软件分析与验证前沿

苏亭

软件科学与技术系

Who am I?

- 苏亭(教授/博导)，软件科学与技术系，软件工程学院
- 个人主页：http://tingsu.github.io
- 研究方向
  - 软件分析、测试、验证、安全
  - 软件与系统的质量和安全保障
- 教育/工作背景
  - ECNU (B.S. & PhD) -» UCD (Visiting PhD) -» NTU (Postdoc) -» ETH (Postdoc) -» ECNU (Professor)
- 联系方式
  - 理科楼B1103，tsu@sei.ecnu.edu.cn

What is Program Analysis?

What you probably know

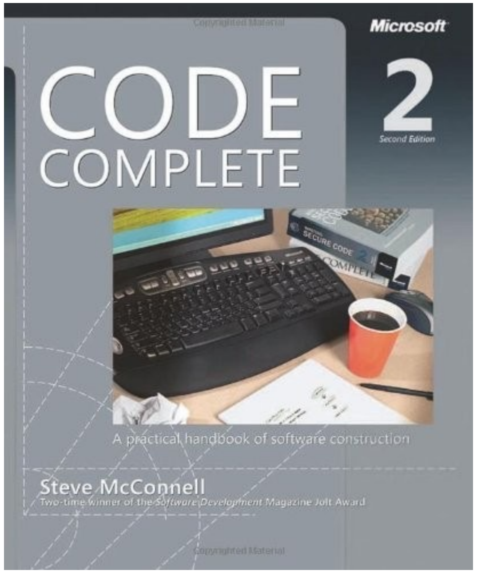
- Manual testing or semi-automated testing:
  - JUnit, Selenium, etc.
- Manual "analysis" of programs:
  - Code inspection, debugging, etc.

***Focus of this course*:**

Automated program analysis

Why Do We Need it?

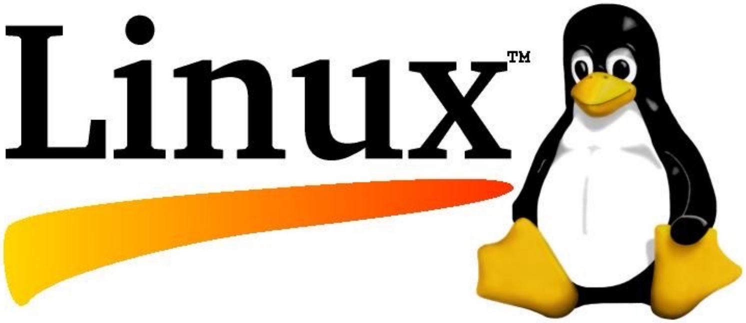
- All software has bugs
- Bugs are hard to find
- Bugs cause serious harm


Copyrighted Material
Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award
Copyrighted Material

0.5-25/KLoC in delivered software

Why Do We Need it?

- All software has bugs
- Bugs are hard to find
- Bugs cause serious harm



1.5 years to find a bug [Palix2011]

Why Do We Need it?

- All software has bugs
- Bugs are hard to find
- Bugs cause serious harm

Ariane 5

003/45/7844
ISAT GeoStar 45
23:15 EST 14 Aug. 2003

Northeast blackout

Therac-25

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
  - $370M's for loss of mission
  - Multi-year setback to the Ariane program

- Read more at https://www.bugsnag.com/blog/bug-day-ariane-5-disaster

Security Vulnerabilities

- Exploits of errors in programs
- Widespread problem
  - Moonlight Maze (1998)
  - Code Red (2001)
  - Titan Rain (2003)
  - Stuxnet Worm（蠕虫病毒，震网）
- Getting worse …

Do you want to install this application?

Hardware controls
record audio
intercept outgoing calls, read phone
System tools
prevent phone from sleeping
Install
Cancel

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?

- Discover useful *facts* about programs
- Broadly classified into three kinds:
  - Static (compile-time)
  - Dynamic (execution-time)
  - Hybrid (combining dynamic and static)

Static vs. Dynamic Analysis

- **Static**
  - Infer facts by inspecting *source or binary code*
  - Typically:
    - Consider *all* inputs
    - *Overapproximate* possible behavior

E.g., compilers, lint-like tools

- **Dynamic**
  - Infer facts by monitoring *program executions*
  - Typically:
    - Consider *current* input
    - *Underapproximate* possible behavior

E.g., automated testing tools, profilers

Example

```
//JavaScript
var r = Math.random();//value in [0,1)
var out = "yes";
if(r < 0.5)
  out = "no";
if(r === 1)
  out = "maybe";
console.log(out);
```

What are the possible outputs?

Example

```
//JavaScript
var r = Math.random();//value in [0,1)
var out = "yes";
if(r < 0.5)
  out = "no";
if(r === 1)
  out = "maybe";//infeasible path
console.log(out);
```

Overapproximation: ”yes”, ”no”, ”maybe”

- Consider all paths (that are feasible based on limited knowledge)

Example

```
//JavaScript
var r = Math.random();//value in [0,1)
var out = "yes";
if(r < 0.5)
  out = "no";
if(r === 1)
  out = "maybe";//infeasible path
console.log(out);
```

Underapproximation: ”yes”

- Execute the program once

Example

```
//JavaScript
var r = Math.random();//value in [0,1)
var out = "yes";
if(r < 0.5)
  out = "no";
if(r === 1)
  out = "maybe";//infeasible path
console.log(out);
```

Sound and complete: ”yes”, ”no”

□ For this example: Can explore both feasible paths

Another Example

```
//JavaScript
var r =Math.random();//value in [0,1)
var out = r *2;
console.log(out);
```

What are the possible outputs?

Another Example

```
//JavaScript
var r =Math.random();//value in [0,1)
var out = r *2;
console.log(out);
```

Overapproximation: Any value

- Consider all paths (that are feasible based on limited knowledge about `random()`)

Another Example

```
//JavaScript
var r =Math.random();//value in [0,1)
var out = r *2;
console.log(out);
```

Underapproximation:

Some number in [0,2), e.g., 1.234

- Execute the program once

Another Example

```
//JavaScript
var r =Math.random();//value in [0,1)
var out = r *2;
console.log(out);
```

Sound and complete?

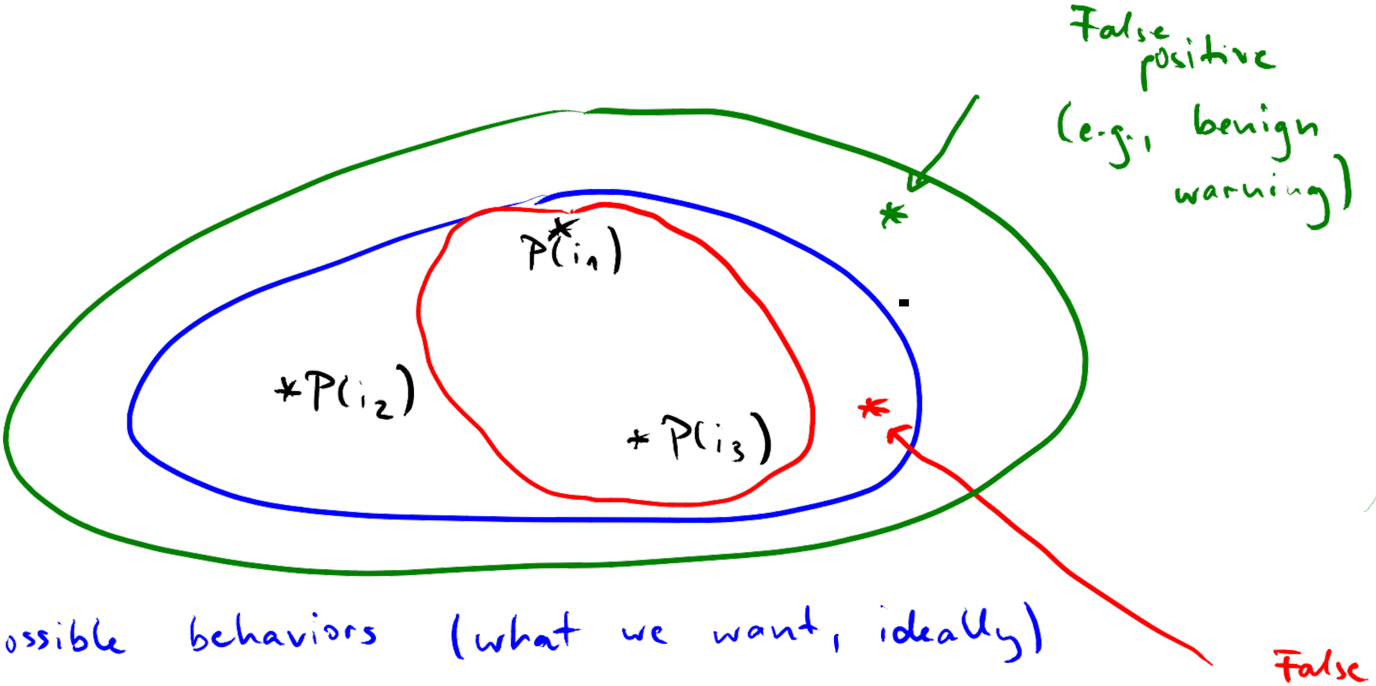
- Exploring all possible outputs:
  Practically impossible
- This is the case for most real-world programs

Terminology

- Over-approximation *v.s.* Under-approximation
- False positives *v.s.* False negatives
- Soundness *v.s.* Completeness
- Precision *v.s.* Recall

Under- & Over-approximation

Program P, Input i, Behavior P(i)



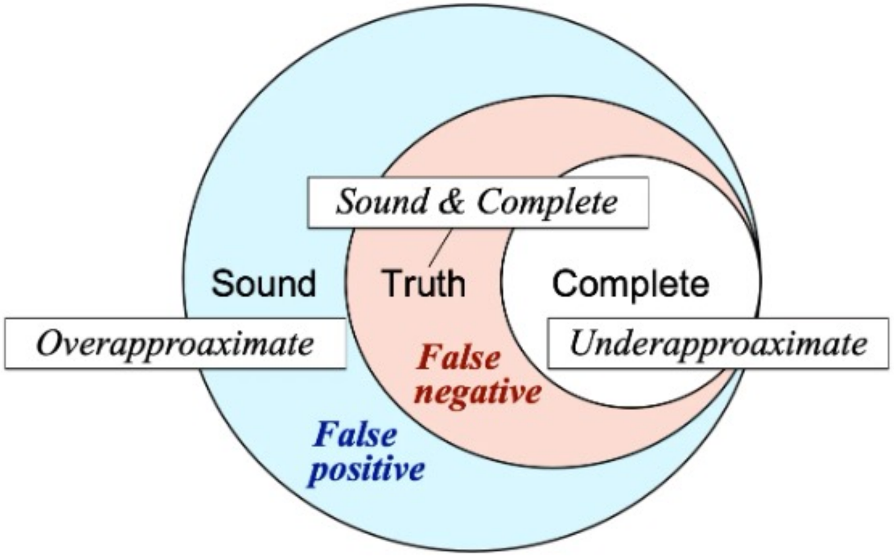
False negative (e.g., missed bugs)

All possible behaviors (what we want, ideally)

Underapproximation (e.g., testing, dynamic analysis)

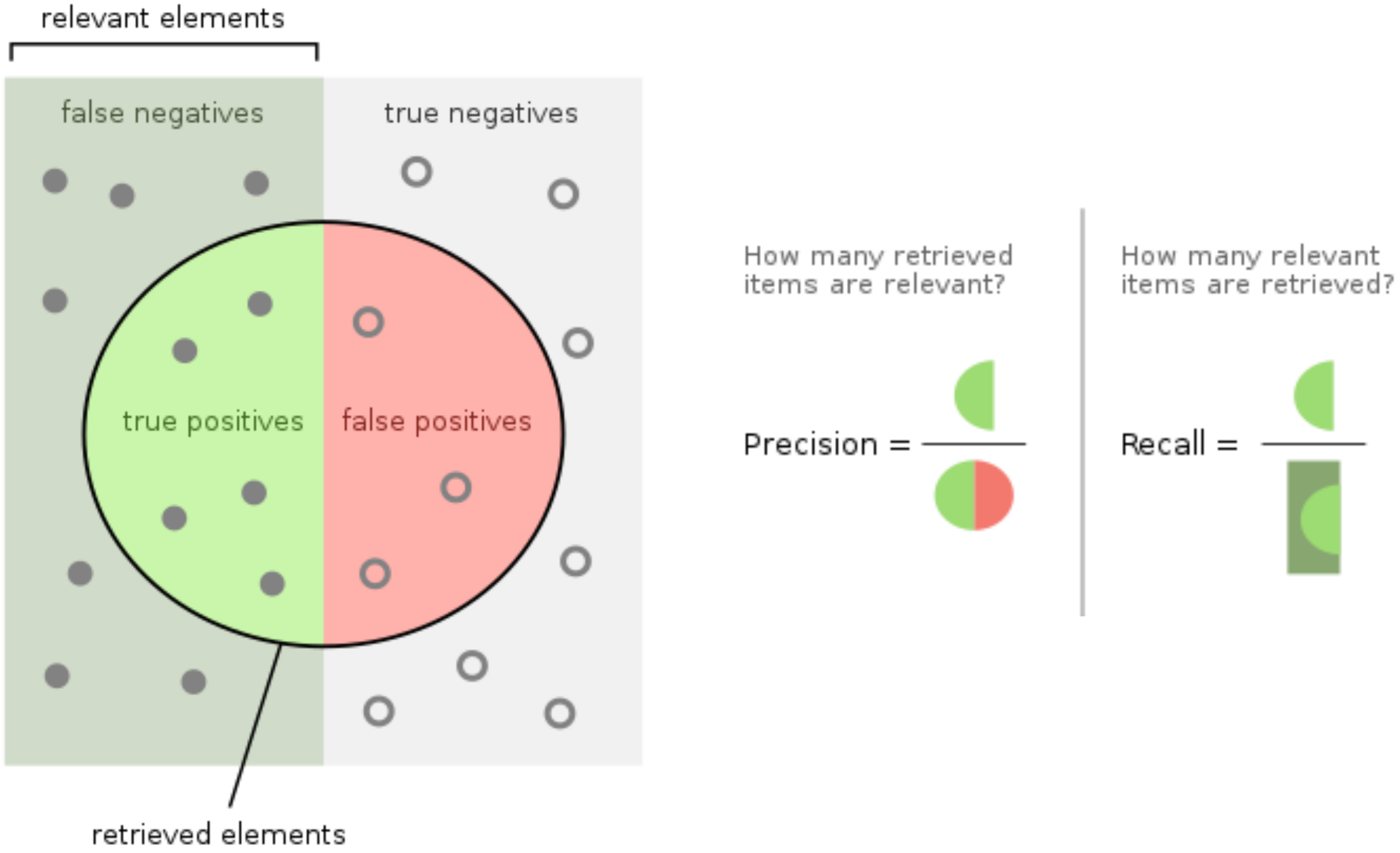
Overapproximation (most static analysis)

Soundness & Completeness



- Truth: all possible behaviors
- 按照程序设计语言领域中主流文献与资料
  - Soundness -> 妥协soundness, 我们会有false negatives
  - Completeness -> 妥协completeness，我们会有false positives
- 按照软件工程等研究领域的资料和文献
  - Sound (欠近似) -> 无误报
  - Complete (过近似) -> 无漏报

Precision & Recall



Example: 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 = ?


}
```

Example: 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

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

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 analysis, testing and verification
- Finding Jobs & Do research

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 analysis, testing and verification

- Finding Jobs & Do research

华为、阿里巴巴（蚂蚁金服）、腾讯、字节、网易、美团、中国航天研究院、中国电信研究院、国家电网研究院……

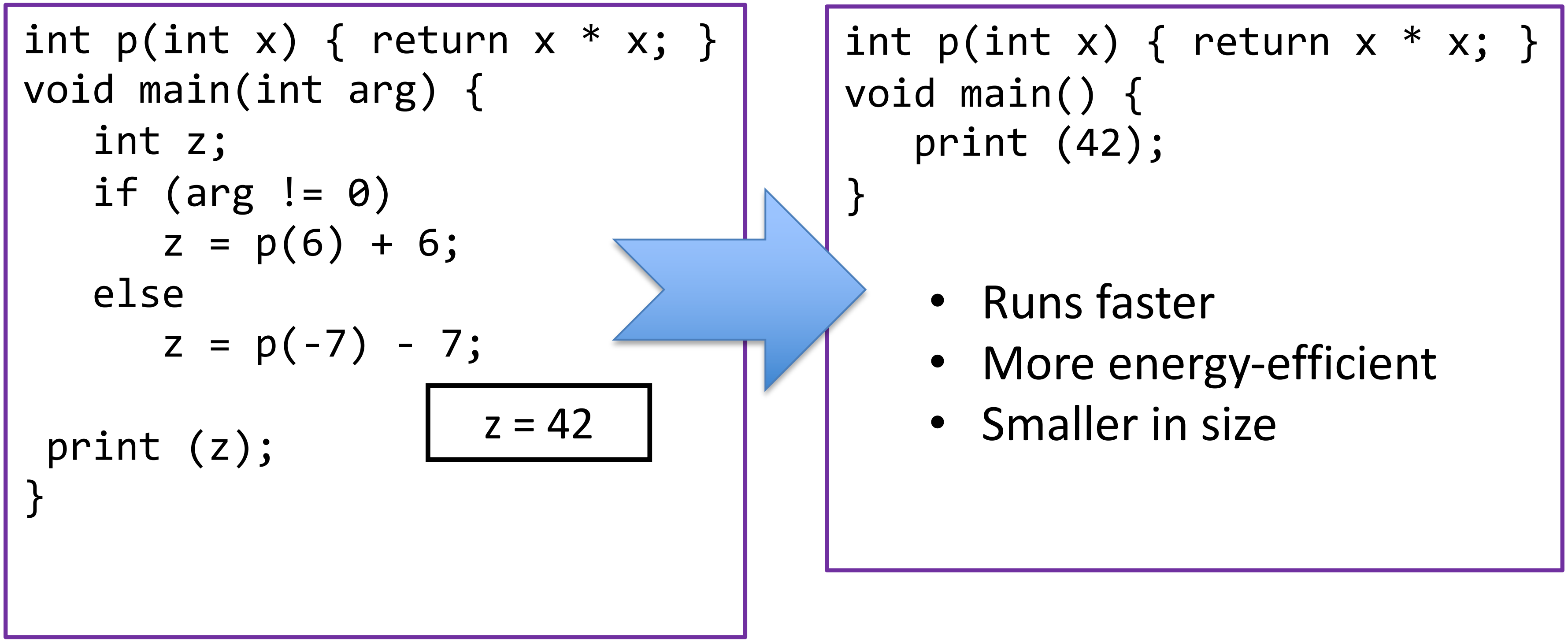
Who Needs Program Analysis?

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

Example: Software Quality Tools

- Static Program Analysis

Suspicious error patterns
Lint, SpotBugs, Coverity

Memory leak detection
Facebook Infer

Checking API usage rules
Microsoft SLAM

Verifying invariants
ESC/Java

The Coverity Platform - From a Developer's Perspective:
https://www.youtube.com/watch?v=_Vt4niZfNeA

Example: Software Quality Tools

- Dynamic Program Analysis

Array bound checking
Purify

Datarace detection
Eraser

Memory leak detection
Valgrind

Finding likely invariants
Daikon

Integrated Development Environments

- Examples: Eclipse and VS Code

- Use program analysis to help programmers:
  - Understand programs
  - Refactor programs
    - Restructuring a program without changing its behavior

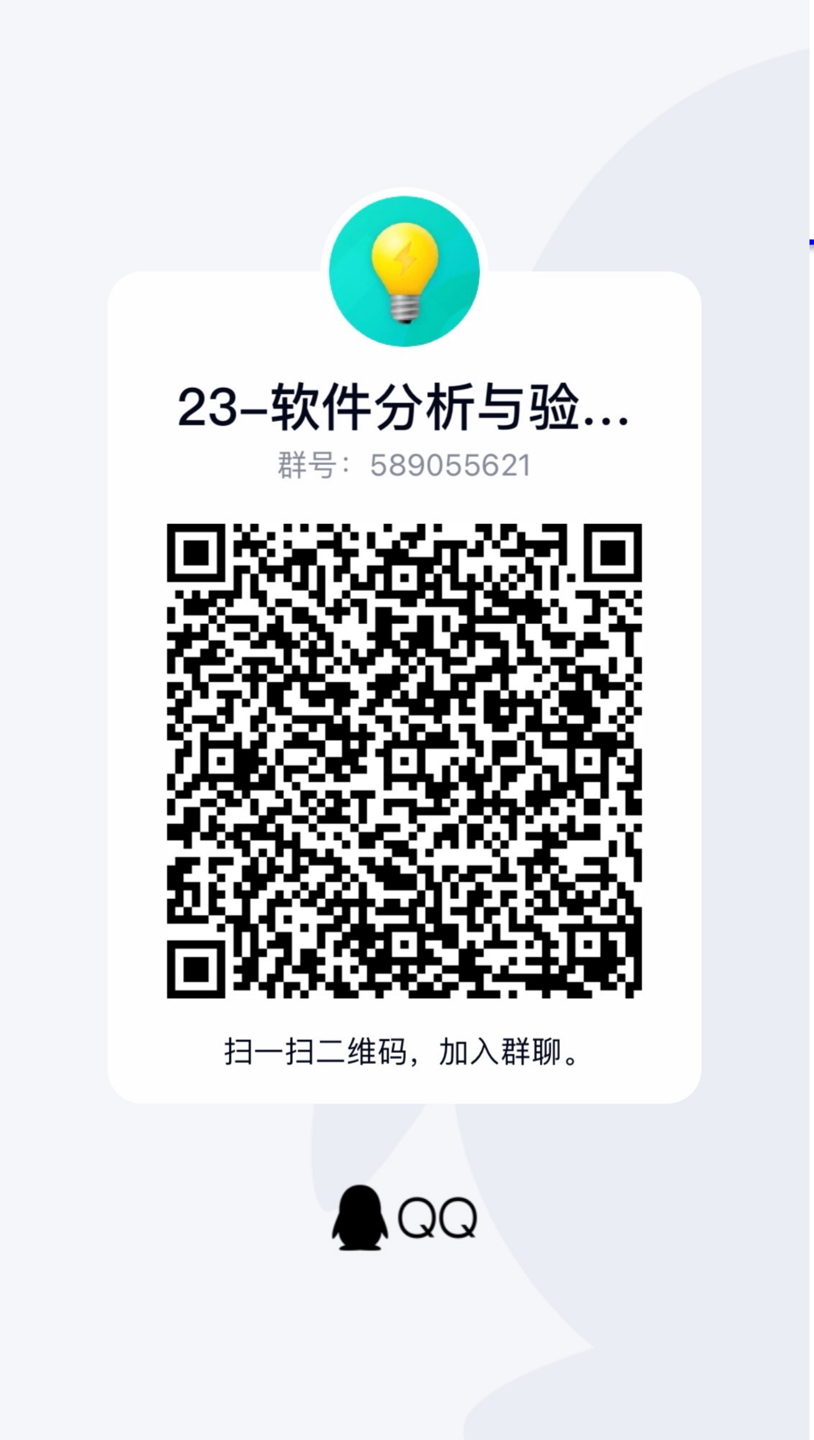
- Useful in dealing with large, complex programs

Course Information

- 课程目标
  1. 掌握软件分析、测试与验证的基本理论和技术原理
  2. 了解相关的前沿研究进展
- 课程信息
  1. 理论课: 每周二下午第9-10节课（下午14:50-16:25）
  2. 课程讲义：大夏学堂
  3. 上课地点: 教书院226
  4. 考核形式: 出勤*20%、平时课堂表现*30%、课程项目（形式: 阅读研究论文、工具调研等）*50%

课程网站: https://tingsu.github.io/files/courses/pa2023.html (TODO)

助教: 姜嘉仪



23–软件分析与验…

群号：589055621

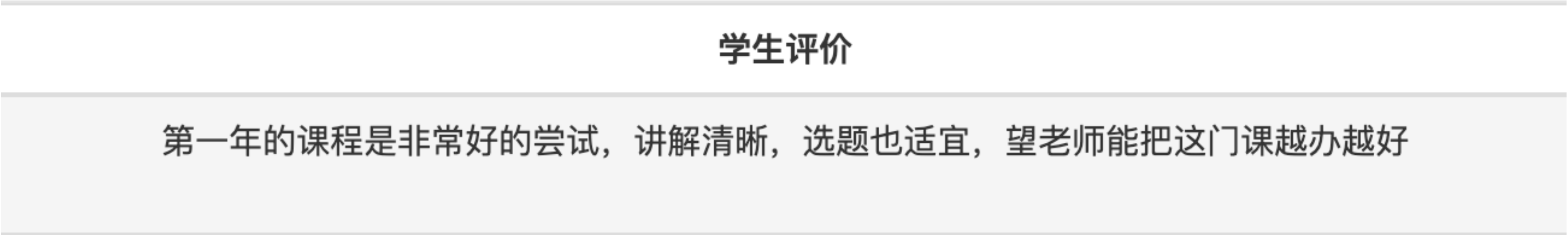
扫一扫二维码，加入群聊。

Course Topics (Tentative)

- Data-flow Analysis
- Pointer Analysis
- Formal verification (model checking)
- Random Testing & Fuzzing
- Symbolic Execution
- Metamorphic & Property-based Testing
- Security Analysis
- Delta debugging
- ……

Course History

- Pre 2022 - 软件分析与验证工具 (郭建)
- 2022- 软件分析与验证前沿
- ……



学生评价

第一年的课程是非常好的尝试，讲解清晰，选题也适宜，望老师能把这门课越办越好

Supplementary Materials

- Mayur Naik (University of Pennsylvania)
- Michael Pradel (University of Stuttgart)
- 南京大学（李樾、 谭添老师）的程序分析课程
- 北京大学（熊英飞老师）的程序分析课程（本科）
- 国防科大（陈立前老师）的程序分析课程

- Static Program Analysis, Anders Møller and Michael I. Schwartzbach https://cs.au.dk/~amoeller/spa/

What Have We Learned?

- What is program analysis?
- Dynamic vs. static analysis: pros and cons
- Terminologies in program analysis
- Undecidability => program analysis cannot ensure termination + soundness + completeness
- Why we need to learn program analysis?

Additional Links

- **What is soundness (in static analysis)?**

  - http://www.pl-enthusiast.net/2017/10/23/what-is-soundness-in-static-analysis/

- **What is static program analysis?**
  - https://matt.might.net/articles/intro-static-analysis/

- **Precision and Recall**
  - https://en.wikipedia.org/wiki/Precision_and_recall