TD5 - Thuesday, November 9

1 Intermediate Representation

Note : This TD has been prepared by Mioara Joldes for the 2009-2010 Compilation course

Exercise 1 (Dataflow analysis 1) Reaching definitions :

A definition d reaches a point p if there exists a path from the point immediately following d to p such that d is not killed (overwritten) along that path. A definition of a variable is killed between 2 points when there is another definition of that variable along the path e.g. a = b + c kills previous definitions of a.

Problem statement : For each point in the program, determine if each definition in the program reaches the point.

Consider the following code.

```
x := 10 ;
WHILE x > 0 DO
BEGIN
   i := i * 10 ;
   IF i > 20 THEN
      x := x - 1 ;
   ELSE
      i := x + 100 ;
   x := x - 1 ;
END ;
x := x - 1 ;
```

Build the CFG (nodes= basic blocks, edges= control flow) for it.

Consider the following sets :
- `Gen[b]` : definitions generated in block b ;
- `Kill[b]` : definitions killed in block b ;
- `In[b]` : definitions reaching the entry in block b ;
- `Out[b]` : definitions reaching the exit of block b ;

Set up a set of equations between in[b] and out[b] for each basic block b. What is the effect of code on in[b] and out[b] on each basic block ? What is the effect of control flow ?

Based on these equations, give an iterative algorithm for solving the reaching definition problem. Apply this algorithm for the code given [1].

1. If you need a little help for finding the equations, here is a link : suif.stanford.edu/~courses/cs243/lectures/l2.pdf

Exercise 2 (Dataflow analysis 2)

You've seen the dataflow analysis and the relevant dataflow problem formulation in class. Here you will study a specific dataflow analysis problem : the live variable analysis problem. Consider the code shown below.

```
 1. a := 0
2. L1: b := a+1
3. c := c+b
4. a := b*2
5. if a<10 goto L1
6. return c
```

The value of a variable X is live at point P if it may be used along some directed path in the CFG that starts at P. In this case we also say that "the variable X is live at point P".

1. Using this definition, show the live range for variables a,b,c for the code shown. You should derive these ranges intuitively. Later you will be asked to solve this problem in a more formal way.
2. Let us now define the following sets associated with a basic block B.
   - `Local sets` :
     - `def[B]` = set of variables v, such that there is at least one definition (write) of v in the basic block B.
     - `use[B]` = set of variables v , such that there is at least one use (read) of v in basic block B, and the first read of v occurs before any write of v in B.
   - `Global sets` :
     - `in[B]` = set of variables live at the entry to basic block B.
     - `out[B]` = set of variables live on exit from basic block B.

   (a) Can you formulate the live variable analysis problem by defining a set of recursive data flow equations which define in[B] and out[B] using def [B] and use[B] ?

   (b) Give an iterative algorithm for solving the live variable analysis problem.

   (c) Draw the CFG of the code given.

   (d) Can you construct the def and use sets for the basic blocks in the CFG above ?

   (e) Use your algorithm to perform the live variable analysis for the CFG above.

   (f) Consider a program with N variables. What is the worst-case complexity of liveness computing for all the variables using your algorithm ?

Exercise 3 (SSA form)

Suppose we have the following code.

```
a := 200
b := a - 2;
i := 0;
L: if (a > i) goto E
b := b- 2;
i := i + 3;
c := a + b;
goto L
E: g := 2 * b
```

1. Build control-flow graph for the above code.

2. Compute the set of live-variables at the entry and exit of each basic block through each iteration of the live-variable analysis algorithm.

3. Convert the CFG into SSA form manually. Put the ϕ functions at the necessary places, and understand their meaning.

4. Establish a connection between the dominance frontiers of the variables and the place where the ϕ functions were added. Apply the Iterated Dominance Frontier Algorithm for passing the code to SSA form and apply it to the given code.

5. The ϕ functions are not machine instructions, obviously. So, before translating a program in SSA to machine code, we have to replace these functions but to keep the same semantics. Propose a solution. What problems appear ? What solutions can we find to them ?

6. Give the conversion ouf of the SSA form for this program.