Data Flow

Arnav Gupta

April 5, 2024

Contents

1 Data Flow Analysis

For CFG paths that are significant for data flow in the program:

- focuses on the assignment of values to variables and their use
- done by analyzing occurrences of variables

  - definition occurence: value bound to variable
  - use occurrence: value of variable is referred
    * predicate use: variable used to decide whether predicate is true
    * computational use: compute a value for defining other variables or output value

2 Definition and Uses

Program contain variables: defined by assigning values and using in expressions

With pointers, both the pointer variable and the data it points to must be considered.

With arrays, index variables must be considered, though only the part of the array used neds to be considered (same for fields in a class).

2.1 Uses

2.1.1 c-use

Refers to uses of a variable that occur within an expression as part of an assignment statement, in an output statement, as a parameter within a function call, and in subscript expressions. (computational)

2.1.2 p-use

Refers to the occurrence of a variable in an expression used as a condition in a branch statement such as `if` or `while`. (predicate)

2.2 Formalization

Variable definition $d(v, n)$: value is assigned to v at node n (LHS of assignment, input statement)

Variable use

- c-use(v, n): variable v used in a computation at node n (RHS of assignment, argument of procedure call, output statement)

- p-use(v, m, n): variable v used in predicate from node m to n (as part of an expression used for a decision)

Variable kill $k(v, n)$: variable v deallocated at node n

3 Data Flow Action

3.1 Checklist

For each successive pair of actions:

- suspicious: dd
- likely defect: dk, kk
- defect: ku
- okay: du, kd, ud, uk, uu

First occurrence of action on a variable:

- suspicious: k, u (unless global)
- okay: d

Last occurrence of action on a variable:

- suspicious: d (defined but never used after)
- okay: k, u (but maybe deallocation forgotten)

3.2 Data Flow Graph

Captures the flow of definitions and uses across basic blocks in a program. Similar to a CFG in that nodes, edges, and paths in the CFG are preserved in the DFG.

- annotate nodes with def and c-use as needed, and each edge with p-use as needed
- label each edge with the condition which when true causes the edge to be taken

3.2.1 Paths and Pairs

Complete path: initial node is start node, final node is exit node

Simple path: all nodes except possibly first and last are distinct

Loop free path: all nodes are distinct

def-clear path with respect to v: any path starting from a node at which variable v is defined and ending at a node at which v is used, without redefining v anywhere else along the path

du-pair with respect to v: (d, u) such that

- d is the node where v is defined
- u is the node ehrtr v is used
- def-clear path with respect to v from d to u
- also referred to as reach: def of v at d reaches the use at u

du-path with respect to v: a path $P = \langle n_1, n_2, \ldots, n_j, n_k \rangle$ such that $d(v, n_1)$ and either one of the following two cases:

- c-use of v at node n_k and P is a def-clear simple path with respect to v
- p-use of v on edge n_j to n_k and $\langle n_1, n_2, \ldots, n_j \rangle$ is a def-clear loop-free path

4 Data Flow Coverage

4.1 All-Definitions

At least one def-clear path from every defining node of v to at least one use of v, be that a c-use or p-use

4.2 All-Uses

At least on def-clear path form every defining node of v to every (reachable) use of v

4.3 All-P-Uses/Some-C-Uses

At least one def-clear path from every defining node of v to every reachable p-use of v

- if none, then to a c-use

4.4 All-C-Uses/Some-P-Uses

At least one def-clear path from every defining node of v to every reachable c-use of v

- if none, then to a p-use

4.5 All-DU-Paths

All du paths covered