Análise Sintática

Escrevendo uma Gramática

Escrevendo uma Gramática

Seja a Gramática

$$E \rightarrow E\mathbf{+}E \mid E\mathbf{*}E \mid \mathbf{(}E\mathbf{)} \mid \mathbf{id} \mid \mathbf{num}$$

Uma derivação posível para $E \rightarrow^{*} id*id+id$ é:

$E \rightarrow \underline{E}\mathbf{+}E \rightarrow \underline{E}*E+E \rightarrow id*\underline{E}+E \rightarrow id*id+\underline{E} \rightarrow id*id+id$

Outra derivação possível é

$E \rightarrow \underline{E}\mathbf{*}E \rightarrow id*\underline{E} \rightarrow id*\underline{E}+E \rightarrow id*id+\underline{E} \rightarrow id*id+id$

Ambiguidade

Uma gramática que produz mais do que uma árvore de derivação para a mesma cadeia é dita ambígua.

Eliminando a ambiguidade

Algumas vezes uma gramática ambígua pode ser reescriva para eliminar a ambiguidade.

Observe a gramática

stmt → **if expr then** *stmt* |
if expr then *stmt* **else** *stmt* |
other

Pode gerar a cadeia

stmt → ***if expr then if expr then*** **other else other**

Eliminando a Ambiguidade

Pode ser gerada assim :

stmt → **if expr then** *stmt* →
if expr then *if expr then* *stmt* **else** *stmt* →
if expr then if expr then **other else** *stmt* →
if expr then if expr then **other else other**

Ou assim

stmt → ***if expr then*** *stmt* **else** *stmt* →
if expr then if expr then *stmt* **else** *stmt* →
if expr then if expr then **other else** *stmt* →
if expr then if expr then **other else other**

Eliminando a ambiguidade

Mas se reescrevermos a gramática como abaixo não há mais como existir duas derivações para a cadeia do exemplo.

stmt → *matched_stmt* |
unmatched_stmt

matched_stmt → **if expr then** *matched_stmt* **else** *matched_stmt* |
other

unmatched_stmt → **if expr then** *matched_stmt* **else** *unmatched_stmt* |
other

Eliminando Recursão a Esquerda

Uma gramática é dita Recursiva à Esquerda se ela tem um não-terminal A tal que existe uma derivação $A \rightarrow^{*} A\alpha$ para qualquer cadeia α .

Métodos descendentes de análise sintática não podem manipilar gramáticas recursivas à esquerda. Assim uma transformação que elimine a recursão à esquerda é necessária.

Eliminando a Recursão à Esquerda

Existem duas formas de recursão à esquerda:

- DIRETA
  - $E \rightarrow E + T$
- INDIRETA
  - $S \rightarrow Aa \mid b$
  - $A \rightarrow Sc \mid d$

Eliminando a Recursão Direta

Passo 1: Para cada não-terminal A recursivo crie um novo não-terminal A'

Passo 2: O não-terminal A passa a produzir somente as produções não-recursivas concatenadas com A' no final.

Passo 3: O não-terminal A' passa a produzir as produções recursivas de A sem o primeiro símbolo e concatenadas com A' no final

Passo 4: O não-terminal A' deve produzir ε .

Eliminando a Recursão Direta

A gramática recursiva :

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

Eliminado a recursão fica

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

Eliminando a Recursão Indireta

Passo 1: Arrange os não-terminais em ordem $\{A_1, A_2, \ldots, A_n\}$

Passo 2:

Para i = 1 até n faça

Para j = 1 até i-1 faça

Substitua cada produção $A_i \rightarrow A_j\gamma$

por $A_i \rightarrow \delta_1\gamma \mid \delta_2\gamma \mid \ldots \mid \delta_k\gamma$

onde $A_i \rightarrow \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$

são todas as atuais A_j-produções

fim_para

Elimine as recursões imediatas entre as A_i-produções

Fim_para

Fatoração à Esquerda

Fatoração à esquerda é uma transformação que é usada para produzir uma gramática apta para o analisador sintático preditivo(ou de predicados).

A ideia básica é não ter dúvidas sobre qual produção de um não-terminal A deve ser usada. Ou seja, parsers preditivo têm problema quando um não-terminal tem duas ou mais produções com prefixo idêntico.

Fatoração à Esquerda

Exemplo: Na gramática abaixo as duas primeiras produções de *stmt* compartilham um prefixo.

stmt → **if expr then** *stmt* |
if expr then *stmt* **else** *stmt* |
other

Fatoração à Esquerda

A transformação consiste em:

Passo 1: Para todo não-terminal A com produções compartilhando um prefixo faça:

- Encontre o maior prefixo comum α entre duas ou mais produções;
- Substitua todas as produções

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_n \mid \gamma$$

- por

$$A \rightarrow \alpha A' \mid \gamma$$

$$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

Passo 2: Aplique o passo 1 até que não haja mais duas produções de A compartilhando um prefixo comum

Fatoração à Esquerda

stmt → **if expr then** *stmt stmt'* |
other

stmt' → ε |
else *stmt*