

# Introduction to Software Analysis

CS 6340

# Why Take This Course?

---

- Learn methods to improve software quality
  - reliability, security, performance, etc.
- Become a better software developer/tester
- Build specialized tools for software diagnosis and testing
- For the war stories

# The Ariane Rocket Disaster (1996)

---



# Post Mortem

---

- Caused due to numeric overflow error
  - Attempt to fit 64-bit format data in 16-bit space
- Cost
  - \$100M's for loss of mission
  - Multi-year setback to the Ariane program
- Read more at <http://www.around.com/ariane.html>

# Security Vulnerabilities

---

- Exploits of errors in programs
- Widespread problem
  - Moonlight Maze (1998)
  - Code Red (2001)
  - Titan Rain (2003)
  - Stuxnet Worm
- Getting worse ...

## 2011 Mobile Threat Report (Lookout™ Mobile Security)

- 0.5-1 million Android users affected by malware in first half of 2011
- 3 out of 10 Android owners likely to face web-based threat each year
- Attackers using increasingly sophisticated ways to steal data and money



# What is Program Analysis?

---

- Body of work to discover useful facts about programs
- Broadly classified into three kinds:
  - Dynamic (execution-time)
  - Static (compile-time)
  - Hybrid (combines dynamic and static)

# Dynamic Program Analysis

---

- Infer facts of program by monitoring its runs
- Examples:

Array bound checking

*Purify*

Datarace detection

*Eraser*

Memory leak detection

*Valgrind*

Finding likely invariants

*Daikon*

# Static Analysis

---

- Infer facts of the program by inspecting its source (or binary) code

- Examples:

Suspicious error patterns

*Lint, FindBugs, Coverity*

Memory leak detection

*Facebook Infer*

Checking API usage rules

*Microsoft SLAM*

Verifying invariants

*ESC/Java*



# QUIZ: Program Invariants

---

An invariant at the end of the program is  $(z == c)$  for some constant  $c$ . What is  $c$ ?

```
int p(int x) { return x * x; }
```

```
void main() {
```

```
    int z;
```

```
    if (getc() == 'a')
```

```
        z = p(6) + 6;
```

```
    else
```

```
        z = p(-7) - 7;
```

z = ?

```
}
```

# QUIZ: Program Invariants

---

An invariant at the end of the program is  $(z == c)$  for some constant  $c$ . What is  $c$ ?

Disaster averted!

```
int p(int x) { return x * x; }
```

```
void main() {
```

```
    int z;
```

```
    if (getc() == 'a')
```

```
        z = p(6) + 6;
```

```
    else
```

```
        z = p(-7) - 7;
```

```
    if (z != 42)
```

```
        disaster();
```

```
}
```

$z = 42$

# Discovering Invariants By Dynamic Analysis

---

$(z == 42)$  *might be* an invariant

$(z == 30)$  is *definitely not* an invariant

```
int p(int x) { return x * x; }
```

```
void main() {  
    int z;  
    if (getc() == 'a')  
        z = p(6) + 6;  
    else  
        z = p(-7) - 7;  
    if (z != 42)  
        disaster();  
}
```

$z = 42$

# Discovering Invariants By Static Analysis

---

*is definitely*  
(z == 42) ~~might be~~ an  
invariant

(z == 30) *is definitely*  
*not* an invariant

```
int p(int x) { return x * x; }
```

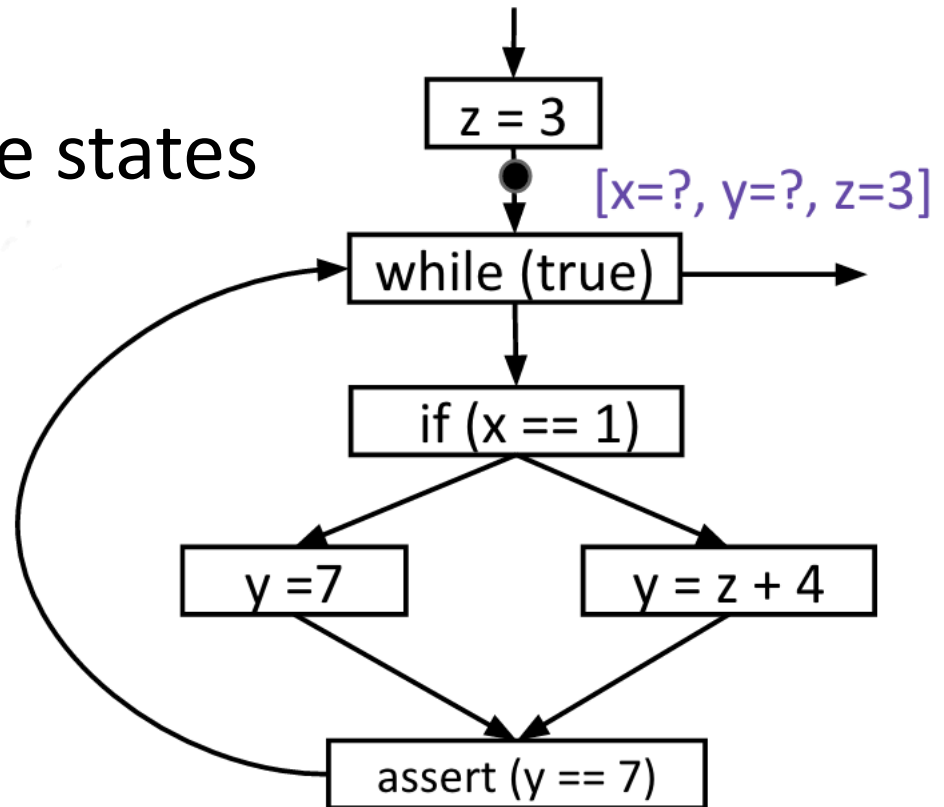
```
void main() {  
    int z;  
    if (getc() == 'a')  
        z = p(6) + 6;  
    else  
        z = p(-7) - 7;  
    if (z != 42)  
        disaster();  
}
```

z = 42

# Terminology

---

- Control-flow graph
- Abstract vs. concrete states
- Termination
- Completeness
- Soundness



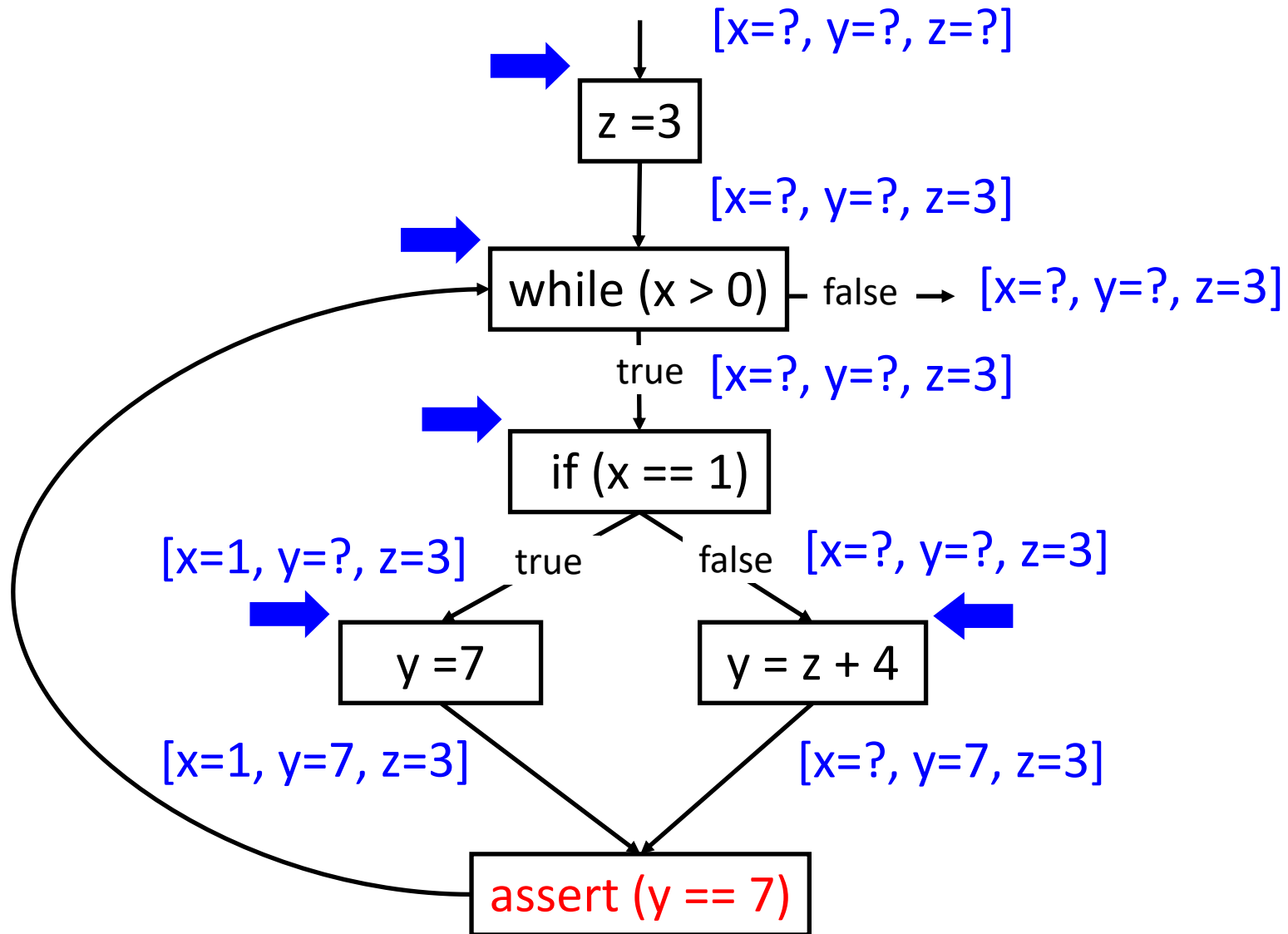
# Example Static Analysis Problem

---

- Find variables that have a constant value at a given program point

```
void main() {  
    z = 3;  
    while (true) {  
        if (x == 1)  
            y = 7;  
        else  
            y = z + 4;  
        assert (y == 7);  
    }  
}
```

# Iterative Approximation



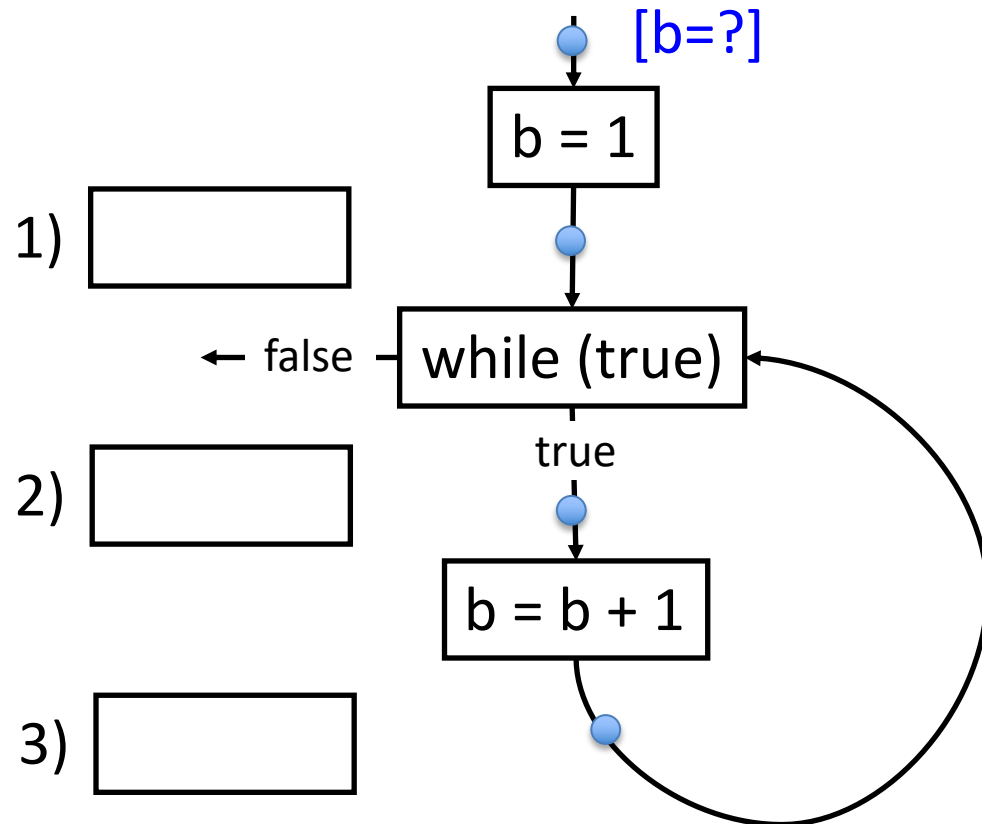
# QUIZ: Iterative Approximation

---

Fill in the value of variable  $b$  that the analysis infers at:

- 1) the loop header
- 2) entry of loop body
- 3) exit of loop body

Enter “?” if a definite value cannot be inferred.



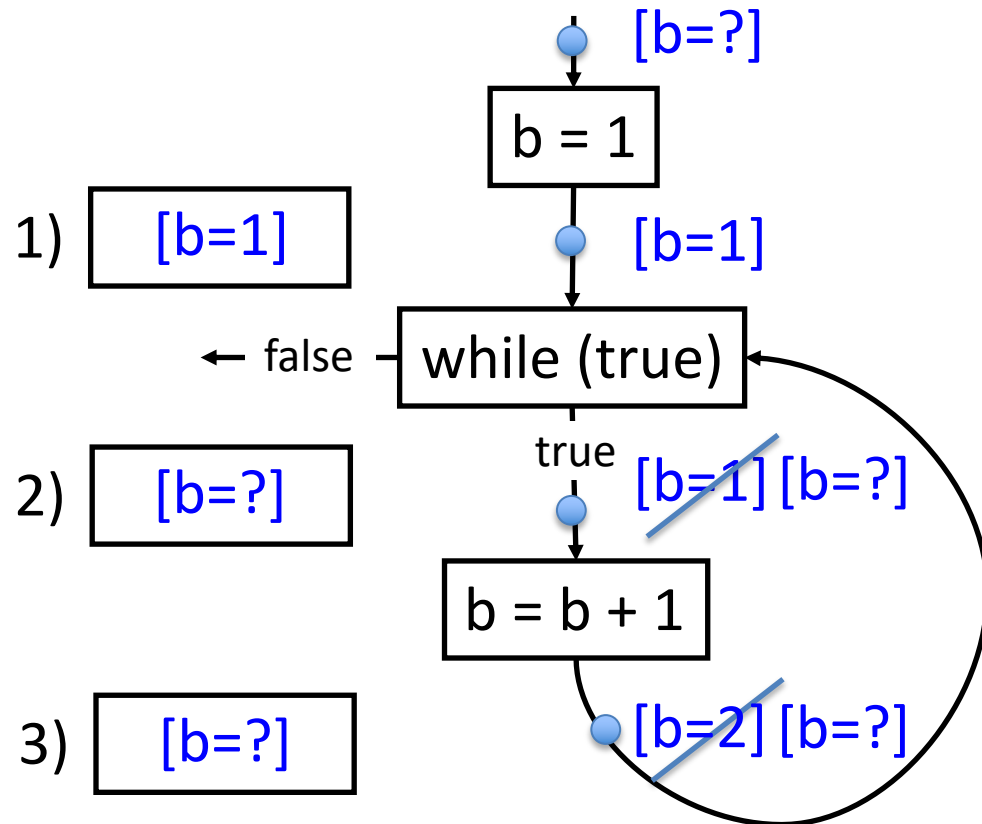


# QUIZ: Iterative Approximation

Fill in the value of variable  $b$  that the analysis infers at:

- 1) the loop header
- 2) entry of loop body
- 3) exit of loop body

Enter “?” if a definite value cannot be inferred.



# QUIZ: Dynamic vs. Static Analysis

---

Match each box with its corresponding feature.

	Dynamic	Static
Cost		
Effectiveness		

- A. Unsound  
(may miss errors)
- B. Proportional to  
program's execution  
time
- C. Proportional to  
program's size
- D. Incomplete  
(may report  
spurious errors)

# QUIZ: Dynamic vs. Static Analysis

---

Match each box with its corresponding feature.

	Dynamic	Static
Cost	B. Proportional to program's execution time	C. Proportional to program's size
Effectiveness	A. Unsound (may miss errors)	D. Incomplete (may report spurious errors)

# Undecidability of Program Properties

---

- Can program analysis be **sound** and **complete**?
  - Not if we want it to **terminate**!
- Questions like “is a program point reachable on some input?” are **undecidable**
- Designing a program analysis is an art
  - **Tradeoffs** dictated by consumer

# Who Needs Program Analysis?

---

Three primary consumers of program analysis:

- Compilers
- Software Quality Tools
- Integrated Development Environments (IDEs)

# Compilers

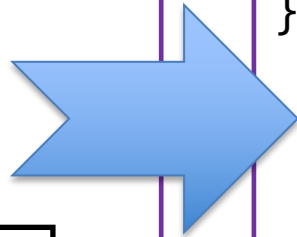
---

- Bridge between high-level languages and architectures
- Use program analysis to generate efficient code

```
int p(int x) { return x * x; }
void main(int arg) {
    int z;
    if (arg != 0)
        z = p(6) + 6;
    else
        z = p(-7) - 7;

    print (z);
}
```

z = 42



```
int p(int x) { return x * x; }
void main() {
    print (42);
}
```

- Runs faster
- More energy-efficient
- Smaller in size

# Software Quality Tools

---

- Primary focus of this course
- Tools for testing, debugging, and verification
- Use program analysis for:
  - Finding programming errors
  - Proving program invariants
  - Generating test cases
  - Localizing causes of errors
  - ...

```
int p(int x) { return x * x; }

void main() {
    int z;
    if (getc() == 'a')
        z = p(6) + 6;
    else
        z = p(-7) - 7;

    if (z != 42)
        disaster();
}
```

$z = 42$

# Integrated Development Environments

---

- Examples: Eclipse and Microsoft Visual Studio
- Use program analysis to help programmers:
  - Understand programs
  - Refactor programs
    - Restructuring a program without changing its behavior
- Useful in dealing with large, complex programs



# What Have We Learned?

---

- What is program analysis?
- Dynamic vs. static analysis: pros and cons
- Program invariants
- Iterative approximation method for static analysis
- **Undecidability** => program analysis cannot ensure  
termination + soundness + completeness
- Who needs program analysis?