

Lecture 0 – Introduction

SWS128: Software Analysis

Jihyeok Park



2026 Fall

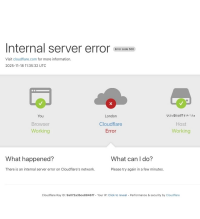
- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Associate Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office:** 507, 정운오IT교양관
 - **Email:** jihyeok_park@korea.ac.kr
- **Class:** SWS128: Software Analysis
- **Lectures:** 20:20–21:50 Mondays @ B101, 우정정보관
- **Homepage:** <https://plrg.korea.ac.kr/courses/sws128/>

#	Date	Title
0	09/07	Introduction
1	09/14	Syntax and Semantics
2	09/21	Control Flow Graphs
3	09/28	Dataflow Analysis
4	10/05	Taint Analysis (recorded)
5	10/12	Abstract Interpretation (recorded)
10/19		Midterm Week (No Class)
6	10/26	Widening and Narrowing
7	11/02	Pointer Analysis
8	11/09	Inter-procedural Analysis
9	11/16	Constraint-based Analysis
10	11/23	Symbolic Execution
11	11/30	Dynamic Analysis
12	12/07	Real-World Tools and Practice
12/14		Final Week (No Class)

No offline class on **October 5** and **October 12**. Recorded lectures will be uploaded to the [LMS](#) instead.

- **3 Homework Assignments: 90%** (30% each)
 - Submitted on the [LMS](#).
 - Each one is due two weeks after it is handed out.
- **Attendance: 10%**
 - All or nothing: the full 10% with at least 2/3 of the checks, and 0% otherwise.
 - No attendance check today; it starts on September 14.
- No midterm, no final, no term project.

Unexpected faults in software cause serious problems:

 June 4, 1996: Ariane-5 explodes after lift off Today in History: June 4, 1996: Ariane-5 explodes when lift off Facebook, June 05, 2016 at 10:42 Ariane 5, Ariane 5, Ariane 5			
Rocket Ariane 5 (1996)	Security Heartbleed (2014)	Global IT CrowdStrike (2024)	Cloud Cloudflare (2025)

Can we **automatically check** whether a program has a fault?

How do we know whether a software is correct?



VS.



Empiricists (경험주의자)

Francis Bacon

*It is correct because I **TESTED**
several times but no error was found!*

Rationalists (합리주의자)

René Descartes

*It is correct because I formally
PROVED that no error exists!*

An **analyzer** takes a program and a property, and decides whether the program satisfies the property.



- **Dynamic analysis** — by executing the program.
- **Static analysis** — without executing it.

Dynamic Analysis	Static Analysis
Empiricists (경험주의자)	Rationalists (합리주의자)
Under-approximation	Over-approximation
False Negatives (Missed Errors)	False Positives (False Alarms)

Every non-trivial semantic property of programs is
undecidable.

— Rice's Theorem (1953)

- No analyzer can be sound (no missed errors), complete (no false alarms), and always terminate.
- So every analysis gives something up. This course is about **which**, and **how**.

To learn the **methodologies** for analyzing software, and to **implement** them.

You will learn how to:

- define what a program **means**, and build its **control flow graph**;
- compute facts about **all executions** at once (dataflow analysis, abstract interpretation);
- track **where data flows** (pointer analysis, taint analysis);
- find an **input** that reaches a given point (symbolic execution, dynamic analysis);
- and judge how far each result can be **trusted**.

- Syntax and Semantics

Jihyeok Park

`jihyeok_park@korea.ac.kr`

`https://plrg.korea.ac.kr`