

# PA4 实验报告

2017011307 张晨

## 实验原理

如果赋值语句的被赋值者不在 `liveOut` 里，那么这条赋值语句可以删除。

具体来说，需要分以下两种情况讨论：

1. 一般赋值。包括 `Assign` / `LoadVTbl` / `LoadImm4` / `LoadStrConst` / `Unary` / `Binary` / `MemoryLoad`。直接判断即可。
2. 函数调用。因为函数调用可能有副作用，不能直接把TAC删了，因此只删对返回值的赋值。删了之后会导致 `Simulator` 崩溃，因此需要稍微改一下 `Simulator`。

此外，如果删除某条代码，也应删除其带来的异常检查代码。但是这不好在TAC层面处理，因为在TAC层并不知道某条TAC是由异常检查生成的。所以，我删掉了原有框架里的所有异常检查的内容。

## 测试结果

给的测试样例都没有刻意删除的死代码。

我构造了一个函数

```
1 static void call(class Main m) {
2     int a1 = m.call1();
3     int a2 = call2();
4     int a3 = m.x;
5     int a4 = a3;
6     int a5 = a4;
7     string a6 = "abcd";
8     bool a7 = true;
9     bool a8 = !a7;
10    int a9 = a4 + a5;
11 }
```

这里面全是死代码，得到的TAC是

```
1 FUNCTION<Main.call>:
2     _T2 = *(_T0 + 0)
3     _T3 = *(_T2 + 8)
4     parm _T0
5     call _T3
6     call FUNCTION<Main.call2>
7     return
```

为了公平，我将我的运行结果与删除异常处理的原始框架进行比较。算上运行的上下文，原始框架使用了35条tac，我的实现使用了21条tac。该测例的完整版见 `TestCases/S4/test.decaf`