

# Data Flow Testing

Mohammad Mousavi

Halmstad University, Sweden

`http://ceres.hh.se/mediawiki/DT8021\_Ed\_2015`

Testing and Verification of Embedded Systems (DT8021),  
April 12, 2015

## Path Testing: Pros and Cons

### Pros:

- ▶ Abstract **adequacy** criterion
- ▶ A measure of **testability**: program complexity

### Cons:

- ▶ Too abstract: not clear **when** to use **which** criterion
- ▶ No use **data flow** and variable **dependencies**
- ▶ No use of **specification**

# Flow Graphs

## Defining nodes

Abstract models of program (control) structure.

Assume for the rest:

- ▶ a single **start** (entry) node:  $n_s$ ,
- ▶ a single **termination** (exit) node  $n_t$ ,
- ▶ one component, no dead code

## Structured Loops

- ▶ a loop  $L$  is a strongly connected component
- ▶ entry node of a loop  $L$  is node  $n \notin nodes(L)$ , such that  $n \rightarrow m$  for some  $m \in nodes(L)$

## Structured Loops

- ▶ a loop  $L$  is a strongly connected component
- ▶ entry node of a loop  $L$  is node  $n \notin nodes(L)$ , such that  $n \rightarrow m$  for some  $m \in nodes(L)$
- ▶ **exit** node of a loop  $L$  is node  $n \notin nodes(L)$ , such that  $m \rightarrow n$  for some  $m \in nodes(L)$
- ▶ **structured** loops: **unique entry** and **exit** nodes

# Annotated Flow Graphs

## Defining nodes

$DEF(n, v)$  holds (for a var.  $v$  and a node  $n$ ), when  $n$  defines  $v$ .

Examples:

- ▶  $input(v)$ , or
- ▶  $v := exp$

$$DEF(n) = \{v \mid DEF(n, v)\}$$

# Annotated Flow Graphs

## Using nodes

$USE(n, v)$  holds (for a var.  $v$  and a node  $n$ ), when  $n$  uses the values of  $v$ . Examples:

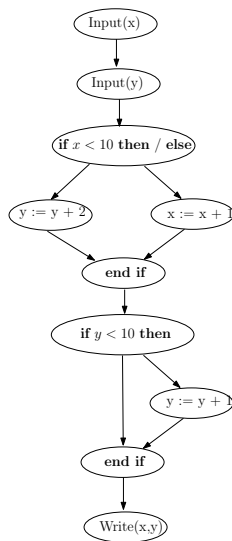
- ▶  $output(v)$ ,
- ▶  $x := exp(v)$ ,
- ▶ *if*  $cond(v)$  *then*, or
- ▶ *while*  $cond(v)$  *do*, ...

$$USE(n) = \{v \mid USE(n, v)\}$$

Also  $REF(n, v)$  in the literature

## Definitions and Uses: An Example

- 1: Input(x) {DEF(1) = {x}}
- 2: Input(y) {DEF(2) = {y}}
- 3: **if** x < 10 **then**
- 4:   y := y + 2 {DEF(4) = USE(4) = {y}}
- 5: **else**
- 6:   x := x + 1
- 7: **end if**
- 8: **if** y > 20 **then**
- 9:   y := y + 1;
- 10: **end if**
- 11: Write(x,y) {USE(11) = {x,y}}
- 12: **end**

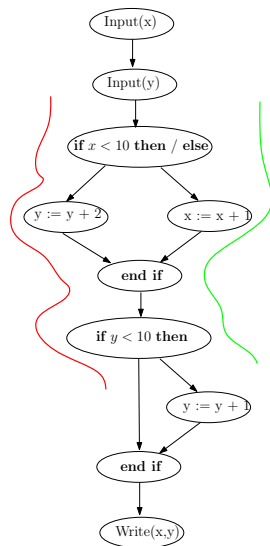




## DC Paths

A definition-clear path (**DC-path**) wrt.  $v$  is a path  $m, P, m'$  such that for all  $n \in \text{nodes}(P)$ ,  $v \notin \text{DEF}(n)$ .

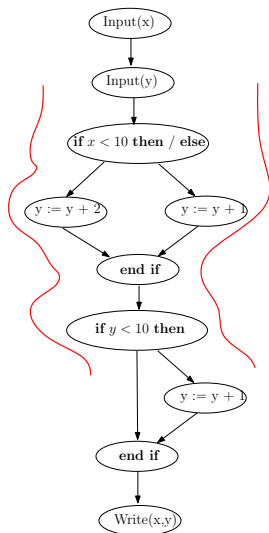
$\text{DEF}(m, v)$  reaches  $\text{USE}(n, v)$  when there is **DC-path**  $m, \dots, n$ .



## DC Paths

A definition-clear path (**DC-path**) wrt.  $v$  is a path  $m, P, m'$  such that for all  $n \in \text{nodes}(P)$ ,  $v \notin \text{DEF}(n)$ .

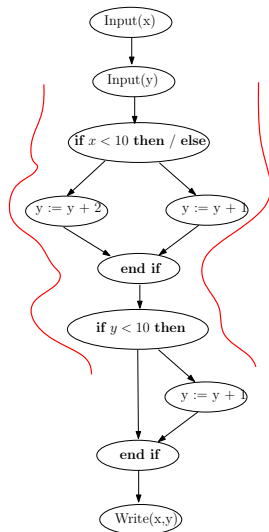
$\text{DEF}(m, v)$  reaches  $\text{USE}(n, v)$  when there is **DC-path**  $m, \dots, n$ .



# Anomalies

Unused definition: A  $DEF(m, v)$  that does not reach any  $USE(n, v)$ .

Undefined usages: A DC-path wrt.  $v$  from  $n_s$  to  $n$  such that  $USE(n, v)$ .

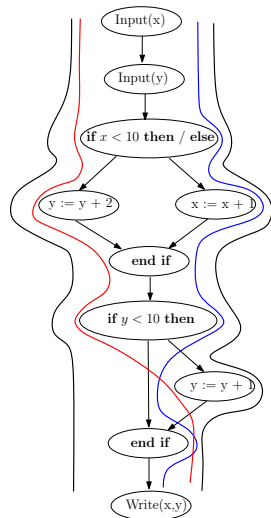


## DU-Path Testing

A test-set  $S$  satisfies **DU-path** when for each variable  $v$ , for each nodes  $m, n, n'$  such that

1.  $DEF(m, v)$  and  $USE(n, v)$ ,
2.  $DEF(m, v)$  reaches  $USE(n, v)$ , and
3.  $n \rightarrow n'$

then **all** simple DC-paths (wrt.  $v$ )  $m, \dots, n, n'$  are covered by a test-case in  $S$ .



## All-Uses Testing

A test-set  $S$  satisfies **All Uses** when  
for each variable  $v$ , for each nodes  $m, n, n'$  such that

1. for each  $DEF(m, v)$  and  $USE(n, v)$ ,
2.  $DEF(m, v)$  reaches  $USE(n, v)$ , and
3.  $n \rightarrow n'$

then **at least one** simple DC-paths (wrt.  $v$ )  $m, \dots, n, n'$   
is covered by a test-case in  $S$ .

## Def-Paths

For each variable  $v$  and node  $m$  such that  $DEF(m, v)$ , the Def-Path set of  $DEF(m, v)$ , denoted by  $Def - Paths(m, v)$  is the set of all DC-paths  $m, \dots, n, n'$  such that:

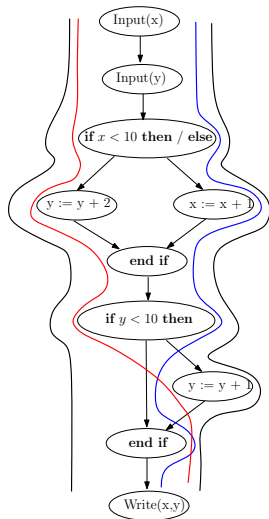
1.  $USE(n, v)$ ,
2.  $DEF(m, v)$  reaches  $USE(n, v)$ , and
3.  $n \rightarrow n'$

## All-Defs Path Testing

A test-set  $S$  satisfies **All-Defs** when for each variable  $v$ , for each nodes  $m, n, n'$  such that

1. for each  $DEF(m, v)$ ,  $USE(n, v)$ ,
2. for each path  $p$  in  $Def - Paths(n, v)$ ,

there is a test case in  $S$  covering  $p$ .



# Subsumption Relation

