Departamento de Ciência de Computadores FCUP

Modelos de Computação CC1004 2020/2021

Folha Prática 9

- Uma gramática independente de contexto $G = (V, \Sigma, P, S)$
  - está na **forma normal de Greibach** sse as regras são da forma $A \to a\gamma$, com $\gamma \in V^\star$ e $a \in \Sigma$;
  - está na **forma normal de Chomsky** sse as regras são da forma $A \to a$ ou $A \to BC$, com $B, C \in V$ e $a \in \Sigma$.

- **Observação:** Se G for uma GIC na forma normal de Chomsky ou na forma normal de Greibach, $\varepsilon \notin \mathcal{L}(G)$. Existem extensões destas formas que permitem ter a regra $S \to \varepsilon$ (para o símbolo inicial) se S não ocorrer no lado direito de nenhuma regra. Pode-se provar que qualquer LIC pode ser gerada por uma GIC nessas condições. A prova é construtiva: dada uma GIC que gere L, descreve o algoritmo de conversão dessa GIC à forma normal pretendida.

- **As linguagens reconhecidas por APs são LICs e as LICs são reconhecidas por APs:**
  - *Transformação de AP em GIC:* Seja $A = (Q, \Sigma, \Gamma, \delta, s_0, Z_0, \{\})$ um AP com aceitação por pilha vazia. A GIC G assim definida gera $\mathcal{L}(A)$: as variáveis de G são representadas por ternos $[q, Z, q']$, com $q, q' \in Q$ e $Z \in \Gamma$, excepto o símbolo inicial que será S, e as regras de G são
    * $S \to [q_0, Z_0, q]$, para todo $q \in Q$;
    * $[q, Z, q'] \to a$, se $(q', \varepsilon) \in \delta(q, a, Z)$, com $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$ e $q, q' \in Q$;
    * $[q, Z, q_n] \to a[q', X_1, q_1] \cdots [q_{n-1}, X_n, q_n]$, se $(q', X_1 \cdots X_n) \in \delta(q, a, Z)$, com $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, e $X_1, \ldots, X_n \in \Gamma$, para todas as sequências $(q_1, \ldots, q_{n-1}, q_n) \in Q^n$.
  - *Transformação de GIC em AP:* Usando a forma normal de Greibach, podemos provar que qualquer LIC pode ser reconhecida por um autómato de pilha. Dada uma GIC $G = (V, \Sigma, P, S)$ na forma normal de Greibach, a linguagem $\mathcal{L}(G)$ é aceite por pilha vazia pelo autómato de pilha $A = (\{q\}, \Sigma, V, \delta, q, S)$, com $\delta(q, a, X) = \{(q, \gamma) \mid (X \to a\gamma) \in P\}$.

- O **algoritmo CYK**, de **C**ocke-**Y**ounger-**K**asami, permite **decidir** em $\Theta(n^3|G|)$ se uma palavra $x \in \Sigma^\star$ com $|x| = n$ pertence à linguagem gerada por uma **gramática G na forma normal de Chomsky**. Para cada gramática G fixa, a complexidade é $\Theta(n^3)$.

  Quando aplicado à palavra $x = x_1 x_2 \ldots x_n$, com $x_i \in \Sigma$, para todo i, constrói uma tabela como a que se indica a seguir, em que a entrada assinalada por $x_i \cdots x_{i+t}$ tem o conjunto das categorias possíveis para a subpalavra $x_i \cdots x_{i+t}$ de x, de acordo com a gramática (ou seja, o conjunto de variáveis da gramática que podem gerar essa sequência de terminais).

#n	$x_1 \cdots x_n$					
#$n{-}1$	$x_1 \cdots x_{n-1}$	$x_2 \cdots x_n$				
$\vdots$	$\vdots$	$\vdots$	$\vdots$			
#3	$x_1x_2x_3$	$x_2x_3x_4$	$\cdots$	$x_{n-2}x_{n-1}x_n$		
#2	x_1x_2	x_2x_3	$\cdots$	$x_{n-2}x_{n-1}$	$x_{n-1}x_n$	
#1	x_1	x_2	$\cdots$	x_{n-2}	x_{n-1}	x_n
	x_1	x_2	$\cdots$	x_{n-2}	x_{n-1}	x_n

O algoritmo CYK usa uma técnica de desenvolvimento de algoritmos designada por *programação dinâmica*: a tabela é construída por linhas, de baixo para cima (o que corresponde à procura de árvores de derivação usando uma estratégia de construção *bottom-up*).

Para determinar as categorias de uma subpalavra y, analisam-se todas as partições de y como zw, com $z \neq \varepsilon$ e $w \neq \varepsilon$, e, considerando **as categorias anteriormente encontradas** para z e w, obtêm-se as categorias possíveis para zw. A palavra zw (isto é, y) pode ser do tipo T sse Z e W forem admissíveis para z e w e existir a regra $T \to ZW$.

Por exemplo, se a sequência fosse $x_3x_4x_5x_6$, seriam consideradas as partições $x_3|x_4x_5x_6$, $x_3x_4|x_5x_6$, e $x_3x_4x_5|x_6$, **Consultar exemplos nos slides.**

Dada $G = (V, \Sigma, P, S)$ e $x = x_1x_2 \ldots x_n$, seja $N[i, i+t]$ o conjunto de variáveis que geram $x_i \ldots x_{i+t}$, i.e., $N[i, i+t] = \{A \mid A \in V,\ A \Rightarrow^\star_{\mathcal{G}} x_i \ldots x_{i+t}\}$.

```
ALGORITMOCYK(x)
| Para i ← 1 até n fazer N[i,i] := {A | A ∈ V, A → x_i};
| Para todo (i,j), com i ≠ j e 1 ≤ i ≤ n e 1 ≤ j ≤ n, fazer N[i,j] := ∅;
| Para t ← 1 até n − 1 fazer
|     Para i ← 1 até n − t fazer
|         Para k ← i até i + t − 1 fazer
|             Para todo BC ∈ N[i,k]N[k+1,i+t] fazer
|                 Para todo A tal que (A → BC) ∈ P fazer
|                     N[i,i+t] := N[i,i+t] ∪ {A}
| Se S ∈ N[1,n] então retorna "Sim"; senão retorna "Não";
```

Exercícios

1. Considere o autómato $\mathcal{A} = (\{s_0, s_1\}, \{\mathtt{0}, \mathtt{1}\}, \{\mathtt{Z}, \mathtt{A}, \mathtt{B}\}, \delta, s_0, \mathtt{Z}, \{\ \})$, com aceitação por pilha vazia, com $\delta(s, a, X) = \emptyset$ exceto para

$$\begin{array}{lll}
\delta(s_0, \mathtt{0}, \mathtt{Z}) = \{(s_0, \mathtt{BZ})\} & \delta(s_0, \mathtt{1}, \mathtt{Z}) = \{(s_0, \mathtt{AZ})\} & \delta(s_0, \mathtt{0}, \mathtt{B}) = \{(s_0, \mathtt{BB})\} \\
\delta(s_0, \mathtt{1}, \mathtt{A}) = \{(s_0, \mathtt{AA})\} & \delta(s_0, \mathtt{0}, \mathtt{A}) = \{(s_0, \varepsilon)\} & \delta(s_0, \mathtt{1}, \mathtt{B}) = \{(s_0, \varepsilon)\} \\
\delta(s_0, \varepsilon, \mathtt{A}) = \{(s_1, \varepsilon)\} & \delta(s_1, \varepsilon, \mathtt{A}) = \{(s_1, \varepsilon)\} & \delta(s_1, \varepsilon, \mathtt{Z}) = \{(s_1, \varepsilon)\}
\end{array}$$

o qual reconhece a linguagem das palavras de $\{\mathtt{0}, \mathtt{1}\}^\star$ que têm menos $\mathtt{0}$'s do que $\mathtt{1}$'s.

Por aplicação do método de conversão, determine uma GIC que gere $\mathcal{L}(\mathcal{A})$.

2. Seja $\mathcal{G} = (\{A, B, C\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, P, A)$, com

$$P = \{A \to \mathtt{aa},\ A \to C,\ A \to \mathtt{bba}A,\ C \to \mathtt{cc}C,\ C \to \mathtt{cc}B,\ C \to \mathtt{cc},\ B \to \mathtt{b}B,\ B \to \mathtt{ba}A,\ B \to \mathtt{b}\}.$$

a) Converta a gramática $\mathcal{G}$ à forma normal de Greibach.

b) Usando a gramática que obteve em **2a)**, determine o AP para reconhecer $\mathcal{L}(\mathcal{G})$, por pilha vazia, que se obtém pelo método de construção definido acima.

c) Converta a gramática $\mathcal{G}$ para a forma normal de Chomsky.

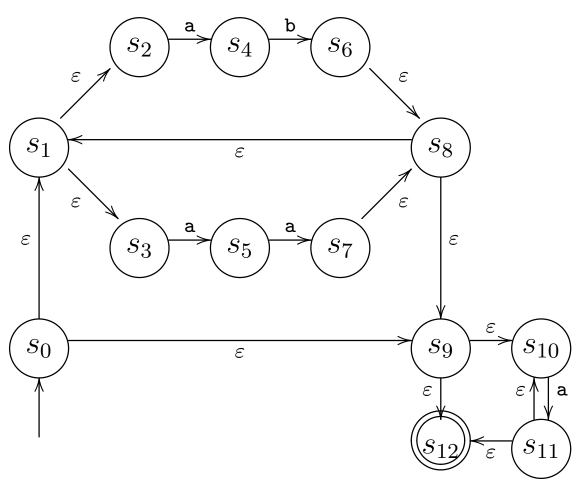
d) Aplique o algoritmo CYK para mostrar que $\mathtt{bbaccb}$ pertence à linguagem gerada pela gramática que obteve em **2c)** e, portanto, a $\mathcal{L}(\mathcal{G})$. Por análise da tabela construída, determine se $x \in \mathcal{L}(\mathcal{G})$ para cada subpalavra x de $\mathtt{bbaccb}$, com $x \neq \varepsilon$.

Linguagens regulares geradas por GICs lineares à direita (ou lineares à esquerda)

- Uma gramática independente de contexto $G = (V, \Sigma, P, S)$
  - é **linear à direita** sse qualquer regra é da forma $A \to w$ ou $A \to wB$, com $w \in \Sigma^\star$ e $B \in V$;
  - é **linear à esquerda** sse qualquer regra é da forma $A \to w$ ou $A \to Bw$, com $w \in \Sigma^\star$ e $B \in V$;
- **GICs lineares à direita (esquerda) geram linguagens regulares, e vice-versa:**
  - Dado um AFD $A = (S, \Sigma, \delta, s_0, F)$, a linguagem $\mathcal{L}(A)$ é gerada pela gramática linear à direita $G = (\{V_s \mid s \in S\}, \Sigma, P, V_{s_0})$, em que P tem a regra $V_s \to aV_{s'}$ sse $\delta(s, a) = s'$ e ainda uma regra $V_f \to \varepsilon$, **por cada estado final** $f \in F$. O mesmo algoritmo pode ser trivialmente adaptado se A for um AFND ou AFND-ε. Um percurso no autómato do estado s_0 até um estado final, em que se consumiu a palavra x, corresponde a uma derivação da palavra x em G a partir de V_{s_0}, e vice-versa.
  - A gramática obtida por aplicação do método de conversão a um **AFD é não ambígua**. Portanto, qualquer linguagem regular é não ambígua.
  - Dada uma GIC G linear à direita, podemos obter um AFND-ε que reconhece $\mathcal{L}(G)$ pelo algoritmo seguinte. Começamos por substituir todas as regras da forma $X \to w$ por $X \to wX_f$, em que X_f é uma variável nova (e a mesma para todas as regras), e será a única com produção $X_f \to \varepsilon$ (o AFND-ε terá um único estado final). Depois, cada regra da forma $X \to wY$, com $|w| \geq 2$, é substituída por $|w|$ regras, $X \to a_1Y_1$, $Y_1 \to a_2Y_2, \ldots, Y_{k-1} \to a_kY$, sendo $w = a_1a_2 \ldots a_k$, e $Y_1, \ldots Y_{k-1}$ variáveis novas (criadas para cada regra). O conjunto de estados do AFND-ε corresponde ao conjunto de variáveis da nova gramática e X_f ao estado final. Cada regra $X \to \alpha Y$, com $\alpha \in \Sigma \cup \{\varepsilon\}$, define uma transição do estado X para o estado Y por α.

Exercícios

3. Seja L a linguagem de alfabeto $\{\mathtt{a}, \mathtt{b}\}$ aceite pelo AFND-ε $A = (S, \Sigma, \delta, s_0, \{s_{12}\})$ representado.



a) Determine a gramática que resulta da aplicação do algoritmo de transformação ao autómato A.

b) Mostre que a gramática que obteve em **3a)** é ambígua.

c) Sabendo que o AFND-ε A resultou da aplicação do algoritmo de Thompson a uma expressão regular r que não contém ε, determine r. A seguir, indique r na forma abreviada.

d) Por aplicação do método de conversão (de um AFND-ε para AFD), determine um AFD A' equivalente ao AFND-ε A.

e) Determine a GIC linear à direita não ambígua que resulta da aplicação do algoritmo de transformação ao AFD A' que obteve na alínea anterior.

f) Analise a forma das palavras de L e determine uma GIC linear à esquerda, não ambígua, que gere L.

4. Considere a gramática independente de contexto $\mathcal{G} = (\{A, B, C\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, P, A)$, com

$$P = \{A \to \mathtt{aa},\ A \to C,\ A \to \mathtt{bba}A,\ C \to \mathtt{cc}C,\ C \to \mathtt{cc}B,\ B \to \mathtt{b}B,\ B \to \mathtt{ba}A,\ B \to \varepsilon\}.$$

a) Justifique que $\mathcal{G}$ é linear à direita. Conclua que $\mathcal{L}(\mathcal{G})$ é regular.

b) Determine o AFND-ε que se obtém por aplicação do algoritmo de conversão a $\mathcal{G}$.

c) Determine uma expressão regular que descreva $\mathcal{L}(\mathcal{G})$.

d) [*] Averigue se a gramática $\mathcal{G}$ é ambígua. Justifique.

5. Seja L a linguagem alfabeto $\{\mathtt{0}, \mathtt{1}\}$ descrita pela expressão $\mathtt{0}^\star\mathtt{1}^\star(\mathtt{00} + \mathtt{11})^\star$.

a) Analise a expressão e determine uma GIC que gere L e não seja linear à direita (nem linear à esquerda).

b) Por análise da expressão, determine uma GIC que gere L e seja linear à direita.

c) Determine o AFD mínimo que aceita L e, por aplicação do método de conversão a esse AFD, determine uma GIC não ambígua que gere L.

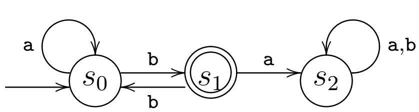
6. Seja $L = \mathcal{L}(\mathtt{0}^\star\mathtt{1}^\star(\mathtt{00} + \mathtt{11})(\mathtt{00} + \mathtt{11})^\star)$. Determine uma GIC não ambígua que gere L, baseando-se:

a) numa caraterização da estrutura das palavras de L, de forma não ambígua.

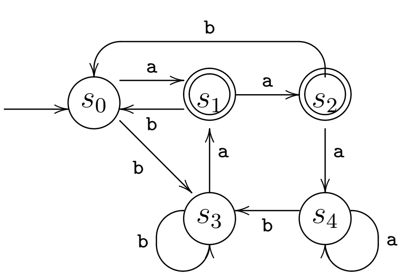
b) na construção de um AFD para L e sua conversão para uma GIC linear à direita não ambígua.

7. Por aplicação de algoritmos de conversão, determine uma GIC não ambígua que gere a linguagem aceite pelo autómato finito representado em cada alínea, para $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

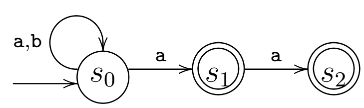
a)



b)



c)



d)