Mx* Compiler: Mid-term Report

郭文轩

April 4, 2019

1 心路历程

现在已经完成了整个大作业的一小半。回头来看，前期走了不少弯路。

第一点是入门资料的选择。从零开始接触编译器，找到的资料主要是龙书虎书这样的大部头。翻开试读几页，算是了解了编译器开发的整体流程，但再往后的内容是深究细节的前端设计。就完成这项大作业而言，这些繁琐的事情交给工具效率更高，这些编译知识不如通过研究 ANTLR 等工具的文档学习来得具体形象，但当时也硬着头皮读了不少，最后于语义分析作罢。

后来找到了学长们留下的参考资料，包括书籍、设计思路等等，最开始拿着陈乐群学长的“big picture”pdf 从更贴近实际的角度去了解我到底要写什么，怎么写，看了个大概，又看了看学长的代码框架，自己便开始动手。然而有一些很基本的问题都没有弄清楚，比如 AST 和 parse tree 的区别、符号表的设计等等，基本是不明所以地照抄。前面稀里糊涂地写，到语义分析的时候就发现寸步难行，觉得思路和代码一样混乱不堪。于是我又停下来，冷静思考了一段时间，找了些别的资料，重头开始。这次写的感觉思路流畅清晰，很清楚自己在做什么。只用了三四天的时间就把代码写好了，再结合数据修补了几处，挺顺利地通过了 semantic test。

现在我最大的体会是：想清楚再动手，包括 g4 的语法设计。另一个心得是，ANTLR 可以研究的东西不少，虽然我没有深入探索，但从文档中可以看出它的功能很强大，如果善用应该能进一步提高效率。当然，学习成本也需要考虑。

2 遇到的问题

说一说在具体的设计中几处模糊的地方。

首先是 AST 与 ANTLR 生成的 parse tree。后者应该属于 CST(concrete syntax tree)，我要做的第一步是从这棵树自己建出一颗更灵活的 AST，方便后面的检查、优化。

然后是 “type node” 和 “type” 与 “symbol”。前者是 AST 上的一种节点，后者可以用《编程语言实现模式》一书的结构来解释.

最后一个深切的体会：没事不要用 private 给自己找麻烦。

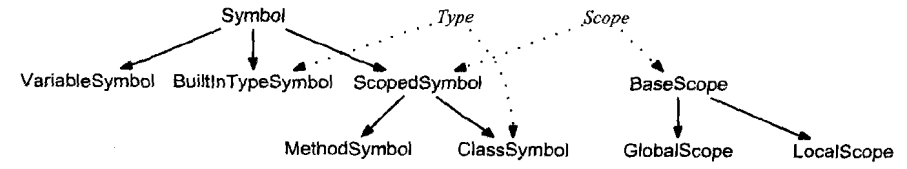

图 7.4 符号表管理相关的类继承图

3 建议

OJ 的使用说明和 git 上的 testcase 要及时更新呀。

4 推荐资料

design of Lequn Chen

Language Implementation Patterns

Engineering a Compiler

The Definitive ANTLR 4 Reference