Capítulo 6:

Linguagens livres de contexto e autômatos de pilha

José Lucas Rangel, maio 1999

6.1 - Introdução.

Os aceitadores, ou reconhecedores, das linguagens livres de contexto são os chamados *autômatos de pilha* ou *ap*'s. Usaremos aqui o modelo mais geral de ap, o *ap não determinístico*, ou *apnd*, que consiste basicamente de um autômato finito não determinístico, com uma memória adicional, em forma de pilha. Numa pilha, símbolos novos só podem ser acrescentados no topo da pilha; apenas o último símbolo armazenado, o símbolo que se encontra no topo da pilha pode ser consultado; esse símbolo deve ser retirado para que os demais possam ser alcançados.

Neste capítulo vamos provar que a classe de linguagens reconhecidas pelos apnd's é exatamente a classe das llc.

6.2 - Autômatos de pilha não determinísticos

Definição. Definimos um autômato de pilha não determinístico (apnd) como uma construção (tupla) $A = < K, \Sigma, \Gamma, \delta, i, I, F >$, formada pelos seguintes elementos:

K - conjunto (finito) de estados

Σ - alfabeto de entrada

Γ - alfabeto da pilha

δ - função de transição

$\delta: K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(K \times \Gamma^*)$

i - estado inicial

$i \in K$

I - símbolo inicial da pilha

$I \in \Gamma$

F - conjunto de estados finais

$F \subseteq K$

O alfabeto Γ contém os símbolos que podem ser armazenados na pilha, ou seja, *empilhados*. Para simplificar a especificação da função de transição δ, consideramos que a cada passo, um símbolo é lido (e retirado) do topo da pilha, e, no mesmo passo, uma seqüência de comprimento qualquer pode ser empilhada. Assim, se queremos retirar um símbolo do topo da pilha, basta que a sequência a ser empilhada seja a seqüência vazia ε. Por outro lado, se quisermos manter o símbolo do topo da pilha, ele deve ser re-empilhado.

Escolher de forma não determinística, uma opção $(p, \alpha) \in \delta(q, a, Z)$ quer dizer que a partir do estado q, lendo a da entrada, e lendo Z do topo da pilha, uma transição

possível para A terá como próximo estado p, e a seqüência α será empilhada, depois da remoção de Z da pilha.

Da mesma forma que num autômato finito não determinístico (afnd), temos a possibilidade de executar uma transição sem avançar na leitura dos símbolos da entrada, e essa ação é representada pela leitura da cadeia vazia ε. Por essa razão, o segundo argumento de δ pode ser ε, além de poder ser um símbolo de Σ.

Usamos $P(K \times \Gamma^*)$ para representar o conjunto potência de $K \times \Gamma^*$, ou seja o conjunto de todos os subconjuntos de $K \times \Gamma^*$. Como se trata aqui de um conjunto infinito, é necessário especificar que, em qualquer caso, $\delta(q, a, Z)$ será sempre um conjunto finito, de forma a manter finita a descrição do apnd A.

Para descrever os passos de uma computação realizada por um apnd, utilizamos *configurações* $[q, y, \gamma] \in K \times \Sigma^* \times \Gamma^*$. Os três elementos de uma configuração são o estado corrente $q \in K$, a parte da entrada ainda a ser lida, $y \in \Sigma^*$, e o conteúdo $\gamma \in \Gamma^*$ da pilha. Por convenção, *o topo da pilha fica à esquerda*, isto é, o primeiro símbolo de γ é considerado como sendo o símbolo do topo da pilha. A configuração inicial correspondente à entrada x é [i, x, I].

Vamos definir a relação *mudança de configuração*, representada por $\vdash$ ou por $\vdash_A$, se quisermos explicitar o apnd A considerado. Para isso utilizamos a função de transição δ. Suponha uma configuração $[q, ax, Z\gamma]$, com $(p, \alpha) \in \delta(q, a, Z)$. Escolhida esta opção, o próximo estado será p, e a cadeia α deve ser empilhada, após a leitura de Z. Assim, podemos passar da configuração $[q, ax, Z\gamma]$para a configuração $[p, x, \alpha\gamma]$, em que o estado passou a ser p, o "símbolo" a da entrada e o símbolo Z da pilha foram lidos, e a seqüência α foi empilhada. Ou seja,

se $(p, \alpha) \in \delta(q, a, Z)$

então $[q, ax, Z\gamma] \vdash [p, x, \alpha\gamma]$

Temos duas situações em que se pode dizer que um apnd aceita sua entrada, que correspondem a duas definições independentes do que se entende por linguagem reconhecida por um apnd. A primeira é semelhante à usada nos afnd: o apnd aceita se, após ler sua entrada, *o estado é um estado final*; a segunda é mais característica dos apnd: o apnd aceita se, após ler sua entrada, *a pilha está vazia*. Assim, podemos definir a linguagem de um apnd A (a linguagem aceita, ou reconhecida por A) de duas maneiras diferentes:

- aceitação por estado final:

  $L_{ef}(A) = \{ x \in \Sigma^* \mid [i, x, I] \vdash^* [f, \varepsilon, \gamma], \text{ com } f \in F \text{ e } \gamma \in \Gamma^* \text{ qualquer } \}$

- aceitação por pilha vazia:

  $L_{pv}(A) = \{ x \in \Sigma^* \mid [i, x, I] \vdash^* [f, \varepsilon, \varepsilon], \text{ com } q \in K \text{ qualquer } \}$

  São, portanto, duas as definições de configuração final:

  $[q, \varepsilon, \gamma]$ é uma configuração final para aceitação por estado final se $q \in F$;

  $[q, \varepsilon, \gamma]$ é uma configuração final para aceitação por pilha vazia, se $\gamma = \varepsilon$.

No primeiro caso, o conteúdo γ da pilha é irrelevante; no segundo caso, o estado q é irrelevante. Note-se que o conjunto de estados finais F de um apnd A não é

utilizado na definição da linguagem que A aceita por pilha vazia. Por essa razão costuma-se, neste caso, fazer F = ∅, sem perda de generalidade.

Um ponto importante a observar é que, no caso geral, as linguagens $L_{ef}(A)$ e $L_{pv}(A)$ podem ser distintas.

Exemplo 6.1: Seja o apnd A = < K, Σ, Γ, δ, i, I, F >, com

K = {0, 1, 2} i = 0

Σ = {a, b} I = X

Γ = {X, A} F = {2}

A função $\delta:\{0,1,2\}\times\{a,b,\varepsilon\}\times\{X,A\} \rightarrow P(\{0,1,2\}\times\{X,A\}^*)$ é dada por

$\delta(0, a, X) = \{(0, AX)\}$ $\delta(1, b, A) = \{(1, \varepsilon)\}$

$\delta(0, a, A) = \{(0, AA)\}$ $\delta(1, \varepsilon, X) = \{(2, X)\}$

$\delta(0, b, A) = \{(1, \varepsilon)\}$

Convencionamos, neste exemplo, e nos seguintes, que o valor da função δ, para as combinações de valores não especificadas, é sempre o conjunto vazio ∅. No caso presente, portanto,

$\delta(0, b, X) = \delta(0, \varepsilon, X) = \delta(0, \varepsilon, A) = \delta(1, a, X) = \delta(1, a, A) = \delta(1, b, X)$

$= \delta(1, \varepsilon, A) = \delta(2, a, X) = \delta(2, a, A) = \delta(2, b, X) = \delta(2, b, A)$

$= \delta(2, \varepsilon, X) = \delta(2, \varepsilon, A) = \emptyset$.

Suponhamos agora que a entrada do apnd A é x=aaabbb. A configuração inicial correspondente é [0, aaabbb, X]. A partir dessa configuração, os seguintes passos são possíveis:

$[0, aaabbb, X] \vdash [0, aabbb, AX] \vdash [0, abbb, AAX] \vdash [0, bbb, AAAX]$

$\vdash [1, bb, AAX] \vdash [1, b, AX] \vdash [1, \varepsilon, X] \vdash [2, \varepsilon, X]$

A última configuração indica que o estado final 2 foi atingido e que toda a entrada foi lida. Podemos assim dizer que $aaabbb \in L_{ef}(A)$.

Se examinarmos o funcionamento de A para a entrada acima, e, através da função δ o funcionamento para as demais sequências em {a, b}*, podemos verificar:

- o primeiro símbolo deve ser um a
- o número de a's deve ser igual ao de b's.
- após o primeiro b, nenhum outro a pode ser aceito.
- após o último b, uma transição-ε leva A para uma configuração em que o estado é 2, e nenhuma transição adicional é possível.

Portanto, $L_{ef}(A) = \{ a^j b^j \mid j \geq 1 \}$. Por outro lado, nenhuma das transições prevê a retirada de X do fundo da pilha, de forma que $L_{pv}(A)=\emptyset$.

Exemplo 6.2: Seja o apnd $A = < K, \Sigma, \Gamma, \delta, i, I, F >$, com

$K = \{0, 1\}$ $\quad i = 0$

$\Sigma = \{a, b\}$ $\quad I = X$

$\Gamma = \{X, A\}$ $\quad F = \varnothing$

sendo a função $\delta:\{0,1\}\times\{a,b,\varepsilon\}\times\{X,A\} \rightarrow P(\{0,1\}\times\{X,A\}^*)$ dada por:

$\delta(0, a, X) = \{(0, A)\}$ $\quad \delta(0, b, A) = \{(1, \varepsilon)\}$

$\delta(0, a, A) = \{(0, AA)\}$ $\quad \delta(1, b, A) = \{(1, \varepsilon)\}$

Com entrada aaabbb, os seguintes são passos possíveis:

$$[0, aaabbb, X] \vdash [0, aabbb, A] \vdash [0, abbb, AA] \vdash [0, bbb, AAA]$$

$$\vdash [1, bb, AA] \vdash [1, b, A] \vdash [1, \varepsilon, \varepsilon]$$

Como toda a entrada foi lida e a pilha está vazia, temos $aaabbb \in L_{pv}(A)$.

Examinando o funcionamento de A, podemos verificar que $L_{pv}(A) = \{ a^j b^j \mid j \geq 1 \}$, ou seja, que a linguagem aceita por pilha vazia por este autômato de pilha é idêntica à linguagem aceita por estado final pelo autômato do exemplo anterior. Por outro lado, $F=\varnothing$ faz com que $L_{ef}(A)=\varnothing$.

Exercício 6.1: Construa um autômato de pilha A que aceita a linguagem dos dois exemplos anteriores, simultaneamente, por pilha vazia e por estado final, ou seja, construa um apnd A tal que $L_{ef}(A) = L_{pv}(A) = \{ a^i b^i \mid i \geq 1 \}$

Exemplo 6.3: Seja a glc $G_0 = <N, \Sigma, P, S>$, com $N = <\{E, T, F\}$,

$\Sigma = \{+, *, (,), a\}, P, E>, S=E$ e P composto pelas regras:

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid a$

O apnd A, a seguir, aceita $L(G_0)$, por pilha vazia:

$A = < \{q\}, \Sigma, N \cup \Sigma, \delta, q, E, \varnothing >$,

sendo δ dada por:

$\delta(q, \varepsilon, E) = \{(q, E+T), (q, T)\}$

$\delta(q, \varepsilon, T) = \{(q, T*F), (q, F)\}$

$\delta(q, \varepsilon, F) = \{(q, (E)), (q, a)\}$

$\delta(q, +, +) = \delta(q, *, *) = \delta(q, (, () = \delta(q,),)) = \delta(q, a, a) = \{(q, \varepsilon)\}$

De acordo com a especificação da função δ acima, vemos que há dois tipos de transições possíveis neste apnd, conforme o símbolo de $N \cup \Sigma$ que aparece no topo da pilha:

- se o símbolo do topo da pilha é um nãoterminal de G_0, nenhum símbolo da entrada é lido, e há uma possibilidade para cada regra do nãoterminal.

  O apnd A depende do não-determinismo para escolher entre as diversas possibilidades aquela que corresponde à regra de G_0 usada no passo correspondente numa derivação esquerda da entrada x.

- se o símbolo do topo da pilha é um terminal de G_0, e o mesmo símbolo é lido da entrada.

  Esta transição só pode ser feita se a escolha (não-determinística) das transições do primeiro tipo foi feita de forma correta para a entrada.

Por exemplo, para a entrada (a+a)*a, temos a derivação esquerda

$$E \Rightarrow T \Rightarrow T*F \Rightarrow F*F \Rightarrow (E)*F \Rightarrow (E+T)*F \Rightarrow (T+T)*F \Rightarrow (F+T)*F$$
$$\Rightarrow (a+T)*F \Rightarrow (a+F)*F \Rightarrow (a+a)*F \Rightarrow (a+a)*a$$

A mesma seqüência de regras é usada, e passos dos dois tipos são intercalados, de acordo com a natureza do símbolo que aparece no topo da pilha:

$$[q, (a+a)*a, E] \vdash [q, (a+a)*a, T] \vdash [q, (a+a)*a, T*F] \vdash [q, (a+a)*a, F*F]$$
$$\vdash [q, (a+a)*a, (E)*F] \vdash [q, a+a)*a, E)*F] \vdash [q, a+a)*a, E+T)*F]$$
$$\vdash [q, a+a)*a, T+T)*F] \vdash [q, a+a)*a, F+T)*F]$$
$$\vdash [q, a+a)*a, a+T)*F] \vdash [q, +a)*a, +T)*F] \vdash [q, a)*a, T)*F]$$
$$\vdash [q, a)*a, F)*F] \vdash [q, a)*a, a)*F] \vdash [q,)*a,)*F] \vdash [q, *a, *F]$$
$$\vdash [q, a, F] \vdash [q, a, a] \vdash [q, \varepsilon, \varepsilon]$$

(Usamos aqui colchetes para representar as configurações, para evitar confusão com os parênteses de Σ.)

Veremos posteriormente que é possível construir, para qualquer glc G, de forma semelhante à utilizada neste último exemplo, um apnd que aceita L(G) por pilha vazia.

6.3 - Equivalência das duas formas de aceitação

Mostraremos agora que as duas formas de aceitação são equivalentes, no sentido de que definem a mesma classe de linguagens. Naturalmente, isto não quer dizer que para o mesmo apnd A as linguagens $L_{ef}(A)$ e $L_{pv}(A)$ são iguais. A idéia aqui é que se $L = L_{ef}(A)$, então existe um apnd B (possivelmente diferente de A) tal que $L_{pv}(B) = L$, e vice-versa.

Teorema 6.1: Para qualquer apnd A, existe um apnd B tal que B aceita por estado final a mesma linguagem que A aceita por pilha vazia.

Dem.: Seja o apnd $A = <K, \Sigma, \Gamma, \delta, i, I, F >$.

Defina $B = < K \cup \{i', f'\}, \Sigma, \Gamma \cup \{I'\}, \delta', i', I', \{f'\} >$, com δ' dada por

$\delta'(i', \varepsilon, I') = \{(i, I\ I')\}$

para $q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, $\delta'(q, a, Z) = \delta(q, a, Z)$

para $q \in K$, $\delta'(q, \varepsilon, I') = \{(f', \varepsilon)\}$

A idéia fundamental da construção de B é que, com entrada x,

- B passa de sua configuração inicial [i', x, I'] para a configuração [i, x, I I'], idêntica à configuração inicial de A, exceto pelo símbolo I' acrescentado no fundo da pilha;
- B simula todas as transições de A (exceto pela presença de I' no fundo da pilha), e quando A atinge uma configuração [q, ε, ε], indicando a aceitação de x, B atinge a configuração [q, ε, I'], com apenas o símbolo I' na pilha;
- B passa então dessa configuração para a configuração [f', ε, ε], que indica a aceitação da entrada x.

Esquematicamente, temos:

configurações de A	configurações de B
	[i', x, I']
[i, x, I]	[i, x, I I']
....	....
[q, ?, ?]	[q, ?, I']
	[f', ?, ?]

Portanto, se A aceita x por pilha vazia, B aceita x por estado final. Para a recíproca, observamos que A pode esvaziar a pilha sem completar a leitura de sua entrada x, atingindo uma configuração [q, y, ε], com y≠ε, que não é uma configuração final. Simulando A, B pode atingir configurações [q, y, I'] e [f', y, I'], mas esta última, apesar do estado final, não é uma configuração de aceitação, pela mesma razão anterior.

Teorema 6.2: Para qualquer apnd A, existe um apnd B tal que B aceita por pilha vazia a mesma linguagem que A aceita por estado final.

Dem.: Seja $A = < K, \Sigma, \Gamma, \delta, i, I, F >$.

Defina $B = < K \cup \{i', d'\}, \Sigma, \Gamma \cup \{I'\}, \delta', i', I', \varnothing >$, com δ' dada por

$\delta'(i', \varepsilon, I') = \{(i, I\ I')\}$

para $q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, $\delta'(q, a, Z) = \delta(q, a, Z)$

para $f \in F$, $Z \in \Gamma \cup \{I'\}$, $\delta'(f, \varepsilon, Z) = \{(d', Z)\}$

para $Z \in \Gamma \cup \{I'\}$, $\delta'(d', \varepsilon, Z) = \{(d', \varepsilon)\}$

A idéia fundamental da construção de B é que, com entrada x,

- B passa de sua configuração inicial [i', x, I'] para a configuração [i, x, I I'], idêntica à configuração inicial de A, exceto pelo símbolo I' acrescentado no fundo da pilha;
- B simula todas as transições de A (exceto pela presença de I' no fundo da pilha), e quando A atinge uma configuração [f, ε, γ], indicando a aceitação de x, B atinge a configuração [f, ε, γI'], com o símbolo I' acrescentado no fundo da pilha;

- B passa então dessa configuração para a configuração [d', ε, γI'], preparando-se para esvaziar a pilha, para indicar a aceitação da entrada x.
- A única ação possível no estado d' é a remoção do símbolo do topo da pilha, de forma que B atinge finalmente a configuração de aceitação [d', ε, ε].

Esquematicamente, temos

configurações de A	configurações de B
	[i', x, I']
[i, x, I]	[i, x, I I']
....	....
[f, ε, γ]	[f, ε, γI']
	[d', ε, γI']
	….
	[d', ε, I']
	[d', ε, ε]

Portanto, se A aceita x por estado final, B aceita x por pilha vazia. Para a recíproca, observamos que A pode esvaziar a pilha sem aceitar sua entrada x, atingindo uma configuração [q, y, ε], com $q \in F$, que não é uma configuração final. Simulando A, B pode atingir a configuração [q, y, I'], mas, como q não é final, o estado d' não pode ser atingido, e I' não pode ser removido da pilha.

6.4 - Equivalência das classes de linguagens definidas por apnd's e glc's

Esta seção mostra que os apnd's reconhecem exatamente as llc's, isto é, as linguagens definidas por glc's. Devido aos resultados da seção anterior, basta considerar aqui uma forma de aceitação, no caso, a aceitação por pilha vazia. O primeiro resultado corresponde ao Teorema abaixo.

Teorema 6.3: Seja L uma llc. Então existe um apnd que aceita L por pilha vazia.

Dem.: Se L é uma llc, então existe uma glc $G = < N, \Sigma, P, S >$ tal que L=L(G). A demonstração é feita através da construção de um apnd A que aceita L por pilha vazia a partir da gramática G, de forma semelhante à utilizada em um dos exemplos vistos anteriormente.

O apnd em questão tem apenas um estado: $A = < \{q\}, \Sigma, N \cup \Sigma, \delta, q, S >$, com δ como se segue:

- para cada nãoterminal A, se as regras de A são $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2$, … $A \rightarrow \alpha_n$, temos $\delta(q, \varepsilon, A) = \{ (q, \alpha_1), (q, \alpha_2), \ldots (q, \alpha_n) \}$.
- para cada terminal a, temos $\delta(q, a, a) = \{ (q, \varepsilon) \}$.

A prova de que $L_{pv}(A) = L(G)$ se baseia na relação entre uma computação de A (uma seqüência de configurações sucessivas de A) que aceita uma cadeia x, e uma derivação esquerda da mesma cadeia x. Deixamos como exercício completar os

detalhes da demonstração. O Exemplo 6.3 permite ver, para um caso particular, a correspondência entre os passos da derivação de uma cadeia x em G e as configurações assumidas por A até a aceitação de x.

O segundo resultado corresponde ao Teorema 6.4. A demonstração encontrada na referência citada anteriormente difere ligeiramente da apresentada aqui.

Teorema 6.4: Se L é uma linguagem aceita por pilha vazia por um apnd A, então L é uma llc.

Dem.: o teorema decorre dos Lemas 1 e 2 a seguir.

Lema 1: Para cada apnd A, existe um apnd B satisfazendo as duas condições:

- L_{pv} (B) = L_{pv} (A).
- B só tem um estado.

Dem.: Seja A = < K, Σ, δ, i, I, >. Vamos construir o apnd equivalente

B = <{ u }, Σ, Γ', δ', u, X, ∅ >, com Γ' = { X } ∪ (K×Γ×K).

Como o estado do apnd B não contém nenhuma informação (é sempre u), durante a simulação de A por B, a informação sobre o estado de A deve ser anotada na pilha de B. À primeira vista, bastaria anotar o estado s_1 de A no topo da pilha de B, no momento do empilhamento de um símbolo Z, juntamente com esse símbolo: [s_1, Z]. Entretanto, neste caso, quando Z fosse desempilhado, toda a informação sobre o estado de A se perderia. Por essa razão, vamos anotar também, com cada símbolo Z, o estado s_2, o estado assumido por A quando esta particular instância de Z for desempilhada. Note que se Z é substituído por outro símbolo Z', o mesmo estado s_2 será descoberto quando Z' for retirado.

Os símbolos do alfabeto Γ' são triplas [s_1, Z, s_2], com um símbolo Z de Γ e dois estados s_1 e s_2. O símbolo X de B será usado apenas como símbolo inicial, e não precisa ser tratado da mesma forma.

Assim, a cada símbolo Z são acrescidos dois estados s_1 e s_2: s_1 será o estado de A quando esta ocorrência de Z aparecer no topo; s_2 será o estado de A quando o símbolo do topo passar a ser aquele imediatamente abaixo de Z. Para garantir a correção da simulação de A, dois símbolos [s_1, Z_1, s_2] e [s_3, Z_2, s_4], que apareçam em posições consecutivas na pilha de B, devem sempre satisfazer a propriedade $s_2 = s_3$. Assim, quando Z_2 se tornar o estado do topo, o estado será s_2, aquele estado previsto (de forma não deterministica) por ocasião do empilhamento de Z_1.

Adicionalmente, se a pilha só contem um símbolo [s_1, Z, s_2], s_2 será o estado de A em que a pilha ficará vazia. A definição da função de transição δ' de B garante estas propriedades:

- transição geral:

  Se (q_1, Z_1Z_2 … Z_n) ∈ δ(q, a, Z),

  então (u, [q_1, Z_1, q_2][q_2, Z_2, q_3] … [q_n, Z_n, q_{n+1}]) ∈ δ'(u, a, [q, Z, q_{n+1}]),

  para quaisquer valores de q_2, …, q_n escolhidos em K.

- caso particular n = 0:

  Se (q_1, ε) ∈ δ(q, a, Z), então (u, ε) ∈ δ'(u, a, [q, Z, q_1]).

- início da simulação:

para cada $q \in K$, $(u, [i, I, q]) \in \delta'(u, \varepsilon, X)$.

Note que B prepara a simulação, colocando na pilha de B um símbolo [i, I, q], com a seguinte informação: o estado inicial i de A, o símbolo inicial I de A, e a indicação de que eventualmente, quando a pilha se tornar vazia, o estado será o estado q. O estado q é escolhido de forma não determinística, razão pela qual todas as possibilidades estão previstas no caso do início da simulação.

O restante da demonstração consiste em mostrar que a cada computação de A corresponde uma computação de B, e vice versa, de forma que A e B aceitam a mesma linguagem por pilha vazia. Novamente, os detalhes são deixados como exercício.

Exemplo 6.4: Seja o apnd $A = < K, \Sigma, \Gamma, \delta, i, I, \varnothing >$, com $K=\{0,1\}$, $\Sigma=\{a, b\}$ e $\Gamma=\{I, a, b\}$, sendo δ dada pela tabela a seguir. Cada linha descreve um elemento (p, α) de $\delta(q, a, Z)$. As linhas estão numeradas para referência posterior.

	q	a	Z	p	α
1	0	a	I	0	a
2	0	b	I	0	b
3	0	a	a	0	aa
4	0	a	b	0	ab
5	0	b	a	0	ba
6	0	b	b	0	bb
7	0	ε	I	1	ε
8	0	ε	a	1	a
9	0	ε	b	1	b
10	1	a	a	1	ε
11	1	b	b	1	ε

O apnd A é essencialmente não determinístico, e aceita por pilha vazia a linguagem $\{ xx^R \mid x \in \{a, b\}^* \}$. Lembramos que x^R indica a cadeia reversa ou refletida de x: se tivermos $x = x_1x_2\ldots x_n$, então $x^R = x_n\ldots x_2x_1$.

Vamos construir o apnd B que deve aceitar por pilha vazia a mesma linguagem. Temos $B = <\{ u \}, \Sigma, \Gamma', \delta', u, X, \varnothing >$, onde

$\Gamma' = \{ X, [0, I, 0], [0, I, 1], [1, I, 0], [1, I, 1], [0, a, 0], [0, a, 1],$

$[1, a, 0], [1, a, 1], [0, b, 0], [0, b, 1], [1, b, 0], [1, b, 1] \},$

e δ' é descrita pela tabela a seguir:

q	a	Z	p	α	comentários
u	ε	X	u u	[0, I, 0] [0, I, 1]	iniciação
u u	a a	[0, I, 0] [0, I, 1]	u u	[0, a, 0] [0, a, 1]	1
u u	b b	[0, I,0] [0, I, 1]	u u	[0, b, 0] [0, b, 1]	2

q	a	Z	p	α	comentários
u	a	[0, a, 0]	u	[0, a, 0] [0, a, 0]	3
			u	[0, a, 1] [1, a, 0]	
		[0, a, 1]	u	[0, a, 0] [0, a, 1]	
			u	[0, a, 1] [1, a, 1]	
u	a	[0, b, 0]	u	[0, a, 0] [0, b, 0]	4
			u	[0, a, 1] [1, b, 0]	
u	a	[0, b, 1]	u	[0, a, 0] [0, b, 1]	
			u	[0, a, 1] [1, b, 1]	
u	b	[0, a, 0]	u	[0, b, 0] [0, a, 0]	5
			u	[0, b, 1] [1, a, 0]	
u	b	[0, a, 1]	u	[0, b, 0] [0, a, 1]	
			u	[0, b, 1] [1, a, 1]	
u	b	[0, b, 0]	u	[0, b, 0] [0, b, 0]	6
			u	[0, b, 1] [1, b, 0]	
u	b	[0, b, 1]	u	[0, b, 0] [0, b, 1]	
			u	[0, b, 1] [1, b, 1]	
u	ε	[0, I, 1]	u	ε	7
u	ε	[0, a, 0]	u	[1, a, 0]	8
u	ε	[0, a, 1]	u	[1, a, 1]	
u	ε	[0, b, 0]	u	[1, b, 0]	9
u	ε	[0, b, 1]	u	[1, b, 1]	
u	a	[1, a, 1]	u	ε	10
u	a	[1, b, 1]	u	ε	11

A última coluna indica a associação das linhas da tabela de δ' com as linhas da tabela de δ.

Suponhamos agora que as máquinas A e B recebem como entrada x = aabaabaa. Vamos mostrar as configurações assumidas pelas duas máquinas, durante a análise de x.

A	B
	[u, aabaabaa, X]
[0, aabaabaa, I]	[u, aabaabaa, [0, I, 1]]
[0, abaabaa, a]	[u, abaabaa, [0, a, 1]]
[0, baabaa, aa]	[u, baabaa, [0, a, 1][1, a, 1]]
[0, aabaa, baa]	[u, aabaa, [0, b, 1][1, a, 1][1, a, 1]]
[0, abaa, abaa]	[u, abaa, [0, a, 1][1, b, 1][1, a, 1][1, a, 1]]
[1, abaa, abaa]	[u, abaa, [1, a, 1][1, b, 1][1, a, 1][1, a, 1]]
[1, baa, baa]	[u, baa, [1, b, 1][1, a, 1][1, a, 1]]
[1, aa, aa]	[u, aa, [1, a, 1][1, a, 1]]
[1, a, a]	[u, a, [1, a, 1]]
[1, ε, ε]	[u, ε, ε]

Como se pode ver, cada terceira componente de símbolo da pilha de B corresponde exatamente ao estado de A, no momento em que o símbolo (ou aquele que o substituiu) é retirado da pilha.

Lema 2: Se o apnd A tem apenas um estado, L_{pv} (A) é uma llc.

Dem.: Seja A = < { u }, Σ, Γ, δ, u, I, ∅ > um apnd com apenas um estado u. Construímos uma glc G tal que L(G) = L_{pv}(A):

G = < N, Σ, P, I >

sendo as seguintes as regras de P:

se (u, γ) ∈ δ(u, a, Z), então P tem uma regra Z → aγ.

A demonstração consiste na verificação de que a cada computação (sequência de configurações) de A, corresponde uma derivação (esquerda) de G.

Exemplo 6.5: Considere o apnd B do Exemplo 6.4. A gramática G pode ser construída como se segue:

X → [0, I, 0] | [0, I, 1]

[0, I, 0] → a [0, a, 0]

[0, I, 1] → a [0, a, 1]

[0, I, 0] → b [0, b, 0]

[0, I, 1]→ b [0, b, 1]

[0, a, 0] → a [0, a, 0] [0, a, 0] | a [0, a, 1] [1, a, 0]

[0, a, 1] → a [0, a, 0] [0, a, 1] | a [0, a, 1] [1, a, 1]

[0, b, 0] → a [0, a, 0] [0, b, 0] | a [0, a, 1] [1, b, 0]

[0, b, 1] → a [0, a, 0] [0, b, 1] | a [0, a, 1] [1, b, 1]

[0, a, 0] → b [0, b, 0] [0, a, 0] | b [0, b, 1] [1, a, 0]

[0, a, 1] → b [0, b, 0] [0, a, 1] | b [0, b, 1] [1, a, 1]

[0, b, 0] → b [0, b, 0] [0, b, 0] | b [0, b, 1] [1, b, 0]

[0, b, 1] → b [0, b, 0] [0, b, 1] | b [0, b, 1] [1, b, 1]

[0, I, 1] → ε

[0, a, 0] → [1, a, 0]

[0, a, 1] → [1, a, 1]

[0, b, 0] → [1, b, 0]

[0, b, 1] → [1, b, 1]

[1, a, 1] → a

[1, b, 1] → b

Assim, podemos derivar x=aabaabaa, pela derivação correspondente às computações de A e de B vistas no exemplo anterior:

X ⇒ [0, I, 1] ⇒ a [0, a, 1] ⇒ a a [0, a, 1] [1, a, 1]

⇒ a a b [0, b, 1] [1, a, 1] [1, a, 1]

⇒ a a b a [0, a, 1] [1, b, 1] [1, a, 1] [1, a, 1]

$\Rightarrow$ a a b a [1, a, 1] [1, b, 1] [1, a, 1] [1, a, 1]

$\Rightarrow$ a a b a a [1, b, 1] [1, a, 1] [1, a, 1]

$\Rightarrow$ a a b a a b [1, a, 1] [1, a, 1]

$\Rightarrow$ a a b a a b a [1, a, 1]

$\Rightarrow$ a a b a a b a a

Exercício 6.3: Verificar se a gramática acima tem regras inúteis, isto é, regras que nunca são usadas na derivação de alguma sentença da linguagem.

Um assunto que não será discutido neste capítulo é a definição de um *autômato de pilha determinístico* (*apd*), a partir do qual são definidas as *linguagens determinísticas*. Esta classe é um subconjunto próprio da classe das (de todas as) llc's. A importância das linguagens determinísticas, pode ser deduzida do fato de que todas as linguagens de programação são tratadas como linguagens determinísticas pelos compiladores.

(maio 99)