

CAPÍTULO 2 - AUTOMATOS FINITOS E LINGUAGENS REGULARES

definição 2.1 - um automato de estados finitos é uma 5-upla $A = (Q, \Sigma, \delta, q_0, F)$ onde :

Q - conjunto finito não vazio (estados)

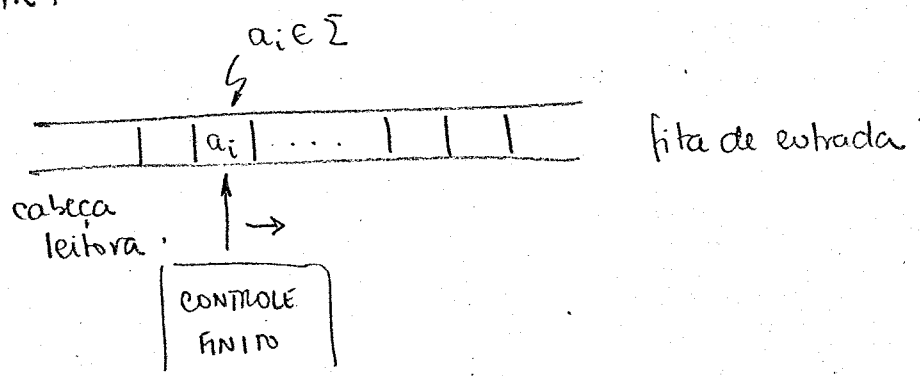
Σ - alfabeto (de entrada) ; $\Sigma \cap Q = \emptyset$

$q_0 \in Q$ (estado inicial)

$F \subseteq Q$ (conjunto de estados finais)

$\delta: Q \times \Sigma \rightarrow Q$ (função de transição de estados)

Simbolicamente :



$\delta(q, a) = q' \iff$ autômato estando no estado q e lendo o símbolo a na fita de entrada, move a cabeça leitora uma posição para a direita e vai para o estado q' .

função δ ESTENDIDA : $\delta : Q \times \Sigma^* \rightarrow Q$

$$\delta(q, \epsilon) = q$$

$$\delta(q, xa) = \delta(\delta(q, x), a) \quad ; \quad x \in \Sigma^*, a \in \Sigma$$

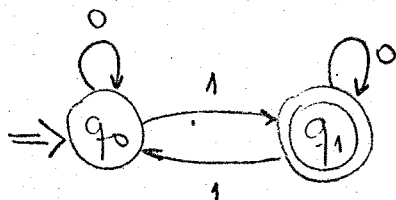
$\delta(q, x) = q' \Leftrightarrow$ autômato estando no estado q e lendo o símbolo mais a esquerda de x , vai estar no estado q' após todos os símbolos de x terem sido lidos.

definição 2.2 - Sejam A um autômato finito e $x \in \Sigma^*$.
 x é aceita por A se $\delta(q_0, x) \in F$.

definição 2.3 - Seja A um autômato finito. A linguagem reconhecida por A é:

$$L(A) = \{x \in \Sigma^* / \delta(q_0, x) \in F\}$$

Exemplo:



(diagrama de estados)

$$A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

$$L(A) = \{x \in \Sigma^* / x \text{ contém n}^\circ \text{ ímpar de 1's}\}$$

RELACÕES DE EQUIVALÊNCIA E AUTÔMATOS FINITOS

definição 2.4 . Uma relação binária R sobre um conjunto S é um conjunto de pares de elementos de S .

NOTAÇÃO :

se $(a, b) \in R$ então $a R b$.

definição 2.5 Uma relação binária R sobre S é uma relação de equivalência se

- 1) R é reflexiva $(\lambda R \lambda, \forall \lambda \in S)$
- 2) R é simétrica $(\lambda, t \in S, \lambda R t \Rightarrow t R \lambda)$
- 3) R é transitiva $(\lambda, t, u \in S, \lambda R t, t R u \Rightarrow \lambda R u)$

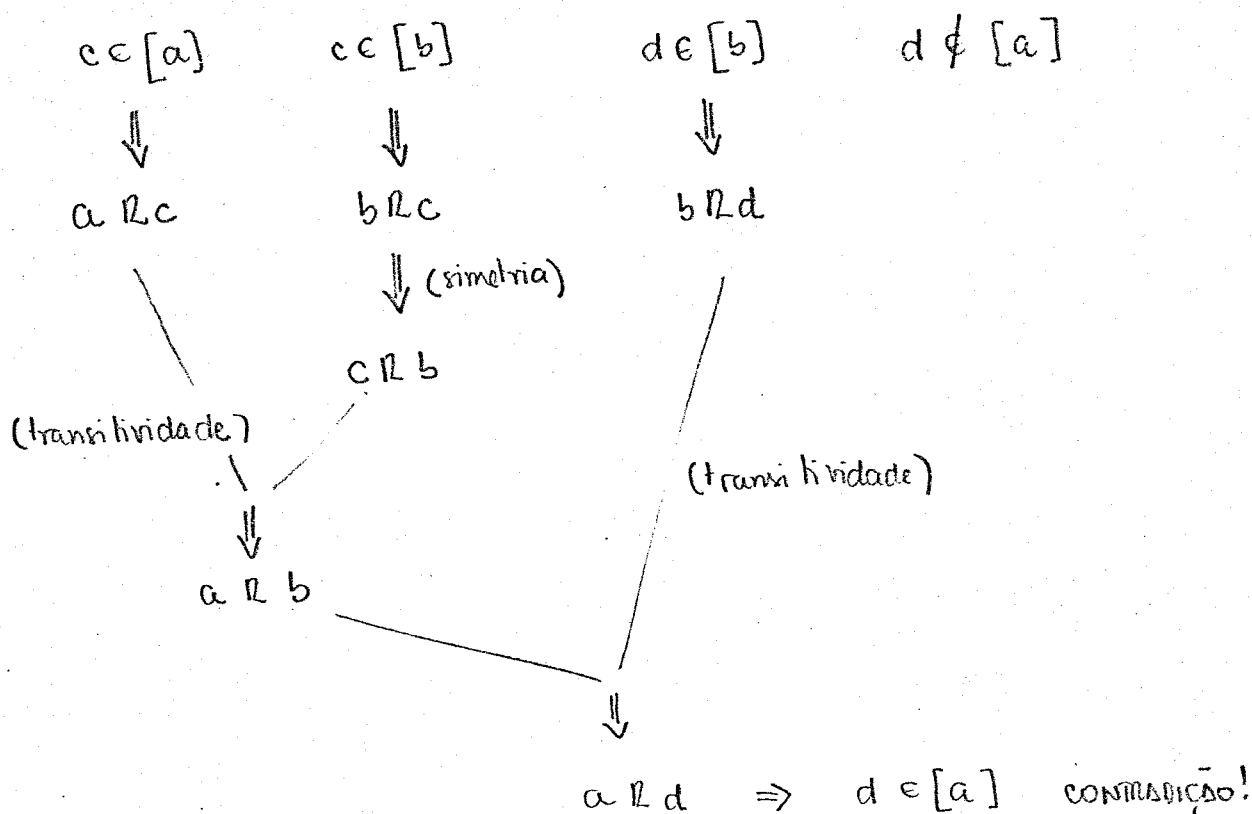
Teorema 2.1 . Se R é uma relação de equivalência sobre S então é possível dividir S em k subconjuntos distintos ($k \geq 1$) chamados CLASSES DE EQUIVALÊNCIA tais que
 $a R b \Leftrightarrow a$ e b pertencem ao mesmo subconjunto

(ou seja : uma relação de equivalência R sobre S INDUZ uma partição de S)

Prova : seja $[a] = \{ b / a R b \}$. O teorema garante que

$$a, b \in S \Rightarrow \begin{array}{ll} \text{i) } [a] = [b] & \text{se} \\ \text{ii) } [a] \cap [b] = \emptyset & \text{se} \end{array}$$

Admitindo, por absurdo, que isto não vale, tem-se :



Por exemplo:

R sobre $N = \{1, 2, 3, \dots\}$ tal que

$$i R j \Leftrightarrow |i - j| \text{ é divisível por } 3$$

CLASSES DE EQUIVALÊNCIA

$$c_1 = \{1, 4, 7, \dots\}$$

$$c_2 = \{2, 5, 8, \dots\}$$

$$c_3 = \{3, 6, 9, \dots\}$$

definição 2.6 : O índice de uma relação de equivalência R (representado por $i(R)$) é o número de

classes de equivalência de R .

definição 2.7 Sejam R_1 e R_2 relações de equivalência. R_1 refina R_2 se $R_1 \subseteq R_2$ (isto é, $x R_1 y \Rightarrow x R_2 y$).

Exercício: Mostre que se R_1 refina R_2 então $i(R_1) \geq i(R_2)$!

definição 2.8 Seja R , relação de equivalência sobre Σ^* .
 R é uma relação de congruência à direita se
 $x R y \Rightarrow (\forall z \in \Sigma^*) xz R yz$

Exemplo: Seja R sobre Σ^* definida por

$$x R y \Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$$

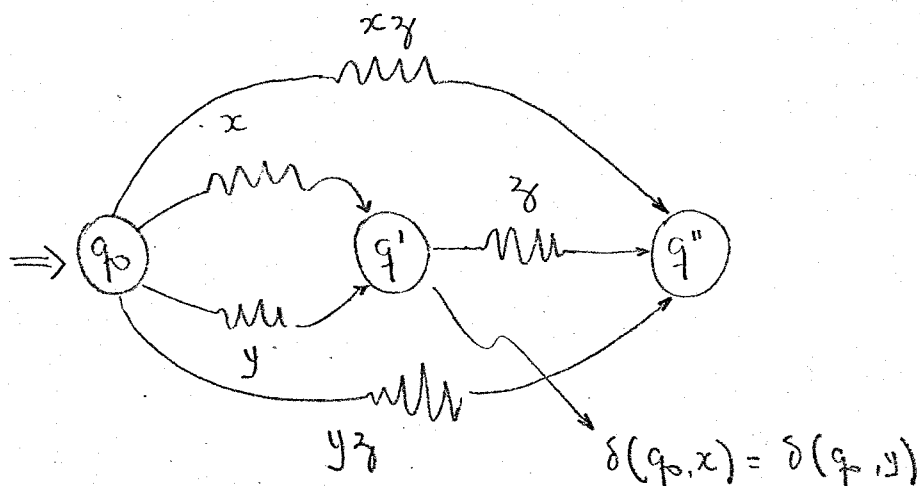
(Exercício: Mostre que R é uma relação de equivalência!)

$$x R y \Rightarrow \delta(q_0, x) = \delta(q_0, y) = q'$$

$$\begin{aligned} \text{Mas: } \delta(q_0, xz) &= \delta(\delta(q_0, x), z) = \\ &= \delta(q', z) = \delta(\delta(q_0, y), z) \\ &= \delta(q_0, yz) \end{aligned}$$

$$\Rightarrow xz R yz$$

Intuitivamente: R é a resposta do autômato à cadeia



definição 2.9. R , relação de equivalência sobre Σ^* . $L \in \Sigma^*$.
 R refina L se $x R y \Rightarrow (x \in L \Leftrightarrow y \in L)$

(ou seja: se R refina L , L é a união de algumas (ou todas) as classes de equivalência de R)

definição 2.10. Seja $L \in \Sigma^*$. A relação R_L é definida por:

$$x R_L y \Leftrightarrow (\forall z \in \Sigma^*) (xz \in L \Leftrightarrow yz \in L)$$

Teorema 2.2. (1) R_L é uma relação de congruência à direita
 (2) R_L refina L
 (3) se R é uma relação de congruência à direita que refina L então $R \subseteq R_L$

(ou seja: R_L é a menor (menos classes de equivalência) relação de congruência à direita que refina L)

Prova:

(1) exercício!

$$(2) \quad x R_L y \Rightarrow (\forall z \in \Sigma^*) (xz \in L \Leftrightarrow yz \in L)$$

Seja $z = \lambda$. Logo $x \in L \Leftrightarrow y \in L$.

(3) seja R uma relação de congruência à direita que refina L

$$x R y \Rightarrow (\forall z \in \Sigma^*) (xz R yz) \quad (R \text{ é r.c.d.})$$

$$\Rightarrow (\forall z \in \Sigma^*) (xz \in L \Leftrightarrow yz \in L) \quad (R \text{ refina } L)$$

$$\Rightarrow x R_L y$$

$$\text{Logo:} \quad R \subseteq R_L.$$

Teorema 2.3 (lema de Nerode). Seja $L \subseteq \Sigma^*$. São equivalentes:

(1) L é regular

(2) Existe uma relação de congruência à direita R sobre Σ^* , que refina L e de índice finito

(3) $i(R_L)$ é finito

Prova:

$$(1) \Rightarrow (2)$$

L é regular \Rightarrow existe (ver lema 2.8) autômato finito $A = (Q, \Sigma, \delta, q_0, F)$ tal que $L = L(A)$.

Seja R uma relação definida por

$$x R y \Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$$

Como se mostrou anteriormente (ver exemplo, página 16), esta relação é relação de congruência à direita e (obviamente!) refina L . Se x e y estão na mesma classe de equivalência ($x R y$) então $\delta(q_0, x) = \delta(q_0, y)$. Como o autômato tem um número finito de estados, $i(R) \leq \#Q$ ($\#Q$ indica a cardinalidade de Q). Logo, $i(R)$ é finito.

(2) \Rightarrow (3). Pelo lema 2.2 tem-se: $R \subseteq R_L$. Logo $i(R) \geq i(R_L)$. Como $i(R)$ é finito, $i(R_L)$ é finito.

(3) \Rightarrow (1). Deve-se construir $A = (Q, \Sigma, \delta, q_0, F)$ tal que $L = L(A)$. Seja $[x]$ a classe de equivalência de R_L que contém $x \in \Sigma^*$.

Então:

$$Q = \{ [x] \mid x \in \Sigma^* \}$$

$$q_0 = [1]$$

$$F = \{ [x] \mid x \in L \}$$

$$\delta([x], a) = [xa]; \quad x \in \Sigma^*, a \in \Sigma$$

(note-se que, como R_L é r.e.d., só existe uma classe de equivalência que contém xa , $\forall x \in \Sigma^*$, $\forall a \in \Sigma$. Portanto, a definição de δ é consistente)

$$\text{Logo: } \delta(q_0, x) = \delta([1], x) = [x]$$

$$x \in L(A) \Leftrightarrow [x] \in F \Leftrightarrow x \in L. \text{ Logo } L = L(A)$$

O leorema de Nerode é muito útil para provar que certas linguagens não são regulares.

EXEMPLO: $L = \{ a^i b^j \mid j \geq i \geq 0 \}$

Seja R_L a relação de equivalência de Nerode:

$$x R_L y \stackrel{\Delta}{\iff} (\forall z \in \Sigma^*) (xz \in L \iff yz \in L)$$

Para mostrar que L é regular, deve-se mostrar que $i(R_L)$ não é finito.

Seja $P^n = \{ a^{i+n} b^i \mid i \geq 0 \}$ ($n \geq 1$). (P^n é o conjunto

das cadeias que são prefixo de alguma cadeia de L e que tem n a's a mais do que b's)

Seja $P = \bigcup_{i \geq 1} P^i$. Será mostrado que as classes de equivalência de R_L são:

- 1) L
- 2) $\Sigma^* - L - P$
- 3) P^i , $i \geq 1$

ou seja, para $x, y \in \Sigma^*$, se $x R_L y$ então

- a) $x, y \in L$ ou
- b) $x, y \in \Sigma^* - L - P$ ou
- c) $x, y \in P^i$ para algum $i \geq 1$

(a) x para todo $z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$, então, em particular, para $z = \lambda$, $x \in L \Leftrightarrow y \in L$.

(b) $x \in \Sigma^* - L - P$. Então x não é prefixo de qualquer cadeia de L . Portanto, não existe $z \in \Sigma^*$ tal que $xz \in L$. Como $xz \in L \Leftrightarrow yz \in L$, não existe $z \in \Sigma^*$ tal que $yz \in L$. Logo, y não é prefixo de qualquer cadeia de L . Portanto, $y \in \Sigma^* - L - P$.

(c) seja $x \in P^l$, ou seja: $x = a^n b^m$ tal que $n - m = l$. Deve-se mostrar que $y \in P^l$. Seja, por absurdo, $y \notin P^l$. Então:

1) seja y tal que $y = a^{n'} b^{m'}$ com $n' - m' = l' \neq l$ (ou seja, $y \notin P^l$). Seja (sem perda de generalidade) $l' < l$.

Seja $z = b^{l'}$. Então $yz = a^{n'} b^{m'+l'} = a^{n'} b^{n'} \in L$.

Por outro lado, $xz = a^n b^{m+l'} \notin L$ (pois $n > m+l'$).

Logo, $y \notin P^{l'}$, $l' \neq l$. CONTRADIÇÃO!

2) seja $y \in \Sigma^* - L - P$. Logo $\nexists z$ tal que $yz \in L$. No entanto, como $x \in P^l$ é possível encontrar $z \in \Sigma^*$ tal que $xz \in L$. Logo $y \notin \Sigma^* - L - P$ CONTRADIÇÃO!

3) seja $y \in L$. Como $\forall z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$, em particular deve valer para $z = \lambda$.

Mas com $z = \lambda$, $xz \notin L$ e $yz \in L$. Logo $y \notin L$.
CONTRADIÇÃO!

Portanto, como $y \notin p^{\ell'}$, $\ell' \neq \ell$; $y \notin \Sigma^* - L - p$ e $y \notin L$
então $y \in p^{\ell}$.

Desta forma R_L tem infinitas classes de equivalência. Pelo lema de Nerode, L não pode ser regular.

MINIMIZAÇÃO DE ESTADOS DE UM AUTÔMATO FINITO.

definição 2.11. Seja $A = (Q, \Sigma, \delta, q_0, F)$. Um estado $q \in Q$ é acessível se $\exists x \in \Sigma^*$ tal que $q = \delta(q_0, x)$.

definição 2.12. O autômato anexo associado a $A = (Q, \Sigma, \delta, q_0, F)$ é definido por:

$$A^c = (Q', \Sigma, \delta', q_0, F') \text{ onde}$$

$$Q' = \{q \in Q \mid q \text{ é acessível}\}$$

$$F' = F \cap Q'$$

$$\delta' = \delta \cap (Q' \times \Sigma \times Q')$$

definição 2.13. Dois autômatos A_1 e A_2 são equivalentes se $L(A_1) = L(A_2)$.

Teorema 2.4 . Para todo automato finito A , A e A^c são equivalentes.

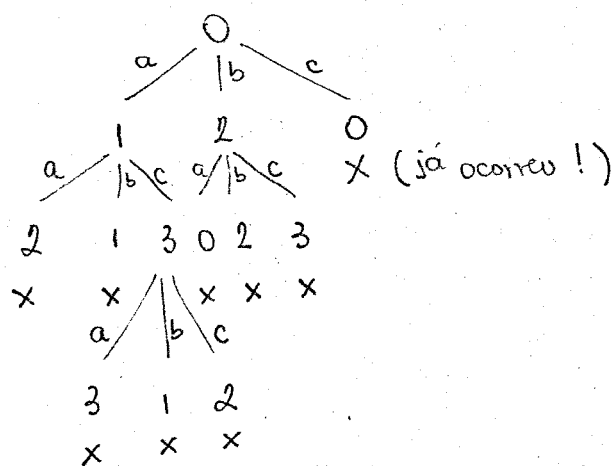
Prova : exercício!

A determinação de estados acessíveis pode ser mecanizada organizando a função δ em forma de árvore e verificando os estados que ocorrem.

EXEMPLO: $A = (Q, \Sigma, \delta, q_0, F)$ com $\Sigma = \{a, b, c\}$;

$Q = \{0, 1, 2, 3, 4, 5\}$; $q_0 = 0$ e δ dada por :

δ	a	b	c
0	1	2	0
1	2	1	3
2	0	2	3
3	3	1	2
4	5	1	2
5	0	4	5



Logo : somente 0, 1, 2 e 3 são acessíveis.

definição 2.14 . Dois estados q e q' são equivalentes ($q \equiv q'$) SSE $(\forall x \in \Sigma^*) (\delta(q, x) \in F \Leftrightarrow \delta(q', x) \in F)$

exercício: Mostre que \equiv é relação de equivalência!

A relação \equiv particiona o conjunto de estados Q em classes de equivalência. A importância dessa partição é que ela permite identificar elementos redundantes em Q (do ponto de vista de reconhecimento da linguagem), pois se $q \equiv q'$ (q e q' na mesma classe de equivalência) não irá fazer diferença se o autômato se encontra no estado q ou no estado q' quando uma cadeia $x \in \Sigma^*$ começar a ser processada. Logo o autômato pode ser minimizado escolhendo-se apenas um elemento de cada uma das classes de equivalência da relação \equiv .

definição 2.15. q é k -equivalente a q' ($q \stackrel{k}{\equiv} q'$) sse
 $(\forall x \in \Sigma^*, |x| \leq k \Rightarrow (\delta(q, x) \in F \Leftrightarrow \delta(q', x) \in F))$

Desta definição segue que se $q \stackrel{k}{\equiv} q'$ então, $q \stackrel{k'}{\equiv} q'$, para todo $k' \leq k$.

$$\begin{aligned} (q \stackrel{k}{\equiv} q' \Rightarrow (\forall x \in \Sigma^*, |x| \leq k' \leq k \Rightarrow \\ \delta(q, x) \in F \Leftrightarrow \delta(q', x) \in F)) \\ \Rightarrow q \stackrel{k'}{\equiv} q') \end{aligned}$$

Portanto, π_k (partição de Q induzida por $\stackrel{k}{\equiv}$) refina $\pi_{k'}$,

$$(\pi_k \subseteq \pi_{k'}, k' \leq k)$$

Teorema 2.5. $q \stackrel{k+1}{\equiv} q' \Leftrightarrow q \stackrel{k}{\equiv} q' \text{ e } \delta(q, a) \stackrel{k}{\equiv} \delta(q', a), \forall a \in \Sigma$.

Prova. (\Rightarrow) $q \stackrel{k+1}{\equiv} q'$. Logo $q \stackrel{k}{\equiv} q'$ pois $\pi_{k+1} \subseteq \pi_k$.

Além disso, para toda cadeia ax , $a \in \Sigma$, $x \in \Sigma^*$, tal que $|ax| \leq k+1$, $k \equiv x$:

$$\delta(q, ax) \in F \Leftrightarrow \delta(q', ax) \in F \quad (q \stackrel{k+1}{\equiv} q')$$

Logo: $\delta(\delta(q, a), x) \in F \Leftrightarrow \delta(\delta(q', a), x) \in F$

e portanto, como $|x| \leq k$, $\delta(q, a) \stackrel{k}{\equiv} \delta(q', a)$.

(\Leftarrow) exercício!

O leorema 2.5 pode ser usado para mostrar (por indução) que se existe k tal que $\pi_k = \pi_{k+1}$, então $\pi_k = \pi_{k+m}$, para todo $m \geq 0$, ou seja, não existem mais refinamentos de π_k . Neste caso, como $\pi \subseteq \pi_k$, $k \equiv x$ que $\pi = \pi_k$.

(para uma prova formal, ver BOOTH, T.L - "SEQUENTIAL MACHINES AND AUTOMATA THEORY")

Teorema 2.6. Seja $A = (Q, \Sigma, \delta, q_0, F)$, com $\#Q = p$ ($p \geq 2$). Então existe n tal que $\pi_n = \pi_{n+1}$ e $n \leq p-2$.

Prova.

Seja $\# \pi_0 = 1$. Neste caso $\pi_1 = \pi_0$ (exercício!) e

$$n=0 \leq p-2.$$

(note que

$$q \stackrel{0}{\equiv} q' \Leftrightarrow \delta(q, \epsilon) \in F \Leftrightarrow \delta(q', \epsilon) \in F)$$

Seja então $\# \pi_0 \geq 2$. Se $\pi_i \neq \pi_{i+1}$ então $\# \pi_{i+1} > \# \pi_i$.

Como cada classe de equivalência de \equiv deve conter pelo menos um estado e como $\# Q = p$,

$$2 \leq \# \pi_i < p.$$

Logo, existirá n tal que, no pior caso:

$$2 = \# \pi_0 < \# \pi_1 < \dots < \# \pi_n = \# \pi_{n+1} = p$$

e neste caso $n = p-2$. Portanto, em geral, existe $n \leq p-2$ tal que $\pi_n = \pi_{n+1}$.

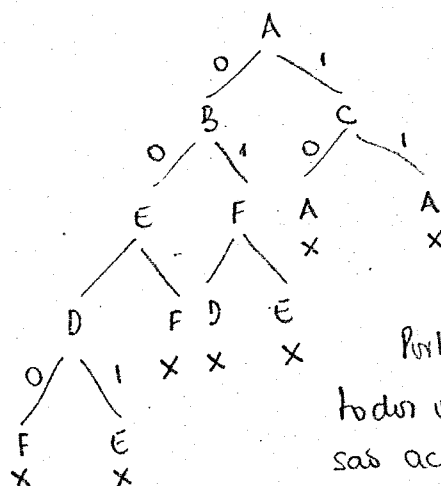
O leorema 2.5 garante que se pode evanhar a relação \equiv considerando-se sucessivamente as relações $\overset{0}{\equiv}$, $\overset{1}{\equiv}$, $\overset{2}{\equiv}$ e assim por diante até que $\pi_k = \pi_{k+1}$ ($k \geq 0$).

O leorema 2.6 garante que este processo sempre pára, isto é, que pode ser construído um algoritmo para construir a relação \equiv .

EXEMPLO: $A = (\{A, B, C, D, E, F\}, \{0, 1\}, \delta, A, \{E, F\})$ com δ

dada por:

δ	0	1
A	B	C
B	E	F
C	A	A
D	F	E
E	D	F
F	D	E



Portanto:
todas as estados
são acessíveis.

Construção de \equiv :

$$\pi_0 = (A \ B \ C \ D) \ (E \ F)$$

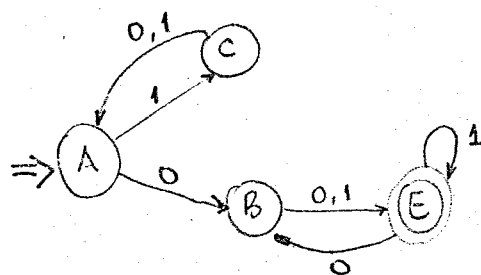
$$\pi_1 = (A \ C) \ (B \ D) \ (E \ F)$$

$$\pi_2 = (A) \ (C) \ (B \ D) \ (E \ F)$$

$$\pi_3 = (A) \ (C) \ (B \ D) \ (E \ F)$$

Logo $\pi = \pi_3$ Assim :

δ'	0	1
A	B	C
B	E	E
C	A	A
E	B	E



(Autômato mínimo e equivalente a \mathbb{A})

AUTÔMATO FINITO NÃO DETERMINÍSTICO

definição 2.16. um autômato finito não determinístico é uma 5-upla $A = (Q, \Sigma, \delta, q_0, F)$, com Q, Σ, q_0 e F definidos como no caso determinístico e δ dada por:

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

com $\mathcal{P}(Q)$ conjunto dos subconjuntos de Q (conjunto potência)

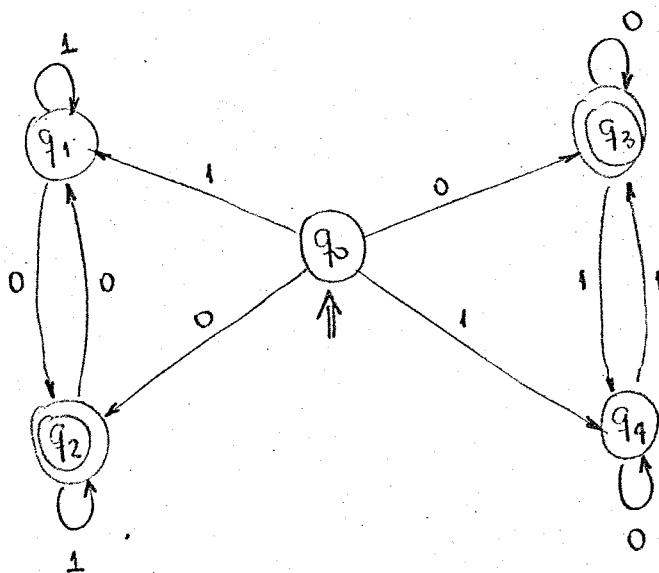
$\delta(q, a) = \{q_1, \dots, q_n\} \Leftrightarrow$ o autômato estando no estado

q e lendo o símbolo a na fita de entrada, move sua cabeça leitora uma posição para a direita e escolhe qualquer um dos q_i , $(1 \leq i \leq n)$ como seu próximo estado (conceitualmente é equivalente a imaginar que o autômato se subdivide em n cópias, cada uma das quais estando em q_i , $i = 1, \dots, n$).

definição 2.17. Seja $x \in \Sigma^*$. x é aceita por a.f.n.d. $A = (Q, \Sigma, \delta, q_0, F)$ se $\delta(q_0, x) \cap F \neq \emptyset$.

(portanto, $L(A) = \{x \in \Sigma^* \mid \delta(q_0, x) \cap F \neq \emptyset\}$)

Por exemplo:



Exercício: determinar $L(A)$!

Teorema 2.7. Seja L um conjunto aceito por um a.f.n.d. Então existe um autômato finito determinístico que aceita L .

Prova (INFORMAL).

$$A = (Q, \Sigma, \delta, q_0, F) \quad (\text{a.f.n.d.})$$

$$A' = (Q', \Sigma, \delta', q'_0, F') \quad (\text{a.f.d.})$$

Como construir A' a partir de A ?

$$1) \quad Q' = \mathcal{P}(Q)$$

2) F' = conjunto formado pelos estados de Q' que possuem pelo menos um estado em F

3) um elemento de Q' será representado por $[q_1, q_2, \dots, q_i]$ onde $q_1, \dots, q_i \in Q$. $q'_0 = [q_0]$.

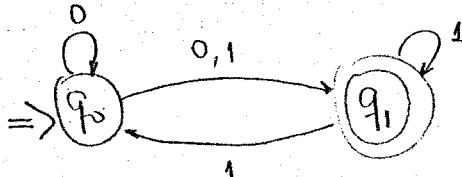
$$4) \quad \delta'([q_1, q_2, \dots, q_i], a) = [r_1, r_2, \dots, r_j] \Leftrightarrow$$

$$\Leftrightarrow \delta(q_1, a) \cup \delta(q_2, a) \cup \dots \cup \delta(q_i, a) = \{r_1, r_2, \dots, r_j\}$$

Exercício : Mostre que $L(A') = L(A)$!

Exemplo : Seja $A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ com δ

dada por :



O autômato A' equivalente a A será :

$\Lambda' = (Q, \{0,1\}, \delta', [q_0], F)$ tal que:

$$Q = \{ \phi, [q_0], [q_1], [q_0q_1] \}$$

$$F = \{ [q_1], [q_0q_1] \}$$

Com δ' dada por:

$$\delta'(\phi, 0) = \delta'(\phi, 1) = \phi$$

$$\delta(q_0, 0) = \{q_0, q_1\} \Rightarrow \delta'([q_0], 0) = [q_0q_1]$$

$$\delta(q_0, 1) = \{q_1\} \Rightarrow \delta'([q_0], 1) = [q_1]$$

$$\delta(q_1, 0) = \phi \Rightarrow \delta'([q_1], 0) = \phi$$

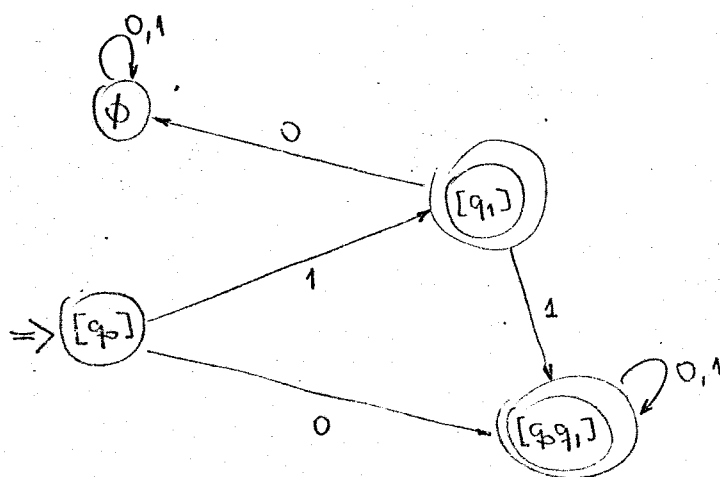
$$\delta(q_1, 1) = \{q_0, q_1\} \Rightarrow \delta'([q_1], 1) = [q_0q_1]$$

e ainda: $\delta'([q_0q_1], 0) = [q_0q_1]$

$$\delta'([q_0q_1], 1) = [q_0q_1]$$

Portanto:

Λ' :



Teorema 2.8 . Seja $G = (N, \Sigma, P, S)$, com $V = N \cup \Sigma$, uma gramática regular. Então existe autômato finito $A = (Q, \Sigma, \delta, q_0, F)$ tal que $L(G) = L(A)$.

Prova (INFORMAL . Ideia da prova : fazer as transições do autômato simularem as derivações na gramática)

Construção de A (um a.f.n.d.)

1) $Q = N \cup \{q\}$, $q \notin N$

2) $q_0 = S$

3) $F = \begin{cases} \{S, q\} & \text{se } S \rightarrow _ \in P \\ \{q\} & \text{caso contrário} \end{cases}$

4) δ definida por :

a) $q \in \delta(B, a)$ se $B \rightarrow a \in P$

b) coloca em $\delta(B, a)$ os estados C tais que $B \rightarrow aC \in P$

c) $\delta(q, a) = \emptyset$, $\forall a \in \Sigma$.

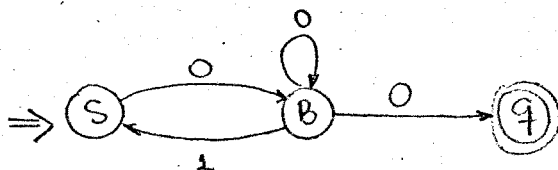
Exercício : Mostre que $L(G) = L(A)$!

Exemplo :

$G = (\{S, B\}, \{0, 1\}, P, S)$ com

$P = \{ S \rightarrow 0B, B \rightarrow 0B, B \rightarrow 1S, B \rightarrow \emptyset \}$

Então :



Notação: transição no autômato finito

$$(q, ax) \vdash (q', x) \Leftrightarrow \delta(q, a) = q'$$

Com essa notação, pode-se escrever:

$$L(A) = \{ x \in \Sigma^* \mid (q_0, x) \vdash^* (q, \epsilon) \text{ , } q \in F \}$$

Exemplo: Para a gramática e autômato do exemplo anterior, pode-se escrever:

$$S \Rightarrow 0B \Rightarrow 00B \Rightarrow 001S \Rightarrow 0010B \Rightarrow 00100$$

$$(S, 00100) \vdash (B, 0100) \vdash (B, 100) \vdash (S, 00) \vdash (B, 0) \vdash (q, \epsilon)$$

Teorema 2.9. Seja $A = (Q, \Sigma, \delta, q_0, F)$ um autômato finito. Então existe uma gramática regular G tal que $L(G) = L(A)$.

Prova: Seja a gramática $G = (N, \Sigma, P, S)$ tal que:

- 1) $N = Q$
- 2) $S = q_0$
- 3) P construído por:

$$a) \quad q \rightarrow aq' \in P \quad \text{se} \quad \delta(q, a) = q' \quad ; \quad q, q' \in Q, \quad a \in \Sigma$$

$$b) \quad q \rightarrow \epsilon \in P \quad \text{se} \quad \delta(q, a) \in F$$

Deve-se mostrar que a gramática G assim construída é tal que

$L(q) = L(\lambda)$, ou seja, $\forall w \in \Sigma^*$,

$$q \xRightarrow{*} w \iff (q, w) \vdash^* (q', \lambda), q' \in F$$

Inicialmente, será mostrado que para $\forall q \in Q$

$$q \xRightarrow{*} w \iff (q, w) \vdash^* (q', \lambda), q' \in F$$

por indução no comprimento de w .

Base. $i = 0$ (ou seja, $w = \lambda$)

$$q \Rightarrow \lambda \iff (q, \lambda) \vdash (q, \lambda), q \in F$$

(pois de (3b) tem-se: $q \rightarrow \lambda \in P \iff \delta(q, \lambda) \in F$)

Hipótese indutiva $|w| = i$, $i \geq 0$

$$q \xRightarrow{i} w \iff (q, w) \vdash^i (q', \lambda), q' \in F$$

Passo da indução. Seja $w = ax$ tal que $|x| = i$, $a \in \Sigma$.

$$q \xRightarrow{i+1} w \iff q \Rightarrow aB \xRightarrow{i} ax$$

ou seja,

$$1) B \xRightarrow{i} x$$

$$2) \delta(q, a) = B \quad (\text{pois } q \rightarrow aB \in P)$$

Mas, pela hipótese indutiva,

$$B \stackrel{i}{\Rightarrow} x \Leftrightarrow (B, x) \vdash^i (q', \perp) ; q' \in F$$

Logo:

$$\begin{aligned} q \stackrel{i+1}{\Rightarrow} w &\Leftrightarrow (q, ax) \vdash (B, x) \vdash^i (q', \perp) ; q' \in F \\ &\Leftrightarrow (q, w) \vdash^{i+1} (q', \perp) ; q' \in F \end{aligned}$$

Portanto: $\forall q \in Q$

$$q \stackrel{*}{\Rightarrow} w \Leftrightarrow (q, w) \vdash^* (q', \perp) ; q' \in F.$$

Como esse resultado vale para todo $q \in Q$, particularizando para $q = q_0$ tem-se:

$$q_0 \stackrel{*}{\Rightarrow} w \Leftrightarrow (q_0, w) \vdash^* (q', \perp) ; q' \in F$$

$$w \in L(q) \Leftrightarrow w \in L(A)$$

Logo: $L(q) = L(A).$

EXPRESSÕES REGULARES

definição 2.18: Seja Σ um alfabeto. As expressões regulares sobre Σ e os conjuntos que elas representam são definidas recursivamente como:

- a) ϕ é uma expressão regular e representa o conjunto vazio ϕ
- b) \perp é uma expressão regular e representa o conjunto $\{\perp\}$
- c) se $a \in \Sigma$ então a é uma expressão regular e representa o conjunto $\{a\}$
- d) se r e s são expressões regulares representando, respectivamente, os conjuntos R e S , então $(r+s)$, (rs) , (r^*) são expressões regulares representando $R \cup S$, RS e R^* , respectivamente.

Para poder eliminar alguns parênteses, assume-se que a prioridade das operações é:

- 1) fecho
- 2) concatenação
- 3) soma

Exemplo: $((0(1^*)) + 0)$ pode ser reescrita como $01^* + 0$.

Teorema 2.10. Seja r uma expressão regular e $L(r)$ a linguagem representada por r . Então existe um a.f.n.d. A tal que $L(A) = L(r)$.

Prova (por indução sobre o número de operadores na expressão r)

Base: r tem 0 operadores. Neste caso r pode ser ϕ , \perp ou a ($a \in \Sigma$). Mas essas expressões representam conjuntos que são reconhecidos, respectivamente, por:

$$(r = \phi) \Rightarrow \textcircled{q_0} \quad \textcircled{q}$$

$$(r = \lambda) \Rightarrow \textcircled{q_0}$$

$$(r = a, a \in \Sigma) \Rightarrow \textcircled{q_0} \xrightarrow{a} \textcircled{q}$$

Hipótese indutiva : o lema é válido para expressões regulares com menos do que i operadores, $i \geq 1$

Passo da indução : r tem i operadores. Neste caso pode-se assumir que :

- 1) $r = r_1 + r_2$ ou
- 2) $r = r_1 r_2$ ou
- 3) $r = r_1^*$

Seja então :

1) $r = r_1 + r_2$. Logo r_1 e r_2 tem menos do que i operadores.

Portanto existem $A_1 = (Q_1, \Sigma, \delta_1, q_1, \{f_1\})$ e $A_2 = (Q_2, \Sigma, \delta_2, q_2, \{f_2\})$ tais que $L(A_1) = L(r_1)$ e $L(A_2) = L(r_2)$.

Deve-se construir A tal que $L(A) = L(r)$. Seja então

$$A = (Q_1 \cup Q_2 \cup \{q_0, f_0\}, \Sigma, \delta, q_0, \{f_0\}) \text{ com } \delta$$

definida por :

- a) $\delta(q_0, \lambda) = \{q_1, q_2\}$

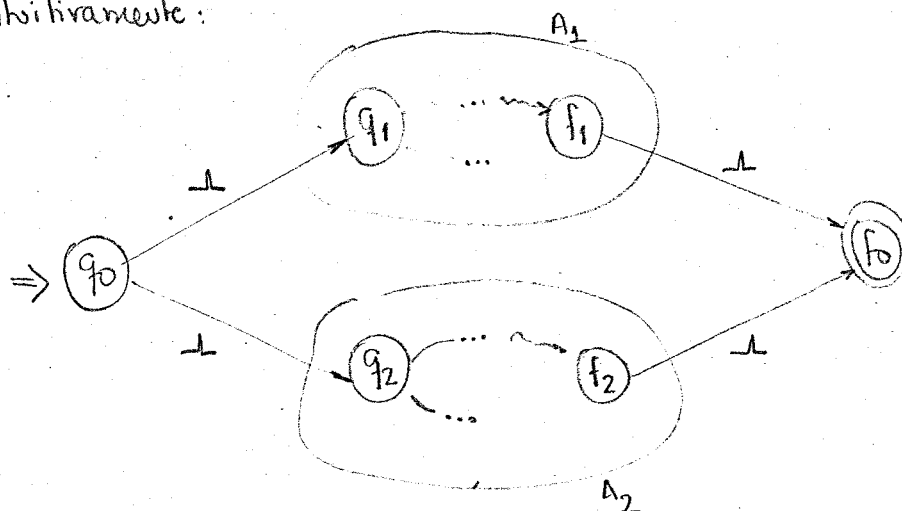
$$b) \delta(q, a) = \delta_i(q, a), \quad q \in Q_i - \{f_i\}$$

$$a \in \Sigma \cup \{\lambda\}$$

$$c) \delta(q, a) = \delta_2(q, a) \quad q \in Q_2 - \{f_2\}; a \in \Sigma \cup \{\perp\}$$

$$d) \delta(f_1, \perp) = \delta(f_2, \perp) = \{f_0\}$$

Inicialmente:



Então:

$$x \in L(r) \Leftrightarrow x \in L(r_1) \cup L(r_2) \Leftrightarrow$$

$$\Leftrightarrow (q_1, x) \vdash^* (f_1, \perp) \text{ ou } (q_2, x) \vdash^* (f_2, \perp)$$

$$\Leftrightarrow (q_0, x) \vdash (q_1, x) \vdash^* (f_1, \perp) \vdash (f_0, \perp) \text{ ou}$$

$$(q_0, x) \vdash (q_2, x) \vdash^* (f_2, \perp) \vdash (f_0, \perp)$$

$$\Leftrightarrow x \in L(A) \quad \text{Logo} \quad L(A) = L(r)$$

Exercício: Complete a prova deste teorema, isto é, considere os casos:

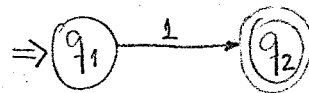
$$2) r = r_1 r_2 \quad e$$

$$3) r = r_1^*$$

Exemplo: $r = 01^* + 1$. Construir automato A tal que $L(A) = L(r)$.

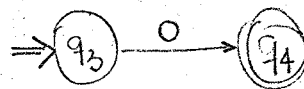
Pode-se escrever: $r = r_1 + r_2$ com $r_1 = 01^*$ e $r_2 = 1$

autômato para r_2 :



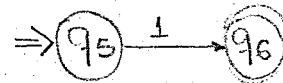
Seja $r_1 = r_3 r_4$ com $r_3 = 0$ e $r_4 = 1^*$

autômato para r_3 :



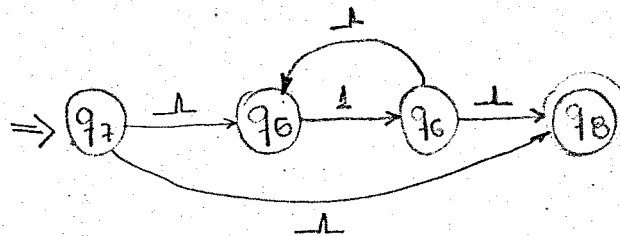
Seja $r_4 = r_5^*$ com $r_5 = 1$

autômato para r_5 :

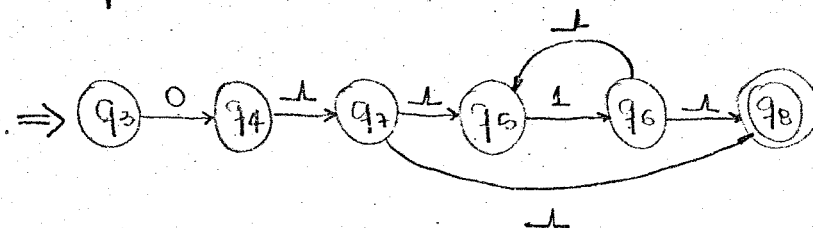


Então:

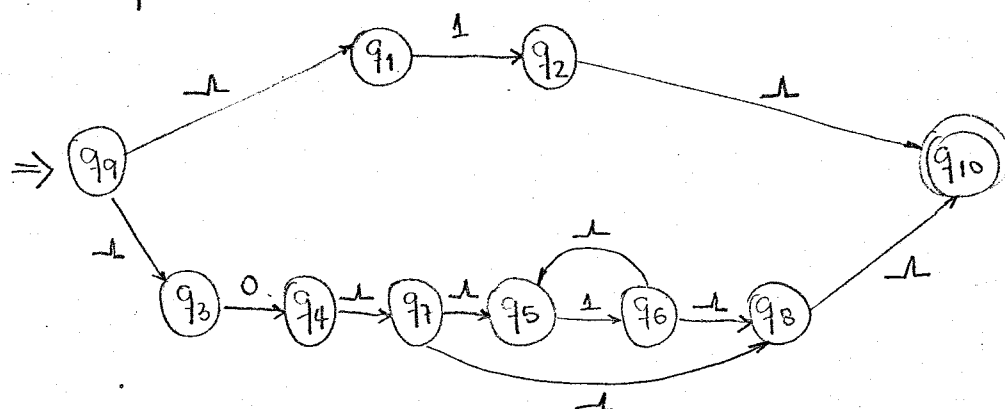
autômato para r_4 :



autômato para r_1 :



autômato para r :



Teorema 2.11. Seja $L = L(A)$ para algum autômato finito (determinístico) A . Então $L = L(r)$ para alguma expressão regular r .

Prova. Seja $A = (\{q_1, \dots, q_n\}, \Sigma, \delta, q_1, F)$.

Seja R_{ij}^k o conjunto das cadeias $x \in \Sigma^*$ tais que $\delta(q_i, x) = q_j$ e x

$\delta(q_i, y) = q_i$ para y prefixo de x ($y \neq \lambda, y \neq x$)

estão $l \leq k$. (Intuitivamente, R_{ij}^k é o conjunto das cadeias que fazem o autômato ir do estado q_i para o estado q_j sem passar (isto é, entrar e sair) por qualquer estado de número maior do que k).

(Note que i e/ou j podem ser maiores do que k)

(Note também que, como não existe estado de número maior do que n , R_{ij}^n é o conjunto de todas as cadeias que levam o autômato do estado q_i para o estado q_j)

Pode-se definir R_{ij}^k recursivamente como:

$$R_{ij}^0 = \begin{cases} \{a \in \Sigma / \delta(q_i, a) = q_j\} & \text{se } i \neq j \\ \{a \in \Sigma / \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{se } i = j \end{cases}$$

$$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$$

Deve-se mostrar que para todo i, j, k , existe uma expressão regular r_{ij}^k que representa R_{ij}^k . Será mostrado por indução em k .

Base: $k=0$.

R_{ij}^0 é um conjunto de símbolos x tais que $x \in \Sigma$ ou $x = \epsilon$.
ou então $R_{ij}^0 = \emptyset$.

Se $R_{ij}^0 = \{a_1, a_2, \dots, a_p\}$, $a_i \in \Sigma \cup \{\epsilon\}$ então

$r_{ij}^0 = a_1 + a_2 + \dots + a_p$. Se $R_{ij}^0 = \emptyset$ então $r_{ij}^0 = \emptyset$.

Passo: a definição recursiva R_{ij}^k envolve apenas quadros de expressões regulares: união, concatenação e fecho. Por hipótese indutiva, para todo m, n existe uma expressão regular

$$r_{mn}^{k-1} \text{ tal que } R_{mn}^{k-1} = L(r_{mn}^{k-1})$$

Portanto, $r_{ij}^k = r_{ij}^{k-1} + (r_{ik}^{k-1})(r_{kk}^{k-1})^*(r_{kj}^{k-1})$

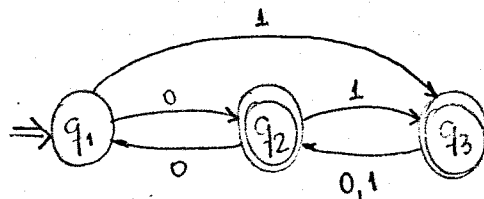
Para completar a prova, deve-se observar que:

$$L(A) = \bigcup_{q_j \in F} R_{ij}^n$$

e portanto, $L(A)$ pode ser representado por

$$r_{ij_1}^n + \dots + r_{ij_p}^n \quad \text{tal que} \quad F = \{q_{j_1}, \dots, q_{j_p}\}$$

Exemplo: Seja o autômato



Neste caso $L(A) = R_{12}^3 \cup R_{13}^3 = r_{12}^3 + r_{13}^3$

	$k=0$	$k=1$	$k=2$	$k=3$
r_{11}^k	1	$r_{11}^1 = r_{11}^0 + (r_{11}^0)(r_{11}^0)^*(r_{11}^0) = 1$		
r_{12}^k	0	$r_{12}^1 = r_{12}^0 + (r_{11}^0)(r_{11}^0)^*(r_{12}^0) = 0 + 0 = 0$		
r_{13}^k	1			
r_{21}^k	0			
r_{22}^k	1			
r_{23}^k	0			
r_{31}^k	0			
r_{32}^k	0+1			
r_{33}^k	1			

Exercício: completar essa tabela e determinar $L(A)$!

"PUMPING LEMMA" PARA CONJUNTOS REGULARES

O teorema a seguir estabelece uma condição necessária sobre as cadeias de um conjunto regular. Ele é uma ferramenta importante para provar que certos conjuntos não são regulares.

Teorema 2.12. Seja $A = (Q, \Sigma, \delta, q_0, F)$ um autômato finito com n estados. Seja $w \in L(A)$ tal que $|w| \geq p$. Se $p \geq n$ então existem $x, y, z \in \Sigma^*$ tais que $w = xyz$, $y \neq \epsilon$ e para todo $k \geq 0$, $xy^kz \in L(A)$.

Prova.

Sejam $p = n$, $w \in L(A)$, $|w| \geq n$. Logo, durante o processamento de w o autômato utiliza pelo menos $(n+1)$ estados. Como $\#Q = n$, deve haver repetição de pelo menos um estado. Seja q um estado repetido.

Sejam então x, y e z tais que, $w = xyz$ e

$$(q_0, xyz) \vdash^* (q, yz) \vdash^+ (q, z) \vdash^* (q', \epsilon), \quad q' \in F$$

Portanto $(q, yz) \vdash^+ (q, z)$ então

$$(q, y^k z) \vdash^+ (q, y^{k-1} z) \vdash^+ \dots \vdash^+ (q, z) \in$$

portanto

$$xy^k z \in L(A), \quad k \geq 0$$

Exemplo:

$$L = \{ 0^n 1^n / n \geq 1 \} \text{ não é regular.}$$

Admitindo que L é regular, existe autômato finito $A = (Q, \Sigma, \delta, q_0, F)$ com p estados tal que $L = L(A)$. Como o número p de estados do autômato é finito e como em L existem cadeias arbitrariamente grandes, será sempre possível encontrar um n tal que

$$|0^n 1^n| \geq p$$

Neste caso, pelo "PUMPING LEMMA", existem $x, y, z \in \{0, 1\}^*$ tais que

$$0^n 1^n = xyz, \quad y \neq \epsilon \quad \text{e} \quad xy^k z \in L \quad (k \geq 0)$$

Mas se $xyz = 0^n 1^n$ então y pode ser de apenas três formas:

a) $y \in L(0^+)$ e neste caso, $xz \notin L$, por exemplo

b) $y \in L(1^+)$ e neste caso, $xz \notin L$, por exemplo

c) $y \in L(0^+ 1^+)$ e neste caso $xy^2 z \notin L$, por exemplo

Portanto, a hipótese de L ser regular contradiz o "PUMPING LEMMA". Logo, L não é regular.

PROPRIEDADES DE CONJUNTOS REGULARES.

Teorema 2.13 A classe de linguagens regulares é fechada sob complementação, ou seja, se L é uma linguagem regular sobre Σ^* então $\Sigma^* - L$ é linguagem regular.

Prova: Como L é regular, existe $A = (Q, \Sigma, \delta, q_0, F)$ tal

que $L = L(A)$.

Seja $A' = (Q, \Sigma, \delta, q_0, Q-F)$. Neste caso:

$$x \in \Sigma^* - L \Leftrightarrow x \notin L \Leftrightarrow \delta(q_0, x) \notin F \Leftrightarrow$$

$$\Leftrightarrow \delta(q_0, x) \in Q-F \Leftrightarrow x \in L(A')$$

Logo: $L(A') = \Sigma^* - L$ e portanto $\Sigma^* - L$ é regular.

Teorema 2.14. A classe de linguagens regulares é fechada sob união, ou seja, se X e Y são conjuntos regulares, então $X \cup Y$ é conjunto regular.

Prova. Sejam $A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$

$B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ tais que

$$X = L(A) \text{ e } Y = L(B).$$

$$\text{Seja } q' \notin Q_A \cup Q_B \text{ e } C = (Q_A \cup Q_B \cup \{q'\}, \Sigma, \delta_C, q', F_C)$$

(a.f.n.d.) tal que

$$F_C = \begin{cases} F_A \cup F_B & \text{se } \lambda \notin L(A) \cup L(B) \\ F_A \cup F_B \cup \{q'\} & \text{se } \lambda \in L(A) \cup L(B) \end{cases}$$

e δ_C definida por:

$$\delta_C(q', a) = \{ \delta_A(q_{0A}, a) \} \cup \{ \delta_B(q_{0B}, a) \}, a \in \Sigma$$

$$\delta_c(q, a) = \delta_A(q, a) \quad , \quad q \in Q_A ; a \in \Sigma \cup \{\epsilon\}$$

$$\delta_c(q, a) = \delta_B(q, a) \quad , \quad q \in Q_B ; a \in \Sigma \cup \{\epsilon\}$$

Seja $x \in X \cup Y$. Dois casos podem ocorrer:

(a) $x = \epsilon$. Logo $\epsilon \in L(A)$ ou $\epsilon \in L(B)$
Portanto

$$\delta_c(q', \epsilon) = q' \in F_c$$

Logo: $\epsilon \in L(c)$.

(b) $x \neq \epsilon$. Seja $x = aw$, $a \in \Sigma$, $w \in \Sigma^*$

Então:

$$aw \in X \cup Y \Leftrightarrow aw \in X \text{ ou } aw \in Y$$

$$\Leftrightarrow (q_{0A}, aw) \vdash (q_{1A}, w) \vdash^* (q_{fA}, \epsilon) , q_{fA} \in F_A \quad \text{ou}$$

$$(q_{0B}, aw) \vdash (q_{1B}, w) \vdash^* (q_{fB}, \epsilon) , q_{fB} \in F_B$$

$$\Leftrightarrow (q', aw) \vdash (q_{1A}, w) \vdash^* (q_{fA}, \epsilon) \quad \text{ou}$$

$$(q', aw) \vdash (q_{1B}, w) \vdash^* (q_{fB}, \epsilon)$$

$$\Leftrightarrow \delta(q', aw) \in F_A \cup F_B \Leftrightarrow aw \in L(c)$$

Logo: $L(c) = X \cup Y$

e $X \cup Y$ é regular.

Teorema 2.15. Se X e Y são conjuntos regulares, então $X \cap Y$ é um conjunto regular.

Prova. Exercício!

PROBLEMAS DE DECISÃO

Um problema de decisão tem solução se e somente se existe um algoritmo que resolve o problema para todas as possíveis condições. Se tal algoritmo existe, o problema é DECIDÍVEL; caso contrário o problema é INDECIDÍVEL.

1. "A linguagem reconhecida por um autômato A é vazia?" é decidível.

$A = (Q, \Sigma, \delta, q_0, F)$ qualquer. Verificar se $\exists q \in Q$ tal que $q \in F$ e q é acessível.

2. "Sejam X e Y conjuntos regulares. $X \subseteq Y$ ou $X = Y$?" é decidível.

Sejam $X = L(A)$ e $Y = L(B)$ para dois autômatos finitos A e B . Então existe um algoritmo para verificar se A e B são equivalentes:

$A \equiv B$ se para todo estado q_i de A existe um estado q_j de B tal que $q_i \equiv q_j$ e reciprocamente. Basta então construir o autômato que reconhece $X \cup Y$ e aplicar o algoritmo de minimização de estados.

Então: $X=Y$ se $A=B$.

Por outro lado, se $X \subseteq Y$ então $X \cap Y = X$. Logo para saber se $X \subseteq Y$, basta construir o autômato que reconhece $X \cap Y$ e verificar se ele é equivalente a A .

(uma outra ideia para resolver este problema é mostrada em
HANNA, Z. - "MATHEMATICAL THEORY OF COMPUTATION")

3. "Seja X conjunto regular. X é finito ou infinito?" é decidível.

Exercício! (sugestão: "PUMPING LEMMA")

AUTÔMATOS FINITOS BILATERAIS ("TWO-WAY FINITE AUTOMATA")

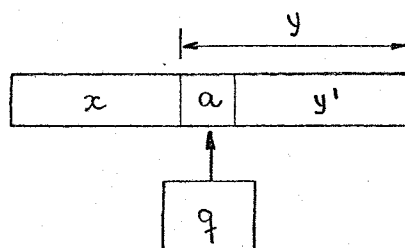
definição 2.19 Um autômato finito bilateral é uma 5-upla $A = (Q, I, \delta, q_0, F)$ onde Q, I, q_0 e F são definidos como no caso de autômato finito e δ é uma função

$$\delta: Q \times I \rightarrow \{-1, 0, 1\} \times Q$$

$\delta(q, a) = (d, q')$ \Leftrightarrow autômato estando no estado q e lendo o símbolo a , move sua cabeça uma posição para a esquerda se $d = -1$, permanece estacionário se $d = 0$ ou move sua cabeça uma posição para a direita se $d = 1$ e vai para o estado q' .

definição 2.20

uma configuração de um a.f.b $A = (Q, \Sigma, \delta, q_0, F)$ é um elemento de $\Sigma^* Q \Sigma^*$, $(x q y)$, interpretado como:



definição 2.21

$$a_1 \dots a_{i-1} q a_i \dots a_n \vdash a_1 \dots q' a_{i+d} \dots a_n \Leftrightarrow$$

$$\Leftrightarrow \delta(q, a_i) = (d, q')$$

definição 2.22

A linguagem reconhecida por um a.f.b $A = (Q, \Sigma, \delta, q_0, F)$ é definida como:

$$L(A) = \{ x \in \Sigma^* / q_0 x \vdash^* x q, q \in F \}$$

Um autômato finito bilateral rejeita cadeias de uma das seguintes maneiras:

- (a) saindo à esquerda
- (b) entrando em "loop"
- (c) saindo à direita para um estado não final

Exemplo:

δ	0	1
$\Rightarrow q_0$	$(1, q_1)$	$(-1, q_1)$
q_1	$(-1, q_0)$	$(0, q_2)$
(q_2)	$(-1, q_1)$	$(1, q_2)$

Para este autômato, tem-se :

$$1) \quad q_0 011 \vdash 0q_1 11 \vdash 0q_2 11 \vdash 01q_2 1 \vdash 011q_2$$

e como $q_2 \in F$, $011 \in L(A)$

$$2) \quad q_0 00 \vdash 0q_1 0 \vdash q_0 00 \vdash 0q_1 0 \vdash \dots \quad \text{"loop"}$$

portanto $00 \notin L(A)$

Teorema 2.16 . Para todo a.f.b $A = (Q, \Sigma, \delta, q_0, F)$, $L(A)$ é um conjunto regular.

Prova (INFORMAL)

Para todo $x \in \Sigma^*$, definir a função

$$T_x : Q \cup \{0\} \rightarrow Q \cup \{0\} \quad 0 \notin Q$$

da seguinte maneira :

$$a) \quad \text{se } x = x'a \quad (a \in \Sigma, x' \in \Sigma^*) \text{ e } x'qa \vdash^* x'aq' \\ \text{então } T_x(q) = q' \\ \text{Caso contrário, } T_x(q) = 0$$

$$b) \quad \text{se } q_0 x \vdash^* xq' \text{ então } T_x(0) = q'$$

$$\text{Caso contrário, } T_x(0) = 0.$$

Utilizando a relação de Nerode

$$x R_{L(A)} y \iff (\forall z \in \Sigma^*) (xz \in L(A) \iff yz \in L(A))$$

pode-se provar que: $\hat{T}_x = \hat{T}_y \Rightarrow x R_{L(A)} y$

Deste resultado segue que:

$$i(R_{L(A)}) \leq \#\{\hat{T}_x \mid x \in \Sigma^*\}$$

Mas se $\#Q = n$, então o número de funções \hat{T}_x diferentes é, no máximo,

$$(n+1)^{n+1}$$

Portanto, $i(R_{L(A)})$ é finito e $L(A)$ é regular.

A ideia da prova deste leorema é de:

SHEPHERDSON, J.C. "The reduction of two-way automata to one-way automata", IBM Journal of Research and Development, 3, 1959

Para uma prova formal ver também:

HARRISON, M.A. "Introduction to formal language theory"
pg. 65-70.