MLIR 编译框架的使用与探索

第二部分：语法分析

上海交通大学计算机系

1　内容简介

在第二部分，我们需要构建一个语法分析器，将获得的词法单元序列构建成一棵抽象语法分析树 (AST)。具体包括解析函数的声明和调用，Tensor 变量的声明以及 Tensor 的二元运算表达式等，并针对非法格式输出错误信息。

2　功能实现

在开始本次大作业前，请先在（docker 容器外的）`pony_compiler` 主目录下执行 `git pull` 拉取更新!!!

注意事项：

- 在 Parser.h 搜索"TODO"，可以看到需要实现的相关函数以及具体要求；
- 在处理非法情况时，要求编译器在终端输出尽可能详细的错误信息；
- 在实现具体功能之前，须阅读 AST.h，该文件定义了 ExprAST 类及其各种子类。

2.1　语法分析功能实现

文件地址：/pony_compiler/pony/include/pony/Parser.h

要求实现以下功能：

1). 解析变量的声明，实现成员函数 parseDeclaration()，要求:

- 语法变量必须以“var”开头，后面为变量名及 tensor shape；
- 语法分析器已支持以下两种初始化形式，以一个二维矩阵为例：
  - var a = [[1, 2, 3], [4, 5, 6]]；
  - var a<2,3> = [1, 2, 3, 4, 5, 6]；
  - 需要同学们额外支持第三种新的形式：
    var<2,3> a = [1, 2, 3, 4, 5, 6]。

2). 解析函数内的部分常用表达式，具体要求为：

- 解析标识符语句，其可以是简单的变量名，也可以用于函数调用。要求实现成员函数 parseIdentifierExpr()；
- 解析矩阵的二元运算表达式,需要考虑算术符号的优先级。要求实现成员函数 parseBinOpRHS()；
- 解析一种新的二元运算“二维矩阵乘法”：
  - 该运算的运算符为 @，优先级与矩阵点乘（*），即矩阵对应位置元素相乘相同；
    例如：var<2,3> a = b @ c ；
  - 可能涉及到的成员函数有：parseBinOpRHS() 和 getTokPrecedence() 等；
    注意：在本次作业中，我们只要求语法分析器能识别这种运算，生成对应的合法 AST 即可。如果想要查看输入程序的运行结果，则需拓展 Pony dialect 以支持此新增运算形式，这部分将在大作业第三部分作为进阶内容发布。

3　实验验证

在对语法分析器构建完毕后，可以通过运行测试用例 test_8 至 test_12 来检查语法分析器的正确性。以 test_8 为例，验证语法分析器功能的正确性，输出 AST（-emit=ast）：

```
$ cmake --build . --target pony
$ ../build/bin/pony ../test/test_8.pony -emit=ast
```

同学们可以根据输出 AST 的结构来判断语法分析器功能的正确性。如果执行结果如图3所示，表示语法分析器解析正确。

```
root@4dae1f2a64aa:/home/workspace/pony_compiler/build# ./bin/pony --emit=ast ../
test/test_8.pony
  Module:
    Function
      Proto 'main' @../test/test_8.pony:4:1
      Params: []
      Block {
        VarDecl <> @../test/test_8.pony:6:3
          Literal: <2, 3>[ <3>[ 1.000000e+00, 2.000000e+00, 3.000000e+00], <3>[
4.000000e+00, 5.000000e+00, 6.000000e+00]] @../test/test_8.pony:6:11
        VarDecl <2, 3> @../test/test_8.pony:7:3
          Literal: <6>[ 1.000000e+00, 2.000000e+00, 3.000000e+00, 4.000000e+00,
5.000000e+00, 6.000000e+00] @../test/test_8.pony:7:17
        Print [ @../test/test_8.pony:8:3
          var: a @../test/test_8.pony:8:9
        ]
        Print [ @../test/test_8.pony:9:3
          var: b @../test/test_8.pony:9:9
        ]
      } // Block
```

图 1. test_8 的正确执行结果

另外，同学们也可以执行以下指令查看输入程序的运行结果（-emit=jit）：

```
$ cmake --build . --target pony
$ ../build/bin/pony ../test/test_8.pony -emit=jit
```

以 test_9 为例，验证语法分析器能否正确解析新的变量声明形式：

```
$ cmake --build . --target pony
$ ../build/bin/pony ../test/test_9.pony -emit=ast
```

如果执行结果如图3所示，表示语法分析器正确解析了新的变量声明形式。
注：var a[2][3] = ... 为我们新增的声明及定义方式，在本次作业中对于此种形式，我们只要求语法分析器能识别这种表示，生成对应的合法 AST 即可。如果想要查看输入程序的运行结果，则需拓展 Pony dialect 以支持此新增表示，感兴趣的同学可以自己尝试。
以 test_10 为例，验证语法分析器能否正确解析新的矩阵乘法运算：

```
$ cmake --build . --target pony
$ ../build/bin/pony ../test/test_10.pony -emit=ast
```

```
[root@4dae1f2a64aa:/home/workspace/pony_compiler/build# ./bin/pony --emit=ast ../]
test/test_9.pony
  Module:
    Function
      Proto 'main' @../test/test_9.pony:3:1
      Params: []
      Block {
        VarDecl <2, 3> @../test/test_9.pony:5:3
          Literal: <6>[ 1.000000e+00, 2.000000e+00, 3.000000e+00, 4.000000e+00,
5.000000e+00, 6.000000e+00] @../test/test_9.pony:5:17
        Print [ @../test/test_9.pony:6:3
          var: b @../test/test_9.pony:6:9
        ]
      } // Block
```

图 2. test_9 的正确执行结果

如果执行结果如图3所示，表示语法分析器正确解析了新的矩阵乘法运算。

```
[root@4dae1f2a64aa:/home/workspace/pony_compiler/build# ./bin/pony --emit=ast ../]
test/test_10.pony
  Module:
    Function
      Proto 'main' @../test/test_10.pony:3:1
      Params: []
      Block {
        VarDecl <2, 3> @../test/test_10.pony:5:3
          Literal: <6>[ 1.000000e+00, 2.000000e+00, 3.000000e+00, 4.000000e+00,
5.000000e+00, 6.000000e+00] @../test/test_10.pony:5:17
        VarDecl <3, 2> @../test/test_10.pony:6:3
          Literal: <6>[ 1.000000e+00, 2.000000e+00, 3.000000e+00, 4.000000e+00,
5.000000e+00, 6.000000e+00] @../test/test_10.pony:6:17
        VarDecl <> @../test/test_10.pony:7:3
          BinOp: @ @../test/test_10.pony:7:13
            var: a @../test/test_10.pony:7:11
            var: b @../test/test_10.pony:7:15
        Print [ @../test/test_10.pony:8:3
          var: c @../test/test_10.pony:8:9
        ]
      } // Block
```

图 3. test_10 的正确执行结果