I. LINGUAGENS REGULARES E AUTÔMATOS FINITOS

1.1. A Primeira Linguagem

A teoria moderna das linguagens formais vem de duas fontes: a caracterização precisa da estrutura das linguagens naturais, como o inglês e o francês, proposta pelo lingüista americano Noam Chomsky, de acordo com regras matemáticas formais, e o desenvolvimento de uma especificação formal para a linguagem de computador ALGOL 60. O trabalho de Chomsky procurava descrever a sintaxe da linguagem natural de acordo com regras de substituição simples e transformações. Chomsky considerava várias formas de restrições de regras, e uma delas deu origem a uma classe de gramáticas conhecidas como *gramáticas livres de contexto* (GLC). O desenvolvimento do ALGOL 60, que seguiu a caracterização de linguagem natural de Chomsky, demonstrou que as GLCs são sistemas razoavelmente adequados para descrever a sintaxe básica de muitas linguagens de programação.

Como um primeiro exemplo de linguagem livre de contexto (LLC), considere a "linguagem" de parênteses casados. Esta linguagem tem um papel importante na ciência da computação como uma notação de escopo de expressões matemáticas e linguagens de programação.

1. Exemplo. Parênteses casados incluem todas as cadeias balanceadas de parênteses esquerdo e direito - por exemplo, (()), ()() e (()(())). A seguinte especificação descreve a linguagem de parênteses casados intuitivamente:

(1) A cadeia () é bem formada.
(2) Se a cadeia de símbolos A é bem formada, então a cadeia (A) também é.
(3) Se as cadeias A e B são bem formadas, então a cadeia AB também é.

Podemos agora seguir esta definição intuitiva para obter um sistema de rescrita - uma gramática - que gera exatamente o conjunto de cadeias legais de parênteses casados.

$$(1)\ S \rightarrow ()$$
$$(2)\ S \rightarrow (S)$$
$$(3)\ S \rightarrow SS$$

Estas três regras de rescrita são chamadas de *produções*. Elas dizem "dada uma cadeia, você pode formar uma nova cadeia substituindo um S por () ou (S) ou SS." Note que (3) usa uma notação diferente da parte (3) da definição intuitiva dada acima. Para gerar (())(()()), ocorrem as seguintes substituições:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (())S \Rightarrow (())(S) \Rightarrow (())(SS) \Rightarrow (())(()S) \Rightarrow (())(()())$$

Aqui, a produção $S \rightarrow SS$ é aplicada primeiro, e então os dois novos Ss são independentemente rescritos como as cadeias (()) e (()()).

Podemos resumir a derivação anterior com a notação

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

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

A gramática apresenta dois tipos diferentes de símbolos: os *não-terminais*, ou *variáveis*, que são os símbolos que podem aparecer em derivações mas não aparecem nas cadeias finais (no exemplo, S é a única variável); e os *terminais*, que são os símbolos que formam a cadeia que se deseja produzir (no exemplo, "(" e ")" são símbolos terminais).

O estilo de linguagem de programação para escrever gramáticas, a chamada forma de Backus-Naur, ou BNF, usa uma notação levemente diferente. Na BNF as variáveis são fechadas em $\langle\rangle$, e $\rightarrow$ é substituído pelo símbolo "::=". Portanto, a segunda produção do Exemplo 1 é escrita na notação BNF como

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

1.2. Gramáticas e Linguagens

Seja Σ um conjunto finito não vazio de símbolos terminais, ou *alfabeto terminal*. Seja Σ^* o conjunto de todas as cadeias de comprimento finito sobre Σ. Este conjunto infinito de cadeias inclui a cadeia vazia, que é denotada pelo símbolo λ. Se $\Sigma = \{(,)\}$, então Σ^* é o conjunto de todas as cadeias finitas de parênteses.

Uma *linguagem L* sobre Σ é qualquer subconjunto de Σ^*. Então a linguagem de parênteses casados, M, é uma linguagem porque $M \subset \{(,)\}^*$. Cada cadeia w em L tem um comprimento finito, $|w|$, com $|\lambda| = 0$. Se $w = x_1x_2...x_n$, cada $x_j \in \Sigma$, temos $|w| = n$. Podemos escrever Σ^+ para denotar a linguagem de cadeias não vazias sobre Σ. Se $w = x_1x_2...x_n$ e $w' = x'_1x'_2...x'_m$ são duas cadeias quaisquer, podemos *concatená-las* para obter

$$ww' = x_1x_2...x_n\, x'_1x'_2...x'_m$$

Estabelecemos que $w\lambda = \lambda w = w$.

No exemplo dos parênteses casados considerou-se regras de rescrita que envolviam caracteres de alfabetos terminal e não terminal. Estes três elementos mais um quarto, uma variável distinta que é usada como um símbolo inicial, formam uma *gramática de estrutura de frase*, ou simplesmente, uma *gramática*.

1. Definição. Uma *gramática de estrutura de frase G* é uma quádrupla (Σ, V, S, P) onde Σ e V são alfabetos finitos disjuntos e
(1) Σ é o *alfabeto terminal* para a gramática;
(2) V é o *alfabeto não terminal* ou *alfabeto de variável* para a gramática;
(3) S é o *símbolo inicial* para a gramática; e

(4) P é conjunto de *produções* da gramática. P é um conjunto de pares (v,w) com v uma cadeia em $(\Sigma \cup V)$ contendo no mínimo um símbolo não terminal, enquanto que w é um elemento arbitrário de $(\Sigma \cup V)^*$. Um elemento (v,w) de P é usualmente escrito como $v \rightarrow w$.

A gramática para a linguagem de parênteses casados pode ser escrita como

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

Em geral usa-se letras maiúsculas para variáveis e dígitos decimais ou letras minúsculas para símbolos terminais e para cadeias de símbolos terminais.

2. Definição. Seja G uma gramática e sejam y e z cadeias finitas de $\Sigma \cup V$. Escreve-se $y \Rightarrow z$ e diz-se que *y deriva diretamente z*, ou *z é diretamente derivado de y*, se z puder ser obtido de y substituindo-se uma ocorrência em y do "lado esquerdo" de alguma produção por seu "lado direito," por exemplo, se G tem uma produção $v \rightarrow u$, tal que y possa ser escrito como pvr e z como pur. Escreve-se $y \Rightarrow^* z$, e diz-se que *y deriva z*, ou z é derivável a partir de y, se $y = z$ ou existir alguma seqüência de cadeias $w_1, w_2...w_n$, com $w_1 = y$ e $w_n = z$ tal que para todo i, w_i deriva diretamente w_{i+1}.

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

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

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

4. Exemplo. Sejam $V = \{S\}$, $\Sigma = \{a,b\}$, $P = \{S \rightarrow aSb, S \rightarrow ab\}$. Neste caso $L(G) = \{a^n b^n \mid n > 0\}$. Ou seja, $L(G)$ é o conjunto de todas as seguintes cadeias: um bloco não vazio de a's, seguido por um bloco de b's do mesmo comprimento.

Nota: pode-se simplificar a notação combinando produções que têm o mesmo lado esquerdo, usando o símbolo |. Por exemplo, pode-se escrever a gramática do exemplo 4 como $S \rightarrow aSb \mid ab$.

5. Exemplo. A gramática

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$$

gera o conjunto de todos os palíndromos sobre o alfabeto terminal $\{a,b\}$. Um *palíndromo* é uma cadeia que é a mesma lida da esquerda para a direita e da direita para a esquerda. O conjunto de palíndromos sobre o alfabeto terminal $\Sigma = \{a, ..., z\}$ inclui várias curiosidades lingüísticas:

Raul ama luar

Socorram-me subi no ônibus em Marrocos.

6. Exemplo. A seguinte gramática gera todas as cadeias sobre o alfabeto terminal {0, 1} com um número igual de 0's e 1's.

$$V = \{S, A, B\}$$
$$\Sigma = \{0,1\}$$

P compreende as seguintes produções:

$$S \to 0B \mid 1A$$
$$A \to 0 \mid 0S \mid 1AA$$
$$B \to 1 \mid 1S \mid 0BB$$

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

$$A \to bC \qquad \text{ou} \qquad A \to b$$

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

8. Exemplo. Considere a gramática linear a direita G_1 com produções

$$S \to \lambda \mid 0 \mid 0S \mid 1 \mid 1S$$

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

Agora, considere a gramática G_2 com produções

$$S \to \lambda \mid 0S \mid 1T$$
$$T \to 0T \mid 1S$$

Pode-se provar que

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

9. Definição. Seja $G = (\Sigma, V, S, P)$ uma gramática. Então G é classificada como tipo i, $i = 0, 1, 2, 3$, de acordo com as seguintes restrições na forma de regras de substituição de P:

(i) Uma gramática é do *tipo 3* se toda produção for da forma $A \to bC$ ou $A \to b$, onde $A, C \in V$ e $b \in \Sigma$ ou $b = \lambda$. Tais gramáticas são também referidas como *gramáticas lineares a direita*. Uma linguagem gerada por tal gramática é chamada uma linguagem do tipo 3 ou linear a direita.

(ii) Uma gramática é do *tipo 2* se toda produção de P for da forma $A \to w$, com $A \in V$ e $w \in (\Sigma \cup V)^*$. Estas, claro, são *gramáticas livres de contexto* e as linguagens que elas geram são linguagens livres de contexto.

(iii) Uma gramática G é do *tipo 1* se toda produção de P for da forma $vAw \to vzw$ onde $z \in (\Sigma \cup V)^+$ (isto é, $z \neq \lambda$). Em adição, permite-se a única regra-λ $S \to \lambda$ no caso onde S não aparece do lado direito de nenhuma produção. As

gramáticas do tipo 1 são precisamente as *gramáticas sensíveis ao contexto* e elas geram as linguagens sensíveis ao contexto.

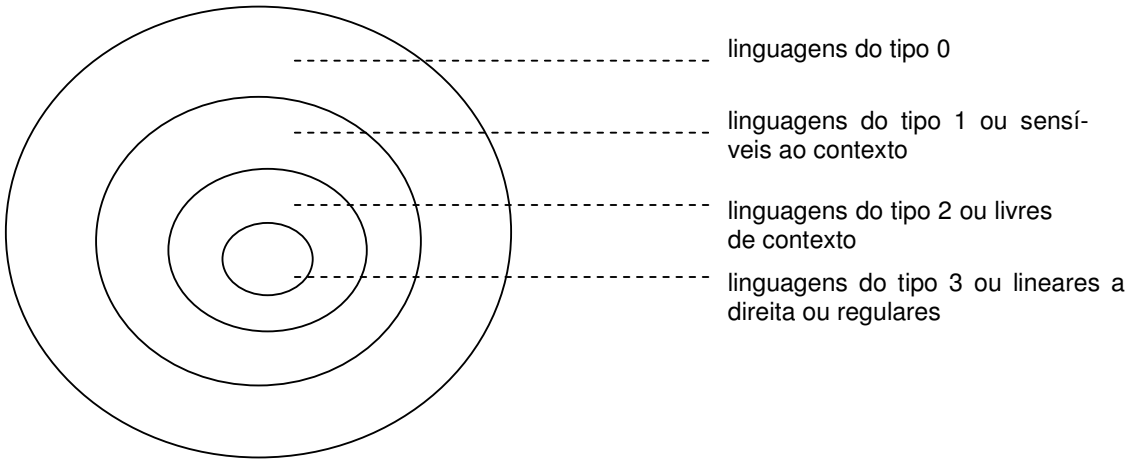
(iv) Qualquer gramática G é do *tipo 0*. Não existem restrições na forma das produções para as gramáticas desta classe.

Vamos usar a notação $\mathbf{L}_i$ para as linguagens de tipo i:

$$\mathbf{L}_i = \{ L \subset \Sigma^* \mid L = L(G) \text{ para alguma gramática } G \text{ de tipo } i \}$$

10. O Teorema da Hierarquia. *A hierarquia de Chomsky é uma hierarquia estrita de classes de linguagem*:

$$\mathbf{L}_3 \subset\neq \mathbf{L}_2 \subset\neq \mathbf{L}_1 \subset\neq \mathbf{L}_0.$$



1.3. Propriedades de Fechamento

Os teóricos da linguagem estão interessados em formas nas quais linguagens dadas (conjuntos de palavras) podem ser combinadas para formar novas linguagens.

1. Definição. Sejam L_1, L_2 duas linguagens sobre o mesmo alfabeto X (isto é, L_1 e L_2 são ambas subconjuntos de X^*). Então define-se

(i) A *união* de L_1 e L_2 é

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

(ii) A *concatenação* de L_1 e L_2 (também chamada "L_1 ponto L_2") é

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

(iii) A *iteração* de L_1 (também chamada "L_1 estrela") é

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

onde se define as potências L_1^n de L_1 para $n \geq 0$ por

Passo Básico: $L_1^0 = \{\lambda\}$

Passo de Indução: $L_1^{n+1} = L_1^n \cdot L_1$ para $n \geq 0$

Então, $L_1^1 = L_1$, $L_1^2 = L_1 \cdot L_1$, etc.

Dizemos que uma *classe* de linguagens é fechada sob uma operação de linguagem se, quando se aplica esta operação às linguagens que pertencem a esta classe, obtêm-se uma outra linguagem pertencente à classe.

2. Teorema. *A família de linguagens de tipo i é fechada sob* $\cup$, $\cdot$ *e* * *para cada i* = 0, 1, 2, 3.

A estratégia geral da prova será a seguinte: Vamos considerar uma operação de cada vez. Pela definitude, considere a concatenação e as LLCs. De G_1 e G_2 construiremos uma nova gramática G^. Vamos então observar que se G_1 e G_2 são livres de contexto então G^ também é, e $L(G\text{^}) = L_1 \cdot L_2$, tal que $\mathbf{L}_2$, a família das L-LCs, é fechada sob concatenação.

3. Proposição. As linguagens lineares a direita (tipo 3) são fechadas sob complemento e interseção.

PROVA: Provaremos o fechamento sob o complemento na próxima seção. Agora, pela Lei de De Morgan,

$$A \cap B = \neg(\neg A \cup \neg B)$$

e portanto as LLDs são fechadas sob interseção também. ◊

4. Teorema. *As linguagens livres de contexto não são fechadas sob interseção ou complemento*.

PROVA: Considere as linguagens

$$L_1 = \{a^n b^n c^m \mid n,m > 0\},$$
$$L_2 = \{a^n b^m c^m \mid n,m > 0\}.$$

L_1 e L_2 são livres de contexto. Mas $L_1 \cap L_2$ não é: sua interseção leva a uma linguagem não livre de contexto

$$\{a^k b^k c^k \mid k > 0\},$$

através da aplicação do lema do bombeamento. Para o complemento aplique a Lei de De Morgan, como na prova anterior. ◊

1.4. Linguagens Regulares e de Estados Finitos

Nesta seção apresentamos uma descrição algébrica de linguagens lineares a direita como as linguagens denotadas como *expressões regulares*. Então, introdu-

zimos uma abordagem de máquina teórica para estas linguagens mostrando que as linguagens lineares a direita são precisamente aquelas aceitas pelos *autômatos finitos*. A equivalência destas três caracterizações é o resultado principal que diz respeito às linguagens lineares a direita.

1. Definição. Seja Σ um alfabeto finito. Uma linguagem $L \subseteq \Sigma^*$ é *regular* se L for finita, ou L pode ser obtida indutivamente usando uma das seguintes operações:
(1) L é $L_1 \cup L_2$, a *união* de L_1 e L_2, onde L_1 e L_2 são regulares; ou
(2) L é $L_1 \cdot L_2$, a *concatenação* de L_1 e L_2, onde L_1 e L_2 são regulares; ou
(3) L é L_1^*, a *iteração* de L_1, onde L_1 é regular.

Note que o conjunto vazio $\varnothing$ e o conjunto $\{\lambda\}$ são ambos regulares, sendo finitos. As linguagens

(i) $\{ab\}^* \cdot \{a\}^*$
(ii) $\{a, b\}^*$

são também regulares. A primeira consiste de todas as cadeias que começam com uma cadeia (possivelmente vazia) de *a*'s e *b*'s alternados, seguida por um bloco (possivelmente vazio) de *a*'s. A segunda linguagem consiste de todas as cadeias finitas de *a*'s e *b*'s.

2. Proposição. *Todo conjunto regular é uma linguagem linear a direita*.

3. Definição. Seja Σ um alfabeto finito. Define-se *expressões regulares* R e suas denotações de conjuntos regulares $S(R)$ indutivamente como:
(i) $\varnothing$ é uma expressão regular com $S(\varnothing) = \varnothing$, o conjunto vazio.
(ii) λ é uma expressão regular com $S(\lambda) = \{\lambda\}$.
(iii) Se $a \in \Sigma$, então a é uma expressão regular com $S(a) = \{a\}$.
(iv) Se R_1 e R_2 são expressões então $(R_1 + R_2)$ é uma expressão regular com $S((R_1 + R_2)) = S(R_1) \cup S(R_2)$.
(v) Se R_1 e R_2 são expressões então $(R_1 \cdot R_2)$ é uma expressão regular com $S((R_1 \cdot R_2)) = S(R_1) \cdot S(R_2)$.
(vi) Se R é regular então $(R)^*$ é regular com $S((R)^*) = (S(R))^*$.

4. Exemplos.

(i) $(a + b)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$
(ii) $(a^* \cdot b^*)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$ (Por que?)
(iii) $(aa + bb)^*$, que abrevia $((a \cdot a) + (b \cdot b))^*$, representa todas as cadeias finitas de *a*'s e *b*'s construídas pela concatenação de pares de *a*'s e pares de *b*'s.
(iv) $aa(b^* + aaa)c + (a + c)^*$ representa a linguagem que é a união de três sublinguagens:

(1) todas as cadeias começando com um *a* duplo, seguido por qualquer número de *b*'s, seguido por um *c*;

(2) a linguagem cadeia $\{aaaaac\}$; e

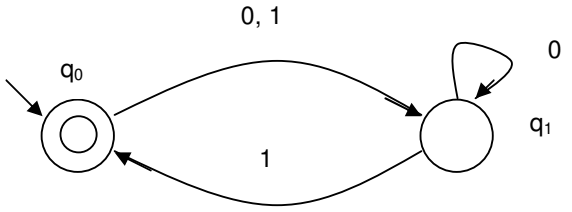
(3) a linguagem construída por todas as cadeias de *a*'s e *c*'s.

5. Exemplo. Considere a expressão regular (b^* + (ab)*). Para achar uma gramática linear a direita para esta expressão, primeiro forme uma gramática para b^*: $S_1 \rightarrow bS_1 \mid \lambda$. Então forme uma gramática (ab): $S_2 \rightarrow aB_2$, $B_2 \rightarrow b$. Isto pode ser estrelado adicionando $S_2 \rightarrow \lambda$, $B_2 \rightarrow bS_2$. Finalmente, a linguagem completa pode ser gerada adicionando $S \rightarrow bS_1 \mid \lambda \mid aB_2$, levando a

$$S \rightarrow bS_1 \mid \lambda \mid aB_2$$
$$S_1 \rightarrow bS_1 \mid \lambda$$
$$S_2 \rightarrow aB_2 \mid \lambda$$
$$B_2 \rightarrow bS_2 \mid b$$

Nesta seção, introduzimos a classe básica de dispositivos de computação chamados de *autômatos finitos*. Estes dispositivos julgam a legalidade de cadeias sobre algum alfabeto finito. Vamos mostrar que a coleção de linguagens que podem ser aceitas por autômatos finitos é exatamente a de linguagens regulares.

6. Exemplo. O autômato finito dado a seguir aceita a linguagem ((0 + 1)0*1)*.



O dispositivo opera da seguinte forma na cadeia 1001: o processamento começa no estado q_0, o *estado inicial* da máquina (indicado pela seta livre). Inicialmente a máquina procura o símbolo mais a esquerda da entrada, neste caso um 1. O arco de q_0 para q_1 que é rotulado com um '1' indica que quando a máquina está no estado q_0 e está procurando um 1, ela deve deslocar-se para o estado q_1 enquanto ela avança a busca de entrada para o próximo símbolo. Agora a máquina está no estado q_1 e está procurando o segundo símbolo, um 0. Neste caso, ela lê o 0 e permanece no estado q_1, já que o arco-0 deixa q_1 e retorna para q_1. Então o segundo 0 da cadeia é lido e outra vez a máquina se desloca de q_1 de volta para q_1. Finalmente, a máquina procura um 1 - o último símbolo - e pára no estado q_0. O estado q_0 é um *estado de aceitação*, como indicado pelo círculo duplo. Então 1001 é aceito por esta máquina, assim como 100001 e 101101. A cadeia 1000 é rejeitada, pois q_1 não é um estado de aceitação.

Vamos mostrar que a linguagem ((0 + 1)0*1)* é o conjunto de aceitação para este dispositivo, através da descrição de todas as cadeias que levam a máquina do estado q_0 para o estado q_0. Como (0 + 1) representa todos os caminhos diretos de q_0 para q_1 e 0*1 descreve todos os caminhos de q_1 para q_0, (0 + 1)0*1 representa todos os caminhos de q_0 para q_0 que visitam q_0 exatamente duas vezes (uma vez no início e uma vez no final). Qualquer número finito de tais caminhos de q_0 para q_0 produzirá cadeias legais, tal que a expressão completa seja ((0 + 1)0*1)*.

7. Definição. Um *autômato finito determinístico* (AFD) M é especificado por uma quíntupla (Q, Σ, δ, q_0, F) onde

Q é um conjunto finito de estados
Σ é o alfabeto terminal
δ: $Q \times \Sigma \rightarrow Q$ é a função de transmissão de estado
$q_0 \in Q$ é o estado inicial
$F \subset Q$ é o conjunto de estados de aceitação.

Então o *grafo de estado* de um AFD tem um nó para cada q em Q, que é rotulado; tem uma seta livre apontando para o nó q_0; tem um círculo duplo para um nó q apenas no caso de q estar em F; e tem um arco rotulado x levando de um nó q para um nó q' apenas no caso de $\delta(q, x) = q$'.

O AFD do Exemplo 6 é representado pela quíntupla ($\{q_0, q_1\}$, {0, 1}, δ, q_0, $\{q_0\}$); a função (finita) δ é especificada pelo seguinte quadro:

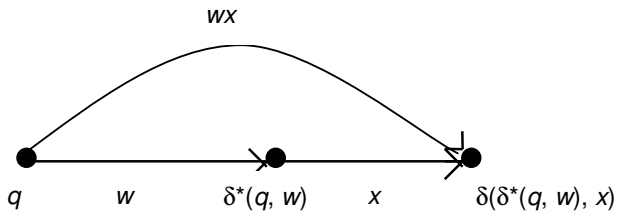
δ	**0**	**1**
$\boldsymbol{q_0}$	q_1	q_1
$\boldsymbol{q_1}$	q_1	q_0

O comportamento do AFD M, numa cadeia w em Σ^* pode ser descrito como o seguinte: M começa no estado q_0 e procura o símbolo mais a esquerda em w. Na base de seu estado corrente - neste caso q_0 - e o símbolo procurado, M desloca-se para algum estado novo. Esta transição é governada pela função de transição δ dada previamente, que mapeia um par estado/símbolo para um estado novo. O segundo símbolo é então procurado, com M no novo estado e novamente uma transição δ ocorre. Este processo continua até que a cadeia de entrada inteira tenha sido lida. Se o estado final é um elemento de F, então a cadeia x é *aceita* por M. Dado um AFD M, deve-se denotar por $T(M)$ o conjunto de cadeias aceitas por M, o conjunto de $w \in \Sigma^*$ que envia M de q_0 para algum estado em F. A meta é caracterizar o conjunto de cadeias que pode ser $T(M)$'s para algum M.

8. Definição. Dado um AFD M, define-se por indução a função δ^*: $Q \times \Sigma^* \rightarrow Q$, tal que $\delta^*(q, w)$ é o estado para o qual M irá na cadeia de entrada w se começado no estado q_0:

Passo Básico: $\delta^*(q, \lambda) = q$ para cada estado q em Q. Se M começou no estado q, então quando ele receber a cadeia de entrada vazia ele deve ainda estar no mesmo estado q.

Passo de Indução: Para cada estado q em Q, cada cadeia de entrada w em Σ^* e cada símbolo de entrada x em Σ, estabelece-se que $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$.



A cadeia *w* envia *M* do estado *q* para o estado $\delta^*(q, w)$, cuja entrada *x* então muda para o estado $\delta(\delta^*(q, w), x)$ - mas este é apenas o estado $\delta^*(q, wx)$ para o qual a cadeia *wx* envia *M* do estado *q*.

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

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

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

Dizemos que um subconjunto *L* de Σ^* é uma *linguagem de estados finitos* (LEF) se for igual a $T(M)$ para algum AFD *M*.

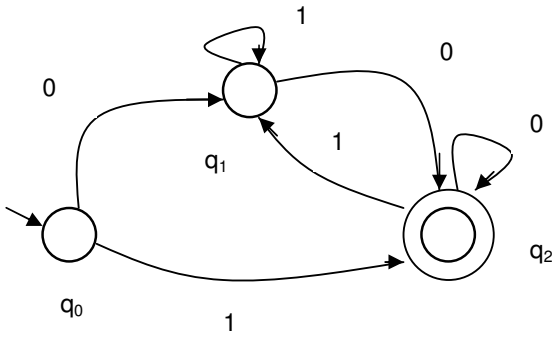
10. Proposição. *Toda LEF é uma linguagem linear a direita. Por isso, as linguagens LEF são fechadas sob complemento*.

11. Exemplo. Para o AFD do Exemplo 6, podemos escrever a seguinte gramática com estado inicial q_0:

$$q_0 \to 0q_1 \mid 1q_1 \mid \lambda$$
$$q_1 \to 0q_1 \mid 1q_0 \mid 1$$

12. Teorema. *Toda LEF é um conjunto regular*.

13. Exemplo. Considere o seguinte *M* dado.



Para gerarmos a expressão regular equivalente a este autômato, precisamos estabelecer todos os caminhos que levam do estado inicial q_0 a um estado de aceitação (neste caso q_2). Os caminhos são:

De q_0 a q_2: 1 . 0*
De q_0 a q_1 a q_2: 0 . 1* . 0 . 0*
De q_0 a q_1 a q_2 a q_1 a q_2: 0 . 1* . 0 . 0* . 1. 1* . 0 . 0*

Observe que há um ciclo se repetindo entre q_2 e q_1, logo pode-se representar os caminhos que levam de q_0 a q_2 passando por q_1 da seguinte forma:

De q_0 a q_1 a q_2 a (q_1 a q_2)*: 0 . 1* . 0 . 0* . (1 . 1* . 0 . 0*)*

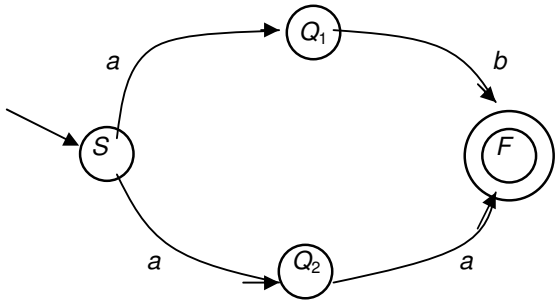
Logo a expressão total fica:

1 . 0* + 0 . 1* . 0 . 0* . (1 . 1* . 0 . 0*)*

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

$$S \to aQ_1, \qquad Q_1 \to bF \mid b$$
$$S \to aQ_2, \qquad Q_2 \to aF \mid a$$

Esta gramática leva à seguinte máquina:



A estrutura obtida não é um AFD, porque dois arcos saindo de *S* têm o mesmo rótulo, *a*. Isto viola a condição da regra de transição para um AFD ser uma função. Quando transições múltiplas deste tipo estão presentes em *M*, dizemos que *M* é uma máquina *não-determinística*.

14. Definição. Um *autômato finito não determinístico* (AFN) *M* é especificado por uma quíntupla (Q, Σ, δ, Q_0, F) onde

Q é um conjunto finito de estados
Σ é o alfabeto terminal
δ: $Q \times \Sigma \to 2^Q$ atribui a cada par estado-entrada (q, x) um *conjunto* $\delta(q, x) \subset Q$ de próximos estados *possíveis* (2^Q é o conjunto potência de Q ou conjunto de todos os subconjuntos de Q)

$Q_0 \subset Q$ é o conjunto de estados iniciais
$F \subset Q$ é o conjunto de estados de aceitação

Vimos que um AFN é como um AFD exceto que existe um *conjunto* de estados iniciais e um *conjunto* de próximos estados possíveis. Estas complicações significam que uma cadeia pode induzir caminhos múltiplos através de um AFN, alguns terminando em estados de aceitação, outros terminando em estados de não aceitação. Lidamos com esta ambigüidade dizendo que $w \in \Sigma^*$ é aceito se existe *pelo menos uma* forma de chegar a um estado de aceitação.

Vamos definir agora δ^*: $2^Q \times \Sigma^* \rightarrow 2^Q$ para um AFN tal que $\delta^*(p, w)$ é o conjunto de estados alcançáveis a partir de algum estado em p pela aplicação da cadeia de entrada w.

15. Definição. Estenda δ: $Q \times \Sigma \rightarrow 2^Q$ para um AFN para δ^*: $2^Q \times \Sigma^* \rightarrow 2^Q$ por:

Passo Básico: $\delta^*(p, \lambda) = p$ para cada $p \subset Q$.
Passo de Indução: $\delta^*(p, wx) = \bigcup\{\delta(q, x) \mid q \in \delta^*(p, w)\}$ para cada w em Σ^*, x em Σ e $p \subset Q$.

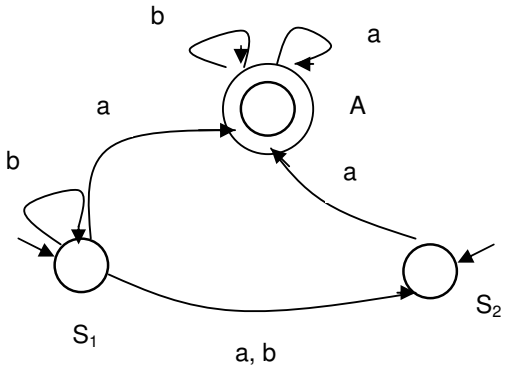
Estabelece-se que

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

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

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

17. Exemplo. Considere o seguinte AFN.



Os estados S_1 e S_2 são estados iniciais, enquanto que o estado A é o único estado de aceitação. Existem $2^{|Q|}$ (neste caso, $2^3 = 8$) estados no AFD equivalente, mas na prática não é necessário considerar todos eles, porque muitos são inalcançáveis. Tira-se vantagem desta observação começando do novo estado inicial e construindo estados adicionais a medida em que eles são gerados. Começa-se com

	a	***b***
{S_1, S_2}	{S_2, A}	{S_1, S_2}

que descreve a ação de δ no estado inicial {S_1, S_2} de entradas a e b, respectivamente.

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

	a	***b***
{S_2, A}	{A}	{A}

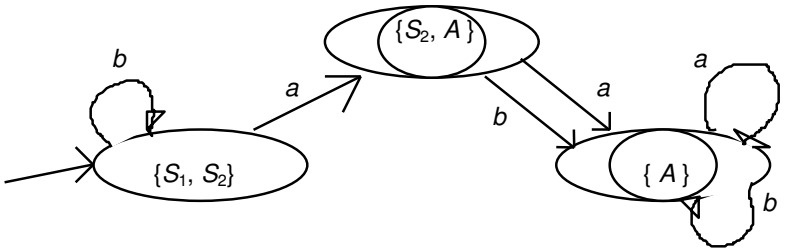
Então tem-se

	a	***b***
{A}	{A}	{A}

A tabela completa do δ fica:

	a	***b***
{S_1, S_2}	{S_2, A}	{S_1, S_2}
{S_2, A}	{A}	{A}
{A}	{A}	{A}

Em forma gráfica o AFD construído é



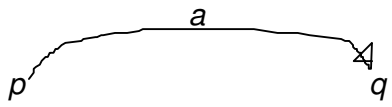
Desenvolvemos maquinaria suficiente para provar que toda linguagem linear a direita pode ser representada por $T(M)$ para algum autômato finito M.

18. Teorema. *Toda linguagem linear a direita L é uma LEF*.

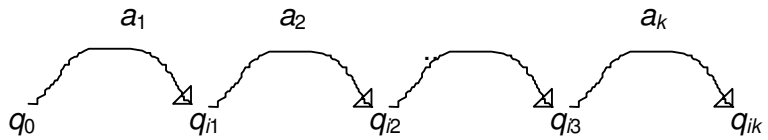
Colocando juntos a Proposição 2, o Teorema 12 e o Teorema 18, conclui-se que as classes de linguagens lineares a direita, linguagens regulares e linguagens de estados finitos coincidem. O próximo teorema, o análogo ao lema do bombeamento para linguagens livres de contexto, permite concluir que as linguagens regulares estão propriamente contidas nas linguagens livres de contexto.

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

PROVA: Vamos usar a seguinte notação



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



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

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

20. Exemplo. A linguagem $L = \{ a^nb^n \mid n \geq 0\}$ não é regular. Suponha que L seja regular. Então $L = T(M)$, onde M tem k estados e considere a^kb^k. Então o teorema se aplica. Se u é construído só de a's ou só de b's, então vw é a^jb^j com $i \neq j$. De outra forma, $u = a^pb^q$, implicando que $uvvw$ tem alternações a-b em excesso.