Lab3 - Interprocedural Analysis

语言特性与项目编译见 Lab1 的文档。建议阅读 PDF 版本。

Motivation & Case

过程内区间分析在涉及函数参数的时候，为保证健全性，会保守地将其假设为⊤，导致结果极其不精确。

因此我们引入过程间分析，通过过程间的控制流边来传递数据流信息，分析被调用过程内的具体变化。

本lab仅考虑函数调用（非递归），不涉及函数返回。

下面展示一段涉及过程间分析的源代码和其对应的 IR

```
function main(){
    a = 20;
    call callee(a);
}

function callee(m){
    m = m + 1;
    check_interval(m, 21, 21);
}
```

```
L0 :  function main(){
L1 :
L2 :  a = 20;                          // a = [20, 20]
L3 :  call callee(a);
L4 :
L5 :  }
L6 :
L7 :  function callee(m){
L8 :                                   // m = [20, 20]
L9 :  m = m + 1;                       // m = [21, 21]
L10:  check_interval(m, 21, 21);       //YES
L11:
L12:  }
L13:
```

目前设计所有instruction的标识唯一，新增位于function内部接口label2inst，用于根据label获取instruction。

Your Task

编写 `analysis/interAnalysis.cpp` 与 `analysis/interAnalysis.h`，你无须修改除此之外的任何文件。

本次分析采用自顶向下分析，从根节点开始分析（⚠️根节点可能会有多个），函数内部的语句分析与Lab1类似，由于引入函数调用，因此需要额外增加对`CallInst`的解释。

完成后你可以通过运行 `build/test/interAnalysisTest` 来检查正确率与召回率，使用方法同 Lab1。

也可以通过以下命令来单独分析某文件

```
fdlang -inter-analysis xxx.fdlang
```

新增测试用例 `call1.fdlang` ~ `call4.fdlang`

要求：能够正常跑通且正确率≥80

所有需要用到的接口、函数等都在 `analysis/interAnalysis.h` 与 `lib/IR/IR.h` 中，无需花时间看其他的代码。

你需要提交：

- 源代码，仅包含 `analysis/interAnalysis.cpp` 与 `analysis/interAnalysis.h`
- 简单的报告，包括方法、分析结果，可以自己构造几个测试用例

除了课上讲的，可以参考的一些资料：

- 北京大学公开课 软件分析技术 熊英飞 https://www.bilibili.com/video/BV1Rt4y1s7tC P8
- 南京大学 软件分析 李樾、谭添 https://www.bilibili.com/video/BV1oE411K79d P7