

Um compilador simples de uma passagem

Um tradutor para expressões simples

Prof. Edson Alves

Faculdade UnB Gama

Tradutor dirigido pela sintaxe

- ▶ Quando não há associação de atributos aos não-terminais, um tradutor dirigido pela sintaxe pode ser construído a partir da extensão de um analisador gramatical preditivo
- ▶ Para isso, inicialmente construa o analisador gramatical preditivo
- ▶ Em seguida, copie as ações sintáticas do tradutor nas posições adequadas no analisador gramatical preditivo
- ▶ Se a gramática tiver uma ou mais produções recursivas à esquerda, é preciso modificar a gramática para eliminar esta recursão antes de proceder com a construção do analisador gramatical preditivo

Transformação de produções recursivas à esquerda

Transformação de produção recursiva à esquerda

Seja $A \rightarrow A\alpha \mid A\beta \mid \gamma$ uma produção recursiva à esquerda. Esta produção equivale às produções recursivas à direita

$$\begin{aligned} A &\rightarrow \gamma R \\ R &\rightarrow \alpha R \mid \beta R \mid \epsilon \end{aligned}$$

onde α e β é uma cadeia de terminais e não-terminais que não começam com A e nem terminam com R .

Exemplo de transformação de produção recursiva à esquerda

$expr \rightarrow expr + digito$

$expr \rightarrow expr - digito$

$expr \rightarrow digito$



$A = expr$

$\alpha = + digito$

$\beta = - digito$

$\gamma = digito$

$R = resto$



$expr \rightarrow digito resto$

$resto \rightarrow + digito resto$

$resto \rightarrow - digito resto$

$resto \rightarrow \epsilon$

Esquema de tradução da gramática para expressões para a notação posfixa

$expr \rightarrow digito\ resto$

$resto \rightarrow +\ digito\ \{imprimir('+\')$

$resto \rightarrow -\ digito\ \{imprimir('-\')$

$resto \rightarrow \epsilon$

$digito \rightarrow 0\ \{imprimir('0\')$

$digito \rightarrow 1\ \{imprimir('1\')$

\dots

$digito \rightarrow 9\ \{imprimir('9\')$

Rotina de extração do próximo token em C++

```
1#include <iostream>
2
3using token = char;
4
5token proximo_token()
6{
7    auto t = std::cin.get();
8
9    return (token) t;
10 }
```

Rotina de tratamento de erro e declaração de *lookahead* em C++

```
12 token lookahead;  
13  
14 void erro()  
15 {  
16     std::cerr << "\nErro de sintaxe! lookahead = " << lookahead << '\n';  
17     exit(-1);  
18 }
```

Rotina de reconhecimento de tokens em C++

```
20 void reconhecer(token t)
21 {
22     if (lookahead == t)
23         lookahead = proximo_token();
24     else
25         erro();
26 }
```


Rotina associada ao não-terminal *expr* em C++

```
52 void expr()  
53 {  
54     digito();  
55     resto();  
56 }
```

Rotina associada ao não-terminal *digito* em C++

```
28 void digito()  
29 {  
30     if (isdigit(lookahead))  
31     {  
32         std::cout.put(lookahead);  
33         reconhecer(lookahead);  
34     } else  
35     {  
36         erro();  
37     }  
38 }
```

Rotina associada ao não-terminal *resto* em C++

```
40 void resto()
41 {
42     if (lookahead == '+' or lookahead == '-')
43     {
44         auto c = lookahead;
45         reconhecer(c);
46         digito();
47         std::cout.put(c);
48         resto();
49     }
50 }
```

Rotina principal do tradutor em C++

```
58 int main()
59 {
60     lookahead = proximo_token();
61
62     expr();
63
64     std::cout.put('\n');
65
66     return 0;
67 }
```

Referências

1. **AHO**, Alfred V, **SETHI**, Ravi, **ULLMAN**, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas*, LTC Editora, 1995.
2. GNU.org. [GNU Bison](#), acesso em 23/05/2022.
3. Wikipédia. [Flex \(lexical analyser generator\)](#), acesso em 23/05/2022.