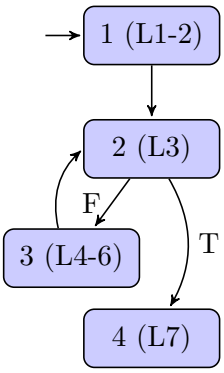
Software Testing, Quality Assurance and Maintenance Winter 2019

Lecture 7 — January 21, 2019

Patrick Lam *version 1*

Basic Blocks. We can simplify a CFG by grouping together statements which always execute together (in sequential programs):



We use the following definition:

Definition 1 *A basic block is a sequence of instructions in the control-flow graph that has one entry point and one exit point.*

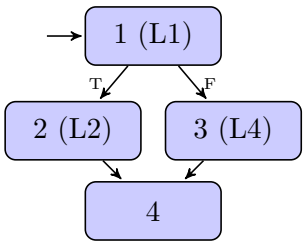
We are usually interested in forming maximal basic blocks. Note that a basic block may have multiple successors. However, there may not be any jumps into the middle of a basic block (which is why statement `l0` has its own basic block.)

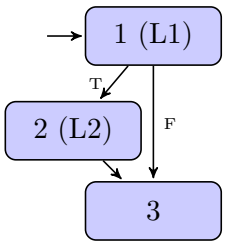
Some Examples

We'll now see how to construct control-flow graph fragments for various program constructs.

if statements: One can put the conditions (and hence uses) on the control-flow edges, rather than in the `if` node. I prefer putting the condition in the node.

```
1  if (z < 17)
2    print (x);
3  else
4    print (y);
```



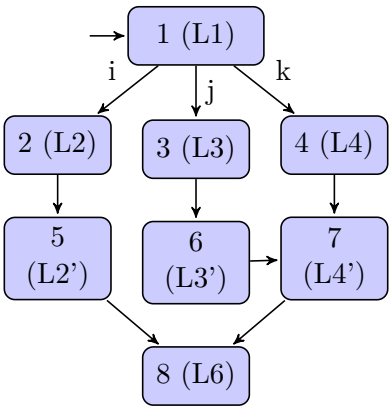



```
1  if (z < 17)
2    print(x);
```

Short-circuit `if` evaluation is more complicated; I recommend working it out yourself.

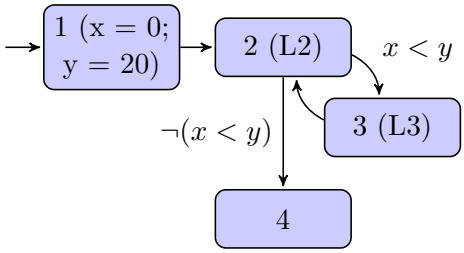
case / switch statements:

```
1  switch (n) {
2    case `I': ...; break;
3    case `J': ...; // fall thru
4    case `K': ...; break;
5  }
6  // ...
```



while statements:

```
1  x = 0; y = 20;
2  while (x < y) {
3    x ++; y --;
4  }
```



Note that arbitrarily complicated structures may occur inside the loop body.

for statements:

```
1  for (int i = 0; i < 57; i++) {
2    if (i % 3 == 0) {
3      print (i);
4    }
5  }
```

(an exercise for the reader;
we saw one earlier!)

This example uses Java's enhanced for loops, which iterates over all of the elements in the `widgetList`:

```
1  for (Widget w : widgetList) {
2    decorate(w);
3  }
```

I will accept the simplified CFG or the more useful one on the right: