Software Development II
Unit 5: Whitebox Testing
Graphs and Programgraphs

Markus Roggenbach

March 2019

You will learn

- what a graph is
- what the “control flow” of a program is
- how graphs can be used to represent the control flow of a program

Graphs

Directed Graphs

A directed graph (digraph) $G = (V, E)$ consists of

- V : a set of nodes (vertices), and
- E : a set of edges.

Each edge e is a pair (n, m) of nodes, i.e. $n, m \in V$.

We say that e goes from n to m.
We call n the start node of e, and m the end node of e.

Indegree/Outdegree of a node

In a directed graph G, the

- indegree of a node n is the number of distinct edges that have n as their end node.
- outdegree of a node n is the number of distinct edges that have n as their start node.

Types of nodes

A node with

- indegree $= 0$ is a source node.
- outdegree $= 0$ is a sink node.
- indegree $\neq 0$ and outdegree $\neq 0$ is a transfer node.

Path and cycle

- A directed path is a sequence of edges such that, for any adjacent pair of edges in the sequence, the end node of the first edge is the start node of the second edge.
- A cycle is a directed path that begins and ends at the same node.

Control Flow

Manual program execution

```
public static int clip(int lower, int upper, int x) {
    if (x < lower) {
        x = lower;
    }
    if (x > upper) {
        x = upper;
    }
    return x;
}
```

Definition: Control flow

Control flow (or flow of control) is the order in which statements are executed.

Programs as graphs

Program graph

Given a Java program, its program graph is a directed graph in which

- nodes are statements (or fragments of statements), and
- edges represent the flow of control.

There is an edge from node i to node j
iff
statement j can be executed directly after statement i.

Program graph for sequences

```
statement1;
statement2;
statement3;
```

Program graphs for conditionals

```
if (condition) {
   statements;
}

if (condition) {
    statements;
} else {
   statements;
}

if (condition) {
   statements;
} else if (condition) {
   statements;
} else if (condition) {
  statements;
}
```

Program graphs for loops

```
for (initialization;condition;update) {
   statements;
}

while (condition) {
   statements;
}

do {
   statements;
} while (condition)
;
```

Program graph - example 1

```
public static void main (String args[]) {
  int a; int b; int c;
  TriangleType result;
  Scanner input = new Scanner(System.in);
  System.out.print("Length of side A: ");
  a = input.nextInt();
  System.out.print("Length of side B: ");
  b = input.nextInt();
  System.out.print("Length of side C: ");
  c = input.nextInt();
  result = classify(a,b,c);
  System.out.println(TriangleType.printTriangleType(result));
}
```

Program graph - example 2

```
public static int clip (int lower, int x) {
    if (x < lower) {
        x = lower;
    }
    return x;
}
```

Program graph - example 3

```
public static int Erna (int a, int b) {
    int d = 1;
    int i = 0;
    while (i <= a) {
        d = d + b;
        i++;
    }
    return d;
}
```

Running a program

Every terminating execution of a program
corresponds to
one path through the program graph –
beginning at the source node and
ending at the sink node of the graph.

What you have learned in this unit

Definitions

- Graph; types of nodes; path; cycle
- Control Flow

You should be able to construct

- The control flow graph of a method.