Universidade Católica de Pelotas
Escola de informática
058814 Linguagens Formais e Autômatos

TEXTO 1

Linguagens Regulares e Autômatos Finitos

Prof. Luiz A M Palazzo
Março de 2008

Hierarquia de Chomsky

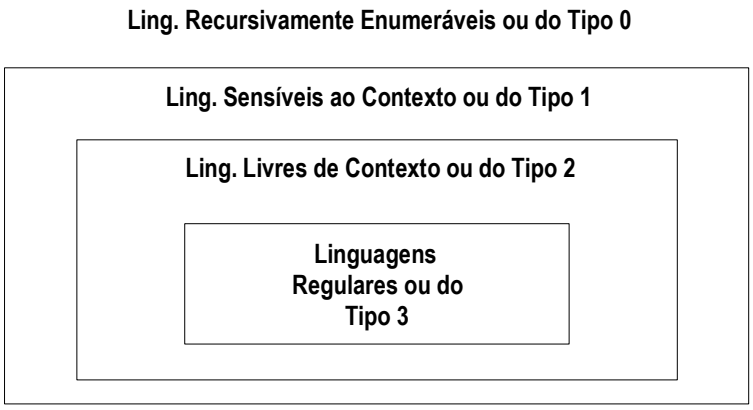


Fig 1: Hierarquia de Chomsky (Lingüística)

O estudo das linguagens regulares (ou linguagens do tipo 3 na Hierarquia de Chomsky, as mais simples, permitindo abordagens de pequena complexidade, grande eficiência e fácil implementação) pode ser abordado através de 3 diferentes formalismos:

- *operacional ou reconhecedor*: Autômato Finito, que pode ser determinístico, não determinístico ou com movimento vazio.

- *axiomático ou gerador*: Gramática Regular

- *denotacional*: Expressão Regular (também pode ser considerado gerador).

Sistemas de Estados Finitos

- Um sistema de estados finitos é um modelo matemático de um sistema com entradas e saídas discretas.

- Pode assumir um número *finito* e *pré-definido* de estados.

- Cada estado resume somente as informações do passado necessárias para determinar as ações para a próxima entrada.

- Podem ser associados a diversos tipos de sistemas naturais e construídos.

- Exemplo: Elevador (1) Não memoriza instruções anteriores. (2) Cada estado sumariza as informações: *andar corrente* e *direção do movimento*. (3) As entradas para o sistema são requisições pendentes.

- Outros exemplos de sistemas de estados finitos: analisadores léxicos e processadores de texto

- SEF de difícil manipulação: (1) O cérebro humano, com 2^{35} células e tamanha complexidade que esta abordagem se torna ineficiente. (2) O computador onde o estudo adequado da computabilidade exige uma memória sem limite pré-definido.

Autômatos Finitos

Um autômato finito determinístico, ou simplesmente autômato finito, pode ser vista como uma máquina composta basicamente por três partes:

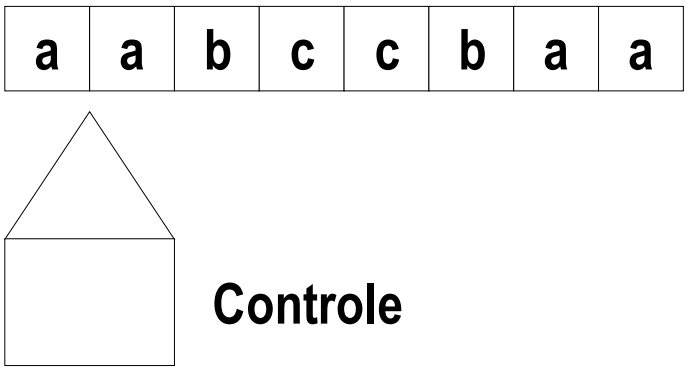


Figura 2: Autômato Finito como uma máquina com controle finito.

a. *Fita*: Dispositivo de entrada que contém a informação a ser processada. A fita é finita à esquerda e à direita. É dividida em células onde cada uma armazena um símbolo. Os símbolos pertencem a um alfabeto de entrada. Não é possível gravar sobre a fita. Não existe memória auxiliar. Inicialmente a palavra a ser processada, isto é, a informação de entrada ocupa toda a fita.

b. *Unidade de Controle*: Reflete o estado corrente da máquina. Possui uma unidade de leitura (cabeça de leitura, que acessa uma unidade da fita de cada vez. Pode assumir um número finito e pré-definido de estados. Após cada leitura a cabeça move-se uma célula para a direita.

c. *Programa ou Função de Transição*: Função que comanda as leituras e define o estado da máquina. Dependendo do estado corrente e do símbolo lido determina o novo estado do autômato. Usa-se o conceito de estado para armazenar as informações necessárias à determinação do próximo estado, uma vez que não há memória auxiliar.

Definição: Autômato Finito Determinístico (AFD)

Um *autômato finito determinístico* (AFD), ou simplesmente *autômato finito* M é uma quíntupla:

$$M = (\Sigma, Q, \delta, q0, F),$$

onde:

- Σ - Alfabeto de símbolos de entrada
- Q - Conjunto finito de estados possíveis do autômato
- δ - Função programa ou função de transição δ: Q x $\Sigma \rightarrow$ Q
- q_0 - Estado inicial tal que $q_0 \in Q$
- F - Conjunto de estados finais, tais que $F \subseteq Q$.

A função programa pode ser representada como um grafo orientado finito conforme representado abaixo:

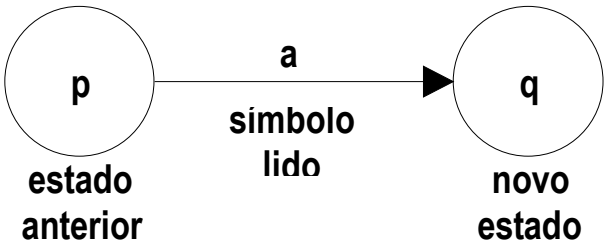


Figura 3: Representação da Função programa como um grafo

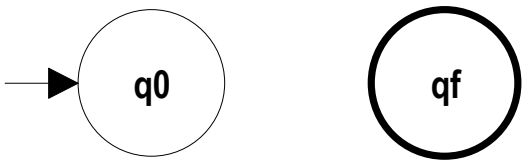


Figura 4: Representação dos estados inicial e final como nodos de um grafo

O processamento de um autômato finito M para uma palavra de entrada w consiste na sucessiva aplicação da função programa para cada símbolo de w, da esquerda para direita, até ocorrer uma condição de parada.

Exemplo: Autômato Finito

O autômato finito $M_1 = (\{a, b\}, \{q_0, q_1, q_2, q_f\}, \delta_1, q_0, \{q_f\})$, onde δ_1 é representada pela tabela abaixo, reconhece a linguagem

L1 = {w | w possui aa ou bb como subpalavra}

δ_1	a	b
q0	q1	q2
q1	qf	q2
q2	q1	qf
qf	qf	qf

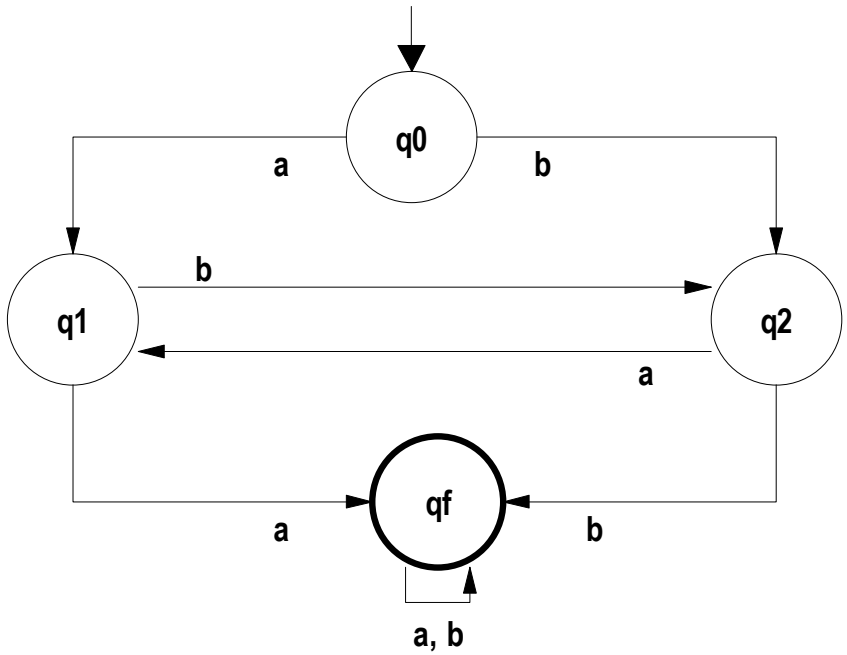


Figura 5: Grafo do autômato finito determinístico

O algoritmo apresentado usa os estados q1 e q2 para "memorizar" o símbolo anterior. Assim q1 representa "o símbolo anterior é a" e q2 representa "o símbolo anterior é b". Após identificar dois aa ou dois bb consecutivos o autômato assume o estado qf (final) e varre o sufixo da palavra de entrada sem

qualquer controle lógico, somente para terminar o processamento. A figura 2.5 ilustra o processamento do autômato finito M1 para a palavra de entrada w = abba, a qual é aceita.

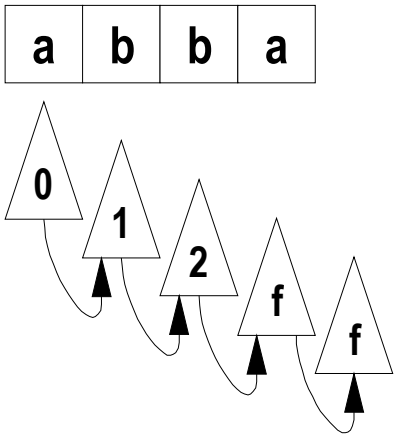


Figura 6: Seqüência de processamento

Note-se que um autômato finito sempre pára ao processar qualquer entrada, pois como toda palavra é finita e como um novo símbolo de entrada é lido a cada aplicação da função programa, não existe a possibilidade de ciclo (loop) infinito. A parada do processamento pode ocorrer de duas maneiras: aceitando ou rejeitando uma entrada w. As condições de parada são as seguintes:

a. Após processar o último símbolo da fita o autômato finito assume um estado final. O autômato para e a entrada w é aceita.
b. Após processar o último símbolo da fita, o autômato finito assume um estado não-final. O autômato para e a entrada w é rejeitada
c. A função programa é indefinida para o argumento (estado corrente e símbolo lido). O autômato para e a entrada w é rejeitada.

Para definir formalmente o comportamento de um autômato finito (ou seja, dar semântica à sintaxe de um autômato finito) é necessário estender a definição da função programa, usando como argumento um estado e uma palavra.

Exercício

Desenvolver AFDs que reconheçam as seguintes linguagens sobre Σ = {a, b}:

a) {w | w possui aaa como subpalavra}

b) {w | o sufixo de w é aa}

c) {w | w possui um número ímpar de a e b}

d) {w | w possui número par de a e ímpar de b ou vice-versa}

e) {w | o quinto símbolo da esquerda para a direita de w é a}

Definição: Função Programa Estendida

Seja M = (Σ, Q, δ, q0, F) um AFD. A *função programa estendida*, denotada por:

$\underline{\delta}$: Q x Σ^* $\rightarrow$ Q

é a função programa δ: Q x Σ $\rightarrow$ Q, estendida para palavras, e é indutivamente definida como se segue:

δ (q, ε) = q

$\underline{\delta}$ (q, aw) = $\underline{\delta}$ (δ (q, a), w)

Exemplo: Função Programa Estendida

Seja o AFD M1 = ({a, b}, {q0,q1,q2,qf}, δ1, q0, {qf}), definida no exemplo anterior. Então a função programa estendida aplicada à palavra abaa a partir do estado inicial q0 é como se segue:

$\underline{\delta}$ (q0, abaa) =	função estendida sobre abaa
$\underline{\delta}$ (δ (q0, a), baa) =	processa abaa
$\underline{\delta}$ (q1, baa) =	função estendida sobre baa
$\underline{\delta}$ (δ (q1, b), aa) =	processa baa
$\underline{\delta}$ (q2, aa) =	função estendida sobre aa
$\underline{\delta}$ (δ (q2, a), a) =	processa aa
$\underline{\delta}$ (q1, a) =	função estendida sobre a
$\underline{\delta}$ (δ (q1, a), ε) =	processa abaa
$\underline{\delta}$ (qf, ε) = qf	função estendida sobre ε. Fim da indução. A palavra é aceita.

Comentários

- Por simplicidade tanto δ quanto $\underline{\delta}$ serão denotadas simplesmente por δ.

- A linguagem aceita por um autômato finito M = (Σ, Q, δ, q0, F) denotada por ACEITA(M), ou L(M), é o conjunto de todas as palavras pertencentes a Σ^* que são aceitas por M, ou seja: ACEITA(M) - {w | δ(q0, w) $\in$ F}.
- Analogamente, REJEITA(M) é o conjunto de todas as palavras pertencentes a Σ^* que são rejeitadas por M.
- As seguintes afirmações são verdadeiras
    a. A intersecção dos conjuntos ACEITA(M) e REJEITA(M) é vazio.
    b. A união dos conjuntos ACEITA(M) e REJEITA(M) é Σ^*.
    c. REJEITA(M) é o complemento de ACEITA(M) em Σ^*
    d. ACEITA(M) é o complemento de REJEITA(M) em Σ^*

Definição: Equivalência de Autômatos Finitos

Dois autômatos M1 e M2 são equivalentes se e somente se:

ACEITA(M1) = ACEITA(M2)

Definição: Linguagens Regulares ou do Tipo 3

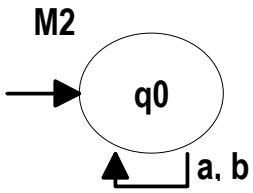
Uma linguagem aceita por um autômato finito é uma *Linguagem Regular* ou do *Tipo 3*.

Exemplo: Autômato Finito

Os autômatos M2 = ({a,b}, {q0}, δ2, q0, $\varnothing$) e M3 = ({a, b}, {q0}, δ3, q0, {q0}) reconhecem respectivamente as linguagens L1 = $\varnothing$ e L2 = Σ^*, onde δ2 e δ3 são representadas abaixo em forma de tabela.

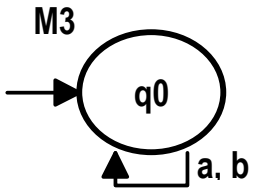
M2

δ2	a	b
q0	q0	q0



M3

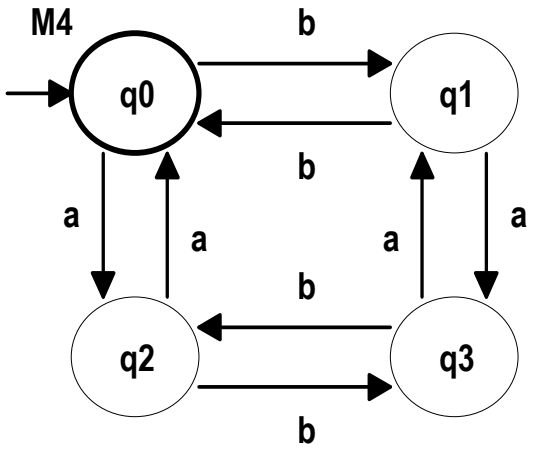
δ3	a	b
q0	q0	q0



Exemplo: Autômato Finito

O autômato M4 = ({a, b}, {q0, q1, qa2, q3}, δ, q0, {q0}), reconhece a linguagem:

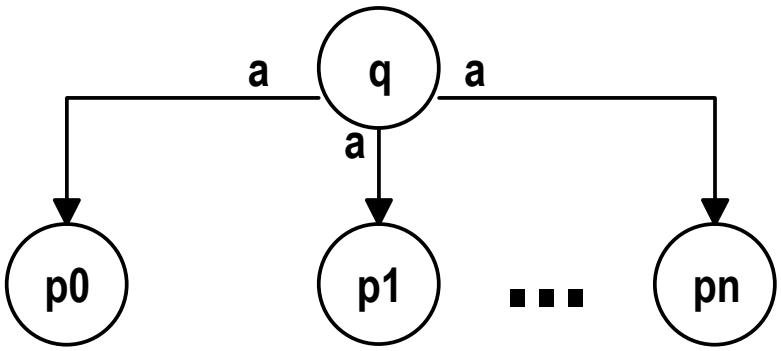
L4 = {w | w possui um número par de a e b}



Autômato Finito Não-Determinístico (AFN)

- Não-determinismo é uma importante generalização dos AF's, essencial para a teoria da computação e para a teoria das linguagens formais.

- Qualquer AFN pode ser simulado por um autômato finito determinístico



- Em AFNs, a função programa leva de um par estado-símbolo a um conjunto de estados possíveis.

- Pode-se entender que o AFN assume simultaneamente todas as alternativas.de estados possíveis {p0, p1, ..., pn} a partir do estado atual ($q \in Q$) e do símbolo recebido ($a \in \Sigma$), como se houvesse uma unidade de controle para processar cada alternativa independentemente, sem compartilhar recursos com as demais.

- Assim o processamento de um caminho não influi no estado, símbolo lido e posição da cabeça dos demais caminhos alternativos.

Definição: Autômato Finito Não-Determinístico (AFN)

Um AFN é uma quíntupla

$$M = (\Sigma, Q, \delta, q0, F),$$

onde:

- Σ - Alfabeto de símbolos de entrada
- Q - Conjunto finito de estados possíveis do autômato
- δ - Função programa ou função de transição δ: Q x $\Sigma \rightarrow 2^Q$, parcial.
- q_0 - Estado inicial tal que $q_0 \in Q$
- F - Conjunto de estados finais, tais que $F \subseteq Q$.

- Portanto os componentes do AFN são os mesmos do AFD, com exceção da função programa (ver figura anterior).
- O processamento de um AFN M para um conjunto de estados ao ler um símbolo, é a união dos resultados da função programa aplicada a cada estado alternativo.

Definição: Função Programa Estendida

Seja M = (Σ, Q, δ, q0, F) um AFN.

A *função programa estendida*, denotada por:

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

é a função programa δ: Qx$\Sigma \rightarrow 2^Q$, estendida para palavras, e é indutivamente definida como se segue:

$$\underline{\delta}\ (P, \varepsilon) = P$$

$$\underline{\delta}\ (P, aw) = \underline{\delta}\ (\cup_{q \in P}\ \delta\ (q, a), w)$$

Assim, tem-se que, para um conjunto de estados {q1, q2, ..., qn} e para um símbolo a:

$$\underline{\delta}(\{q1, q2, ..., qn\}, a) = \delta(q1, a) \cup \delta(q2, a) \cup ... \cup \delta(qn, a)$$

- Por simplicidade tanto δ quanto $\underline{\delta}$ serão denotadas simplesmente por δ.
- A linguagem aceita por um AFN M = (Σ, Q, δ, q0, F) denotada por ACEITA(M), ou L(M), é o conjunto de todas as palavras pertencentes a Σ^* tais que existe pelo menos um caminho alternativo que aceita a palavra, ou seja:

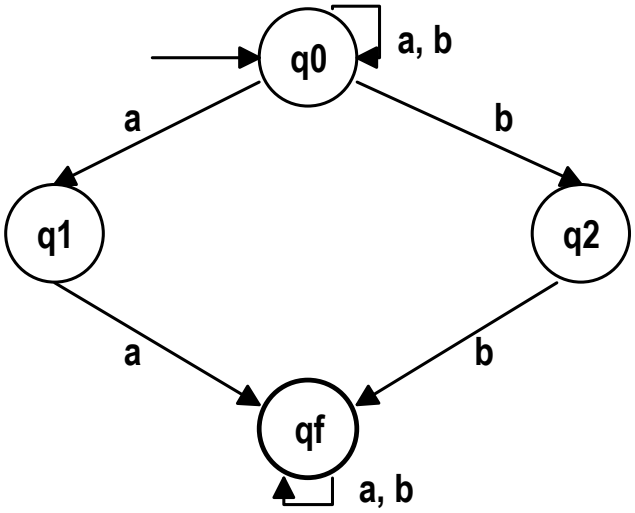
$$\text{ACEITA(M)} = \{w \mid \text{existe } q \in \delta(q0, w) \text{ tal que } q \in F\}.$$

- Analogamente, REJEITA(M) é o conjunto de todas as palavras de Σ^* que são rejeitadas por todos os caminhos alternativos de M a partir de q0.

Exemplo: Autômato Finito Não-Determinístico

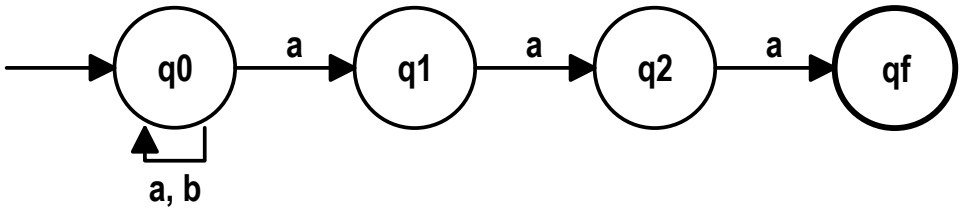
O AFN M5 = ({a, b}, {q0, q1, q2, qf}, δ5, q0, {qf}),
reconhece a linguagem L5 = {w | w possui aa ou bb como sub-palavra},
onde δ5 é dada abaixo, na forma de tabela:

δ_5	a	b
q0	{q0, q1}	{q0, q2}
q1	{qf}	-
q2	-	{qf}
qf	{qf}	{qf}



Exemplo: Autômato Finito Não-Determinístico:

O AFN M6 = ({a,b}, {q0, q1, q2, qf}, δ6, q0, {qf}), representado na figura abaixo reconhece a linguagem L6 = { w | w possui aaa como sufixo }



Teorema: Equivalência entre AFD e AFN

A classe dos AFD é equivalente à classe dos AFN.

- A prova consiste em mostrar que para todo AFN M é possível construir um AFD M' que realiza o mesmo processamento, ou seja, M' simula M.
- A demonstração apresenta um algoritmo para converter um AFN qualquer em um AFD equivalente.
- A idéia central do algoritmo é a construção de estados de M' que simulem as diversas combinações de estados de M.
- A transformação contrária - construir um AFN a partir de um AFD - não necessita ser demonstrada, uma vez que decorre trivialmente das definições (Por quê? Porque a função programa δ do AFN contém a função programa δ' do AFD).

Seja M = (Σ, Q, δ, q0, F) um AFN qualquer e seja M' = (Σ', Q', δ', <q0>, F') um AFD construído a partir de M como se segue:

Q' : Conjunto de todas as combinações, sem repetições, de estados de Q, as quais são denotadas por <q1q2...qn> onde qi $\in$ Q para i em {1, 2, ..., n}. Note-se que a ordem dos elementos não identifica mais combinações. Por exemplo: <quqv> = <qvqu>.

δ' : Tal que δ'(<q1...qn>, a) = <p1...pm> sss δ ({q1, ..., qn}, a) = {p1, ..., pm}, ou seja, um estado de M' representa uma imagem de todos os estados alternativos de M.

<q0>: Estado inicial.

F' : Conjunto de todos os estados <q1q2...qn> $\in$ Q' tal que alguma componente qi $\in$ F, para i $\in$ {1, 2, ..., n}.

PROVA:

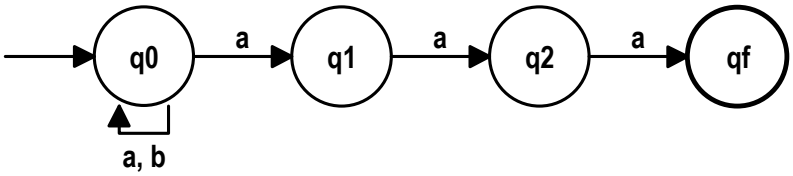
A demonstração de que o AFD M' simula o processamento do AFN M é dada por indução sobre o tamanho da palavra. Deve-se provar que, para uma palavra qualquer w de Σ:

$\delta'(<q0>, w) = <q1...qu>$ sse $\delta(\{q0\}, w) = \{q1, ..., qu\}$

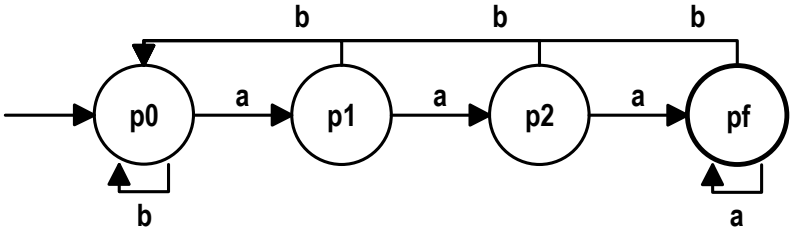
(A prova está no livro, na página 50).

Exemplo: Construção de um AFD a partir de um AFN.

Seja o AFN M6 = ({a,b}, {q0, q1, q2, qf}, δ6, q0, {qf}), dado no exemplo anterior e representado abaixo:



O AFD M6' = ({a, b}, Q', δ', <q0>, F'), construído conforme o algoritmo dado é:



onde:

Q' = {<q0>,<q1>,<q2>,<qf>,<q0q1>,<q0q2>, ...,<q0q1q2qf>}

F' = {<qf>,<q0qf>,<q1qf>,...,<q0q1q2qf>}

δ6' = É tal conforme os valores dados na tabela abaixo:

δ6'	a	b
<q0>	<q0q1>	<q0>
<q0q1>	<q0q1q2>	<q0>
<q0q1q2>	<q0q1q2qf>	<q0>
<q0q1q2qf>	<q0q1q2qf>	<q0>

No grafo que representa M6', acima, p0, p1, p2 e pf denotam respectivaente <q0>, <q0q1>, <q0q1q2>, <q0q1q2qf>.

Autômato Finito com Movimento Vazio

- *Movimentos vazios* constituem uma generalização dos AFN e são transições que ocorrem sem que haja a leitura de símbolo algum
- Os movimentos vazios podem ser interpretados como um não-determinismo interno do autômato, que é encapsulado.
- A não ser por uma eventual mudança de estados, nada mais pode ser observado sobre um movimento vazio..
- Qualquer AFε pode ser simulado por um autômato finito não-determinístico

Definição: Autômato Finito com Movimento Vazio (AFε)

Um autômato finito não-determinístico e com movimento vazio (AFNε), ou simplesmente autômato finito com movimento vazio (AFε), é uma quíntupla:

$$M = (\Sigma, Q, \delta, q0, F),$$

onde:

Σ - Alfabeto de símbolos de entrada

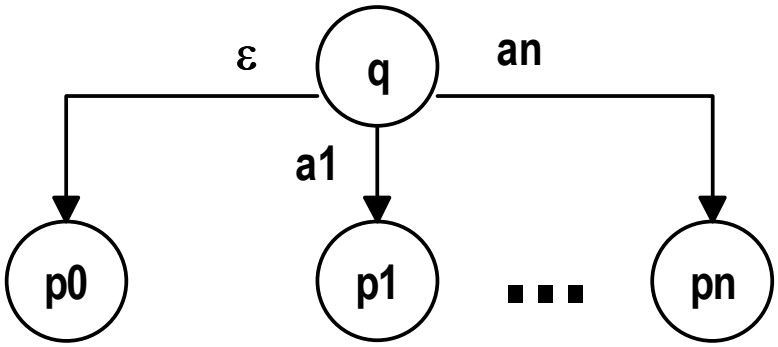
Q - Conjunto finito de estados possíveis do autômato

δ - Função programa ou função de transição δ: Q x ($\Sigma \cup \{\varepsilon\}$) $\rightarrow 2^Q$, parcial.

q_0 - Estado inicial tal que $q_0 \in Q$

F - Conjunto de estados finais, tais que $F \subseteq Q$.

- Portanto os componentes do AFε são os mesmos do AFN, com exceção da função programa (ver figura abaixo).



O processamento dos AFε é similar ao dos AFN. Por analogia o processamento de uma transição para uma entrada vazia também é não-determinística. Assim um AFε ao processar uma entrada vazia assume simultaneamente os estados de origem e destino da transição.

Exemplo: Autômato Finito com Movimento Vazio

O AFε M7 = ({a,b}, {q0, qf}, δ7, q0, {qf}), representado na figura abaixo reconhece a linguagem L7 = { w | qualquer símbolo a antecede qualquer símbolo b }, onde δ7 é representada na forma da tabela:

δ7	a	b	ε
q0	{q0}	-	{qf}
qf	-	{qf}	-