Decaf PA4

计64 翁家翌 2016011446

DU链求解

算法核心可以用如下公式表示：

$$\mathrm{LiveIn}[\mathrm{B}] = \mathrm{LiveUse}[\mathrm{B}] \cup (\mathrm{LiveOut}[\mathrm{B}] - \mathrm{Def}[\mathrm{B}])$$
$$\mathrm{LiveOut}[\mathrm{B}] = \cup_{\mathrm{b}\in \mathrm{S}[\mathrm{B}]}(\mathrm{LiveIn}[\mathrm{b}])$$

其中 S[B] 为 B 的所有后继基本块的集合。对于 DU 链的求解，这里需要多记录一个位置信息。

处理KILL和GEN

Kill 表示每一块有哪些变量定义过，Gen 表示每一个被定义变量的最后定义位置。

修改 `BasicBlock.java`，添加函数 `computeGenAndKill`，位于240行。

首先先复制一遍上面的 `analyzeLiveness` 函数，然后把操作改一下。由于只有 `def op0` 会影响这两个集合，因此根据之前的注释，只需将其统一修改即可，大致代码如下所示：

```
if (tac.op0 != null) {
    if (!kill.contains(tac.op0))
        gen.add(new Pair(tac.id, tac.op0));
    kill.add(tac.op0);
}
```

处理IN和OUT

修改 `FlowGraph.java`，添加函数 `analyzeDUChain`，位于210行。

首先复制一遍上面的 `analyzeLiveness` 函数框架。对于每个BasicBlock求出Gen和Kill集合，然后对着in与out的公式迭代，如果出现相邻两次迭代状态相同，则停止迭代。代码如下：

```
public void analyzeDUChain() {
    for (BasicBlock bb : bbs)
        bb.computeGenAndKill();
    boolean changed = true;
    do {
        changed = false;
        for (BasicBlock bb : bbs) {
            for (int i = 0; i < 2; i++)
                if (bb.next[i] >= 0)
                    bbs.get(bb.next[i]).in.addAll(bb.out);
            int last = bb.out.size();
            bb.out.clear();
            bb.out.addAll(bb.gen);
```

```
            for (Pair item : bb.in)
                if (!bb.kill.contains(item.tmp))
                    bb.out.add(item);
            if (last != bb.out.size())
                changed = true;
            for (int i = 0; i < 2; i++)
                if (bb.next[i] >= 0)
                    bbs.get(bb.next[i]).in.addAll(bb.out);
        }
    } while (changed);
}
```

处理整体 DU 链

修改 `BasicBlock.java`，添加函数 `analyzeDUChain`，位于290行。

还是复制一遍之前的框架之后，对于每一个 Block 先求出 in，作为该 Block 第一个语句的 in。从前到后扫描每一条语句，动态维护该 in 集合，同时建立整体 DU 链。由于是根据位置获得的 DU 链的信息，已经不知道定义位置在哪个 Block 里面了，因此 DU 链集合无法分配到每一块中，只能建立一个整体的 DU 链。

处理局部 DU 链

修改 `BasicBlock.java`，添加函数 `rebuildDUChain`，位于381行。

还是复制一遍框架之后，由于现在已经求得整体的 DU 链，但要将 DU 链存在每个 Block 里面，所有还需再遍历每一个块进行 DU 链的提取。大致代码如下：

```
Temp core = null;
if (tac.opc == ....) core = tac.op0;
if (core != null)
    DUChain.put(new Pair(tac.id, core), parent.DUChain.get(new Pair(tac.id, core)));
```

分析输出的 TAC 序列与 DU 链信息

`t0.du` 的输出结果共包含三个函数，如下所示：

FUNCTION _Main_New

```
FUNCTION _Main_New :
BASIC BLOCK 0 :
1   _T0 = 4 [ 2 ]
2   parm _T0
3   _T1 =  call _Alloc [ 5 6 ]
4   _T2 = VTBL <_Main> [ 5 ]
5   *(_T1 + 0) = _T2
6   END BY RETURN, result = _T1
```

该函数顺序执行，无分支结构，没有出现覆盖定值点等等现象。

FUNCTION main

```
FUNCTION main :
BASIC BLOCK 0 :
7   call _Main.f
8   END BY RETURN, void result
```

该函数无定值点，无 DU 链。

FUNCTION _Main.f

```
FUNCTION _Main.f :
BASIC BLOCK 0 :
9   _T7 = 0 [ 10 ]
10  _T5 = _T7 [ ]
11  _T8 = 1 [ 12 ]
12  _T6 = _T8 [ ]
13  _T10 = 0 [ 14 ]
14  _T9 = _T10 [ 21 24 30 ]
15  _T11 = 2 [ 16 ]
16  _T3 = _T11 [ 18 ]
17  _T12 = 1 [ 18 ]
18  _T13 = (_T3 + _T12) [ 19 ]
19  _T4 = _T13 [ 28 ]
20  END BY BRANCH, goto 1
BASIC BLOCK 1 :
21  END BY BEQZ, if _T9 =
        0 : goto 7; 1 : goto 2
BASIC BLOCK 2 :
22  _T14 = 1 [ 23 ]
23  _T3 = _T14 [ 35 ]
24  END BY BEQZ, if _T9 =
        0 : goto 4; 1 : goto 3
BASIC BLOCK 3 :
25  call _Main.f
26  END BY BRANCH, goto 4
BASIC BLOCK 4 :
27  _T15 = 1 [ 28 ]
28  _T16 = (_T4 + _T15) [ 29 ]
29  _T4 = _T16 [ 28 32 36 ]
30  END BY BEQZ, if _T9 =
        0 : goto 6; 1 : goto 5
BASIC BLOCK 5 :
31  _T17 = 4 [ 32 ]
32  _T18 = (_T4 - _T17) [ 33 ]
33  _T4 = _T18 [ 28 36 ]
34  END BY BRANCH, goto 6
BASIC BLOCK 6 :
35  _T5 = _T3 [ ]
36  _T6 = _T4 [ ]
37  END BY BRANCH, goto 1
BASIC BLOCK 7 :
38  END BY RETURN, void result
```

从该函数中可以看出 _T5 是 `a`，_T6 是 `b`，_T3 是 `i`，_T4 是 `j`。

在2.2节图4中，d_1对应(15-)16，d_2对应(17-)19，d_3对应(22-)23，d_4对应(27-)29，d_5对应(31-)33，d_6对应35，d_7对应36。

验证一致性：（`a [b]` 表示 定值点 a 的 DU 链中包含 b）

- i 在定值点 d_1 的 DU 链为 $\{d_2\}$：16 [18], 18 [19 (d_2)]
- j 在定值点 d_2 的 DU 链为 $\{d_4\}$：19 [28], 28 [29 (d_4)]
- i 在定值点 d_3 的 DU 链为 $\{d_6\}$：23 [35 (d_6)]
- j 在定值点 d_4 的 DU 链为 $\{d_4, d_5, d_7\}$：29 [28, 32, 36 (d_7)], 28 [29 (d_4)], 32 [33 (d_5)]
- j 在定值点 d_5 的 DU 链为 $\{d_4, d_7\}$：33 [28, 36 (d_7)], 28 [29 (d_4)]
- d_6, d_7 的定值点没有被引用，因此其 DU 链为空。