# 编译器中期报告

张文涛 517030910425

2019 年 3 月 27 日

## 1 编译器学习心得及设计理念

### 1.1 antlr

在本次设计中直接使用了 antlr 工具最新版本 4.7.2 直接生成 Parser，虽然提前通过学习虎书学习了一些词法以及语法分析的相关知识，但是也没有底气写出一个能与 antlr 自动生成的 Parser 相媲美的成品。

antlr 通过类似正则表达式以及树的方式定义语法规则，随后生成 java 语言描述的 Parser，通过阅读官方文档，还了解了一些其他的用处。编写 g4 文件的过程比较轻松，之后又因为后续编写的要求对其稍稍进行修改。

### 1.2 抽象语法树 (AST)

通过 Node 节点以及一系列继承关系进行建构，将 antlr 生成的含有冗余信息的 CST 记录为信息简洁的 AST，并同时粗糙建立符号表以便于类型检查，这个过程由于有 IDEA 这样的强大 IDE 的支持，也较为容易。

### 1.3 符号表和类型 (SymbolTable And Type)

编译器的符号表设计参考了 *Language Implementation Patterns: Create Your Own Domain-Specific and General Programming*

采用了树状结构的符号表系统。在建立 AST 的过程中记录符号类型，并进行简单的类型是否定义的检查。在 AST 建立完成后，检查符号表的内容是否自洽，从中推出是否存在语法错误。

对于类型，设计了 MxType，支持内建类型，用户自定义类类型，以及函数类型。并且重载了 Equal 函数，以便于类型检查判断。

### 1.4 类型检查 (Type Check)

对于每个 Expression 都标记了类型，Identifier 从符号表查询得到，其他从 Identfier 推导得到，并将类型记录到对应的 Expression 节点里，便于类型检查。

## 1.5 学习体会

由于我启动这个项目的时间较早，事先学习了虎书的相关内容，并且从陈乐群学长的介绍里进行了学习，因此中期检查的内容完成得较早。通过对这一阶段的编写以及一些刁钻的测试，我更加理解了编译器解析语法的过程，以及编程语言的名字以及作用域规则，有较大的收获。

## 1.6 建议

这里希望对评分规则给出建议，希望能够通过和 gcc O0 以及 O1 的相对性能比较得出最终的成绩，而不是通过榜单的方式。这样会出现性能相差很小但是分数相差很大的情况。

还希望能够在手册中更加详细的描述语法规则，从而便于在设计时考虑周全，而不是长时间占用 OJ 评测得到结果。

希望能够提早编译作业发布的时间，让同学们能在寒假提早进行完成，减轻学期内的负担。