编译原理 PA1-A 实验报告

赵浩宇，2016012390

October 2017

1 实验内容

使用 JFlex,byacc/j 并且结合之前的框架代码，构建抽象语法树。其中需要对 decaf 语言进行一些新增的语言支持。需要完成 5 部分，一是对复数的支持，包括识别常量，新增运算符，以及复数打印语句。二是 Case 语句的支持。三是对 Super 关键字的支持。四是对 dcopy 以及 scopy 两个语句进行支持，即支持对象的复制。五是对串行循环卫士语句进行支持。

2 实验过程

2.1 复数部分

根据实验的 readme 的提示，首先实现复数的功能。对于复数功能的实现，首先加入 imgconst。我选择在 baselexer 文件中加入一个 imgconst 类，并且模仿 intconst 的写法，并且在 lexer.l 与 parser.y 中加入相应的代码，首先对复数进行了识别。之后模仿 intconst 中对整数的处理，在虚数类中也进行处理，则可对其进行报错。下一步就是加入复数的三个新增的运算符。这三个运算符主要是模仿 expr 中的 unary operator，在 lexer.l,parser.y 与 tree.java 中加入相应的代码即可。其中在 lexer.l 中加入关键字的识别，在 lexer.y 中加入规约的规则，在 tree.java 中加入打印规则即可。需要注意的是，需要在 parser.y 中加入相应语句以解决规约冲突。之后再模仿 print 语句，加入 printcomp 语句。这个部分也是需要在 lexer.l,parser.y 与 tree.java 中加入相应的代码。

2.2 Case 语句

Case 语句稍微有一些繁琐。在 lexer.l,parser.y 与 tree.java 中加入识别关键字与 token 这些简单的操作，关键字包括 case,default。之后根据提示，在 tree.java 中加入 ACaseExpr 与 DefaultExpr 两种语句对应的类。ACaseExpr 对应的是一条非 default 语句，而 DefaultExpr 对应的是 default 语句。同时，在 parser.y 中加入相应的规约规则。写好这两个语句之后，由于 ACaseExpr 的数量不确定，模仿已经给出的代码框架中的 ExprList，写出 ACaseExprList。为了使得出错的可能尽量少，所以尽量模仿 ExprList 中的代码，这使得 ACaseExprList 中 ACaseExpr 出现的次数是大于等于 1 的。之后再在 tree.java 中新建一个 Case 类，再在 parser.y 中加入对 Case 语句进行处理的规约动作即可。

2.3 Super,dcopy,scopy

这三个关键字比较简单，做法基本上都是一样的。对于 Super 关键字，只需模仿之前的 this 关键字即可。在 lexer.l 中加入识别关键字的内容，在 parser.y 中加入规约规则，在 tree.java 中模仿 this 类，加入相应的创建以及打印规则即可。对于 dcopy 与 scopy，只需继承给出的 expr 类，并且重写一些函数，注意一下打印格式即可。并且要在 lexer.l 中加入关键字，以及在 parser.y 中加入规约规则。

2.4 do,od 语句

这个部分与之前的 Case 语句比较相似。只不过不同的是，根据语法的提示，do,od 语句有由 stmt 以及 stmtlist 等进行规约。而 Case 语句是根据 expr 进行规约。那么模仿已给代码中的 stmt 以及 stmtlist,block 的写法，即可写出 do,od 语句。和 Case 语句的区别就在于一个用 expr 进行处理，一个使用 stmt 进行处理。

3 总结与收获

至少了解了 JFlex 以及 byacc/j 的使用。并且知道了应用这些自动化工具进行抽象语法树构建的简便之处。同时，参考了之前框架的代码，也对现

阶段抽象语法树构建的过程有所了解。同时，在写代码的同时，由于会出现规约冲突的错误，会对其进行解决，并且加深对文法的理解。