LINGUAGENS FORMAIS E AUTÔMATOS --- LINGUAGEM LIVRE DE CONTEXTO ---

Forma Normal de Greibach

Introdução

As formas normais estabelecem restrições rígidas na definição das produções, mas não reduz o poder de geração das GLCs

A ideia é facilitar a demonstração que toda linguagem livre de contexto pode ser reconhecida por um autômato com pilha

Uma GLC $G = (V, \Sigma, P, S)$ está na Forma Normal de Greibach (FNG) se todas as produções são da forma: $A \rightarrow \boldsymbol{a}\alpha$

Para $A \in V, a \in \Sigma$ e $\alpha \in V^*$

Caso particular: quando $\varepsilon \in L(G)$ permite-se que $S \rightarrow \varepsilon$, mas $S \notin \alpha$ (i.e., S não pode ocrrer do lado direito de uma regra de produção

Toda LLC pode ser gerada por uma GLC na FNG

Introdução

A Forma Normal de Greibach (FNG) é interessante porque, nas derivações, ela introduz os símbolos terminais no início. A FNG estabelece a equivalência entre GLC e AP

Por exemplo, considere a linguagem $L(G) = \{a^n b^n : n \geq 0\}$. Considere que as regras de produção da gramática G, já na Forma Normal de Greibach (FNG), sejam:

P: $S \rightarrow A \mid \varepsilon$

$A \rightarrow aAB \mid aB$

$B \rightarrow b$

Derivação de $\omega = aabb$ $\therefore$ $S \Rightarrow A \Rightarrow aAB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabb$

Sheila Greibach, UCLA

Cadeias de tamanho n são necessários n produções, ou $n + 1$ caso gere a cadeia vazia (ε)

Algoritmo

O algoritmo ocorre em seis etapas:

(1) simplificação da gramática;

(2) renomeação das variáveis em uma ordem crescente;

(3) colocar todas as produções na forma: $A_r \rightarrow A_s\alpha$, em que $r \leq s$

(4) exclusão das recursões à esquerda, da forma: $A_r \rightarrow A_r\alpha$

(5) um terminal no início do lado direito de cada produção

(6) transformação das produções na forma $A \rightarrow a\alpha$, onde α só contém variáveis

Algoritmo

(1) **Simplificação da gramática.** Veja nos slides anteriores

(2) **Renomeação das variáveis em uma ordem crescente.**

Por exemplo, **renomeie** todas as variáveis para $A_1 A_2 \cdots A_n$, sempre iniciando com a variável inicial

Diferente critérios de nomeação geram diferentes gramáticas FNG

Entretanto, todas são equivalentes e geram a mesma linguagem

Considere a seguinte GLC com as seguintes regras de produção:

$S \rightarrow AA \mid a$

$A \rightarrow SS \mid b$, que **renomeando** ficaria:

$A_1 \rightarrow A_2 A_2 \mid a$

$A_2 \rightarrow A_1 A_1 \mid b$

Algoritmo

(3) colocar todas as produções na forma: $A_r \rightarrow A_s\alpha$, em que $r \leq s$

Por exemplo: $A_1 \rightarrow A_3\alpha$

Devemos corrigir produções como: $A_2 \rightarrow A_1\alpha$

O exemplo anterior não está de acordo com esse critério:

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow A_1A_1 \mid b$

A solução acima é trocar "somente" o **primeiro A_1** (que é o que está errado!) pelas suas produções, isto é, $(A_2A_2 \mid a)$. No caso, o resultado ficaria assim:

$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

Algoritmo

A nova gramática ficou:

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

(4) **Exclusão das recursões à esquerda, da forma: $A_r \rightarrow A_r\alpha$**

Principalmente em compiladores, não pode ter recursão à esquerda

Seja uma produção do tipo: $A_1 \rightarrow A_1\alpha$. A Solução é criar uma variável auxiliar, digamos B_1, e incluir as duas regras: $B_1 \rightarrow \alpha$ e $B_1 \rightarrow \alpha B_1$, que agora é uma recursão à direita

Ou seja, elimina-se $A_1 \rightarrow A_1\alpha$ e inclui-se as duas regras: $B_1 \rightarrow \alpha$ e $B_1 \rightarrow \alpha B_1$

Algoritmo

No nosso exemplo a gramática estava:

$A_1 \rightarrow A_2 A_2 \mid a$

$A_2 \rightarrow A_2 A_2 A_1 \mid a A_1 \mid b$

que tem um problema de recursão à esquerda

Para facilitar a explicação, vamos expandir as regras:

$A_1 \rightarrow A_2 A_2$

$A_1 \rightarrow a$

$A_2 \rightarrow A_2 A_2 A_1$

$A_2 \rightarrow a A_1$

$A_2 \rightarrow b$

onde vemos que o problema é só na terceira regra

Algoritmo

Cuja solução é (i) criar a regra: $B_2 \rightarrow A_2A_1$; (ii) criar a regra: $B_2 \rightarrow A_2A_1B_2$; e (iii) eliminar a regra original: $A_2 \rightarrow A_2A_2A_1$

A nova gramática ficará da seguinte forma:

$A_1 \rightarrow A_2A_2$

$A_1 \rightarrow a$

$B_2 \rightarrow A_2A_1$

$B_2 \rightarrow A_2A_1B_2$

$A_2 \rightarrow aA_1$

$A_2 \rightarrow b$

que agora contém **seis regras** e não tem mais problemas de recursão à esquerda

Algoritmo

Caso especial: a regra que foi trocada tinha A_2, certo?

Tem alguma outra regra de A_2 mas que não tinha recursão à esquerda?

Sim, as regras $A_2 \rightarrow aA_1$ e $A_2 \rightarrow b$

O que tem que ser feito é incluir as duas novas regras:

$A_2 \rightarrow aA_1B_2$ e $A_2 \rightarrow bB_2$

A nova gramática ficará:

$A_1 \rightarrow A_2A_2 \mid a$

$B_2 \rightarrow A_2A_1 \mid A_2A_1B_2$

$A_2 \rightarrow aA_1 \mid aA_1B_2 \mid b \mid bB_2$

agora com **oito regras**

Algoritmo

(5) um terminal no início do lado direito de cada produção

Seja uma produção do tipo $A_1 \rightarrow A_2\alpha$. A correção é substituir A_2 por suas produções do tipo: $A_2 \rightarrow aA_1 \mid aA_1B_2 \mid b \mid bB_2$ em que todas começam por um símbolo terminal

Estendendo novamente a gramática anterior:

$A_1 \rightarrow A_2A_2$

$A_1 \rightarrow a$

$B_2 \rightarrow A_2A_1$

$B_2 \rightarrow A_2A_1B_2$

$A_2 \rightarrow aA_1$

$A_2 \rightarrow aA_1B_2$

$A_2 \rightarrow b$

$A_2 \rightarrow bB_2$

Algoritmo

A regra $A_1 \rightarrow A_2A_2$ precisa ser corrigida.

A primeira etapa é eliminar a regra. A segunda etapa é gerar quatro novas regras produções, porque A_2 têm exatamente quatro regras que começam por símbolos terminais. O resultado será:

$A_1 \rightarrow aA_1A_2$

$A_1 \rightarrow aA_1B_2A_2$

$A_1 \rightarrow bA_2$

$A_1 \rightarrow bB_2A_2$

Algoritmo

A regra $B_2 \rightarrow A_2A_1$precisa ser corrigida.

A primeira etapa é eliminar a regra. A segunda etapa é gerar quatro novas regras produções, porque A_2 têm exatamente quatro regras que começam por símbolos terminais. O resultado será:

$B_2 \rightarrow aA_1A_1$

$B_2 \rightarrow aA_1B_2A_1$

$B_2 \rightarrow bA_1$

$B_2 \rightarrow bB_2A_1$

Algoritmo

A regra $B_2 \rightarrow A_2A_1B_2$ precisa ser corrigida.

A primeira etapa é eliminar a regra. A segunda etapa é gerar quatro novas regras produções, porque A_2 têm exatamente quatro regras que começam por símbolos terminais. O resultado será:

$B_2 \rightarrow aA_1A_1B_2$

$B_2 \rightarrow aA_1B_2A_1\,B_2$

$B_2 \rightarrow bA_1B_2$

$B_2 \rightarrow bB_2A_1B_2$

Algoritmo

O resultado ficará:

$A_1 \rightarrow aA_1A_2 | aA_1B_2A_2 | bA_2 | bB_2A_2 | a$

$B_2 \rightarrow aA_1A_1 \mid aA_1B_2A_1 \mid bA_1 \mid bB_2A_1 \mid aA_1A_1B_2$
$\mid aA_1B_2A_1B_2 \mid bA_1B_2 \mid bB_2A_1B_2$

$A_2 \rightarrow aA_1 \mid aA_1B_2 \mid b \mid bB_2$

Agora contendo 17 regras

Algoritmo

(6) **transformação das produções na forma $A \rightarrow a\alpha$, onde α só contém variáveis**

Assuma que exista uma regra $A_1 \rightarrow ab$, o que poderia ser feito? Simples, você alteraria a regra de A_1 para $A_1 \rightarrow aB_1$ e incluiria a regra $B_1 \rightarrow b$

Analisando a gramática, todas as regras já estão no formato exigido pela etapa (6)