Data Flow Testing

CS 3750
Consider a famous book by Niklaus Wirth: *Algorithms + Data Structures = Programs*. What part of “programs” have we considered in white-box testing so far?
Imagine a program graph. We are interested in a node $u$ if it is one of two types:

- a *defining node* for some variable $v$ ($\text{DEF}(v, n)$)
- a *usage node* for some variable $v$ ($\text{USE}(v, n)$)

- We may use a variable in a *predicate use* (P-use) or in a *computation use* (C-use).
- Note the original paper had P-uses on edges and C-uses as nodes. For simplicity, I will “move” P-uses to a node containing the branches.
Types of Paths

We can also consider two particular types of paths between these data nodes:

- A *definition/use path* (du-path) for variable $v$ is a path that begins with a defining node for $v$ and ends with a use node for $v$.

- A *definition-clear path* (dc-path) for variable $v$ is a du-path with only one defining node for $v$ (the initial node).
Example

Find the du-paths and dc-paths for z in the program below.

```java
public int basisExample( int x, int y ) {
    int z = x + y;

    for( int a = 1; a < z; a++, z-- ) {
        if( x > z ) {
            a = a + x;
        }
    }

    return z;
}
```
We can define metrics using these du- and dc-paths to help us guide and/or evaluate our testing! Some of the main metrics are called the “Rapps-Weyuker” metrics (after Sandra Rapps and Elaine J. Weyuker, the inventors of this technique in 1982). A set of paths $T$ can be classified in several ways listed below.

- **All-Defs**: for every variable $v$, $T$ contains a dc-path from every defining node for $v$ to a usage node for $v$
- **All-Uses**: for every variable $v$, $T$ contains a dc-path from every defining node for $v$ to every usage node for $v$
Even More Fun Coverage Metrics

- **All-P-Uses/Some-C-Uses**: for every variable \( v \), \( T \) contains a dc-path from every defining node for \( v \) to every predicate usage node for \( v \). If a defining node for \( v \) does not have a dc-path to a predicate usage, then \( T \) contains a dc-path from that defining node for \( v \) to a computation use node for \( v \).

- **All-C-Uses/Some-P-Uses**: (like the above only with the usages flipped)

- **All-DU-Paths**: for every variable \( v \), \( T \) contains every loop-free dc-path from every defining node for \( v \) to every usage node for \( v \)
A Few Formal Definitions

To describe the coverage metrics above, it is helpful to introduce a few formal definitions.

- $dcu(x, i)$: the set of all nodes $j$ such that $j$ is a $c$-use of $x$ and there is a def-clear path with respect to (wrt) $x$ from $i$ to $j$
- $dpu(x, i)$: the set of all edges $(j, k)$ such that $(j, k)$ is a $p$-use of $x$ and there is a def-clear path wrt $x$ from $i$ to $(j, k)$