Decaf PA 1-A 实验报告*

邓博文

2018 年 10 月 21 日

将修改的文件以及各文件修改思路整理如下。

1 Lexer.l

此文件是词法分析器的源程序。它的主要功能是由给定的正则表达式匹配整个输入流中特定的字符串，得到各个单词记录，供后续语法分析阶段使用。对于新的语言特性，只有出现新的单词记录时，才需要在此文件中添加相对应的正则表达式。本阶段中共添加了 9 个正则表达式，对应 6 个关键字以及 3 个操作符。因为这些正则表达式都只需匹配字面值 (literal)，所以只需将正则表达式的内容定义对应字面值即可，如关键字 scopy 对应正则表达式"scopy"。

2 SemValue.java

此文件定义了文法符号的语义信息。这个类在词法分析器和语法分析器之间起到了桥梁的作用。当词法分析器成功匹配到一个单词记录后，会调用 setSemantic 产生一个 SemValue 类型的变量，同时返回一个由 yacc 指定的整数值。这两部分分别对应单词记录的属性值和词法单元。可见，单词记录的语义信息正是保存在 SemValue 中的。这部分主要是修改了 toString 方法，对所有新添加的单词记录输出对应的字符串信息（虽然根据自己的观察，此方法并没有被调用过）。如 scopy 输出的信息为"keyword：scopy"。

*使用的 JDK 为 JDK11

3 Parser.y

此文件为语法分析器源文件。主要修改有两部分。第一部分是对于词法分析器新增的单词记录要定义对应的终结符。需注意的是，对于操作符，要按照 **Specification on New Features of Decaf** 定义优先级与结合性，以此来避免冲突的产生。第二部分修改是对新增的语言特性添加相应的生成式。生成式写法需参照实验指导书中语法规则说明一节，遵守习惯写法，并从已经实现的语言特性中寻找与类似的参考语法，模仿着写即可。大部分生成式只需要将给出的参考语法直接抄下来即可。这里着重说下自己完成实验时遇到的一个问题。

对于数组常量，自己一开始定义的生成式如下（动作部分省略）：

$$
\begin{aligned}
ArrayConstant \ : \quad & '[' \ ConstantList \ Constant \ ']' \\
| \quad & '[' \quad ']' \\
ConstantList \ : \quad & ConstantList \ Constant \ ',' \\
| \quad & /* \ empty \ */
\end{aligned}
$$

这样定义，在只添加数组常量这一特性时没有问题。但是当加入 comprehension 表达式后，会出现移进/规约冲突。这是因为，根据 comprehension 的生成式：

$$
\begin{aligned}
Expr \ : \quad & '[' \ Expr \ FOR \ IDENTIFIER \ IN \ Expr \ IF \ Expr ']' \\
| \quad & '[' \ Expr \ FOR \ IDENTIFIER \ IN \ Expr \ ']'
\end{aligned}
$$

在执行语法分析时，当已经看到了'['，并且下一个单词记录是 Constant 时（Constant 属于 Expr），则既可以由 ConstantList 的生成式进行规约（即匹配空），或者由 comprehension 的生成式进行移进。由于 yacc 默认对于移进/规约冲突选取移进动作，则 ConstantList 对应的规约动作永远不会被执行，这会使得数组常量的语法分析出错。改正后的写法如下：

$$
\begin{aligned}
ArrayConstant \ : \quad & '[' \ ConstantList \ ']' \\
| \quad & '[' \quad ']' \\
ConstantList \ : \quad & ConstantList \ ',' \ Constant \\
| \quad & Constant
\end{aligned}
$$

这种写法主要的不同在于 ConstantList 不能匹配空。这就使得当看到'[' 并且下一个单词记录是 Constant 时，不能立刻规约，而是会产生一个新的状态，通过再观察下一个单词记录来决定使用哪一个生成式（这个有些类似于消除左公因子的过程）。

4 Tree.java

此文件对应抽象语法树的各种节点。主要修改分为两类，一类是对于原有节点类修改成员变量以及构造函数，另一类是添加新的节点。当某一语法是在原有语法基础上增加了某些信息时，对应于第一种情况。比如 sealed 关键字，只需要在 ClassDef 类中添加成员变量 isSealed 来标识该类是否被封装。当某一语法和已有语法没有关联时，则应考虑定义新的节点。对于新增加的节点，其成员变量应由参考语法得出。比如数组动态下表访问表达式，形如"Expr[Expr default Expr]"，则应有三个成员变量，都是 Expr 类型，分别对应语法中的 3 个 Expr。同时，新定义的节点类必须实现两个抽象方法 printTo 和 accept，可分别参考 ASTPrintFormat.md 和 Vistor 类中其他方法来实现。最后需要注意的是，所有节点类都必须是静态的，因为节点类都是 Tree 类的内部类，而 Tree 类是抽象类，不可实例化，只有静态类可以在不实例化外部类的情况下访问。