde $L_1 = \{w \in \{a,b,c,d\}^* \mid w$ contém a subcadeia "bb"$\}$ e $L_2 = \{w \in \{a,b,c,d\}^* \mid w$ contém a subcadeia "dd"$\}$:

- $L_1 = (a \mid b \mid c \mid d)^* bb (a \mid b \mid c \mid d)^*$
- $L_2 = (a \mid b \mid c \mid d)^* dd (a \mid b \mid c \mid d)^*$

Como L_1 e L_2 são regulares, e a classe das linguagens regulares é fechada em relação às operações de complemento e intersecção, segue que L também é regular.

Exemplo

Exemplo 10.5
Considere-se a linguagem L_1 definida sobre o alfabeto $\{a,b\}$, de tal forma que pertencem a L_1 todas as cadeias que podem ser formadas com os símbolos de seu alfabeto, excetuando-se aquelas que contêm exatamente três símbolos a. L_1 é regular? Não é difícil perceber que $L_1 = \overline{L_2}$, onde $L_2 = b^*ab^*ab^*ab^*$, ou seja, L_1 corresponde à complementação da linguagem que contém todas as cadeias com exatamente três símbolos a (L_2). Portanto, L_1 é regular. Uma expressão regular que representa L_1 é $b^* \mid b^*ab^* \mid b^*ab^*ab^* \mid b^*ab^*ab^*ab^*a(a \mid b)^*$.

Exemplo

Exemplo 10.6
Considere-se o alfabeto $\{a, b, c\}$ e a linguagem L definida de tal forma que suas cadeias satisfazem todas às seguintes regras:

1. Possuem a subcadeia aaa como prefixo;
2. Possuem comprimento total múltiplo de 4;
3. Possuem quantidade par de símbolos c;
4. Não contêm a subcadeia bb.

São exemplos de cadeias pertencentes a L:
$aaabcccc, aaabcbca, aaaa, aaaaaaba, aaaacccbaaac$ etc.

Exemplo

L é regular? Para responder, basta notar que $L = ((L_1 \cap L_2) \cap L_3) \cap \overline{L_4}$, onde:

- L_1 é gerada por $aaa(a \mid b \mid c)^*$:
  Cadeias que possuem aaa como prefixo.
- L_2 é gerada por $((a \mid b \mid c)(a \mid b \mid c)(a \mid b \mid c)(a \mid b \mid c))^*$:
  Cadeias que possuem comprimento total múltiplo de 4.
- L_3 é gerada por $((a \mid b)^* c (a \mid b)^* c (a \mid b)^*)^*$:
  Cadeias que possuem quantidade par de símbolos c.
- L_4 é gerada por $(a \mid b \mid c)^* bb (a \mid b \mid c)^*$:
  Cadeias que contêm a subcadeia bb.

Como L_1, L_2, L_3 e L_4 são regulares, e a classe das linguagens regulares é fechada em relação às operações de intersecção e complementação, conclui-se que L é regular.

Conceito

Quando se diz que um problema (ou questão) desse tipo é **decidível**, isso significa que ele sempre tem solução, qualquer que seja a sua instância considerada (ou argumentos aplicados). Mais do que isso, cada questão decidível é caracterizada pela existência de um algoritmo que permite resolver o problema geral com garantias de obtenção do resultado — afirmativo ou negativo, dependendo do caso.

Linguagem não-vazia

Teorema 11.1 "A linguagem L aceita por um autômato finito com n estados é não-vazia se e somente se o autômato aceita pelo menos uma cadeia $w, |w| < n$."

- A condição necessária ("aceita uma sentença de comprimento inferior a $n \Rightarrow$ linguagem é não-vazia") é óbvia e não necessita ser demonstrada.
- A condição suficiente ("linguagem é não-vazia $\Rightarrow$ aceita uma sentença de comprimento inferior a n") não é tão óbvia, mas pode ser verificada com auxílio do "Pumping Lemma". Considere-se $w \in L(M), |w| = m$.
- Se $m < n$, então nada há para demonstrar, e a hipótese é trivialmente verdadeira.

Linguagem não-vazia

- Se, no entanto, $m \geqslant n$, então w pode ser reescrita como xyz com $xz \in L(M), y \neq \varepsilon, |xz| < m$. Seguem, então, duas possibilidades: $|xz| \geqslant n$ ou $|xz| < n$. Se $|xz| < n$, a hipótese está demonstrada. Se, por outro lado, $|xz| \geqslant n$, pode-se agora considerar $w = xz$ e aplicar o "Pumping Lemma" novamente, desta vez sobre tal cadeia.
- Através da iteração deste passo, é possível obter cadeias de comprimentos sucessivamente menores, enquanto o comprimento da cadeia anterior for maior ou igual a n. Assim, é possível demonstrar a existência de uma sentença de comprimento inferior a n, pertencente a L.

Linguagem não-vazia

A condição suficiente do Teorema 11.1 pode também ser compreendida através do seguinte raciocínio: partindo-se do estado inicial, se o autômato aceitar pelo menos uma cadeia, então a linguagem é não-vazia. Como o autômato possui n estados, então é necessário que pelo menos um desses estados seja final, e também acessível desde o estado inicial.
Se o estado inicial for simultaneamente final, então a cadeia vazia é aceita e a linguagem aceita pelo autômato é não-vazia. Observe-se ainda que $|\varepsilon| < n$, qualquer que seja o valor de n, uma vez que $n \geqslant 1$.

Linguagem não-vazia

Se o estado inicial não for simultaneamente final, então será necessário atingir pelo menos um dos outros $n-1$ estados do autômato, o qual deve também ser final. Para isso, bastam cadeias de comprimento máximo $n-1$, inclusive, já que cadeias de comprimento maior ou igual a n possuem ciclos (conforme o “Pumping Lemma”), e não modificam o conjunto de estados que são acessíveis a partir do estado considerado.

Linguagem não-vazia

Logo, se nenhuma cadeia de comprimento menor que n for aceita pelo autômato, isso significa que:

- Não existem estados finais no autômato, ou
- Os estados finais do autômato não são acessíveis desde o estado inicial

e, portanto, a linguagem por ele aceita é vazia.
Em outras palavras, qualquer estado acessível de um autômato finito com n estados é alcançável por meio de cadeias de comprimento máximo $n-1$. Se algum desses estados for final, então a linguagem aceita é não-vazia. Caso contrário, é vazia.

Linguagem não-vazia

Para determinar se uma linguagem, aceita por um autômato finito com n estados, é não-vazia, basta verificar se o autômato aceita alguma sentença de comprimento entre 0 (inclusive) e $n-1$ (inclusive). Se nenhuma dessas cadeias for aceita, pode-se concluir que a linguagem é vazia, sem testar quaisquer outras cadeias.
Para um autômato finito com n estados, cujo alfabeto de entrada tenha m símbolos, a quantidade de cadeias que devem ser testadas é dada pela fórmula:

$$\sum_{i=0}^{n-1} m^i$$

pois, conforme pode ser verificado na Tabela 51, essa fórmula representa a quantidade total de cadeias distintas cujos comprimentos estão entre 0 (inclusive) e $n-1$ (inclusive), e que podem ser construídas a partir de um alfabeto com m símbolos.

Linguagem não-vazia

Tabela 51: Quantidade de cadeias que podem ser obtidas a partir de um alfabeto com m símbolos, com comprimento entre 0 e $n-1$

Comprimento	Cadeias distintas	Cadeias distintas
0	1	m^0
1	m	m^1
2	$m * m$	m^2
3	$m * m * m$	m^3
...	...	...
$n-1$	$m * m * m * \ldots * m$	m^{n-1}

Exemplo

Exemplo 11.1
Seja L uma linguagem regular sobre o alfabeto $\{a, b, c\}$ e aceita por um autômato finito M com três estados. Então, para determinar se L é não-vazia, basta verificar se alguma das seguintes cadeias é aceita por M:

- Comprimento 0 (uma cadeia): ε
- Comprimento 1 (três cadeias): a, b, c
- Comprimento 2 (nove cadeias): $aa, ab, ac, ba, bb, bc, ca, cb, cc$

Se alguma dessas 13 (= 1 + 3 + 9) cadeias for aceita por M, então L será não-vazia. Caso contrário, será vazia.

Linguagem infinita

Teorema 11.2 "A linguagem L aceita por um autômato finito com n estados é infinita se e somente se o autômato aceitar pelo menos uma cadeia $w \in \Sigma^*, n \leqslant |w| < 2n$."

A condição "se" (aceita pelo menos uma cadeia $w, n \leqslant |w| < 2n \Rightarrow$ linguagem infinita) pode ser facilmente deduzida através do "Pumping Lemma": como $|w| \geqslant n$, então w pode ser reescrita como xyz, e $xy^iz \in L, \forall i \geqslant 0$. Logo, L é infinita.
A condição "somente se" (linguagem infinita $\Rightarrow$ aceita pelo menos uma cadeia $w, n \leqslant |w| < 2n$) é demonstrada, por contradição, a seguir.
Se L é infinita, então com certeza existem cadeias de comprimento maior ou igual a n (pois a quantidade de cadeias com comprimento menor ou igual a n é finita). Considere-se $w \in L(M), |w| \geqslant n$.

Linguagem infinita

Se $|w| < 2n$, então não há nada a demonstrar e a hipótese é trivialmente verdadeira.
Se $|w| \geqslant 2n$, então, de acordo com o "Pumping Lemma", $w = xyz, |xy| \leqslant n, 1 \leqslant |y| \leqslant n$. Logo, a cadeia xz também pertence a $L, |xz| < |w|, |xz| \geqslant n$ (pois, como $|w| \geqslant 2n$ e, na pior das hipóteses, $|y| = n$, então $|xz| = |w| - |y| \geqslant n$).
Duas possibilidades podem então ocorrer com a cadeia xz: ou $|xz| \geqslant 2n$ ou $|xz| < 2n$.
Se $|xz| < 2n$, então a hipótese é verdadeira e o teorema está demonstrado.

Linguagem infinita

Se $|xz| \geqslant 2n$, pode-se considerar agora $w = xz$ e aplicar o "Pumping Lemma" novamente sobre essa cadeia. Através da iteração deste passo, enquanto o comprimento de w for maior ou igual a $2n$, é possível obter cadeias de comprimentos sucessivamente menores, porém sempre de comprimento maior ou igual a n. Logo, necessariamente existe uma cadeia pertencente à linguagem, de comprimento maior ou igual a n e menor que $2n$, e o teorema está demonstrado.
Assim, se a linguagem for infinita, ela deverá obrigatoriamente conter pelo menos uma cadeia de comprimento entre n (inclusive) e $2n$ (exclusive).

Linguagem infinita

A condição “somente se” do Teorema 11.2 pode também ser compreendida da seguinte forma: por se tratar de uma linguagem infinita, e portanto não-vazia, o autômato correspondente aceita pelo menos uma cadeia w_0, $0 \leqslant |w_0| < n$ (ver Teorema 11.1).
Por outro lado, como se trata de uma linguagem infinita, então é fato que este autômato possui pelo menos um ciclo, correspondente à cadeia y, $1 \leqslant |y| \leqslant n$ (conforme o “Pumping Lemma”).
Logo, a combinação desses resultados (ou seja, o “bombeamento” da cadeia y na cadeia w_0, resultando em uma nova cadeia cujo comprimento corresponde à soma dos comprimentos mínimos e máximos das outras duas) garante a existência de pelo menos uma cadeia w_1, $1 < |w_1| < 2n - 1$, que é aceita pelo autômato.

Linguagem infinita

Se $|w_1| \geqslant n$, então a condição está provada. Se $|w_1| < n$, pode-se "bombear" y novamente, desta vez em w_1, resultando na cadeia w_2, $2 < |w_2| < 2n - 1$.
A iteração desse passo, enquanto $w_i < n$, garante a existência de uma cadeia w_j, aceita pelo autômato, tal que $n \leqslant |w_j| < 2n - 1$, como se quer demonstrar.
Em outras palavras, a existência de ciclos acessíveis desde o estado inicial garante que o autômato aceita pelo menos uma cadeia w tal que $n \leqslant |w| < 2n - 1$.

Linguagem infinita

A principal aplicação deste teorema se encerra no algoritmo por ele sugerido, o qual permite determinar se a linguagem aceita por um autômato finito com n estados é infinita ou não: basta verificar se o autômato aceita alguma cadeia de comprimento entre n (inclusive) e $2n-1$ (inclusive). Como a quantidade de cadeias com essa característica é finita, conclui-se ser sempre possível determinar se uma linguagem regular é infinita ou não, bastando para isso analisar, exaustivamente, se alguma dessas cadeias pertence à linguagem definida.

A quantidade de cadeias que devem ser testadas em um autômato com n estados e cujo alfabeto de entrada possui m símbolos é dada pela fórmula:

$$\sum_{i=n}^{2n-1} m^i$$

Exemplo

Exemplo 11.2
Seja L uma linguagem regular sobre o alfabeto $\{a, b\}$ e aceita por um autômato finito M com 2 estados. Então, para saber se L é infinita, basta verificar se alguma das seguintes cadeias é aceita por M:

- Comprimento 2 (quatro cadeias): aa, ab, ba, bb
- Comprimento 3 (oito cadeias): $aaa, aab, aba, abb, baa, bab, bba, bbb$

Se alguma dessas 12 (= 4 + 8) cadeias for aceita por M, então L é infinita. Caso contrário, conclui-se que a linguagem é finita.

Linguagem finita

Teorema 11.3 "A linguagem L aceita por um autômato finito com n estados é finita se e somente se o autômato não aceita nenhuma sentença w tal que $n \leqslant |w| < 2n$."

Decorre diretamente do teorema anterior. L é infinita se e somente se o autômato finito correspondente aceita pelo menos uma cadeia $w, n \leqslant |w| < 2n$. Logo, se não existir nenhuma cadeia que satisfaça a essa condição, a linguagem L será finita. Para determinar se, além de finita, L é não-vazia, basta verificar se o autômato finito correspondente aceita pelo menos uma cadeia de comprimento menor do que n (Teorema 11.1).

Resumo

A Tabela 52 resume os resultados até aqui obtidos.

Tabela 52: Cardinalidade de uma linguagem regular

L: uma linguagem aceita por um autômato finito com n estados		
$\exists w \in L, \lvert w \rvert < n$?	$\exists w \in L, n \leqslant \lvert w \rvert < 2n$?	L é ...
Sim	Sim	**Infinita**
Sim	Não	**Finita, não-vazia**
Não	Não	**Finita, vazia**
Não	Sim	N.A. (contradição)

Exemplo

Exemplo 11.3
Seja uma linguagem L_1 sobre o alfabeto $\{a\}$, aceita por um autômato M_1 com três estados. Para determinar se L_1 é vazia, basta verificar se alguma das cadeias pertencentes ao seguinte conjunto é aceita por M_1: $X = \{\varepsilon, a, aa\}$. Para determinar se L_1 é infinita, deve-se verificar as cadeias do conjunto $Y = \{aaa, aaaa, aaaaa\}$.
Seja o autômato M_1, representado na Figura 63.

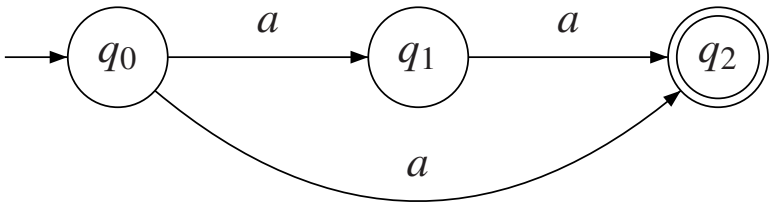


Figura 63: Autômato M_1 que aceita $L_1 = \{a, aa\}$, finita e não-vazia

Exemplo

É fácil perceber, neste caso, que $L_1(M_1) = \{a, aa\}$ é finita e não-vazia. De fato, as cadeias a, aa de X são aceitas por M_1. No entanto, nenhuma das cadeias $aaa, aaaa, aaaaa$ de Y são aceitas por M_1.
Suponha-se agora M_2, correspondente ao autômato da Figura 64.

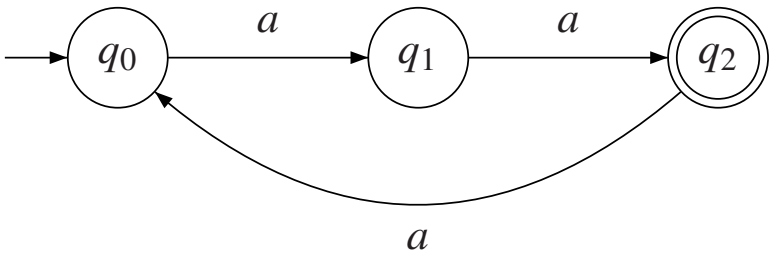


Figura 64: Autômato M_2 que aceita $L_2 = aa(aaa)^*$, infinita

Exemplo

A linguagem $L_2(M_2)$ é, neste caso, $aa(aaa)^*$, e portanto infinita. A infinitude de L_2 é comprovada pelo fato de M_2 aceitar a cadeia $aaaaa$ de Y. O fato de M_2 aceitar aa de X indica que L_2 é não-vazia.
Por último, considere-se M_3 como sendo o autômato da Figura 65.

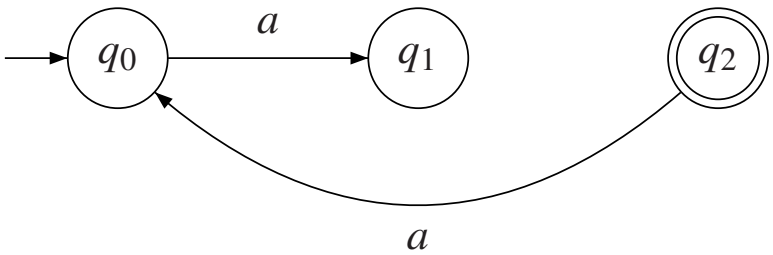


Figura 65: Autômato M_3 que aceita $L = \emptyset$, finita e vazia

Nenhuma das cadeias ε, a, aa de X é aceita por M_3. Logo, como se pode comprovar observando-se a Figura 65, L_3 é vazia (e portanto finita).

Pertencimento

Teorema 11.4 "Seja L uma linguagem regular sobre $\Delta, \Delta \subseteq \Sigma$, e $\alpha \in \Sigma^*$ uma cadeia. Então, a questão $\alpha \in L$ é decidível."

Seja $M = (Q, \Delta, \delta, q_0, F)$ tal que $L = L(M)$. O Algoritmo 11.1 mostra como decidir se a cadeia α pertence ou não à linguagem L.

Pertencimento

Algoritmo 11.1 *"Determina se uma cadeia é sentença da linguagem definida por um autômato finito."*

- *Entrada: um autômato finito* $M = (Q, \Delta, \delta, q_0, F), \Delta \subseteq \Sigma$, *e uma cadeia* $\alpha \in \Sigma^*$;
- *Saída: Se* $\alpha \in L(M)$, *SIM; caso contrário, NÃO;*
- *Método:*
  1. *Obter* $M' = (Q, \Delta, \delta', q_0, F')$, *isento de transições em vazio, tal que* $L(M') = L(M)$;
  2. *Determinar* $\delta'(q_0, \alpha)$. *Se* $\delta'(q_0, \alpha) \in F'$, *então SIM; caso contrário, NÃO.*

Pertencimento

O Algoritmo 11.1 garante que qualquer cadeia pode ser analisada em um número finito de passos (ou tempo finito de processamento) em um autômato finito. Para isso, é suficiente garantir que o mesmo seja isento de transições em vazio, o que implica a inexistência de ciclos formados exclusivamente por transições desse tipo, as quais poderiam, eventualmente, provocar um processamento interminável da cadeia de entrada.

Exemplo

Exemplo 11.4
Considere-se o autômato da Figura 66, que possui um ciclo formado por transições em vazio.

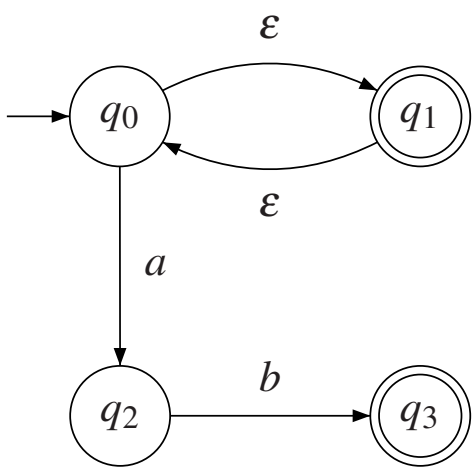


Figura 66: Autômato com ciclo de transições em vazio

Exemplo

Os movimentos executados por esse autômato na análise da cadeia *ba* não permitem que o mesmo pare em qualquer configuração, final ou não-final, como mostra a seguinte seqüência:

$$(q_0, ba) \vdash (q_1, ba) \vdash (q_0, ba) \vdash (q_1, ba) \vdash (q_0, ba) \vdash (q_1, ba) \vdash \ldots$$

O autômato equivalente, da Figura 67, é isento de transições em vazio, e, portanto, de ciclos formados por transições em vazio.

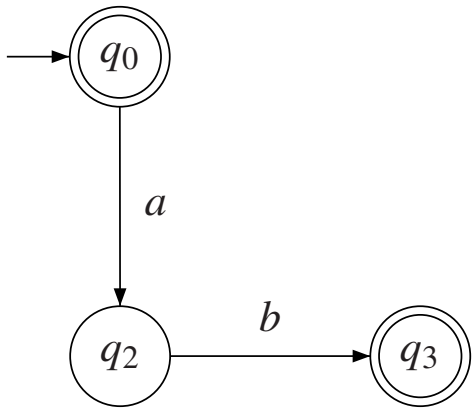


Figura 67: Autômato equivalente ao da Figura 66, porém isento de ciclos formados por transições em vazio

Exemplo

Esse autômato atinge a seguinte configuração de parada, para a mesma cadeia *ba* de entrada, após executar zero movimentações:

$$(q_0, ba)$$

Portanto, por não se tratar de uma configuração final, a cadeia *ba* é rejeitada e não pertence à linguagem definida pelos autômatos das Figuras 66 e 67.

Igualdade

Teorema 11.5 "Sejam L_1 e L_2 duas linguagens regulares quaisquer. Então, a questão $L_1 = L_2$ é decidível."

Considerem-se as linguagens $L_1 = L_1(M_1) \subseteq \Sigma_1^*$ e $L_2 = L_2(M_2) \subseteq \Sigma_2^*$. A condição $L_1 = L_2$ pode também ser formulada como:

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

onde as operações de complementação se referem a qualquer alfabeto Σ tal que $(\Sigma_1 \cup \Sigma_2) \subseteq \Sigma$. Para decidir se $L_1 = L_2$, deve-se executar o Algoritmo 11.2.

Igualdade

Algoritmo 11.2 *"Determina se duas linguagens regulares são idênticas."*

- *Entrada: dois autômatos finitos* $M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ *e* $M_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$*;*
- *Saída: Se* $L_1(M_1) = L_2(M_2)$*, SIM; caso contrário, NÃO;*
- *Método:*
  1. *Basta construir* M_3 *tal que* $L_3(M_3) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$*. Se* $L_3 = \emptyset$*, então SIM; caso contrário, NÃO*

Igualdade

A construção de M_3 pode ser efetuada diretamente a partir dos algoritmos utilizados na apresentação de resultados anteriores (fechamento dos conjuntos regulares em relação às operações de união, complementação e intersecção, respectivamente Teoremas 10.1, 10.2 e 10.3). Além disso, a questão $L_3(M_3) = \emptyset$ pode ser decidida em função do Teorema 11.1.

Totalidade

Teorema 11.6 "Seja M um autômato que aceita L sobre Σ. Então, a questão $L = \Sigma^*$ é decidível."

Esta questão pode ser decidida pelo Algoritmo 11.3.

Totalidade

Algoritmo 11.3 *"Determina se a linguagem aceita por um autômato finito é* Σ^**."*

- *Entrada: um autômato finito* $M = (Q, \Sigma, \delta, q_0, F)$*;*
- *Saída: Se* $L(M) = \Sigma^*$*, SIM; caso contrário, NÃO;*
- *Método:*
  1. *Basta construir* M' *tal que* $L(M') = \Sigma^* - L(M) = \overline{L(M)}$*. Se* $L(M') = \emptyset$*, então SIM; caso contrário, NÃO.*

Subconjunto

Teorema 11.7 "Sejam $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$ duas linguagens regulares. Então, a questão $L_1 \subseteq L_2$ é decidível."

A condição $L_1 \subseteq L_2$ também pode ser formulada como:

$$(\Sigma^* - L_2) \cap L_1 = \overline{L_2} \cap L_1 = \emptyset$$

onde a operação de complementação se refere a um alfabeto Σ tal que $\Sigma_2 \subseteq \Sigma$. Para decidir se $L_1 \subseteq L_2$, basta executar o Algoritmo 11.4.

Subconjunto

Algoritmo 11.4 *"Determina se uma linguagem regular é subconjunto de uma outra linguagem regular."*

- *Entrada: dois autômatos finitos* $M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ *e* $M_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$*;*
- *Saída: Se* $L_1(M_1) \subseteq L_2(M_2)$*, SIM; caso contrário, NÃO;*
- *Método:*
  1. *Basta construir* M_3 *tal que* $L_3(M_3) = \overline{L_2(M_2)} \cap L_1(M_1)$*. Se* $L_3 = \emptyset$*, então SIM; caso contrário, NÃO.*