INSTITUTO FEDERAL
Paraná
Campus Pinhais

Linguagens formais e autômatos

Forma Normal de Chomsky

Gabriel V C Candido
gabriel.candido@ifpr.edu.br

Instituto Federal do Paraná - Pinhais

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Sumário

Relembrando . . .

- Eliminação de recursividade no símbolo de partida
- Eliminação de regras λ
- Eliminação de regras de cadeia
- Eliminação de símbolos inúteis (não geram terminais ou são inalcançáveis)

Nosso objetivo era ter gramáticas com produções que "não encolhiam".

Relembrando . . .

Regras do tipo:

- $S \to \lambda$
- $A \to a$
- $A \to w$, w pode ter variáveis, mas $|w| \geq 2$!!

Forma Normal de Chomsky

GLC com regras do tipo:

- $S \to \lambda$
- $A \to a$
- $A \to BC$, onde B, C são variáveis, e não são S

Note que $|BC| = 2$. E também que não pode haver recursividade no símbolo de partida!

Como transformar em FNC

- Queremos transformar $A \rightarrow w$ em $A \rightarrow BC$

Primeiro passo é substituir os terminais de w por variáveis:

G: $\{A \rightarrow bDcF$

$$G': \begin{cases} A \rightarrow B'DC'F \\ B' \rightarrow b \\ C' \rightarrow c \end{cases}$$

Como transformar em FNC

Agora temos que fazer as regras produzirem exatamente 2 variáveis. Fazemos isso de trás para frente:

G':

$$\begin{cases} A \to B'DC'F \\ B' \to b \\ C' \to c \end{cases}$$

G'':

$$\begin{cases} A \to B'DT_2 \\ B' \to b \\ C' \to c \\ T_2 \to C'F \end{cases}$$

G'':

$$\begin{cases} A \to B'T_1 \\ B' \to b \\ C' \to c \\ T_1 \to DT_2 \\ T_2 \to C'F \end{cases}$$

Exemplo 1

$$
G: \begin{cases} S \to aABC \mid a \\ A \to aA \mid a \\ B \to bcB \mid bc \\ C \to cC \mid c \end{cases}
$$

Exemplo 1

$$G: \begin{cases} S \to aABC \mid a \\ A \to aA \mid a \\ B \to bcB \mid bc \\ C \to cC \mid c \end{cases}$$

$$G': \begin{cases} S \to A'T_1 \mid a \\ A' \to a \\ T_1 \to AT_2 \\ T_2 \to BC \\ A \to A'A \mid a \\ B \to B'T_3 \mid B'C' \\ B' \to b \\ T_3 \to C'B \\ C' \to c \\ C \to C'C \mid c \end{cases}$$

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Exemplo 2

Gramática para expressões de adição.
$\Sigma = \{d, +, (,)\}$.

$$G: \begin{cases} S \to A \\ A \to T \mid A + T \\ T \to d \mid (A) \end{cases}$$

1. Eliminar regras de cadeia.

Exemplo 2

$$G: \begin{cases} S \to A \\ A \to T \mid A + T \\ T \to d \mid (A) \end{cases} \qquad G': \begin{cases} S \to d \mid (A) \mid A + T \\ A \to d \mid (A) \mid A + T \\ T \to d \mid (A) \end{cases}$$

2. Transformar para Forma Normal de Chomsky (FNC).

Exemplo 2

G'':

$$\begin{cases} S \to d \mid (A) \\ S \to A + T \\ A \to d \mid (A) \\ A \to A + T \\ T \to d \mid (A) \end{cases}$$

G''':

$$\begin{cases} S \to d \mid EAD \mid AMT \\ A \to d \mid EAD \mid AMT \\ T \to d \mid EAD \\ M \to + \\ E \to (\\ D \to) \end{cases}$$

Exemplo 2

$$G''': \begin{cases} S \to d \mid EAD \mid AMT \\ A \to d \mid EAD \mid AMT \\ T \to d \mid EAD \\ M \to + \\ E \to (\\ D \to) \end{cases}$$

$$G'''': \begin{cases} S \to d \mid EY \mid AX \\ A \to d \mid EY \mid AX \\ T \to d \mid EY \\ X \to MT \\ Y \to AD \\ M \to + \\ E \to (\\ D \to) \end{cases}$$

Forma Normal de Chomsky

- Árvore de derivação gera sempre um terminal ou dois filhos
- Ainda pode ter recursividade à esquerda

Eliminação de recursividade à esquerda

Queremos evitar laços infinitos na análise sintática por conta de regras do tipo $A \to Aa$.

G:

$$\begin{cases} A \to Aa \mid b \end{cases}$$

G':

$$\begin{cases} A \to bZ \mid b \\ Z \to aZ \mid a \end{cases}$$

Ambas as gramáticas acima fazem ba^*.

Exemplo 3

$(b + c)(a + b)^*$

$$G: \left\{ A \to Aa \mid Ab \mid b \mid c \right.$$

Exemplo 3

$(b + c)(a + b)^*$

$$G: \left\{ A \to Aa \mid Ab \mid b \mid c \right.$$

$$G': \begin{cases} A \to bZ \mid cZ \mid b \mid c \\ Z \to aZ \mid bZ \mid a \mid b \end{cases}$$

Exemplo 4

$(b + c)^* a (b + c)^*$

$$G: \begin{cases} A \to AB \mid BA \mid a \\ B \to b \mid c \end{cases}$$

Exemplo 4

$(b + c)^* a (b + c)^*$

$$G: \begin{cases} A \to AB \mid BA \mid a \\ B \to b \mid c \end{cases}$$

$$G': \begin{cases} A \to BAZ \mid aZ \mid BA \mid a \\ Z \to BZ \mid B \\ B \to b \mid c \end{cases}$$

Conclusão

- FNC ainda tem o problema da recursividade.
- Aprendemos a tratar a recursividade *direta* à esquerda.
- Ainda pode ter recursividade *indireta* à esquerda.