PA4 Report

潘庆霖 计61 2016011388

1、DU链求解的实现过程

- 算法描述

  参考了课件上给出的算法，但具体实现过程中，只是借用并借鉴了原有框架的代码，添加自己的代码。

  所依赖的数据结构：

  在BasicBlock.java中，定义了：

  ```
  public Map<Temp, Set<Integer>> useIn;
  public Map<Temp, Set<Integer>> useOut;
  public Set<Temp> useDef;
  ```

  具体描述：

  useIn中存储了当前基本块中，***未被定义但是已经被引用的变量以及引用的具体位置***。包括temp及其被引用位置的tac的id。

  useOut中存储的是：**当前基本块的所有后继引用块的useIn的并集**。

  useDef中存储了**在当前基本块中被定值过的temp**。

  基于上述数据结构，求解DU链的算法具体可以描述为：

  1. 正确**求解每一个基本块的useOut集合**。首先在单个基本块中计算出正确的useIn集合以及useDef集合。之后，进行如下的迭代：
     1. 遍历每个基本块BB的直接后继基本块BB1的useIn中的key，进行分析，
        1. 如果key存在于BB的useIn中，并且不存在与BB的useDef中，则说明，变量key在BB中被引用，但是未被定义。将BB1中的useIn中对应key的集合，并入BB中useIn的集合。
        2. 如果key不存在于BB的useIn中，且不存在于BB的useDef中，则说明，虽然变量key没有在BB中被引用/定义过，但是key在BB1中，在定义之前被引用了。将BB1的useIn中key对应的集合作为值，key作为键，加入到BB的useIn中。
     2. 如果上一步骤没有导致任何集合发生变化或者任何基本块的useIn发生变化，则跳转到下一步骤。否则重复上一步骤。
     3. 根据每一基本块的直接后继基本块的useIn，计算该基本块的useOut。
  2. 在单个基本块中，以上一步骤中得到的正确的useOut集合为基础，按照向后流的顺序进行分析。
     1. 将沿途的引用点的id加入useOut中对应的temp的集合。
     2. 遇到对某个temp的定值点时，将useOut中对应这一temp的id集合，转移到DU链中，作为该定值点的DU链。将useOut中原有的对应id集合清空。
- 具体实现
  - 在BasicBlock.java中
    - 添加上述三个数据结构，并在构造函数中进行初始化
    - 在原有框架计算liveUse和Def的函数computeDefAndLiveUse中，添加语句，用来计算当前基本块的useIn和useDef。

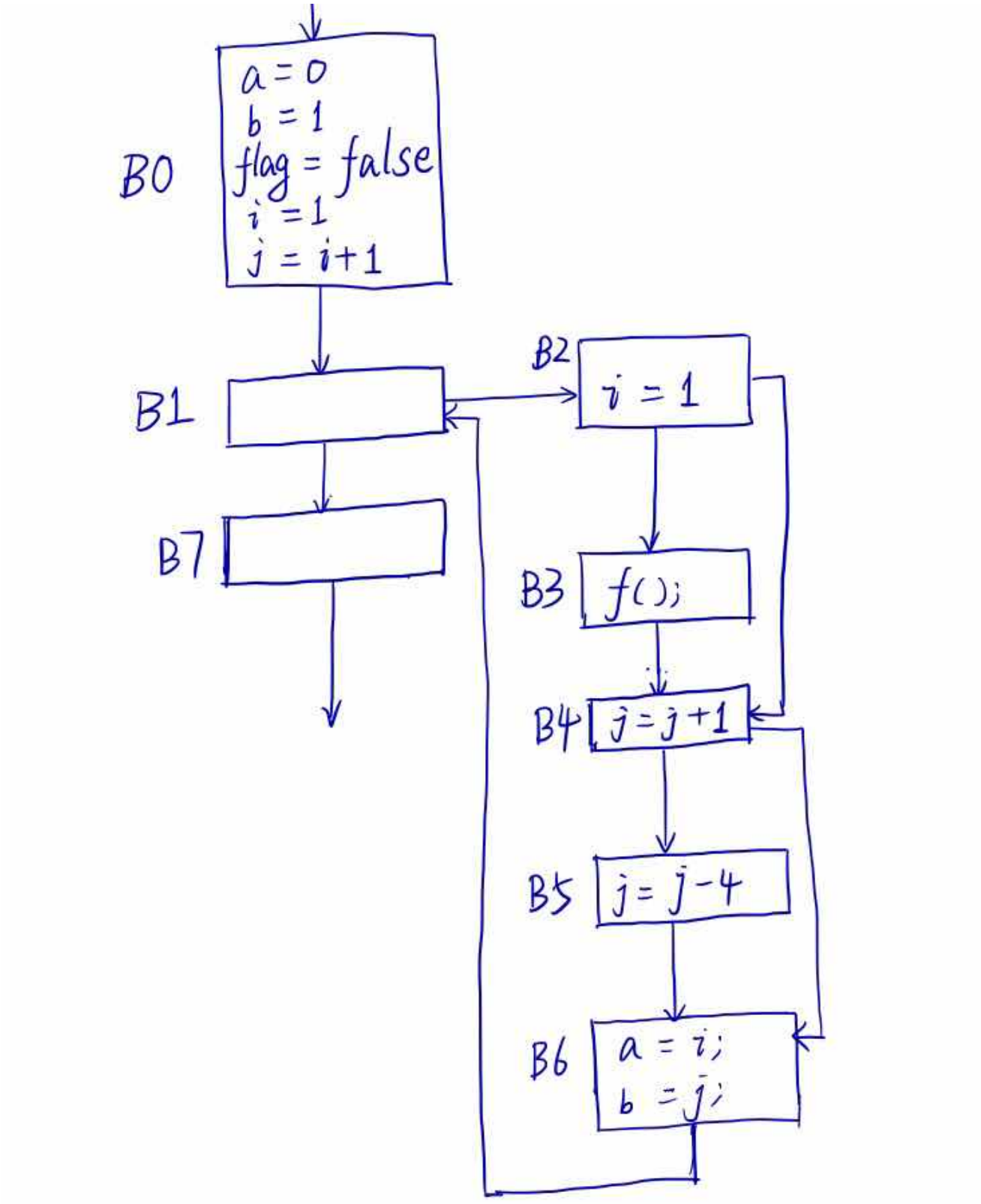
        - 实现了计算DUChain的函数analyzeDUlink。按照向后流的顺序，计算每一个定值点对应的DU链。
    - 在FlowGraph.java中
        - 实现了analyzeDUChain函数。
            - 根据基本块之间的前驱后继关系，补全各个基本块的useIn集合。
            - 根据每个基本块的直接后继基本块的useIn，取并集计算基本块的useOut。

2、举例说明

输出的结果TestCase/S4/output/t0.du 中，对应讲义12中2.2节图4部分的函数f（）。

```
FUNCTION _Main.f :
BASIC BLOCK 0 :
9   _T7 = 0 [ 10 ]
10  _T5 = _T7 [ ]   #a = 0;
11  _T8 = 1 [ 12 ]
12  _T6 = _T8 [ ]   #b = 1;
13  _T10 = 0 [ 14 ]
14  _T9 = _T10 [ 21 24 30 ]     #flag = false;
15  _T11 = 2 [ 16 ]
16  _T3 = _T11 [ 18 ]   #i = 2;
17  _T12 = 1 [ 18 ]
18  _T13 = (_T3 + _T12) [ 19 ]
19  _T4 = _T13 [ 28 ]       #j = i + 1;
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 ]     # def T14
23  _T3 = _T14 [ 35 ]   #i = 1;  use T14
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 ]     #j = j + 1;
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
```

结合注释以及流图，不难得出，下面这个流图的正确性（与2.2节中的流图稍有差别）。进而可验证上面tac码中得到的DU链信息的正确性。



可以计算得到，每个基本块对应的useIn为：

BasicBlock	useIn	useDef
BB0	{}	{T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13}
BB1	{T9:{21}}	{}
BB2	{T9:{24}}	{T3,T14}
BB3	{}	{}
BB4	{T4:{28},T9:{30}}	{T15,T16,T4}
BB5	{T4:{32}}	{T17,T18,T4}
BB6	{*T3:{35},*T4:{36}}	{T5,T6}

经过迭代补全useIn之后，各基本块的useIn以及useOut如下：

BasicBlock	useIn	useOut
BB0	{}	{T4:{28},T9:{30,24,21}}
BB1	{T4:{28},T9:{30,24,21}}	{T4:{28},T9:{30,24}}
BB2	{T4:{28},T9:{30,24}}	{T3:{35},T4:{28},T9:{30}}
BB3	{T3:{35},T4:{28},T9:{30}}	{T3:{35},T4:{28},T9:{30}}
BB4	{T3:{35},T4:{28},T9:{30}}	{*T3:{35},*T4:{32}}
BB5	{*T3:{35},*T4:{32}}	{*T3:{35},*T4:{36}}
BB6	{*T3:{35},*T4:{36}}	{}

以T9为例，T9在B0中仅有一个定值点，根据算出的useOut可以直接得到DU链为{21,24,30}。而从生成的tac码中可以得到验证，T9确实在21,24,30这几个位置有引用，且其余位置没有引用；并且只有一个定值点，求解得到的DU链是正确的。

同理，其他的DU链对应关系也符合正确性。