LAB3 ROSE

ex4

21307099 李英骏

目录

1 递归下降的翻译模式

```
<module> ::= 'MODULE' ID ';' <declarations> <module_key> 'END' '.' ;

<module_key> ::= 'BEGIN' <statement> ;

<declarations> ::= <const_declarations> <type_declarations> <
    var_declarations> <procedure_declarations> ;

<const_declarations> ::= | 'CONST' ID '=' <expression> ';' <
    const_declarations> ;

<type_declarations> ::= | 'TYPE' ID '=' <type> ';' <type_declarations> ;

<var_declarations> ::= | 'VAR' <identifier_list> ':' <type> ';' <
    var_declarations> ;

<procedure_declarations> ::= | 'PROCEDURE' ID <formal_parameters> ';' <
    proc_body> <procedure_declarations> ;

<formal_parameters> ::= '(' <fp_section> ')' ;

<fp_section> ::= | 'VAR' <identifier_list> ':' <type> ';' <fp_section> ;

<proc_body> ::= <declarations> 'BEGIN' <statement> 'END' ID ';' ;

<type> ::= 'INTEGER' | 'BOOLEAN' | <array_type> | <record_type> | ID ;

<array_type> ::= 'ARRAY' <expression> 'OF' <type> ;

<record_type> ::= 'RECORD' <field_list> 'END' ;
<field_list> ::= <identifier_list> ':' <type> ';' <field_list> ;

<statement> ::= <if_statement> | <while_statement> | <assignment> | <
    procedure_call> | <compound_statement> ;

<if_statement> ::= 'IF' <expression> 'THEN' <statement> <else_clause> 'END'
     ;
<else_clause> ::= | 'ELSE' <statement> | 'ELSIF' <expression> 'THEN' <
    statement> <else_clause> ;

<while_statement> ::= 'WHILE' <expression> 'DO' <statement> 'END' ;

<assignment> ::= ID <selector> ':=' <expression> ';' ;
\end{grammar}
```

```
37 <procedure_call> ::= ID '(' <actual_parameters> ')' ';' ;
38
39 <compound_statement> ::= 'BEGIN' <statement> 'END' ;
40
41 <expression> ::= <simple_expression> | <simple_expression> <rel_op> <
       simple_expression> ;
42
43 <simple_expression> ::= <term> | <term> <add_op> <simple_expression> ;
44
45 <term> ::= <factor> | <factor> <mul_op> <term> ;
46
47 <factor> ::= NUMBER | '(' <expression> ')' | <unary_op> <factor> | ID <
       selector> ;
48
49 <selector> ::= | '.' ID <selector> | '[' <expression> ']' <selector> ;
50
51 <rel_op> ::= '=' | '<>' | '<' | '<=' | '>' | '>=' ;
52 <add_op> ::= '+' | '-' | 'OR' ;
53 <mul_op> ::= '*' | 'DIV' | 'MOD' | 'AND' ;
54 <unary_op> ::= '+' | '-' | 'NOT' ;
55
56 <identifier_list> ::= ID | ID ',' <identifier_list> ;
57
58 <actual_parameters> ::= <expression> | <expression> ',' <actual_parameters>
       ;
```

2 自顶向下 vs. 自底向上

由于时间有限, 第四问的代码没有写完, 以下分析可能有偏差.

2.1 分析技术的简单性

递归下降预测分析:

- 实现相对简单, 特别是 LL(1) 文法. 每个非终结符对应一个递归函数按照文法规则编写. 而且可以自动生成, 基本不用写.
- 调试较为容易, 因为分析过程与文法规则直接对应, 可以逐步跟踪函数调用.

自底向上的 LR 分析:

- 实现非常的复杂, 而且调试困难!!

2.2 分析技术的通用性

递归下降预测分析:

- 适用于 LL(k) 文法（特别是 LL(1)）, 不适用于具有左递归和二义性的文法. 需要手动消除左递归和左因子化.
- 能处理的语言范围有限, 主要适用于上下文无关文法.

自底向上的 LR 分析:

- 适用于 LR(k) 文法（如 SLR、LALR、LR(1)）, 可以处理更广泛的上下文无关文法, 包含左递归和更复杂的文法结构, 能处理的语言范围更广.

2.3 语义动作的表达

递归下降预测分析:

- 在递归函数中插入语义动作代码较方便.
- 容易与语法制导翻译结合, 直接在对应的递归函数中进行处理.

自底向上的 LR 分析:

- **语义动作**: 通过状态和动作表实现语义动作, 通常在规约时执行动作. 语义动作的插入位置较为明确.
- **语法制导翻译**: 能够实现复杂的语法制导翻译, 但需要在构造分析表时注意语义动作的顺序和位置.

2.4 出错恢复的实现

递归下降预测分析:

- **出错恢复**: 出错恢复较为复杂, 需要在每个递归函数中手动添加错误处理代码. 通常使用回退或尝试匹配.
- **难度**: 难度较大, 需要仔细设计和实现.

自底向上的 LR 分析:

- 可能出现移进-规约错误和规约-规约错误, 要进行处理.
- **出错恢复**: 通过分析表和状态进行错误处理较为系统化. 常用的策略包括填补错误和跳过错误符号, 直到找到可以继续解析的状态.
- **难度**: 虽然实现复杂, 但更系统化和自动化, 错误处理相对明确.

2.5 分析表大小

递归下降预测分析:

- **分析表**: 通常不使用分析表, 而是直接通过递归函数实现. 表格驱动的 LL 分析器表格较小.
- **优劣**: 不涉及大的分析表, 代码结构直观.

自底向上的 LR 分析:

- **分析表**: 使用状态和动作表.LR 分析器的分析表较大, 特别是 LR(1) 分析器.
- **优劣**: LALR 分析器通过合并状态减少表的大小, 但表格仍然较大.

2.6 分析速度

递归下降预测分析:

- 较快, 直接通过递归函数解析.
- **性能**: 适用于简单文法, 性能较好.

自底向上的 LR 分析:

- 状态转换和规约过程相对较慢, 但对于复杂文法具有更高的解析能力.
- **性能**: 适用于复杂文法, 能够高效处理各种文法结构, 但状态转换的开销较大.

3 代码说明

期末时间有限, 代码没有写完. 仅做了如下工作:

我先参考了 GitHub-WhyUseName-ROSE. 这份代码在我的测例上有 bug:

```
C:\Users\liyj3\Desktop\WhyUseName\ROSE-master\ex4\bin>java -classpath .;..\lib\callgraph.jar;..\lib\flowchart.jar;..\lib\jgraph.jar Main ..\src\testcases\arithmetic.obr
..\src\testcases\arithmetic.obr:
The module is calculate
38 line 12 column
Syntactic Exception: Missing Operand Exception.
```

图 1: Exceptions

以及在某些测例上 (Arithmetic.005) 无法检测出括号缺失错误. 本人重写了 Parser 部分, 使其能够通过所有测例并正确抛出异常. 鉴于代码诚信不上交. 代码已提交到GitHub-KFCFKXQS-ex4_fixed

原先代码的主要 BUG 有以下两个:

1. 这里没获取下一个属性点的 token

```
845         if (m_sym.m_sym == Node.PERIOD) {
846             int match = 0;
847 
848             m_sym = nextNode();
849             for (int i = 0; i < t.m_recordElement.size(); i++) {
850                 Type t1 = t.m_recordElement.elementAt(i);
851                 if (t1.m_name.equals(m_sym.m_name)) {
852                     match = 1;
853                     sy.m_name = sy.m_name + ("." + m_sym.m_name);
854                     selector(t1, sy, level + 1);
855                 }
856             }
857             if (match == 0)
858                 throw new SemanticException("varible ( "+ m_sym.m_name + '
```

图 2: BUG 1

2. 这里读取右括号, 原代码写漏了, 没有再获取下一个 token.

```
814         else if (sym == Node.LPAREN) {
815             sy = expression();
816 
817             if (m_sym.m_sym != Node.RPAREN)
818                 throw new MissingRightParenthesisException();
819 
820             if(neg == 0)
821                 sy.m_name = "( " + sy.m_name + " )";
822             else sy.m_name = "-( " + sy.m_name + " )";
823             m_sym = nextNode();
824         }
```

图 3: BUG 2

其他地方也有一些漏了 throw Exception 和递归写错的问题

修改后运行结果如下:

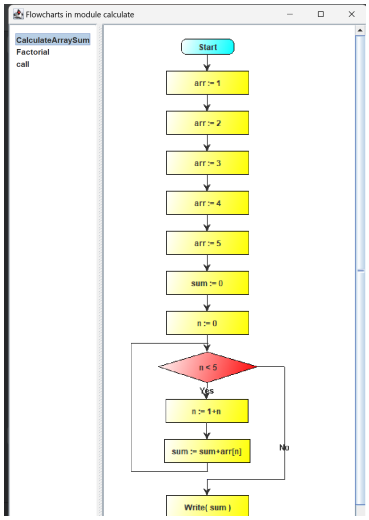

图 4: Arithmetic.obr 1

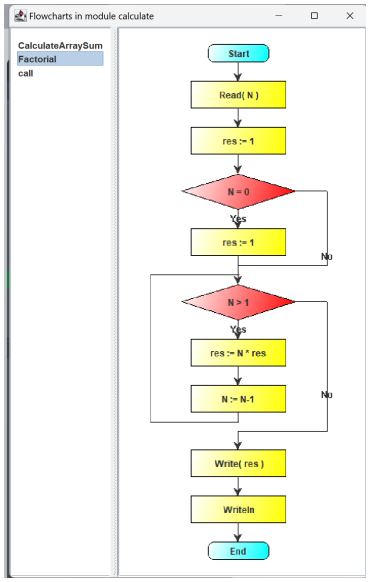

图 5: Arithmetic.obr 2

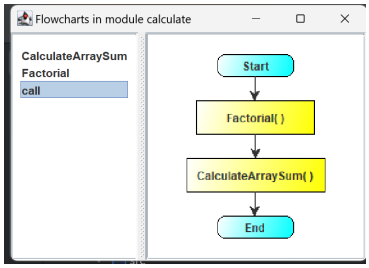

图 6: Arithmetic.obr 3

3 代码说明

错误测例:

```
../testcases/arithmetic.001:
Parsing calculate ...
Error position : Line 28  Column 13 LexicalException :
Illegal Octal number.
08

../testcases/arithmetic.002:
Parsing calculate ...
Error position : Line 28  Column 13 LexicalException :
Illegal IntegerRange: more than 12.
1926081719260817

../testcases/arithmetic.003:
Parsing calculate ...
Error position : Line 24  Column 3 LexicalException :
Illegal Identifier Length: more than 24.
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn

../testcases/arithmetic.004:
Parsing calculate ...
Error position : Line 3  Column 18 LexicalException :
Illegal Symbol.
/

../testcases/arithmetic.005:
Parsing calculate ...
Error position : Line 14  Column 14 Syntactic Exception: Missing Operator Exception.

../testcases/arithmetic.006:
Parsing calculate ...
Error position : Line 14  Column 12 Syntactic Exception: Missing Operand Exception.

../testcases/arithmetic.007:
Parsing calculate ...
Error position : Line 9  Column 9 Syntactic Exception: Missing Right Parenthesis Exception.

../testcases/arithmetic.008:
Parsing calculate ...
Error position : Line 9  Column 8 Syntactic Exception: Missing LeftParenthesis Exception.

../testcases/arithmetic.009:
Parsing calculate ...
Error position : Line 10  Column 8 Semantic Exception: varible ( M ) hasn't been declared!

../testcases/arithmetic.010:
Parsing calculate ...
Error position : Line 11  Column 11 Semantic Exception: Type Mismatched Exception.
```

图 7: Exceptions

3 代码说明