CAP - Exercises: CFG and data-flow analyses (chapter 7)

Laure Gonnord

Nov. 2016

1 Liveness analysis

A variable at the left-hand side of an assignement is *killed* by the block. A variable that appears in this bloc is *generated*.

$$LV_{exit}(\ell) = \begin{cases} \emptyset & \text{if } \ell = \text{final} \\ \bigcup\{LV_{entry}(\ell') | (\ell', \ell) \in flow(G)\} \end{cases}$$

$$LV_{entry}(\ell) = \big(LV_{exit}(\ell) \backslash kill_{LV}(\ell)\big) \cup gen_{LV}(\ell)$$

EXERCISE ▶ **Live variables**
Generate the CFG for the following program :

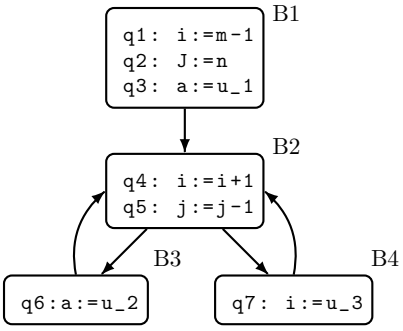
```
while d>0 do {
    a:=b+c;
    d:=d-b;
    e:=a+f;
    if e>0 then {
        f:=a-d;
        b:=d+f;
    }
    else{
        e:=a-c;
        }
    b:=a+c;
}
```

On this CFG :

— Write and solve the equations defining the sets Gen, $Kill$, In and Out for the liveness analysis, pay attention to initialisations.

— Suppress the dead code.

EXERCISE ▶ **Live Variables**
After code generation, we obtain the following graph :



On this graph, perform liveness analysis and suppress the dead code.

2 Available expressions

We recall :

$$AE_{entry}(\ell) = \begin{cases} \emptyset & \text{if } \ell = init \\ \bigcap\{AE_{exit}(\ell') | (\ell', \ell) \in flow(G)\} & \end{cases}$$

$$AE_{exit}(\ell) = \big(AE_{entry}(\ell) \backslash kill_{AE}(\ell)\big) \cup gen_{AE}(\ell)$$

EXERCISE ▶ **Common (sub) expressions**
On the following 3-address code :

```
(1)   a=b+c
(2)   b=a+c
      d=c+e
      si d>7 aller a (8)
      t=a+c
      v=c+e
      aller a (10)
(8)   t=b+c
      v=a+c
(10)  b=a+c
      a=b+c
```

1. Construct the CFG.
2. For each block, compute the sets `gen` and `kill` sets for common subexpressions.
3. Compute the set of all available expressions at the entry and exit of each block.
4. Optimise the code.

EXERCISE ▶ **With a loop**
Same questions with :

```
x:=a+b;
y:=a*b;
while(y>a+b) do
   a:=a+a;
   x:=a+b;
done
```

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$

			Step		Step		Step		Step	
ℓ	$kill(\ell)$	$gen(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$	$In(\ell)$	$Out(\ell)$