Compilers - Exercise 5: Termination analysis

In this exercise we will use data-flow analysis to identify code that may not terminate. The idea is to compute a lower bound on the execution time of the code. Assume that we have as input a control-flow graph (CFG) as described in the lecture notes:

```
> data CFGNode = Node Id Instruction [Register] [Register] [Id]  [Id]
> --                                 defs       uses       succs preds
```

We will pretend that all instructions take 1 unit of time to execute – even function calls. The output of the analysis will be, for every node in the graph, the earliest time at which this node could have been reached, assuming that the start node is reached at time 0. Nodes that are never reached are assigned the special value Infinity. If all final (ie exit) nodes of the graph have the value Infinity, the code will never terminate. In fact, nodes assigned Infinity correspond to unreachable nodes that may be removed from the control flow graph.

1. One of the data-flow equations clearly will take the form

$$\text{timeOut}(n) = \text{timeIn}(n) + 1$$

   (a) What is the equation for defining timeIn in terms of termOut?

   (b) As shown in the lecture notes, we use iteration to solve the system of DFA equations timeIn(n) and timeOut(n) for each node n in the CFG.
   What are the initial assignments for timeIn(n) and timeOut(n)?

   (c) Do any nodes need to be initialized with a value different from the others?

   (d) Is this a forward or backward data-flow analysis?

2. Draw the control flow graph for the following code, and show the operation of the iteration algorithm using the equations of part 1.

```
start:
   a = b + c
   d = a < 10
L0:
   if d goto L1 else L2
   b = a - 1
L1:
   a = f(d,e)
   if a goto L0 else L2
L2:
   a = b
end
```

Paul Kelly Imperial College London 2009. *Acknowledgement: This exercise is loosely based on parts of Homework 4 of Andrew Myer's CS412 Introduction to Compilers course at Cornell.*

Exercise 5: Dataflow analysis and optimization

Sample solutions

Exercise 5.1:

1. One of the data-flow equations clearly will take the form

$$\text{timeOut}(n) = \text{timeIn}(n) + 1$$

   (a) What is the equation for defining timeIn in terms of timeOut?

   Answer:

$$\text{timeIn}(n) = \min_{p \in \text{pred}(n)} timeOut(p)$$

   (b) As shown in the lecture notes, we use iteration to solve the system of DFA equations timeIn(n) and timeOut(n) for each node n in the CFG.

   What are the initial assignments for timeIn(n) and timeOut(n)?

   Answer: *You initialise all nodes (except start) to be Infinity, except for the entry node (*`start` *below), which you initialise to be zero.*

   (c) Do any nodes need to be initialized with a value different from the others?

   Answer: *Yes, the entry node (labelled* `start` *below).*

   (d) Is this a forward or backward data-flow analysis?

   Answer: *It's a forward dataflow problem. Information is propagated from* `start` *forwards. TimeIn*(n) *depends on its predecessor* pred(n).

2. Draw the control flow graph for the following code, and show the operation of the iteration algorithm using the equations of part 1.

```
     start:
 1:     a = b + c
 2:     d = a < 10
     L0:
 3:     if d goto L1 else L2
 4:     b = a - 1
     L1:
 5:     a = f(d,e)
 6:     if a goto L0 else L2
     L2:
 7:     a = b
end
```

Answer: *Control flow graph:*

Node id	Instruction	Uses	Defs	Succs	Preds
0:	start			1	
1:	a = b + c	b,c	a	2	0
2:	d = a < 10	a	d	3	1
3:	if d goto L1 else L2	d		5,7	2
4:	b = a - 1	a	b	5	
5:	a = f(d,e)	d,e	a	6	4,3
6:	if a goto L0 else L2	a		3,7	5
7:	a = b	b	a	8	3,6

I've shown the defs and uses here though we don't need them for this example. Note that I have added an explicit `start` *node because we have to initialise it specially.*

Iteration:

Node id	Preds	Step 0		Step 1		Step 2	
		timeIn	timeOut	timeIn	timeOut	timeIn	timeOut
0:			0	0	0	0	0
1:	0	∞	∞	0	1	0	1
2:	1	∞	∞	1	2	1	2
3:	2	∞	∞	2	3	2	3
4:		∞	∞	∞	∞	∞	∞
5:	4,3	∞	∞	3	4	3	4
6:	5	∞	∞	4	5	4	5
7:	3,6	∞	∞	3	4	3	4