1 Correctness of Kildall’s Algorithm (7 Pts.)

Show that Kildall’s algorithm produces the MOP solution when applied to a distributive monotone dataflow analysis framework. Let $\text{PATH}(n)$ denote the set of all paths from the starting node $n_0$ to a node $n$ in the graph $G$. Then the MOP solution is defined as:

$$fp(n) = \bigcap_{p \in \text{PATH}(n)} fp(p)$$

for all $n$. Kildall’s algorithm formally computes:

**Input:** An instance $I = (G, M)$ of a distributive monotone dataflow analysis framework $D = (\mathcal{L}, \mathcal{F})$ with $\mathcal{L} = (L, \preceq, \cap, 1)$ where $G = (N, E, n_0)$ is a flow graph. $M$ maps each node $n$ in $G$ to the corresponding function $f_n$ in $\mathcal{F}$.

**Init:** $\forall n: A[n] = \bot$

**Iteration:** Visit nodes in any order $n_1, n_2, \ldots$ (with repetitions and not fixed in advance). Whenever visiting a node $n$ set

$$A[n] = \bigcap_{p \in \text{PRED}(n)} fp(A[p])$$

where $\text{PRED}(n) = \{p \mid (p, n) \in E\}$. If there exists a node $n \in N - n_0$ such that equation 1 is not fulfilled after we have visited $n_s$, then there exists $t > s$ such that $n_t = n$. This iteration is repeated until a fixed point is found, i.e. there are no further updates required and equation 1 holds for all $n$.

a) Show that Kildall’s algorithm will always eventually halt.

b) Show that after applying the algorithm, the following invariant holds:

$$A[n] \subseteq \bigcap_{p \in \text{PATH}(n)} fp(\bot)$$

(2)

c) Conclude that the $A[n]$ are the MOP solution of the set of equations.

2 Copy Propagation (5 Pts.)

The copy analysis determines for each program point whether on every execution path leading to it from a copy assignment, e.g. $x := y$, there are no assignments to $y$.

a) Write down the data-flow equations for computing copy propagation information. You may treat the join and transfer function as one.

b) Construct the CFG and then apply the fixed point iteration to the following program.

```
y := z;
a := b;
while b < 0 {
   if a > z
      b := a + 5;
   else
      a := x;
} 
a := z;
```