Introduction to Program Analysis

Lecture 1
CS 6340

Course Staff

- Instructor: Mayur Naik

  Office: KACB 2320
  Email: naik@cc.gatech.edu
  Office hours: Tue, Thu 3:30-4:30

- Teaching Assistant: Ravi Mangal

  Office: Lounge beside KACB 2320
  Office hours: Mon 4-5, Fri 11:30-12:30

Course Communication

- All class materials on course website
  - URL: http://pag.cc.gatech.edu/naik/cs6340/
  - Also linked from https://t-square.gatech.edu
- Annoucements via website and email
- Ask questions on forum in t-square
  - Preferred over email for most questions

Course Content

- Focus on *program analysis*
  - Concerns discovering facts about programs

- Both principles and practice

- Mix of basic and advanced material

Course Structure

- Lectures, from notes and research papers
  - No textbook
- Assignments (60%), of two kinds:
  - Implement program analysis algorithms in Java
  - Use program analysis tools and report findings
- In-class final exam (40%)

Course Pre-Reqs and Credits

- Pre-Reqs: None, but expect to program in Java
- Credits: 3 hours, counts toward:
  - Elective in MS CS program for specializations of “Information Security” and “DB + SE”
  - Breadth component in PhD CS program for areas of “PLC” and “Software Methodologies and Engr.”
- Auditing allowed in exceptional cases

Why Take This Course?

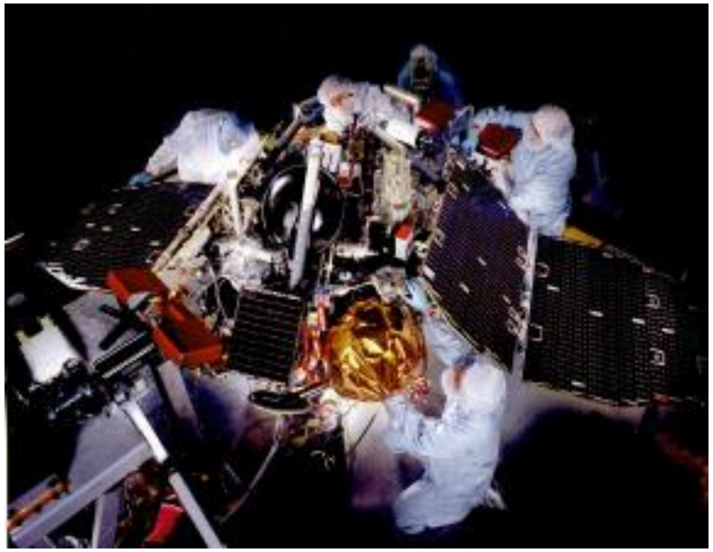
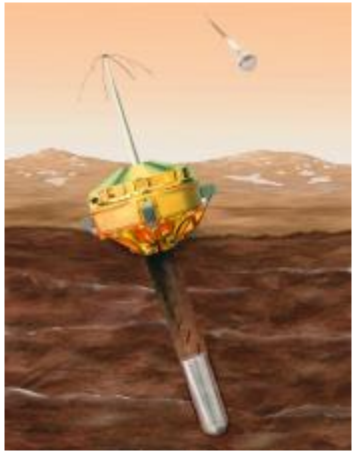
- Prepare for research in program analysis
  - This is where the field is headed
- Apply program analysis to problems in other areas (security, systems, etc.)
- Be a better software developer/tester
- For the war stories

The Ariane Rocket Disaster (1996)

Post Mortem

- Failed due to unhandled floating-point exception
- Cost
  - $100M's for loss of mission
  - Multi-year setback to the Ariane program

Mars Polar Lander (1999)

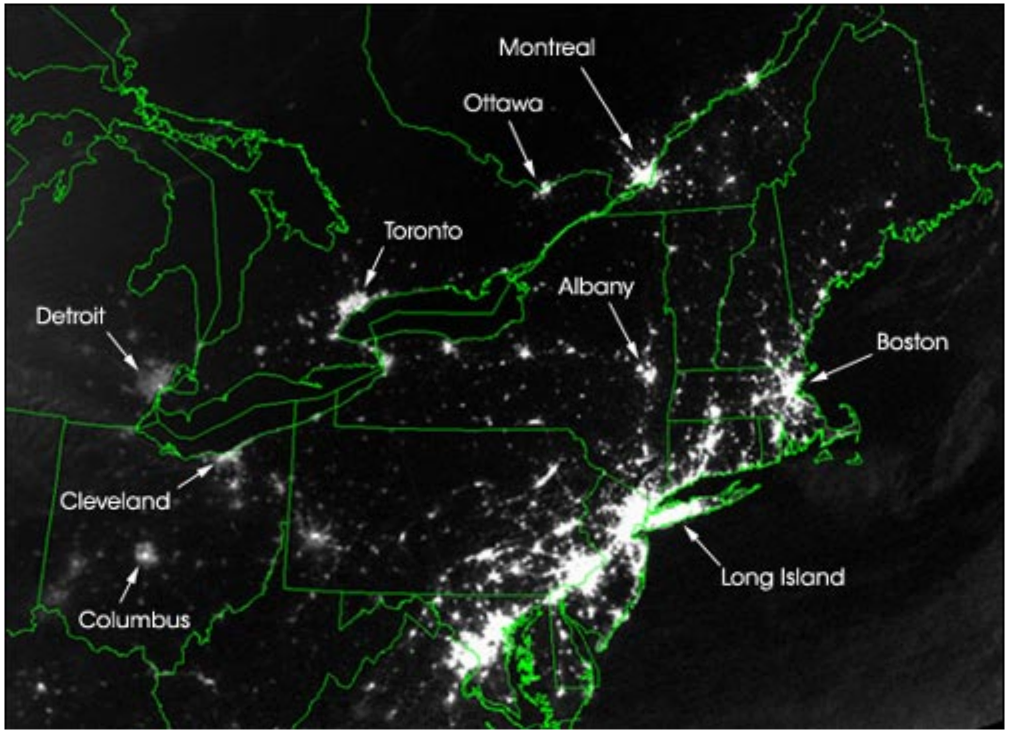



Post Mortem

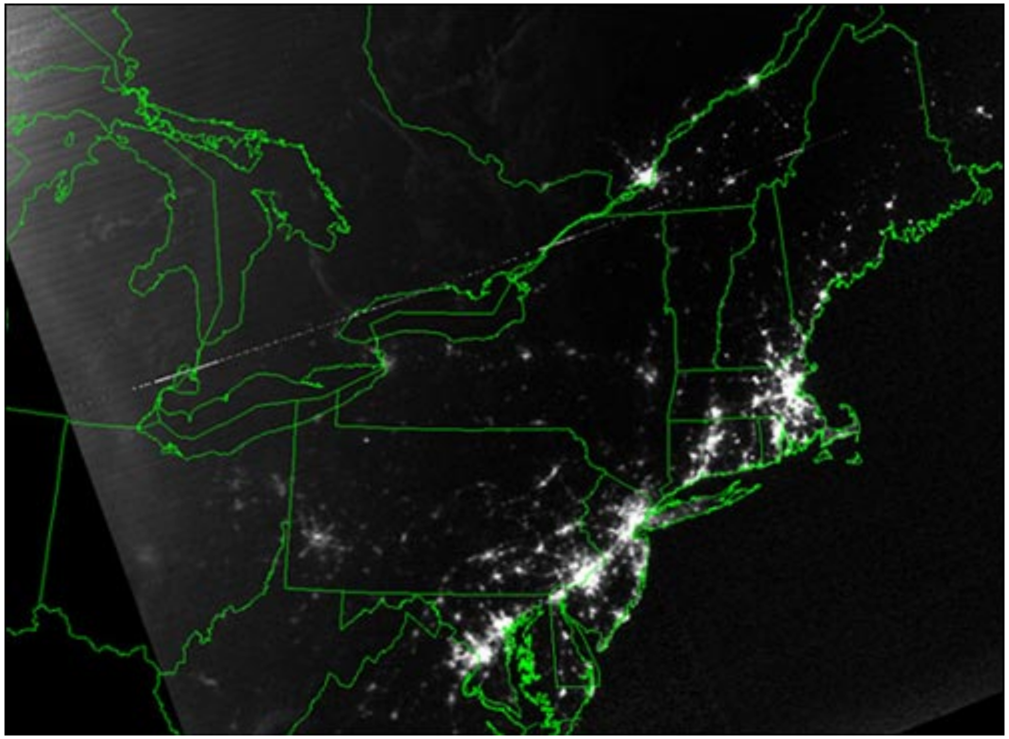
- A units problem
  - Caller expected values in inches/feet
  - Callee assumed values in meters
  - Essentially, a type error

- Total loss of $100M mission

East Coast USA



East Coast USA: 2003 Blackout

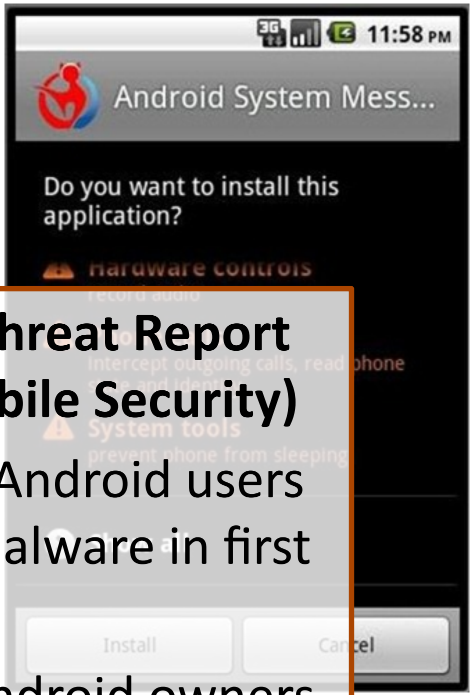


Post Mortem

- Local failure rapidly cascaded through grid
- Major contributing cause was unnoticed crash of automated alarm systems
- 10M’s of people affected

Security Vulnerabilities

- Often exploit bugs in programs
- Widespread problem
  - Code Red
  - Titan Rain
  - Moonlight Maze
  - Operation Orchard
  - Stuxnet Worm
- Getting worse …



Do you want to install this application?

Hardware controls
record audio

intercept outgoing calls, read phone

System tools
prevent phone from sleeping

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 facts about programs
- Broadly classified into three kinds:
  - Dynamic (execution-time)
  - Static (compile-time)
  - Hybrid (combines dynamic and static)
- This course will cover all three kinds

Dynamic Program Analysis

- Infer facts of program by monitoring its runs
- Examples:
  - Array bound checking
    - Purify
  - Memory leak detection
    - Valgrind
  - Datarace detection
    - Eraser
  - Finding likely invariants
    - Daikon

Static Analysis

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

An Example Static Analysis Problem

- Find variables with constant value at a given program location

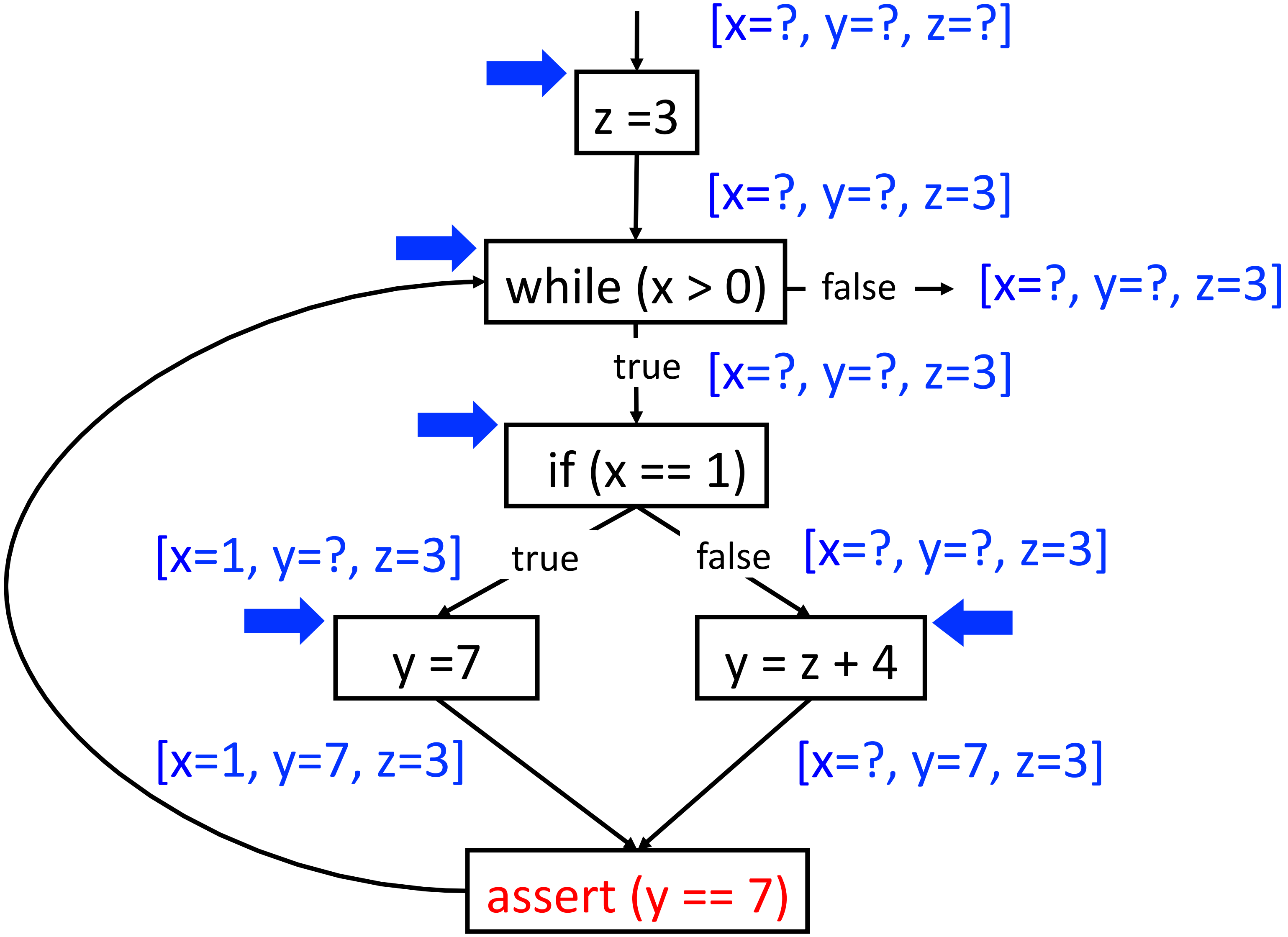
- Example program:

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

void main() {
    int z;
    if  (getc())
        z = p(6) + 8;
    else
        z = p(-7) - 5;
    printf (z);
}
```

$\boxed{z = 44}$

Iterative Approximation



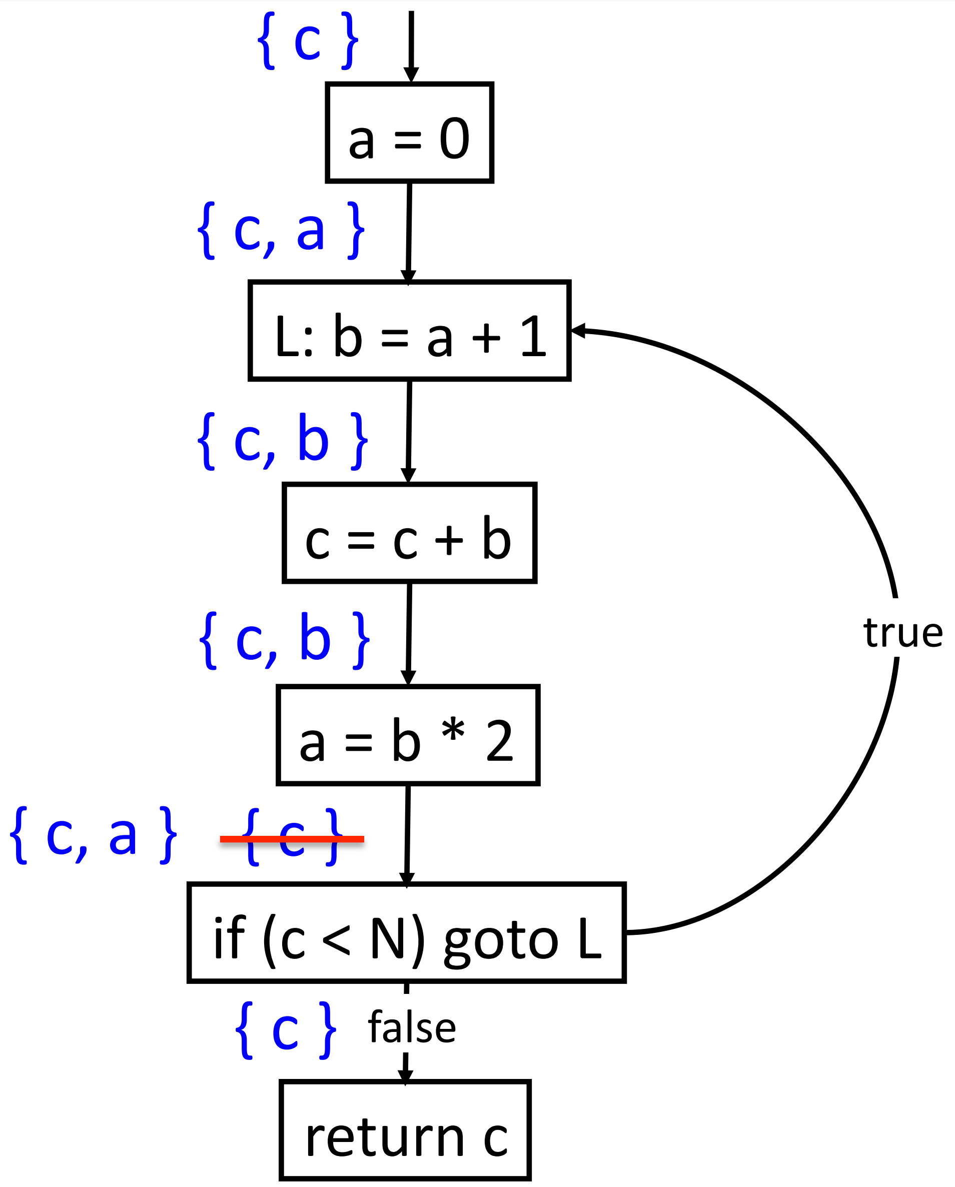
Another Static Analysis Problem

- Liveness Analysis: find which variables are live at each program point

- These are variables that are used before being set on some path from current program point

- Many applications:
  - Compilers: register allocation
  - Software Quality Tools: find uninitialized variable use

Liveness Analysis on Example Program



Other Static Analysis Problems

- Reaching definitions
- Expressions that are "available"
- Dead code
- Pointer variables never point into the same location
- Points in the program in which it is safe to free an object
- An invocation of virtual method whose address is unique
- Statements that can be executed in parallel
- Integer intervals

Dynamic vs. Static Analysis

	Dynamic	Static
Effectiveness		
Cost		

Dynamic vs. Static Analysis

	Dynamic	Static
Effectiveness	Unsound (may miss errors)	Incomplete (may report spurious errors)
Cost	Proportional to program's execution	Proportional to program's size

Undecidability of Program Properties

- Even seemingly simple program properties are undecidable
  - e.g.: is a program point reachable on some input?

=> no program analysis can be sound *and* complete

- Some properties undecidable even if program is simplified (e.g., conditionals are ignored)

Who Needs Program Analysis?

- Compilers
  - Advanced computer architectures
  - High-level programming languages (functional, OO, garbage-collected, concurrent)
- Software Quality Tools (Testing, Verification, Debugging)
  - Generate test cases
  - Find programming errors
  - Generate certification proofs
  - Localize causes of errors
- Program Understanding (e.g., IDEs)

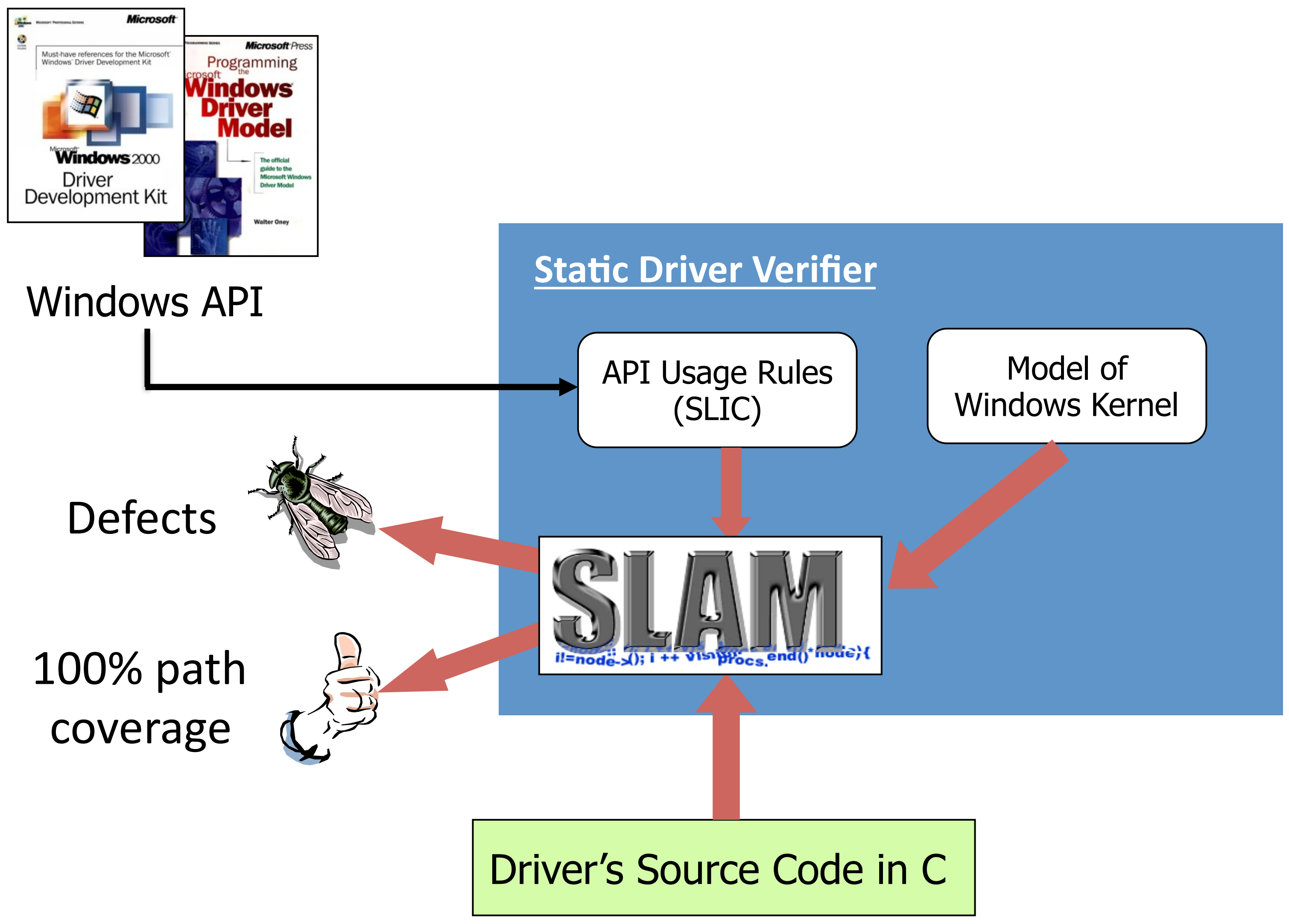
Software Quality Tools

- Detecting hazards (lint)
  - e.g.: Using uninitialized variables

```
a = malloc() ;
b = a;
free(a);
c =  malloc();
if  (b == c)
     printf(“unexpected equality”);
```

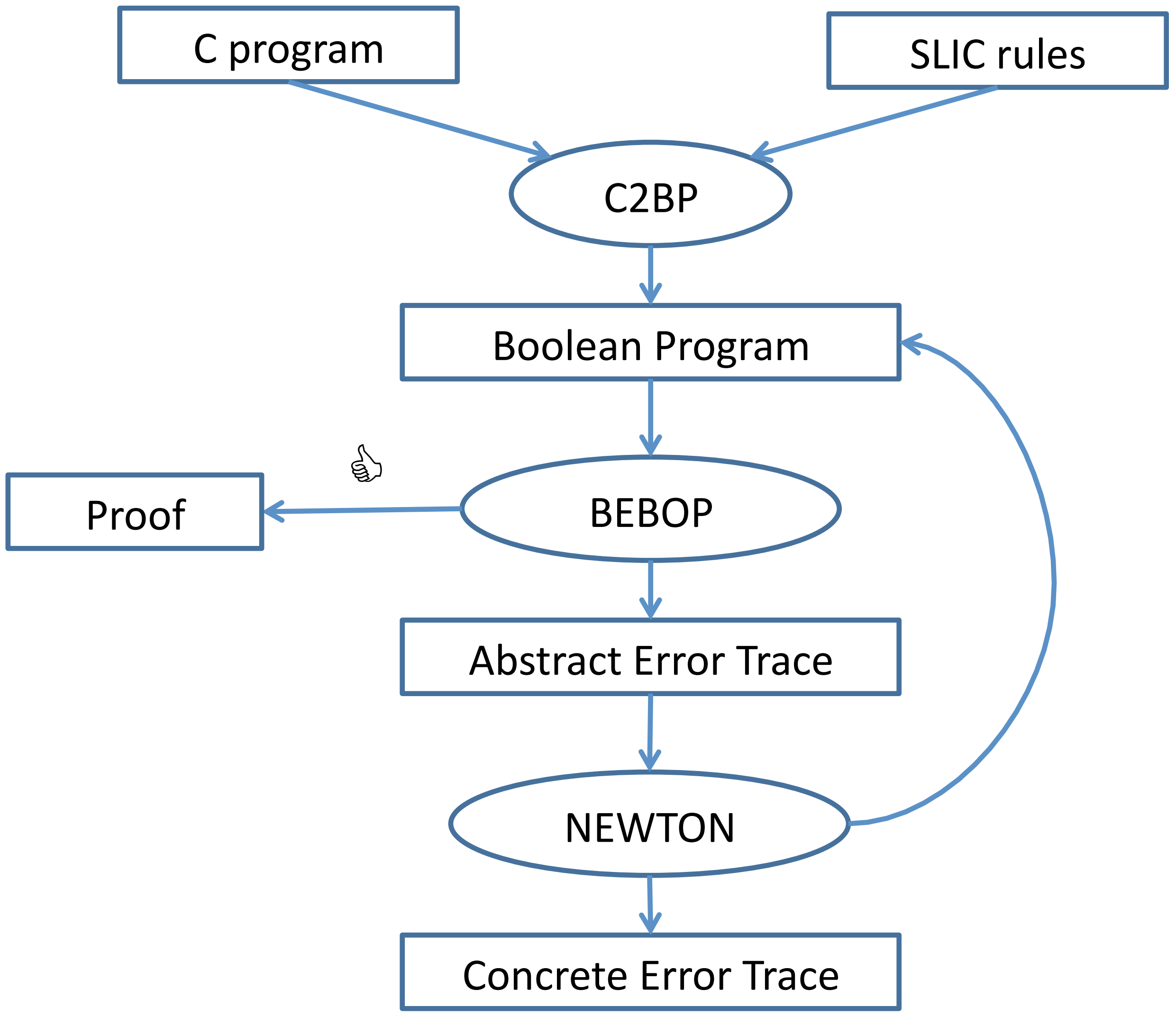
- References outside array bounds
- Memory leaks (even in Java!)

Case Study 1: Static Driver Verifier



Microsoft
Must-have references for the Microsoft Windows Driver Development Kit
Windows 2000 Driver Development Kit
Microsoft Press
Programming the Microsoft Windows Driver Model
The official guide to the Microsoft Windows Driver Model
Walter Oney
Windows API
Static Driver Verifier
API Usage Rules (SLIC)
Model of Windows Kernel
Defects
SLAM
100% path coverage
Driver's Source Code in C

Overview of SLAM Process



Bill Gates' Quote about SLAM

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability."

Bill Gates, April 18, 2002.

Keynote address at WinHec 2002

Case Study 2: ASTRÉE

- Prove absence of bugs in safety critical C code
- ASTRÉE automatically proved absence of bugs in the primary flight control software of Airbus A340's fly-by-wire system
  - Analyzed a program of 132,000 lines of C code