基于 JoeQ 的程序优化

2017011303 林俊峰

实验内容

1. 实现 `submit.FindRedundantNullChecks` ，其功能应为分析并打印输入的所有类名所对应类的各方法的冗余 `NULL_CHECK` Quads。

使用数据流分析框架计算冗余 `NULL_CHECK` Quads。

- V定义为checked变量的集合
- 交汇运算为求交集
- 传递函数为
  $f_s(x) = (x - def_s) \cup check_s$, check集合的元素为 `NULL_CHECK` 语句检查的变量，对于其它语句此集合为空。
- 边界条件 $out[entry] = \{\}$
- 初始化 $out[b] = U$
  算法收敛后，对于每一条 `NULL_CHECK` Quad, 若它的 `IN` 集合中包含它检查的变量，则这条Quad是冗余的，将它打印。

2. 实现
   `submit.Optimize.optimize(List<String> optimizeClasses, boolean nullCheckOnly)` 。其功能应为对根据类名 `Helper.load` 得到的 `clazz` 进行优化；在 `nullChecksOnly` 为 `true` 时，仅移除冗余的 `NULL_CHECK` ，否则也进行其它优化。

基础要求比较简单，在1的基础上把打印操作修改为移除相应的Quad即可

额外优化：和 `null` 进行 `IFCMP_A` 后的以及New后的 `NULL_CHECK` 。

在预处理 `prepocess` 中遍历quad时(需要知道cfg的信息，所以在预处理中做)，在遍历到一条 `IFCMP_A` 的q0时，若是与 `null` 作比较，则对于变量x非null的分支目标q1，branchOut[q0,q1]集合加入x
$branchOut(q0, 01) = branchOut(q0, q1) \cup x$
branOut使用二维map 实现

同时由于引入了一个新的branchOut集合，需要修改数据流框架，变换一下求交汇的姿势
从 $in = \cap out$ 修改为 $in = \cap(out \cup branchOut)$

对于测例 NullTest
消除前

```
 Result of interpretation: Returned: null (null checks: 328 quad count: 1937)
```

消除后

```
 Result of interpretation: Returned: null (null checks: 326 quad count: 1935)
```

可以看到有额外的2条 NULL_CHECK 被消除