编译技术 Project1 报告

刘皓华 黄凯 叶文沁 王胤博

1. 分工

叶文沁：从语法分析树构建中间代码语法树。

黄凯：词法分析和语法分析。

刘皓华：代码生成。

王胤博：整合形成报告。

2. 思路与实现

整体思路

从 json 文件里提取 kernel，为之建立 DAG，然后生成符号表，建立 IR 树，生成代码。

词法分析和语法分析

在进行词法分析之前，需要提取 json 文件中的 kernel 字段作为词法分析程序的输入。由于本次 project 的 json 文件格式固定且较为简单，没有使用第三方工具 jsoncpp，而是自己编写了一个读取 json 文件的函数。该函数会读取 json 文件的各个字段，将其存储在预先定义好的结构体中，并去除 kernel 字段里多余的空格。读取完成后，将 kernel 字段存储在一个中间文件中，以该中间文件作为词法分析程序的输入。我们的词法分析程序借助词法分析工具 Lex 实现，语法分析程序借助语法分析工具 Yacc 实现。词法分析程序向语法分析程序提供 token，语法分析程序在每次需要时调用词法分析程序来获取 token。为了实现的方便，我们对 RHS 的文法进行了调整，将其改写成了 $RHS \to RHS + Term | RHS - Term$ 的形式。借助词法分析程序和语法分析程序，为输入的 kernel 建立了一个 DAG，供代码生成使用。

IR 语法树建立

在分析 json 文件和生成 DAG 后，首先遍历一遍 DAG，生成符号表（代码在 buildTable.h，其中的 indexDom 记录每个 index 的范围，varTable 记录每个数组的大小信息。通过遍历 Tref 或 Sref 节点的左子节点记录数组大小，将信息放入 varTable。遍历 Tref 节点的右子节点，计算下标范围。将每个表达式范围作为继承属性传递给子表达式，并计算子表达式的范围，最后在进入 index 节点后更新 indexDom 中该 index 的范围），然后建立 IR 树（代码在 solution.cc。基本思路：每个等式对应有一个和 LHS 相同大小的临时变量，用于临时记录计算结果。每一个等式会产生一个临时变量声明语句、一个计算临时变量的循环语句、一个将临时变量赋给 LHS 的循环语句。每个等式的 RHS 分成若干 Term 处理，每个 Term 对应一条语句。如果这个 Term 有不在 LHS 的 index，那么该语句就是一条用于求和的循环语句，否则是一条给临时变量赋值的语句）。

代码生成

1. 新的 IR 节点类：向 IR 中添加了一种语句类型，用于变量的声明，命名为 Def。Def 类的实例里只包含一个 Expr 类实例，在创造一个 Def 实例时只需要一个包含了该声明语句的变量的所有信息的 Var 类的实例。

附上 Def 的定义：

```
class Def: public StmtNode, public std::enable_shared_from_this<Def> {
 public:
    Expr var;

    Def(Expr _var) :
        StmtNode(IRNodeType::Def), var(_var) {}

    Stmt mutate_stmt(IRMutator *mutator) const;
    void visit_node(IRVisitor *visitor) const;

    static Stmt make(Expr _var) {
        return std::make_shared<const Def>(_var);
    }

    static const IRNodeType node_type_ = IRNodeType::Def;
};
```

2. IRPrinter：IRPrinter 主要是深度优先遍历 IR 树，并在必要的节点进行打印。其中较为重要的细节包括：

1) 对于每一个运算，在打印时都用括号封装起来，保证 c 程序运行时不出错。

2) 对于结构 Var,在打印参数、打印声明语句与打印普通变量时,使用三种不同的处理,用在 IRPrinter.h 里定义的两个变量控制。

3) 在 Var 的处理中，需要特别注意区分标量，用 shape 进行判断时，考虑是不是 shape 的大小为 1 且第一个的值也为 1；用 args 判断时，只要判断 args 的大小即可。

4) 在 kernel 里要注意一下没有输入参数的情况。

3. 附件说明

在 include 目录下，我们添加了 buildTable.h（作用前已说明）、json.h（用于处理 json 文件）和 node.h（定义了语法树中的节点和操作）、lex.yy.c、lex_yacc.h、y.tab.c、y.tab.h（这四个文件是根据对词法和文法的描述、利用 lex 和 yacc 创建的，用于词法和文法分析）。

在./project1 下执行如下命令编译整个项目：

```
mkdir build
cd build
cmake ..
make -j 4
```

然后运行`./test1` 即可测试所有例子。