

Todos automata finito não determinísticos (AFND) é um automata finito determinísticos (AFD), mas será que vale a volta? Vimos que podemos expressar um AFND por uma função

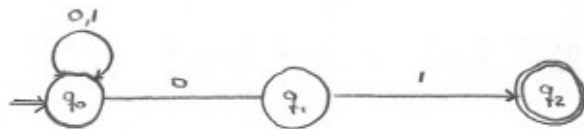
$$\delta: Q \times \Sigma \longrightarrow \mathcal{P}(Q)$$

Podemos criar uma função:

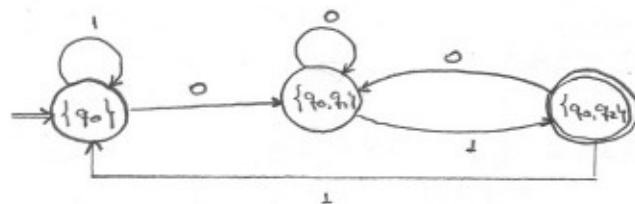
$$\delta': \mathcal{P}(Q) \times \Sigma \longrightarrow \mathcal{P}(Q)$$

onde $\delta'(\Omega, a) = \bigcup_{q \in \Omega} \delta(q, a)$. Assim, podemos pensar em

um AFND como um AFD cujo conjunto de estados é $\mathcal{P}(Q)$, onde Q é o conjunto de estados. Por exemplo, vamos gerar um AFD a partir do automata:

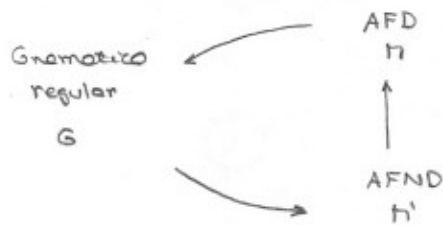


Temos:

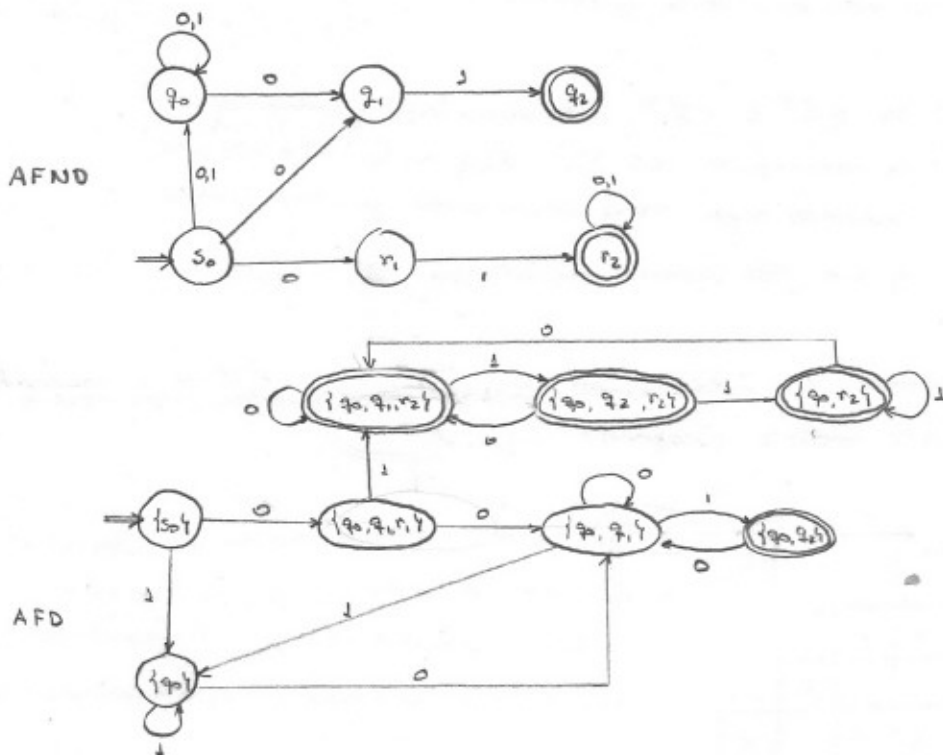


Dado um AFND $M = \langle \Sigma, Q, \delta, q_0, F \rangle$, construímos um AFD $M' = \langle \Sigma, \mathcal{P}(Q), \delta', \{q_0\}, F' \rangle$ onde δ' é definido como acima e $F' = \{X \in \mathcal{P}(Q) \mid X \cap F \neq \emptyset\}$.

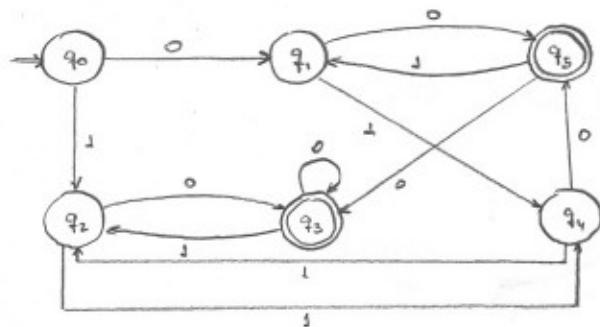
Vimos que dado um AFD M , existe uma gramática regular G tal que $L(G) = T(M)$. E, dado uma gramática regular G , existe um AFND M' tal que $L(G) = T(M')$. Por fim, mostramos que dado um AFND M' , existe um AFD M tal que $T(M) = T(M')$. Assim, mostramos a seguinte sequência de equivalências, que define uma equivalência:



Vejamos outro exemplo de conversão de AFND para AFD. Considere o seguinte autômato que reconhece todas as sequências que começam em 10 ou terminam em 01.



Dado uma linguagem, mostramos que existe um autômato M tal que $L = T(M)$. Na verdade existem vários autômatos que reconhecem esta linguagem. Queremos agora aquele que o menor possível, isto é, aquele com o menor número de estados. Dado um autômato como a abaixo:



São que existem estados que podemos juntar por produzir um autômato com menos estados? Dado um autômato qualquer, e estados q e r , quando não podemos juntar q e r ?

- 1) se $q \in F$ e $r \notin F$, não podemos juntar q e r .
- 2) se houver um $a \in \Sigma$, $\delta(q, a)$ e $\delta(r, a)$ são estados que não podemos juntar então q e r não podem ser juntos.

Analisemos que estados não podem ser juntos. Considere a seguinte matriz diagonal inferior:

1					
2					
3	X	X	X		
4				X	
5	X	X	X		X
	0	1	2	3	4

Às lado estão marcados os estados que não podem ser juntos pois um é final e o outro não é. Analisemos os demais usando a regra recursiva (2).

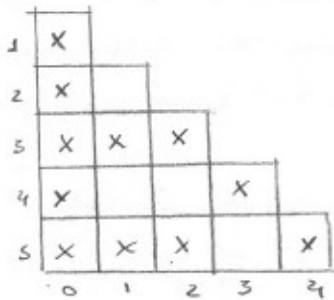
Analisando a junção de (0,1),

temos:

$$q_1 \xleftarrow{0} q_0 \xrightarrow{1} q_2$$

$$q_3 \xleftarrow{0} q_1 \xrightarrow{1} q_4$$

Como q_1 e q_3 não podem ser juntos, q_0 e q_2 também não podem. Prosseguindo pelo primeiro colun, temos como ao lado. O próximo a ser analisado é a junção de (1,2).



$$q_3 \xleftarrow{0} q_1 \xrightarrow{1} q_4$$

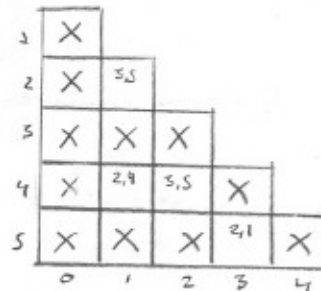
$$q_3 \xleftarrow{0} q_2 \xrightarrow{1} q_4$$

Assim, se (3,5) não podem ser juntos, então (1,2) também não podem ser juntos. Assim, o resultado de (1,2) depende de (3,5) logo colocamos X nos demais quadrad. Quando analisarmos (3,5) vamos:

$$q_3 \xleftarrow{0} q_5 \xrightarrow{1} q_2$$

$$q_3 \xleftarrow{0} q_3 \xrightarrow{1} q_1$$

Assim (3,5) depende de (2,3). Escrevendo equies que sabemos não podem ser juntos e as dependências, temos o gráfico ao lado.

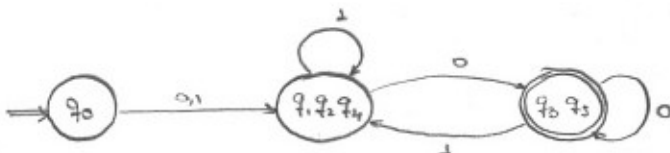


$$(2,1) \longleftrightarrow (3,5)$$

$$(1,4) \longrightarrow (2,4)$$

Até lim, temos os estados que não podem ser juntos e dependências circulares.

Essas dependências circulares correspondem a junções que podem ser feitas (e se é um teorema!) assim, podemos juntar 1,2 e 4 e 3 e 5. Logo, temos:



Expressões regulares: Usamos expressões regulares para representar linguagens regulares. Dado um alfabeto $\Sigma = \{0, 1\}$, temos a seguinte correspondência:

Expressão regular	Linguagem
\emptyset	\emptyset
0	$\{0\}$
1	$\{1\}$
ϵ	$\{\epsilon\}$
01	$\{01\}$
$0+1$	$\{0, 1\}$
$01+1$	$\{01, 1\}$
$0(0+1)$	$\{00, 01\}$
$0(1+\epsilon)$	$\{01, 0\}$
0^*	$\{\epsilon, 0, 00, 000, 0000, \dots\}$
$(01)^*$	$\{\epsilon, 01, 0101, 010101, \dots\}$
$(0+1)^*$	$\{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$
1^+	$\{1, 11, 111, 1111, \dots\}$

Temos as seguintes igualdades:

$$0^* = 0^+ + \epsilon$$

$$0^+ = 0 0^*$$

Podemos representar as linguagens regulares usando expressões regulares. Por exemplo:

- termina com 0: $(0+1)^*0$
- começa com 10: $10(0+1)^*$
- penúltimo termo é 0: $(0+1)^*0(0+1)(0+1)$

Exercício: Escreva uma expressão regular que represente os números na base 4 com o último algarismo aparecendo no máximo 1 vez antes $\Sigma = \{0, 1, 2, 3\}$.

$$\begin{aligned} &(1+2+3)^*(0+E)(1+2+3)^*0 + \\ &(0+2+3)^*(1+E)(0+2+3)^*1 + \\ &(0+1+3)^*(2+E)(0+1+3)^*2 + \\ &(0+1+2)^*(3+E)(0+1+2)^*3 \end{aligned}$$

A definição de expressões regulares é recursiva.
Seja ER o conjunto das expressões regulares:

$$\emptyset, E \in ER \quad \Sigma_i \subseteq ER$$

se $E \in ER$ então $E^*, E^+ \in ER$ e, além disso

$E_1, E_2 \in ER$ então: $E_1 E_2, E_1 + E_2 \in ER$. Cada expressão regular representa uma linguagem. Considere " $E \rightarrow L$ " significa a expressão regular E representa a linguagem L . Temos:

$$\emptyset \rightarrow \emptyset \quad \epsilon \rightarrow \{\epsilon\} \quad a \rightarrow \{a\}, \forall a \in \Sigma$$

$$E_1 \rightarrow L_1 \text{ e } E_2 \rightarrow L_2 \text{ então } E_1 + E_2 \rightarrow L_1 \cup L_2$$

$$\text{se } E_1, E_2 \rightarrow L_i \text{ então } L = \{w_1 w_2 \mid w_1 \in L_1, \text{ e } w_2 \in L_2\}$$

Para E^* podemos dar várias definições equivalentes, feitas de forma recursiva. Uma delas é a de que $E \rightarrow L$ e $E^* \rightarrow L'$ então L' é o menor conjunto tal que.

$$\begin{aligned} \epsilon &\in L' \quad L \subseteq L' \\ w_1, w_2 \in L' &\implies w_1 w_2 \in L' \end{aligned}$$