Lecture 7

Concolic Execution

Zvonimir Rakamarić
University of Utah

Last Time

- Drawbacks of concrete testing
- Symbolic execution
- Solutions for the path explosion problem
  - Structural abstraction
  - Compositional symbolic execution

Symbolic Execution

- Key idea: execution of programs using symbolic input values instead of concrete data
- Concrete vs symbolic
  - Concrete execution
    - Program takes only one path determined by input values
  - Symbolic execution
    - Program can take any feasible path – coverage!

Symbolic Program State

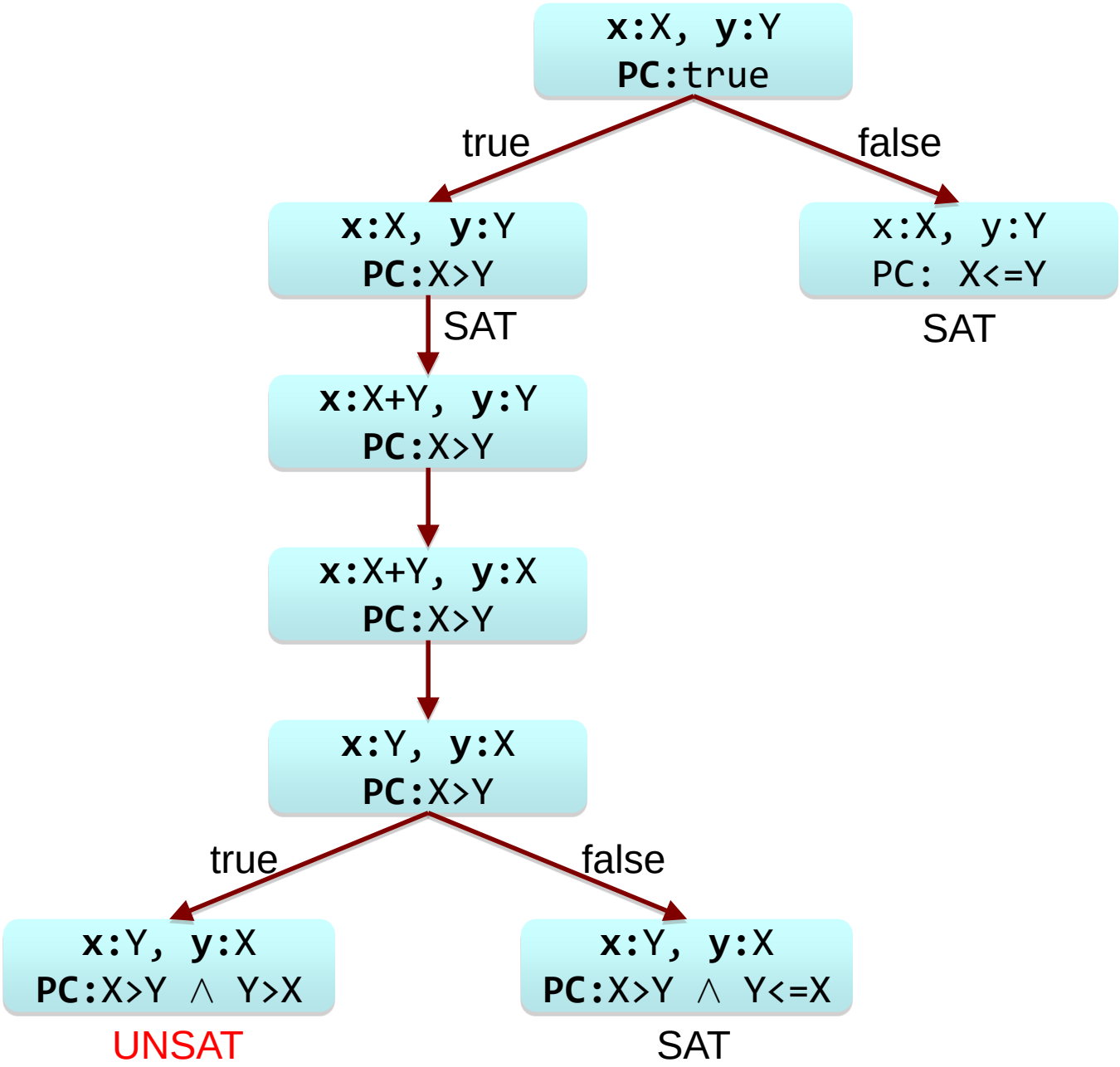
- Symbolic values of program variables
- Path condition (PC)
  - Logical formula over symbolic inputs
  - Accumulates constraints that inputs have to satisfy for the particular path to be executed
  - If a path is feasible its PC is satisfiable
- Program location

Symbolic Execution Tree

- Characterizes execution paths constructed during symbolic execution
- Nodes are symbolic program states
- Edges are labeled with program transitions

Example

```
1) int x, y;
2) if (x > y) {
3)    x = x + y;
4)    y = x - y;
5)    x = x - y;
6)    if (x > y)
7)       assert false;
8) }
```


x:X, y:Y
PC:true
true
false
x:X, y:Y
PC:X>Y
x:X, y:Y
PC: X<=Y
SAT
SAT
x:X+Y, y:Y
PC:X>Y
x:X+Y, y:X
PC:X>Y
x:Y, y:X
PC:X>Y
true
false
x:Y, y:X
PC:X>Y ∧ Y>X
x:Y, y:X
PC:X>Y ∧ Y<=X
UNSAT
SAT

Example

```
int foo(int a, int b) {
  int k = a – b;
  int t = a + b + 3;
  if (a % 2 == 0) {
    a = b++;
    if (t > 0)
      k = t – 2;
  }
  if (a + 6 > k)
    b = 5;
  if (t + a + b == 20)
    assert false;
  return t + a + b;
}
```

Further Limitations of Symbolic Execution

- Limited by the power of constraint solver
  - Cannot handle non-linear and very complex constraints
- Inherently white-box technique
  - Source code (or equivalent) is required for precise symbolic execution
  - Modeling libraries is a huge problem

This Time

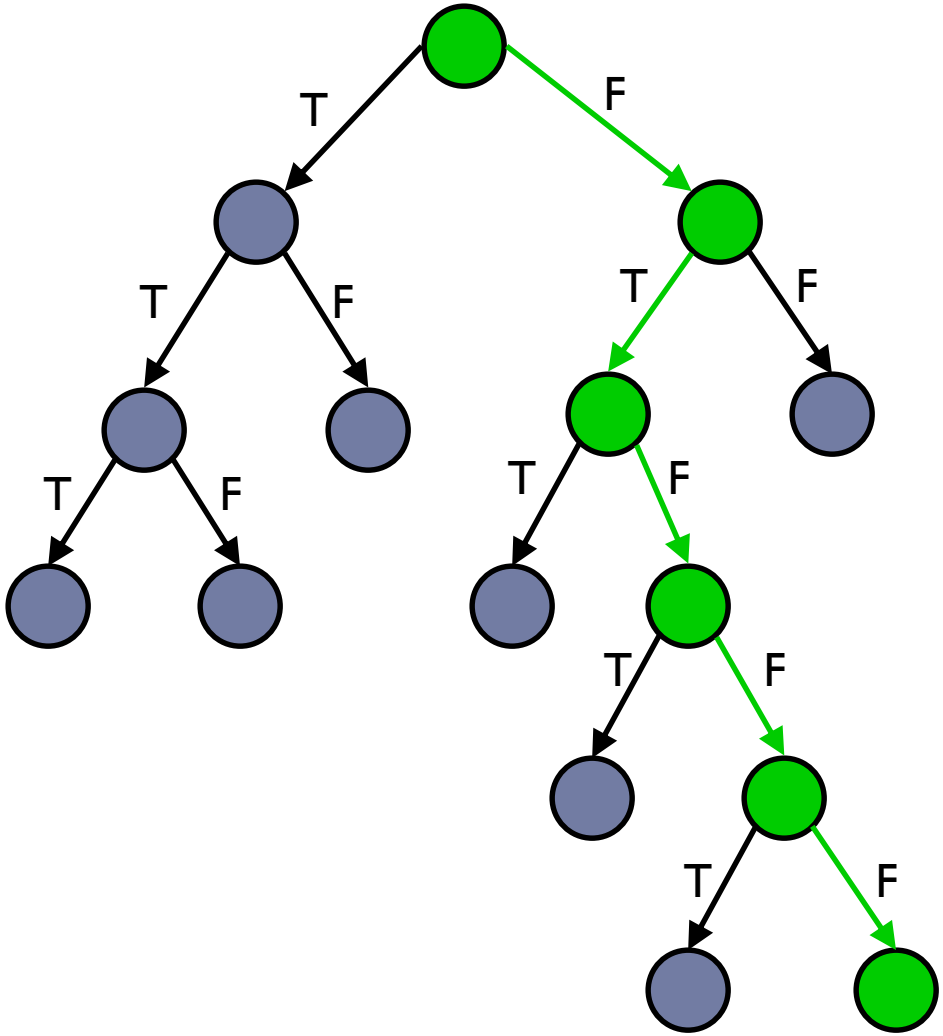
- Combining concrete and symbolic execution
- Many names referring to the same thing:
  - DART (directed automated random testing)
  - Concolic (concrete + symbolic) execution
  - Dynamic symbolic execution

- Playing with KLEE
  - Try aptlab.net

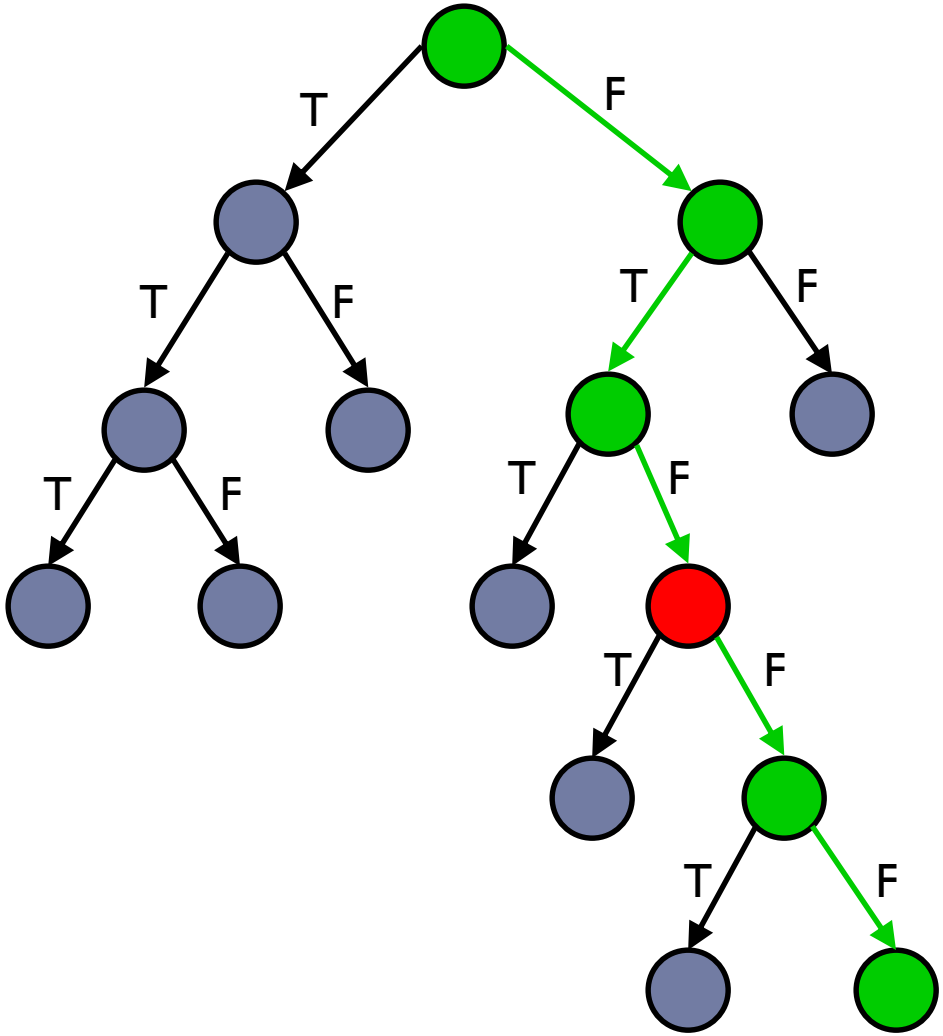
Concolic Execution

- Combination of concrete and symbolic execution to overcome the two weaknesses of classic symbolic execution
- Algorithm
  - Execute program concretely
  - Collect the symbolic path condition along the way
  - Negate a constraint on the path condition after the run and solve it to get a model
  - Execute again with the newly found concrete input values

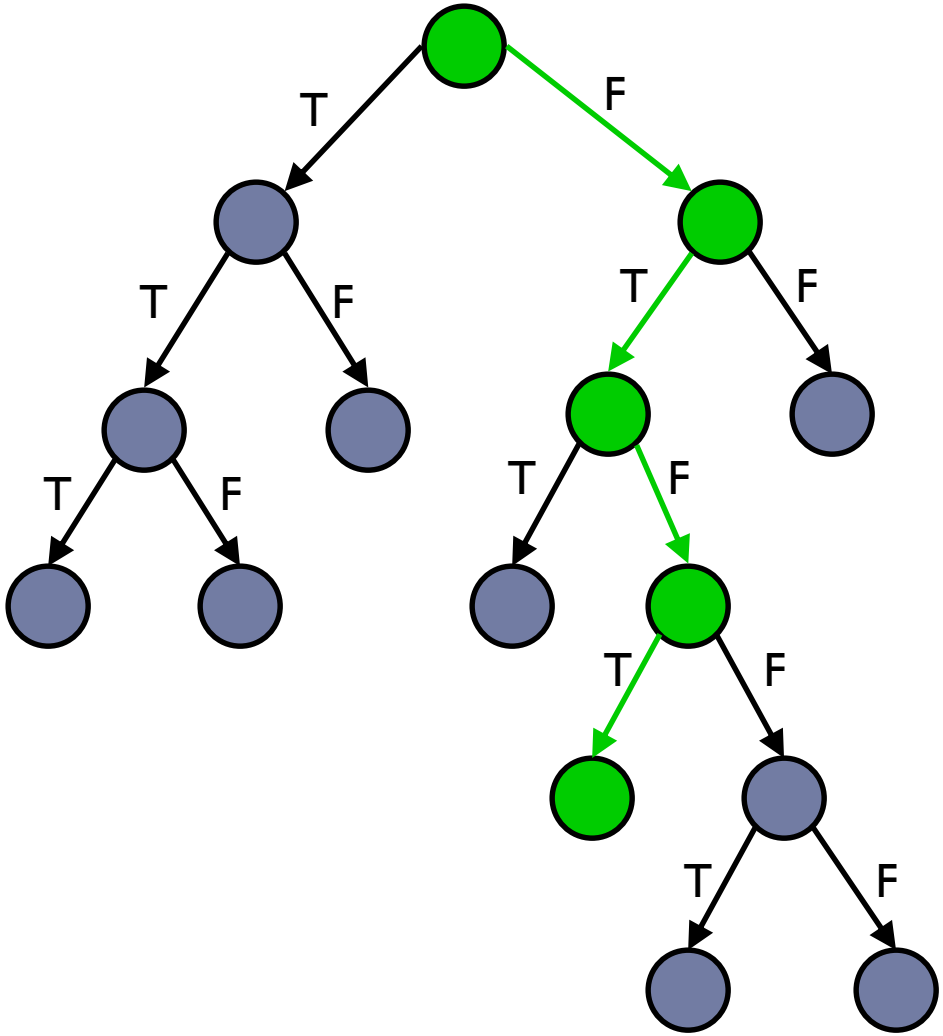
High-Level Picture



High-Level Picture



High-Level Picture



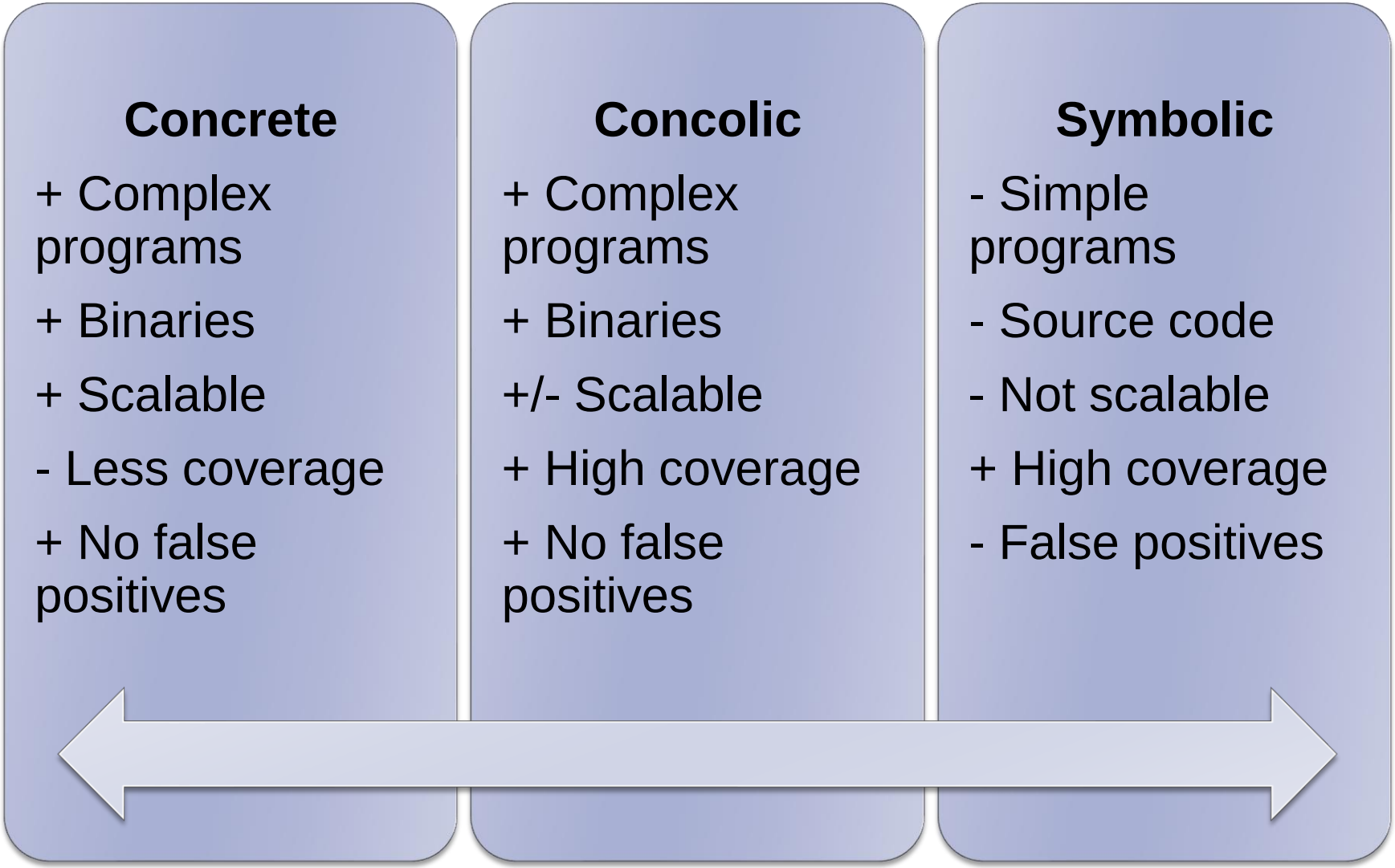
Simple Example I

```
void foo(int x, int y) {
   if (x == y) {
      assert false;
   }
}
```

Simple Example II

```
void foo(int x, int y) {
  if (x == hash(y)) {
    assert false;
  }
}
```

Concolic Covering Middle Ground



Recent Success Stories

- SAGE
  - Microsoft's internal tool for finding security bugs
  - White-box fuzzing
    - Concolic execution for finding bugs in file parsers (jpeg, docx, ppt,…)
  - Last line of defense
  - Big clusters continuously running SAGE
- KLEE
  - Open source concolic executor
  - Runs on top of LLVM
  - Has found lots of problems in open-source software