编译原理Decaf PA1-A 实验报告

计62 李祥凡 2016011262

一. 实验总述

本次实验的任务是编码实现Decaf 语言编译器的词法分析和语法分析部分， 同时生成抽象语法树。 decaf 基本框架已经给出，其中已经完成了《decaf 语言规范》中描述的语法的分析以及错误处理，我的任务是要完成对新语言特性的词法语法分析，以及相关的语法树构建。

实现方法是将在lexer.l中添加必要的识别关键字和操作符的规则，在parser.y中添加语义分析所需的、新语言特性对应的正则表达式，在SemValue.java中添加新成员变量 (如果需要的话)，在Tree.java 中添加新语言特性节点 (以Tree为基类)。

二.具体实现

语法分析的规则基本上是按照 README.pdf 中的 “参考语法” 来写的。

逐个分析新添加的语言特性:

1. 支持对象复制语句。 新增对象浅复制语句，形如 scopy(id, E)。

1) 在Lexer.l中添加识别关键字 “scopy” 的规则

2) 在Parser.y的%token中添上SCOPY, 并添加名为 "OCStmt" 的正则表达式，并使其成为Stmt的一个子集

3) 在Tree.java中添加Tree的子类OCStmt，OCStmt类有两个成员变量，分别是scopy函数参数1的标识符名以及参数2的表达式, 并根据result输出中的语法树打印格式定义了它的printTo函数

2. 引入关键词 sealed 修饰 class，使其不能被继承。

1) 在Lexer.l中添加识别关键字 “sealed” 的规则

2) 在Parser.y的%token中添上SEALED, 并添加名为 “SealedClause” 的正则表达式，用于识别一个可选的"sealed"关键字，同时修改正则表达式ClassDef, 使其能识别“class"关键字前的可选的"sealed”关键字:

3) 修改Tree.java中的ClassDef语法节点，添加一个String型的变量sealed, 用于记录"class"前是否有"sealed"关键字，修改printTo函数，使其能根据sealed变量是否为空选择是否打印"sealed"

3. 支持串行条件卫士语句。串行条件卫士语句的一般形式如

if { E1 : S1 ||| E2 : S2 ||| ... ||| En : Sn }

1）在Lexer.l中添加识别"|||"的规则

2) 在Parser.y中添加正则表达式"GuardedStmt"、"IfBranch" 和 "IfSubStmt", GuardedStmt代表整个条件卫士语句，IfBranch代表任何非空条件卫士语句括号内的部分去掉最后一个分支后剩余的部分，IfSubStmt代表一个分支，即"Expr : Stmt"

3) 在SemValue类中添加成员变量ilist, 这是一个存储IfSubStmt类对象的List

4)在Tree.java中添加静态整型常量GUARDSTMT 和 IFSUBSTMT，并添加新的语法节点IfSubStmt和GuardStmt,

IfSubStmt的成员变量是 ':' 两端的stmt和expr, 它的printTo函数会先打印"guard", 然后调用stmt和expr的printTo函数

GuardStmt类的成员变量lst是一个装IfSubStmt的List, 他的printTo函数会先打印"guarded", 然后依次调用lst中的IfSubStmt对象的printTo函数, 若lst为空，则打印"<empty>"

4. 支持简单的自动类型推导。

1）在Lexer.l中添加识别"var"关键字的规则

2) 在Parser.y的%token中添上VAR，并在LVaue的展开式中添上 “VAR IDENTIFIER”

3) 在Tree.java中添加静态整型常量VARIDENT，并添加LValue的子类VarIdent，代表被"var"捆绑的左值

5. 支持若干与一维数组有关的表达式或语句:

1） 数组常量，形如 [c1, c2, ..., cn]，其中 c1, c2, ..., cn为同一类型的常量， n 不小于 0。

例如：[1,2,3]，[5]，[]，[“how”,”are”,”you”]

- 在Parser.y中新增正则表达式“ConstantList” 和 “ArrayConstant”
- ConstantList代表一个或多个以 ',' 分隔的Constant
- ArrayConstant代表空的一对方括号或被一对方括号括起来的ConstantList
- 在SemValue中添加了一个装Expr的List clist1，用于存储数组常量中的各个常量
- 在Tree.java中添加静态整型常量ARRAYCONSTANT，并添加Expr的子类ArrayConstant，ArrayConstant的成员变量lst用于存储数组常量中的各个常量，ArrayConstant的printTo函数先打印“array const"，然后依次调用数组中各个Expr的printTo函数

2） 数组初始化常量表达式，形如

E %% n

- 在Lexer.l中新增“%%”操作符的识别规则
- 在Parser.y中的%token中添加ARRAY_REPEAT，在正则表达式Expr的右端展开式中添加上 Expr ARRAY_REPEAT Expr
- 在Parser.y中定义 ARRAY_REPEAT 的优先级
- 在Tree.java中添加静态整型常量ARRAYREPEAT，修改Binary类的printTo函数，在tag的case中添上ARRAYREPEAT

3） 数组拼接表达式，形如

E1 ++ E2

- 在Lexer.l中添加操作符"++"的识别规则
- 在Parser.y的%token中添加ARRAY_CONCAT，在正则表达式Expr的右端展开式中添加上 Expr ARRAY_REPEAT Expr
- 在Parser.y中定义 ARRAY_CONCAT 的优先级
- 在Tree.java中添加静态整型常量ARRAYCONCAT，修改Binary类的printTo函数，在tag的case中添上ARRAYCONCAT

4） 取子数组表达式，形如

E [E1 ： E2]

- 在Parser.y中，在正则表达式Expr的展开式中添加子数组表达式 Expr '[' Expr ':' Expr ']'
- 在Tree.java中添加静态整型常量SUBARRAY，并新增Expr的子类SubArray，成员变量array为数组名，

index1为索引1，index2为索引2

5） 数组下标动态访问表达式，形如：

E [E1] default E’

- 在Lexer.l中添加关键字“default"的识别规则
- 在Parser.y的%token中添上DEFAULT，并定义优先级（无结合性，优先级高于其他一切运算符）
- 在Expr的展开式中加上数组下标动态访问表达式：Expr '[' Expr ']' DEFAULT Expr
- 在Tree.java中添加静态整型常量INDEXEDDEFAULT，并定义Expr的子类IndexedDefault，成员变量array为数组名，index为索引，default_val为index非法时返回的默认值

  6）Python 风格的数组 comprehension 表达式，形如

  [E' for x in E if B]

  或者当 B 恒为 true 时，简写为

  [E' for x in E]

- 在Lexer.l中添加识别“in"关键字的规则
- 在Parser.y中Expr的展开式中添加Python风格的数组表达式 '[' Expr FOR IDENTIFIER IN Expr ']' 和 '[' Expr FOR IDENTIFIER IN Expr IF Expr']'
- 在Tree.java中添加静态整型常量PYARRAY，并增添Expr的子类PyArray

7) 数组迭代语句，形如 foreach (var x in E <while B>) S 或者 foreach (Type x in E <while B>) S（这里，Type 为用户指定类型）

- 在Lexer.l中新增识别“foreach"关键字的规则
- 在Parser.y的%token中添上FOREACH，定义正则表达式ForeachStmt，BoundVariable 和 VARTYPE，并在Stmt的展开式中添加ForeachStmt

  在Tree.java中添加静态整型常量FOREACHSTMT和BOUNDVARIABLE，并定义Tree的子类BoundVariable 和ForeachStmt，并在Stmt的展开式中添加ForeachStmt.

- 修改Tree.java中的TypeIdent类的printTo函数，使其在typeTag为VAR时输出“var"