Graph Coverage Applications

Software Testing

(3104313)

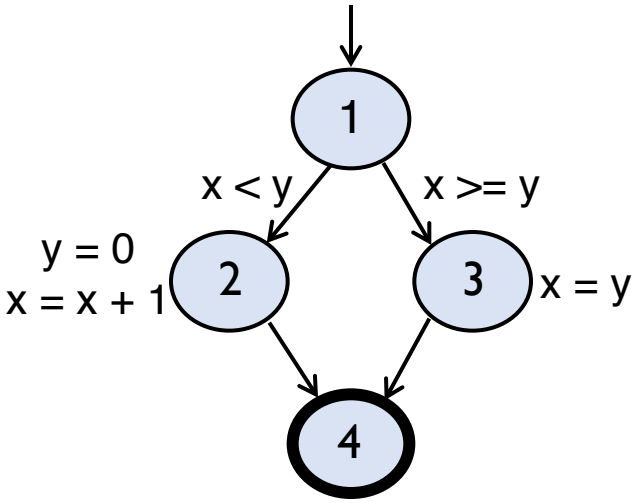
Amirkabir University of Technology
Spring 1399-1400

Control Flow Graph (CFG)

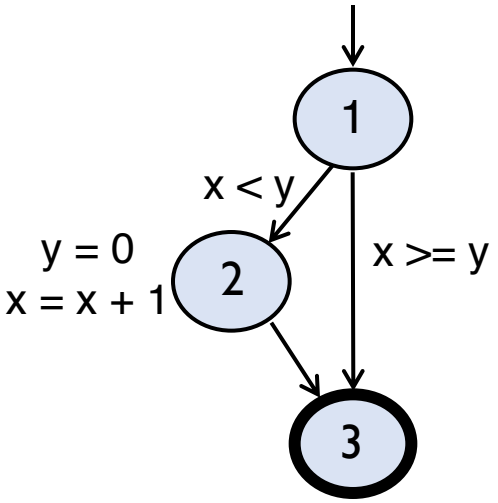
The most common graph for **source code**.

CFG: `if` Statement

```
if (x < y)
{
   y = 0;
   x = x + 1;
}
else
{
   x = y;
}
```




```
if (x < y)
{
   y = 0;
   x = x + 1;
}
```



In-Class Exercise #15

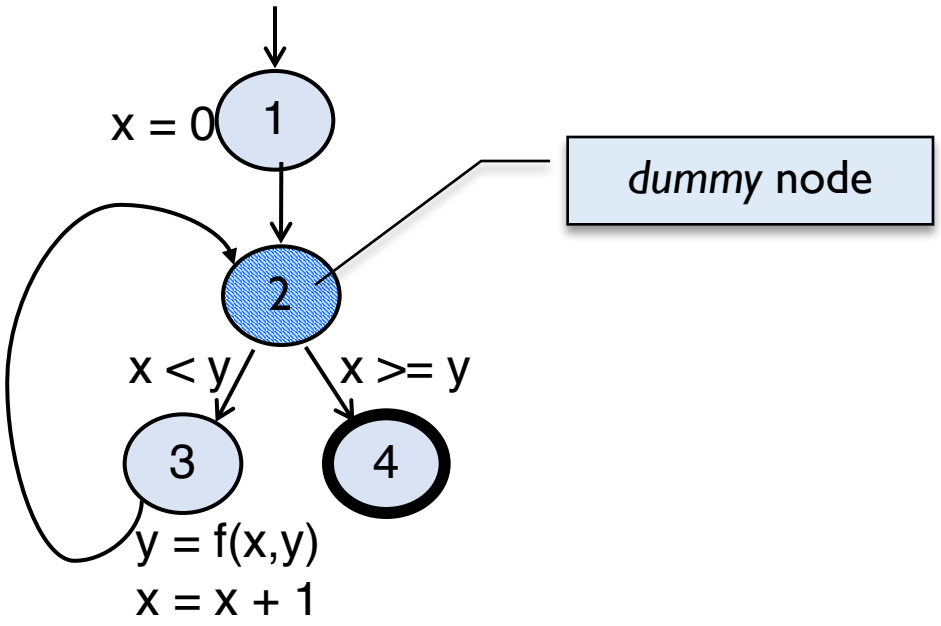
1. Draw the CFG for the given *while* loop. (label the edges)

```
x = 0;
while (x < y){
   y = f (x, y);
   x = x + 1;
}
return (x);
```

- You have ∞ minutes ☺, but now think about 2-3 minutes!
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

CFG: `while` Statement

```
x = 0;
while (x < y){
   y = f (x, y);
   x = x + 1;
}
return (x);
```



In-Class Exercise #15

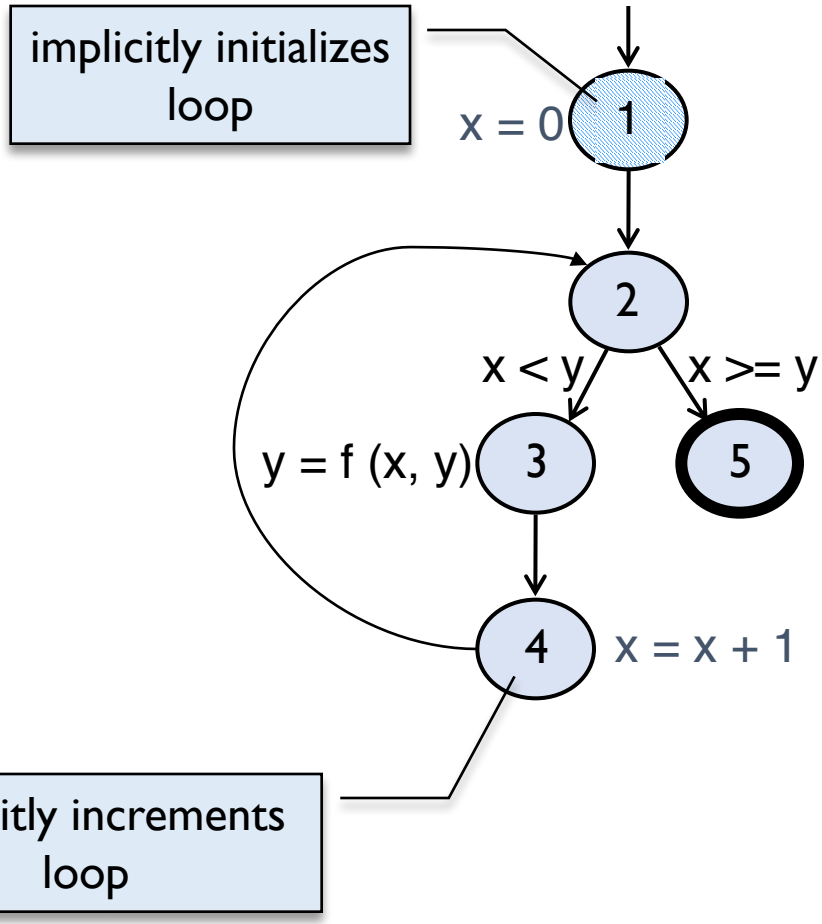
2. Draw the CFG for the given *for* loop. (label the edges)

```
for (x = 0; x < y; x++){
   y = f (x, y);
}
return (x);
```

- You have ∞ minutes ☺
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

CFG: **for** Statement

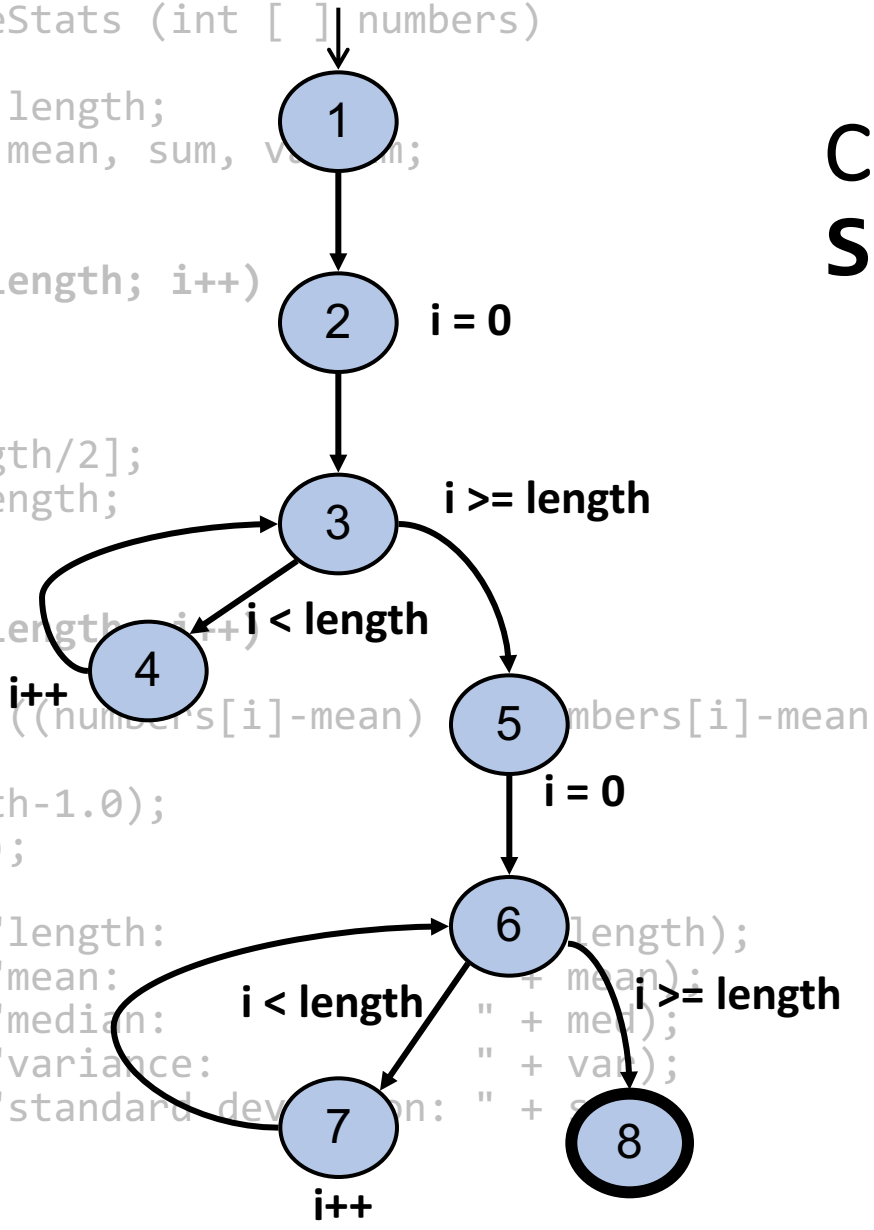
```
for (x = 0; x < y; x++){
   y = f (x, y);
}
return (x);
```



```
public static void computeStats (int [ ] numbers)
{
     int length = numbers.length;
     double med, var, sd, mean, sum, varsum;

     sum = 0;
     for (int i = 0; i < length; i++)
     {
       sum += numbers [i];
     }
     med   = numbers [length/2];
     mean = sum/(double)length;

     varsum = 0;
     for (int i = 0; i < length; i++)
     {
       varsum = varsum  + ((numbers[i]-mean) * (numbers[i]-mean));
     }
     var = varsum / (length-1.0);
     sd  = Math.sqrt (var);

     System.out.println ("length:              " + length);
     System.out.println ("mean:                " + mean);
     System.out.println ("median:              " + med);
     System.out.println ("variance:            " + var);
     System.out.println ("standard deviation: " + sd);
}
```

CFG for **Stats()**

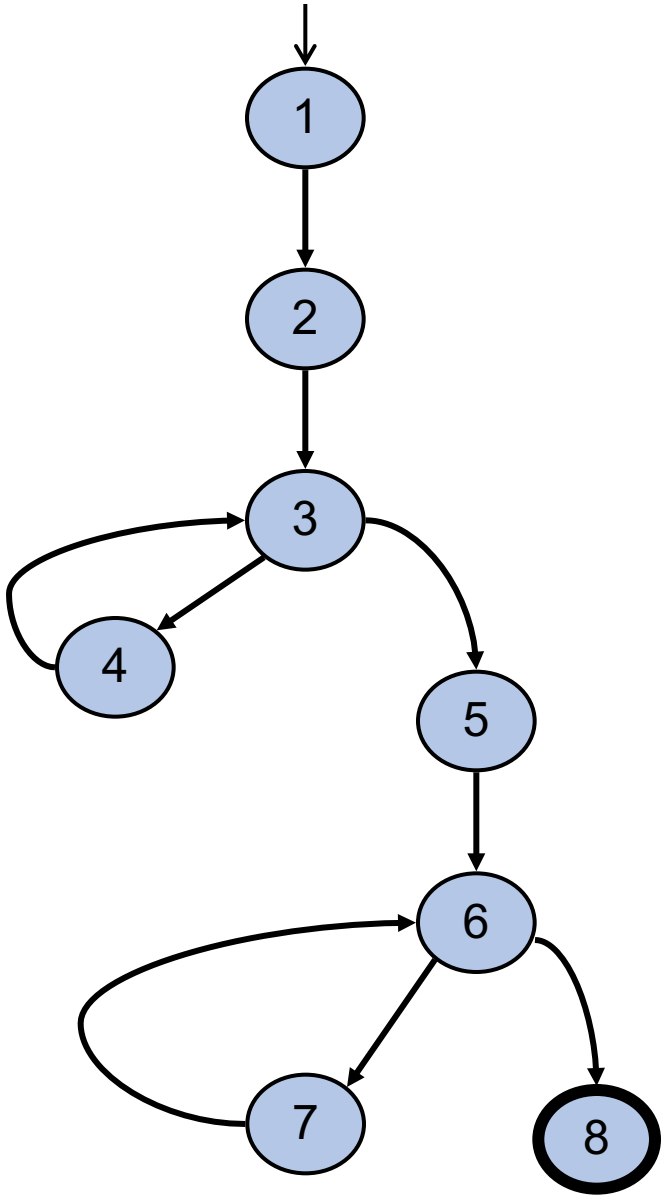
```
public static void computeStats (int [ ] numbers)
{
     int length = numbers.length;
     double med, var, sd, mean, sum, varsum;

     sum = 0;
     for (int i = 0; i < length; i++)
     {
       sum += numbers [i];
     }
     med   = numbers [length/2];
     mean = sum/(double)length;

     varsum = 0;
     for (int i = 0; i < length; i++)
     {
       varsum = varsum  + ((numbers[i]-mean) * (numbers[i]-mean));
     }
     var = varsum / (length-1.0);
     sd  = Math.sqrt (var);

     System.out.println ("length:                " + length);
     System.out.println ("mean:                  " + mean);
     System.out.println ("median:                " + med);
     System.out.println ("variance:              " + var);
     System.out.println ("standard deviation: " + sd);
}
```



CFG for **Stats()**

Graph Coverage for `Stats()`

- Edge Coverage
- Edge-Pair Coverage
- Prime Path Coverage
- ….



After-Class Exercise #16

Give the test requirements and then test paths for the CFG for `Stats()`, applying

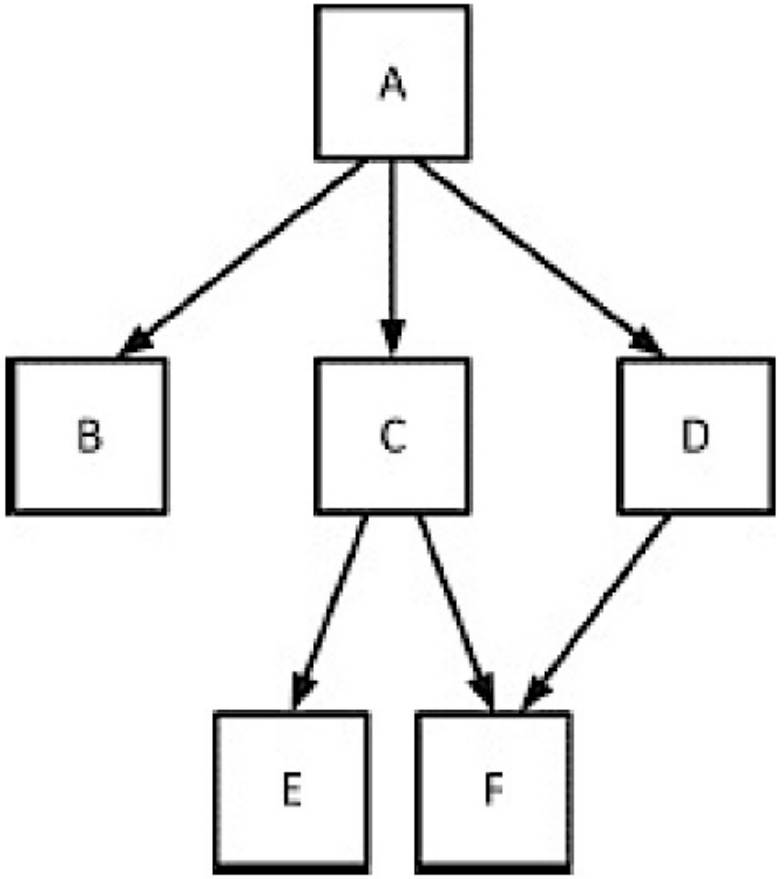
1. Edge-Pair Coverage
2. Prime Path Coverage

- You have 840 minutes!
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

Graph Coverage for Design Elements

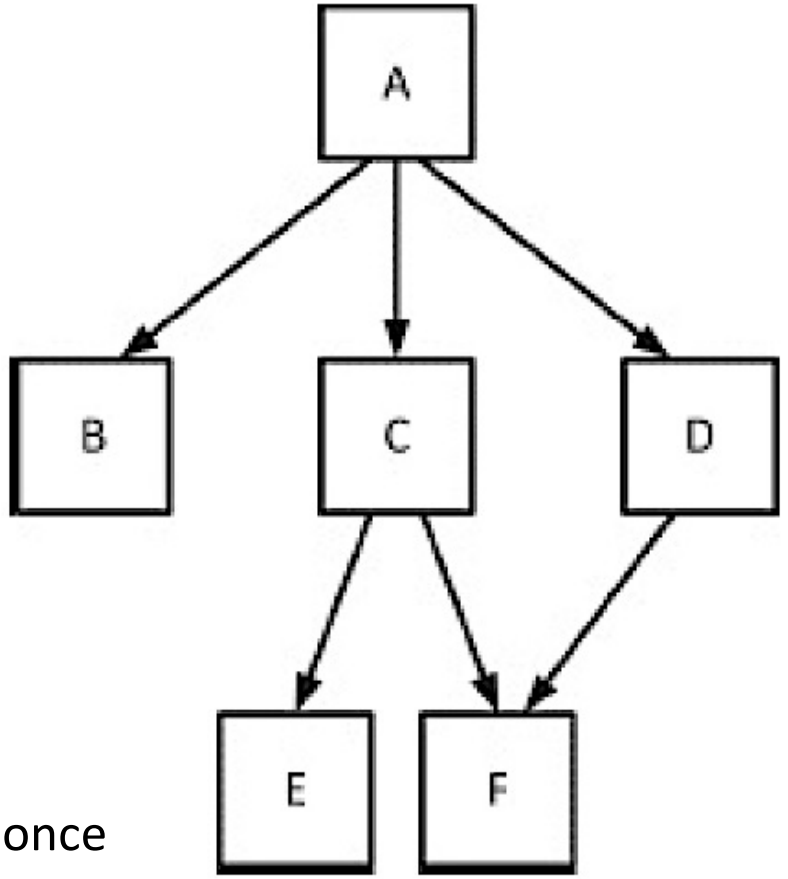
- Data abstraction & OO language features
- Testing software **based on design elements** is becoming more important

Call Graph



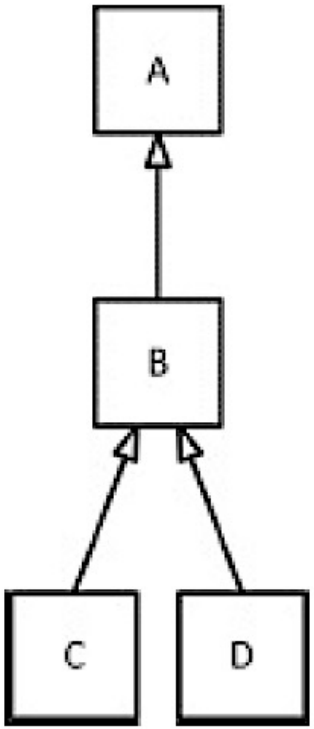
- Node Coverage
- Edge Coverage

Call Graph



- Node Coverage
  - each **method** be called at least once
  - **Method Coverage**
- Edge Coverage
  - each **call** be executed at least once
  - **Call Coverage**

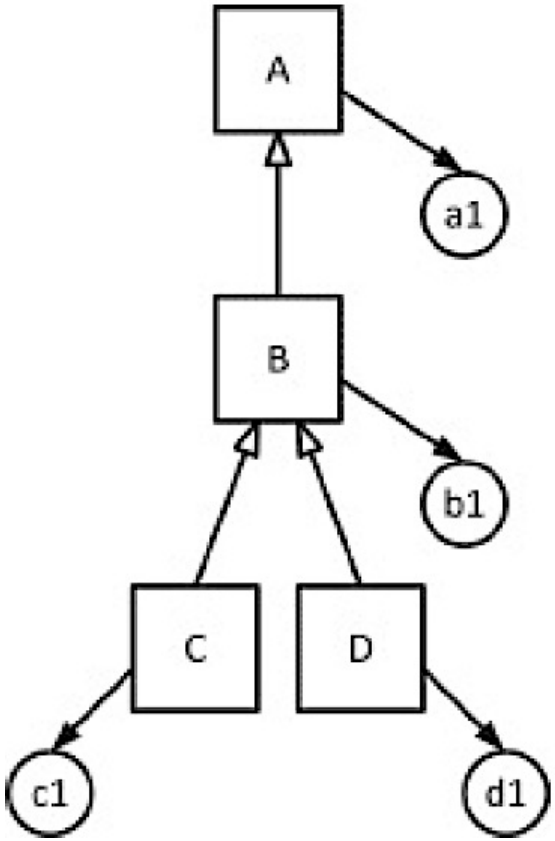
Inheritance Hierarchy



- Inheritance dependency

- Node Coverage?
- Edge Coverage?
- ????

Inheritance Hierarchy



- Inheritance dependency
- Node Coverage?
  - at least **one object be created** for each class.
- Edge Coverage?
  - ?
- OO Call Coverage
  - for each object of each class, the call graph must be covered according to the Call Coverage criterion

break

Graph Coverage for Specifications

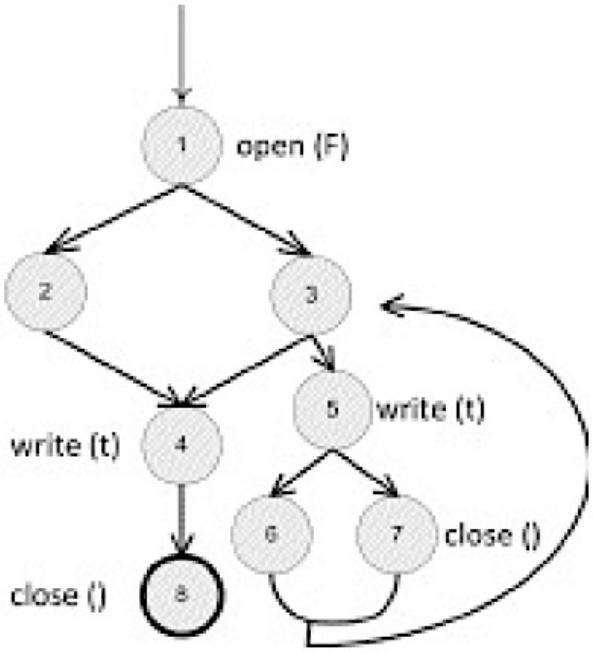
Testing Sequencing Constraints

`FileADT`

- `open (String fName)`
- `close (String fName)`
- `write (String textLine)`

1. An `open(F)` must be executed before every `write(t)`
2. An `open(F)` must be executed before every `close()`
3. A `write(t)` must not be executed after a `close()` unless an `open(F)` appears in between
4. A `write(t)` should be executed before every `close()`
5. A `close()` must not be executed after a `close()` unless an `open(F)` appears in between
6. An `open(F)` must not be executed after an `open(F)` `unless` a `close()` appears in between

Testing Sequencing Constraints

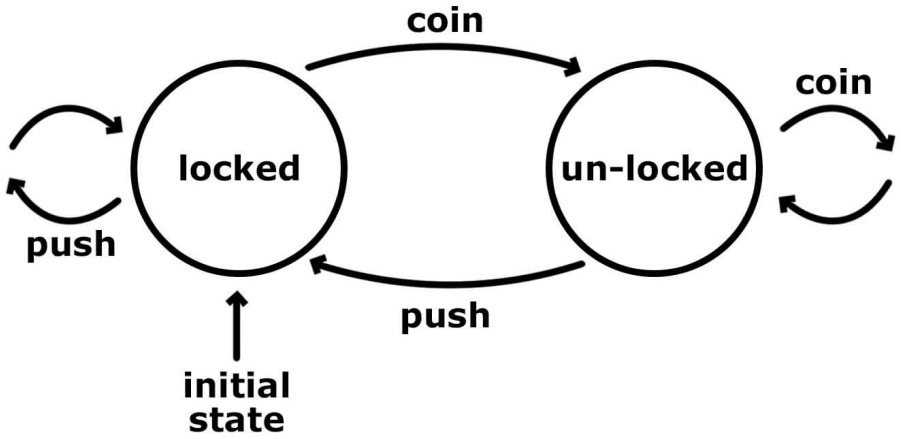


`FileADT`

- `open (String fName)`
- `close (String fName)`
- `write (String textLine)`

1. An `open(F)` must be executed before every `write(t)`
2. An `open(F)` must be executed before every `close()`
3. A `write(t)` must not be executed after a `close()` unless an `open(F)` appears in between
4. A `write(t)` should be executed before every `close()`
5. A `close()` must not be executed after a `close()` unless an `open(F)` appears in between
6. An `open(F)` must not be executed after an `open(F)` `unless` a `close()` appears in between

Testing State Behaviour



1. Combining control flow graphs
2. Using the software structure
3. Modeling state variables
4. Using the implicit or explicit specifications

Graph Coverage for Use Cases

UML Activity Diagram