第一次作业首次提交说明

- 杨子涵
- 2020100373

完成任务：

- 为输入代码创建Call Graph（实例代码中已经给出）
- 为每个Call Graph中节点创建控制流图（Control Flow Graph）
- 遍历Call Graph的每个节点和Control Flow Graph中的每个基本块的语句
- 将CG节点信息，基本块编号，语句内容输出到终端
- 编写说明文档

实验代码及说明：

本次完成实验代码主要添加了两个函数，分别实现Call Graph 和 Control Flow Graph的深度优先搜索

```
public int CG_DFS(CGNode current_node, int level) throws WalaException {

        /*
            This function do dfs search on call graph, and should be invoked
after initialize.
            User can invoke it in doAnalysis function after the try... catch...
statements.
            Besides, this function also invokes CFG_DFS and do DFS on CFG of
each node in CG.
         */

        // ensure that current node has not been visited
        if (cgnode_set.contains(current_node)){
                return 0;
        }
        //print information of ccurrent node
        System.out.println("level" + String.format("%-3d",level) + ":" +
                String.join("", Collections.nCopies(level,"\t")) +
current_node.toString());
        // do cfg dfs for each node in cg
        Set<ISSABasicBlock> bbset = new HashSet<ISSABasicBlock>();
        ControlFlowGraph<SSAInstruction, ISSABasicBlock> cfg =
current_node.getIR().getControlFlowGraph();
        CFG_DFS(bbset, cfg.getNode(0), level+1, cfg);

        //dfs: search unvisited sons
        cgnode_set.add(current_node);
        for (CGNode cn: Iterator2Iterable.make(cg.getSuccNodes(current_node))){
            CG_DFS(cn, level+1);
        }
        cgnode_set.remove(current_node);
        return 0;
}
```

```
public int CFG_DFS(Set<ISSABasicBlock> bb_set, ISSABasicBlock basicBlock, int
level, ControlFlowGraph<SSAInstruction, ISSABasicBlock> cfg){

    /* this function is used to do dfs on cfg, and print the information basic
block and the instructions*/
        //
    if (bb_set.contains(basicBlock)){
            return 0;
    }
    // print information and instructions of the basic block
    SSAInstruction[] instructions = cfg.getInstructions();
    System.out.println("level" + String.format("%-3d",level) + ":" +
                       String.join("", Collections.nCopies(level,"\t"))
+"BB:\t"+basicBlock);
    for(int start=basicBlock.getFirstInstructionIndex(); start <=
basicBlock.getLastInstructionIndex(); start++){
        System.out.println("level" + String.format("%-3d",level+1) + ":" +
             String.join("", Collections.nCopies(level+1,"\t")) +
"Instructions:\t" + instructions[start]);
        }
        // search the unvisited sons
    bb_set.add(basicBlock);
    for (ISSABasicBlock b: Iterator2Iterable.make(cfg.getSuccNodes(basicBlock)))
{
        CFG_DFS(bb_set,b,level+1,cfg);
    }
    bb_set.remove(basicBlock);
    return 0;
}
```

我们在对CG做DFS时，对当前节点进行扩展，生成对应的CFG图，并调用第二个函数对CFG进行深度优先搜索。

代码输出

```
level0  :Node: synthetic < Primordial, Lcom/ibm/wala/FakeRootClass,
fakeRootMethod()V > Context: Everywhere
level1  :   BB: BB[SSA:0..0]0 - com.ibm.wala.FakeRootClass.fakeRootMethod()V
level2  :       Instructions:   invokestatic < Primordial,
Lcom/ibm/wala/FakeRootClass, fakeWorldClinit()V > @0 exception:2
level2  :       BB: BB[SSA:1..1]1 - com.ibm.wala.FakeRootClass.fakeRootMethod()V
level3  :           Instructions:   3 = new <Application,[Ljava/lang/String>@14
level3  :           BB: BB[SSA:2..2]2 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level4  :               Instructions:   5 = new <Primordial,Ljava/lang/String>@2
level4  :               BB: BB[SSA:3..3]3 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level5  :                   Instructions:   arraystore 3[6] = 5
level5  :                   BB: BB[SSA:4..4]4 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level6  :                       Instructions:   invokespecial < Primordial,
Ljava/lang/Object, <init>()V > 3 @4 exception:7
level6  :                       BB: BB[SSA:5..5]5 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level7  :                           Instructions:   invokestatic < Application,
Lcom/example/test/Main, main([Ljava/lang/String;)V > 3 @5 exception:8
```

```
level7  :                            BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level6  :                         BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level5  :                     BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level4  :                 BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level3  :             BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level2  :        BB: BB[SSA:-1..-2]6 -
com.ibm.wala.FakeRootClass.fakeRootMethod()V
level1  :   Node: synthetic < Primordial, Lcom/ibm/wala/FakeRootClass,
fakeWorldClinit()V > Context: Everywhere
......
```

我们截取了一部分代码的输出如上

通过缩进来表现节点和节点之间的层次关系：

- 用level表示层次深度，代码中的fakeroot（而非main）作为CG图的入口。
- 根据调用关系，被调用节点层次高于调用节点层次。
- 节点信息之下，输出CFG上的DFS结果，层次比节点信息加一。
- BB块命令比BB块层次加一，与目标BB块层次相同
- 结果展示了包含CG和CGF所有节点在内的深度优先遍历

此外，我们使用graphviz工具将CG图可视化，生成PDF保存在第一次作业目录下：