编译技术Project1报告

设计思路，实现方法

主要分为两部分，第一部分是json到IRtree的转换，第二部分是通过自己写的IRPrinter对IRtree进行visit打印生成的代码。

第一部分

json到IRtree的转换模仿了gemm.cc的例子，需要通过解析kernel生成循环变量(index)，表达式(move)，循环(loopnest)。

想法是通过对kernel进行两边扫描来完成以上过程。第一遍扫描确定循环变量及范围，运算的张量及其shape。然后对=号左边进行分析，获得左边的式子。最后第二遍扫描完成表达式的合成，并转换为move语句。

第一遍扫描：

第一遍扫描只关注张量名，<>，[]3个内容。

使用到的变量有s2Var，s2indexVar两个map。s2Var是通过张量名来找到张量的范围（比如A<16,32>通过A可以找到{16，32}），s2indexVar是通过找循环变量名找循环变量（是一个indexVar类，里面可以访问循环变量的边界和获得循环变量的exp）。nowVar记录了当前的张量名。

1.当扫描到<时，调用Clistin，Clistin函数主要是对<16，32>这种进行解析，扫描的时候会判断<>前的张量是否加入s2Var中，如果没有就加进去。

2.当扫描到[时，调用Alistin，Alistin函数在第一遍扫描时tag=false并忽略返回值，函数内部会解析[i,j]或者[i+2,j]这种内容，会把没有出现过的循环变量加入s2indexVar，同时联合传入的Clist来更新循环变量的bound。

3.当扫描到张量名时，直接通过while循环一次性获得到nowVar中。

对=号左边的分析

要生成一个基本的exp表达式，和=号右边解析A<16,32>[i,j]的方式差不多，就是通过Alistin的返回值获得下标，这里是i和j，存到leftAlist中，通过Clistin获得{16，32}存到leftClist中，把张量名存到leftVar中。后面要用到的时候通过`Var::make(data_type, leftVar, leftAlist, leftClist)`来生产左边的张量。

第二遍扫描：

第二遍扫描是生成表达式，基本思路和普通的中缀表达式计算差不多，设计了三个栈来完成中缀表达式的计算，一个是符号栈`stack<string> op`，一个是表达式栈`stack<Expr> num`，一个是循环变量栈`stack<set<string> >loopVar`（因为在运算的时候循环变量会逐渐变多）。

1.当扫描到运算符（+，-，*等）的时候

1.1 如果是右括号，和正常的中缀表达式计算一样直接弹栈直到左括号。

1.2 如果是左括号就直接入栈。

1.3 如果是最外层的+和-和;那么根据爱因斯坦求和规范就要生成一个loopnest，此时先把栈内内容全部计算完，然后判断op.top()是否为空来决定使用Add还是Sub（因为如果式子是b-c+d，在生成b的loopnest后那个减号是需要压入op中的，因为c的loopnest生成的时候是用要用Sub的，因此生成的时候需要进行这个判断），最后在`set<string> real`中把左侧的循环变量和当前表达式的循环变量合并，通过`LoopNest::make(index, {main_stmt})`生成循环。

1.4 其他情况就是正常的中缀表达式计算，通过运算符优先级来决定是否继续弹栈，然后把当前扫描到的运算符压栈

2.当扫描到数字的时候，用while循环把数字全部扫入到tmp中，然后压入num栈，此时loopVar栈中需要压入一个空的`set<string>`（常量没有循环变量的增加）

3.遇到<的时候，调用Clistin，这里作用仅仅是跳过<16,32>，并把{16,32}压入Clist中

4.遇到[的时候，通过Alistin并传tag=true来获得返回值，这个返回值记录的是[]内解析出来的表达式，联合Clist可以生成一个新的`Var::make`压入num栈中，loopVar栈中要压入循环变量，已经在Alistin中做完了

5.剩下的只可能是张量名了，直接while扫描完把张量名存到nowVar中

第二部分

IRPrinter的改写通过调用gemm.cc来实现，通过比对gemm.cc中输出的结果，不断改写IRPrinter，使其输出符合C++格式的语句。

改写内容如下：

1.`IRPrinter::visit(Ref<const Kernel> op)`中将`<CPU>`更改为`void`，并改写参数的输出。

2.修改`IRPrinter::visit(Ref<const LoopNest> op)`中输出，使其符合C++格式

3.`IRPrinter::visit(Ref<const Index> op)`中添加`IndexType::INT`

4.`IRPrinter::visit(Ref<const Move> op)`中删除多余输出部分

5.改写所有数组的表示方式，如由`A<2,2>`改写为`A[2][2]`

6.通过判断是否为参数，添加变量类型名或不添加

结果

编译后输入`./project1/test1`，结果如下，正确。

```
(base) zhangkeingdembp:build zkm$ ./project1/test1
Random distribution ready
Example Wrong answer
Case 1 Success!
Case 2 is hidden
Case 3 is hidden
Case 4 Success!
Case 5 Success!
Case 6 Success!
Case 7 Success!
Case 8 is hidden
Case 9 is hidden
Case 10 Success!
```

分工

第一部分：吴裕铖 李宸昊 王思翰

第二部分：张可鸣

接口：王思翰