

语法分析

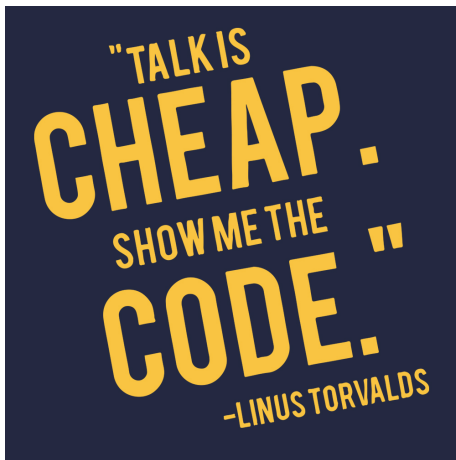
(5. ANTLR 4 语法分析器)

魏恒峰

hfwei@nju.edu.cn

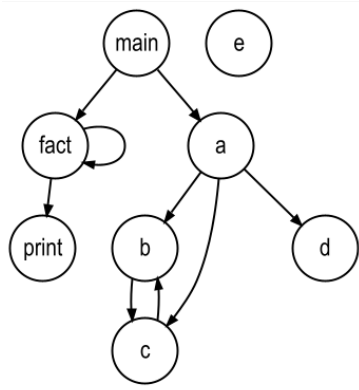
2023 年 03 月 24 日





Cymbol.g4

```
1  int factorial(int n) {  
2      if (n == 1)  
3      then return 1;  
4  
5      return n * factorial(n - 1);  
6  }  
7  
8  int main() {  
9      factorial(5);  
10 }
```



函数调用图 (Function Call Graph)

IfStat.g4



二义性文法

```
stat : 'if' expr 'then' stat  
      | 'if' expr 'then' stat 'else' stat  
      | expr  
      ;
```

IfStat.g4

```
stat : 'if' expr 'then' stat  
      | 'if' expr 'then' stat 'else' stat  
      | expr  
      ;
```

```
stat : matched_stat | open_stat ;
```

```
matched_stat : 'if' expr 'then' matched_stat 'else' matched_stat  
              | expr  
              ;
```

```
open_stat: 'if' expr 'then' stat  
          | 'if' expr 'then' matched_stat 'else' open_stat  
          ;
```

IfStatOpenMatched.g4

Expr.g4

```
expr :  
    | expr '*' expr  
    | expr '-' expr  
    | DIGIT  
    ;
```

运算符的**结合性**带来的**二义性**

ExprAssoc.g4

```
expr: '!' expr  
    | <assoc = right> expr '^' expr  
    | DIGIT  
    ;
```

右结合运算符、前缀运算符与后缀运算符的结合性

Expr.g4

```
expr :  
    | expr '*' expr  
    | expr '-' expr  
    | DIGIT  
    ;
```

运算符的**优先级**带来的**二义性**

ExprLR.g4

```
expr : expr '-' term  
      | term  
      ;
```

```
term : term '*' factor  
      | factor  
      ;
```

```
factor : DIGIT ;
```

左递归 (左结合)

```
expr :  
      | expr '*' expr  
      | expr '-' expr  
      | DIGIT  
      ;
```

ANTLR 4 可以处理该文法

ExprRR.g4

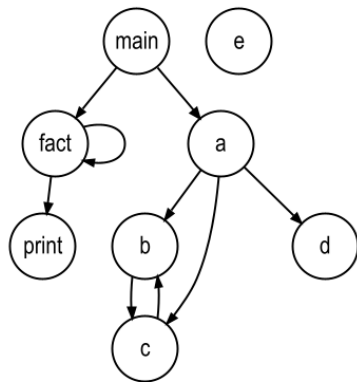
```
expr :  
    | expr '*' expr  
    | expr '-' expr  
    | DIGIT  
    ;
```

```
expr : term expr_prime ;  
expr_prime : '-' term expr_prime  
            |  
            ;  
  
term : factor term_prime ;  
term_prime : '*' factor term_prime  
            |  
            ;  
  
factor : DIGIT ;
```

ANTLR 4 可以处理该语法

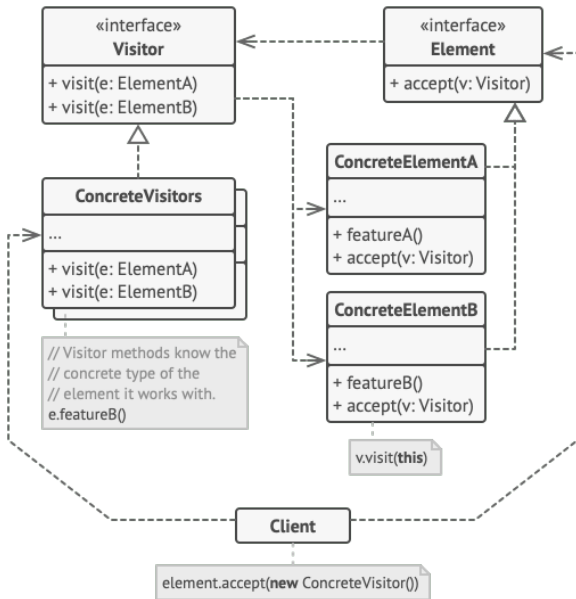
右递归 (右结合)

Call Graphs

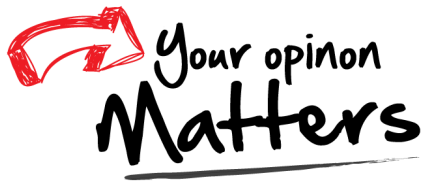


ParseTreeWalker 与 Listener

```
23 public void walk(ParseTreeListener listener, ParseTree t) {
24     if ( t instanceof ErrorNode) {
25         listener.visitErrorNode((ErrorNode)t);
26         return;
27     }
28     else if ( t instanceof TerminalNode) {
29         listener.visitTerminal((TerminalNode)t);
30         return;
31     }
32     RuleNode r = (RuleNode)t;
33     enterRule(listener, r);
34     int n = r.getChildCount();
35     for (int i = 0; i<n; i++) {
36         walk(listener, r.getChild(i));
37     }
38     exitRule(listener, r);
39 }
```



Thank
You!



Office 926

hfwei@nju.edu.cn