Capítulo 10: Linguagens determinísticas e seus aceitadores

José Lucas Rangel

10.1 - Introdução.

Nos capítulos relativos às linguagens livres de contexto, observamos que as classes de linguagens aceitas por autômatos de pilha determinísticos (apd) e não determinísticos (apnd) não são as mesmas, ao contrário do que acontece, por exemplo, com os autômatos finitos. Neste capítulo, vamos apresentar exemplos e algumas propriedades das linguagens determinísticas, isto é, das linguagens aceitas por apd's.

10.2 - Autômatos de pilha determinísticos.

Como observado anteriormente, um apd é um caso particular de apnd, em que, a partir de qualquer configuração existe, no máximo, uma configuração acessível. Em termos da definição de um apd $A = <K, \Sigma, \Gamma, \delta, i, I, F>$, isto significa:

- para quaisquer $q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, $\delta(q, a, Z)$ tem, no máximo, um elemento;
- se, para algum $q \in K$ e algum $Z \in \Gamma$, $\delta(q, \varepsilon, Z)$ não é vazio, então para todos os símbolos $a \in \Sigma$, $\delta(q, a, Z)$ deve ser vazio.

A primeira condição impede a possibilidade de escolha entre duas transições com um símbolo; a segunda condição evita a possibilidade de escolha entre a leitura de um símbolo de Σ, e a leitura de um ε, ou seja, a escolha entre ler e não ler um símbolo da entrada.

Note, entretanto, que, ao contrário do que acontece com os autômatos finitos determinísticos, pode não existir nenhuma configuração alcançável a partir de uma dada configuração, mesmo que a entrada ainda não tenha sido completamente lida. Por exemplo, a pilha pode ficar vazia. Assim, uma entrada x pode ser deixar de ser aceita por um apd sem ter sido completamente lida.

Definimos uma *linguagem determinística* como sendo uma linguagem aceita *por estado final* por um apd. A razão para essa escolha está nas propriedades a seguir, que limitam o interesse das linguagens aceitas por apd's por pilha vazia.

Dizemos que uma cadeia y é um *prefixo* de outra cadeia x se x = y z para algum z; dizemos que y é um *prefixo próprio* de x se y é um prefixo de x, e $y \neq x$.

Dizemos que uma linguagem L tem a *propriedade dos prefixos* se nenhuma cadeia de L tem um prefixo próprio que também pertence a L, ou seja, se dadas duas cadeias x, y pertencentes a L,

$x = yz$ implica $z = \varepsilon$.

Fato: Se $L = L_{pv}(M)$, para algum apd M, então L tem a propriedade dos prefixos.

Dem.: Suponha que $x \in L$. Isto significa que a partir da configuração inicial [i, x, I], correspondente a x, pode ser alcançada uma configuração final (para aceitação por pilha vazia) $[q, \varepsilon, \varepsilon]$. Considere uma cadeia $y=xz$, com $z \neq \varepsilon$. A partir da configuração inicial correspondente a y, [i, y, I] = [i, xz, I] pode ser alcançada a configuração [q, z, ε], sem que a parte z da entrada seja lida. Como a partir desta última

configuração nenhuma outra pode ser atingida, nenhum símbolo de z será lido, e a cadeia y não será aceita.

Exercício: Seja L uma linguagem qualquer num alfabeto Σ, e seja $ um símbolo novo, não pertencente a Σ.

(a) Mostre que a linguagem L•{$} = { x$ | x(L} tem a propriedade dos prefixos;
(b) Mostre também que se L é uma linguagem determinística, ou seja, se L é aceita por estado final por um apd, L•{$} é aceita por um apd por pilha vazia.

Exercício: Mostre que as linguagens { $a^i b^j$ | i=j }, { $a^i b^j$ | i>j }, { $a^i b^j$ | i<j } e { $a^i b^j$ | i≠j }, são determinísticas.

10.3 Propriedades de fechamento da classe das linguagens determinísticas

Para as linguagens determinísticas, não existe um correspondente do Lema do Bombeamento (*pumping lemma*), de forma que a principal maneira de provar que uma linguagem livre de contexto não é determinística é através do uso de propriedades de fechamento. Por exemplo, veremos posteriormente que a classe das linguagens determinísticas é fechada para a operação de complemento. Assim, uma llc cujo complemento não é uma llc não pode ser determinística.

Vamos começar pela propriedade do fechamento com o complemento.

Teorema: A classe das linguagens determinísticas é fechada para o complemento.

Dem.: Seja M um apd que aceita uma linguagem L, por estado final.

Vamos mostrar como construir um apd M' que aceita o complemento de L. A idéia central é semelhante à da prova do fechamento da classe das linguagens regulares para o complemento: vamos trocar os estados finais pelos não finais. Assim, pelo menos em princípio, se L é a linguagem aceita por M = < K, Σ, Γ, δ, i, I, F>, o apd M' = < K, Σ, Γ, δ, i, I, K-F> deve aceitar o complemento de L.

Entretanto, algumas possibilidades devem ser consideradas:

1. M não aceita uma entrada x porque M pára sem que a entrada x tenha sido toda lida. Se construirmos M' apenas pela troca dos estados finais e não finais de M, x continuaria não sendo aceita, o que seria incorreto.
2. M lê toda a entrada x, mas após isso, M executa uma infinidade de passos com entrada ε, sem retirar símbolos da pilha, e, nessa fase, passa por estados finais e não finais. Note que M aceita x, mas se construirmos M' apenas pela troca dos estados finais e não finais, x continuaria sendo aceita, o que seria incorreto.

Para resolver esses problemas, é necessário fazer várias transformações em M antes de trocar os estados finais e não finais. Os detalhes da efetiva construção de M' a partir de M podem ser vistos no livro de Hopcroft e Ullman.

A partir deste resultado, podemos mostrar que a classe das linguagens determinísticas é contida propriamente na classe das llc.

Fato: A llc L = { $a^i b^j c^k$ | ou i≠j ou j≠k } não é determinística.

Dem. Basta mostrar que o complemento não é uma livre de contexto.

Intuitivamente, L "não pode" ser determinística porque comparar i e j envolve um tratamento da pilha (empilhar i símbolos, desempilhar j símbolos) completamente diferente do necessário para comparar j e k (empilhar j símbolos, desempilhar k símbolos).

Fato: A classe das linguagens determinísticas não é fechada para as operações de união e interseção.

Dem.: Para a união, note que a linguagem L do exemplo anterior pode ser vista como a união de duas llc determinísticas:

$L = \{ a^i b^j c^k \mid \text{ou } i \neq j \text{ ou } j \neq k \} = L_1 \cup L_2$,

sendo

$L_1 = \{ a^i b^j c^k \mid i \neq j \}$
$L_2 = \{ a^i b^j c^k \mid j \neq k \}$

Para mostrar que a classe das linguagens determinísticas não é fechada para a interseção, podemos usar os fatos de que a classe não é fechada para a união, mas é fechada para o complemento. Uma das relações de de Morgan é

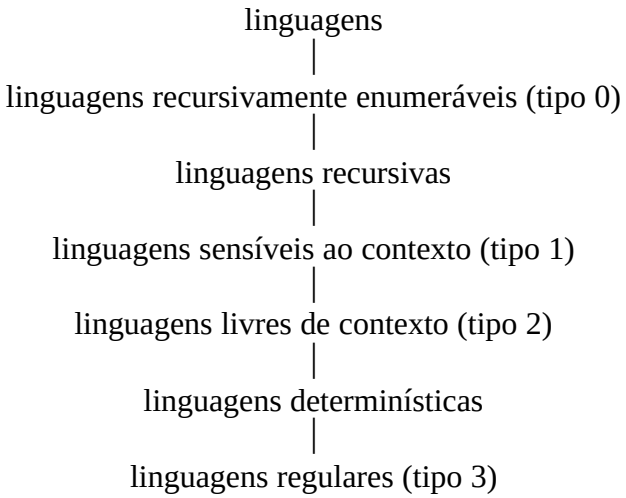
$$L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$$

Assim, se a classe das linguagens determinísticas fosse fechada para a interseção, também seria fechada para a união. Pela contradição, vemos que a classe não pode ser fechada para interseção.

Exercicio: Mostrar que $L = \{ a^i b^j \mid i=j \text{ ou } i=2j \}$ é uma linguagem não determinística, que pode ser escrita como a união de duas linguagens determinísticas.

10.4 Conclusão

Com os exemplos e as propriedades estabelecidas neste capítulo e nos anteriores, a hierarquia das classes de linguagens definidas aqui pode ser representada pela figura abaixo, em que todas as inclusões indicadas são próprias.



(julho 99)