Foundations of Software Science (w1)

Kazutaka Matsuda
Eijiro Sumii

Today's Topic

▶ General information about this lecture

▶ A quick tour (of my part)

▶ Polls

▶ Reviews of preliminary knowledges (on blackboard)

About This Lecture ...

▸ Foundations of Software Sciences

- Part 1: Foundations of Functional Programs and Logics
  - taught by me (Kazutaka Matsuda)

- Part 2: Foundations of Concurrent Programs
  - taught by Prof. Sumii

Lecture Style

- ▸ *I* sometimes use slides
  - *I* will upload the materials *after* the class
    - Don't copy the slides in the class
- ▸ *I* also use the blackboard too
  - Still, don't copy the blackboard
    - taking notes ≠ copying the blackboard
- ▸ *We* will distribute handouts if necessary
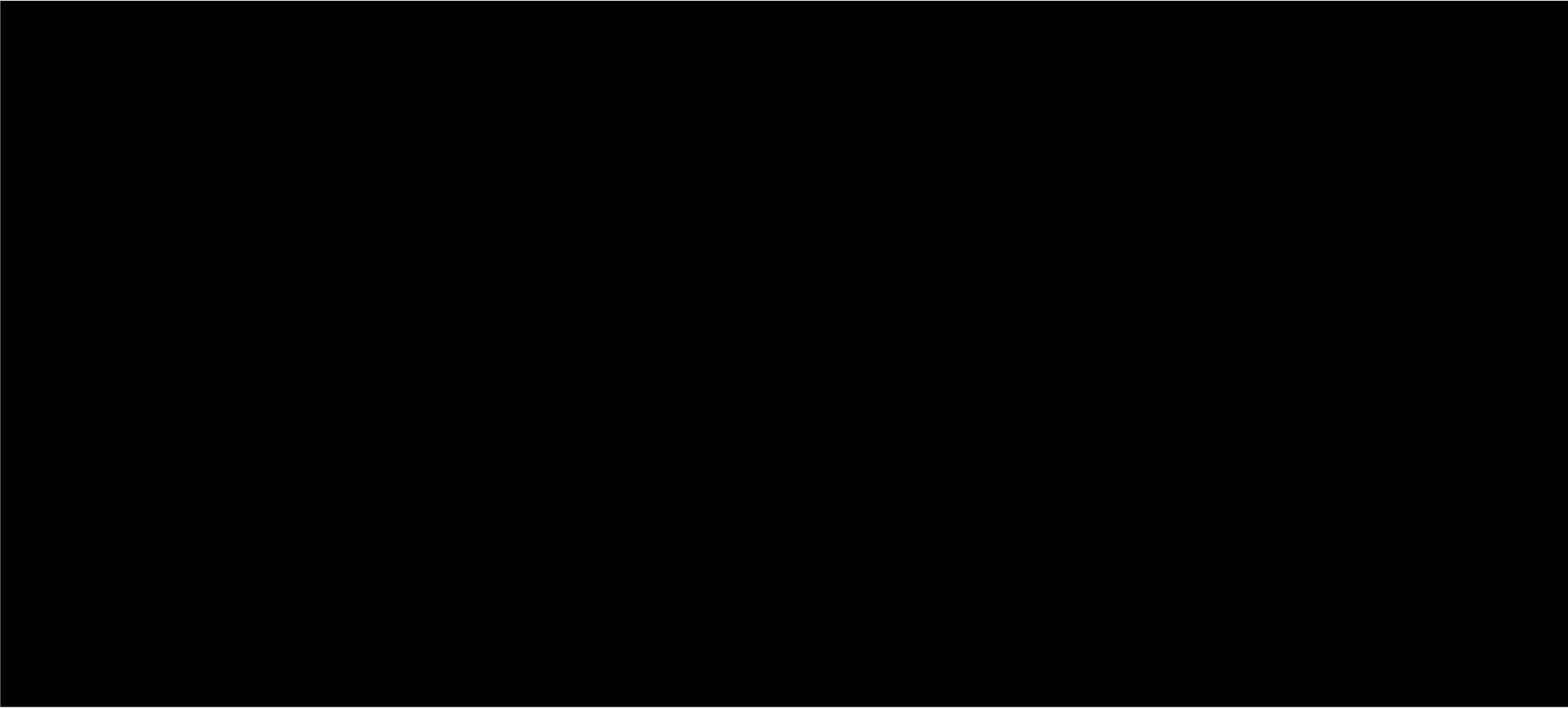 - Such handouts may or may not be uploaded

Notes on Handouts

▸ They may **_not_** be uploaded
▸ They usually be distributed at the beginning of a class
  - If you happen to be late, you may lose a chance to get a handout
▸ Redistribution is not allowed

Textbooks

▸ You ***don't have to*** buy any textbooks ***for my part***
- *I* will distribute handouts if necessary

▸ However, some textbooks ***help*** your understanding a lot

Further Reading (1)

▸ M.H. Sørensen & P. Urzyczyn: Lectures on the Curry-Howard Isomorphism, Elsevier
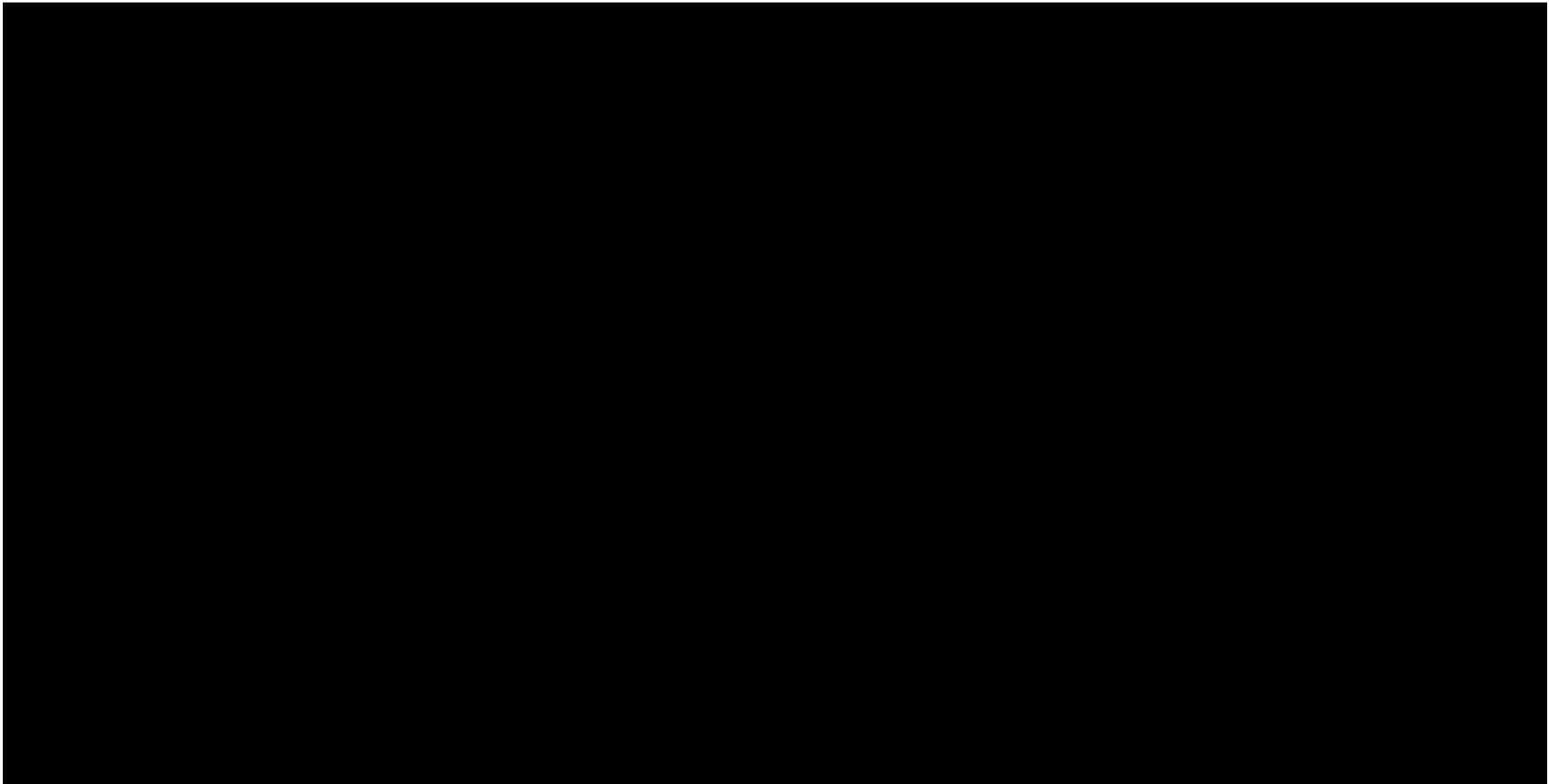
Further Reading (2)

▸ H.P. Barendregt: The Lambda Calculus. Its Syntax and Semantics, Elsevier

Further Reading (3)

▸ J.-Y. Girard, Y. Lafont, P. Taylor:
Proofs and Types

  • http://www.paultaylor.eu/stable/Proofs+Types.html

Further Reading (4)

▸ 高橋正子：計算論. 計算可能性とラムダ計算，近代科学社

Evaluation

‣ Mainly by examinations
  - We will have two examinations
    – for my part
    – for Sumii's part

‣ Partly, by reports
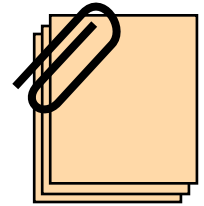 - **_Today, you have report assignments_** (see the handout)

Notes on Reports and Exams

Your Goal

▸ To *tell* us "*you understood* asked points"
- • We are NOT (only) asking correct answers
- • It's a sort of communication

You ← I understood the points!
Give me a credit!

NG: I cannot *believe* it (you may fail the course).

or

OK: I *am convinced* that you really did it! ⟩ Me

One thing you must not to do

▸ Plagiarism

plagiarism | ˈpleɪdʒərɪz(ə)m |

noun [mass noun]
the practice of taking someone else's work or ideas and passing them off as one's own. *there were accusations of plagiarism.* [count noun] : *it claims there are similar plagiarisms in the software produced at the university.*

from Oxford Dictionary of English via 辞書 2.2.1 (Mac App).

- Clarify the origins if you borrow somethings from someone else.
- See [https://www.indiana.edu/~tedfrick/plagiarism/] for details

A Quick Tour
(of My Part)

Background

▸ Software is everywhere
  - PCs
  - servers, clouds
  - game consoles
  - mobile phones
  - medical machines
  - cars
  - plants

Background

▸ Software *bugs* are everywhere
- PCs
- servers, clouds
- game consoles
- mobile phones
- medical machines
- cars
- plants

"History's Worst Software Bugs" from WIRED
http://archive.wired.com/software/coolapps/news/2005/11/69355

SOFTWARE : COOL APPS

History's Worst Software Bugs

Simson Garfinkel ✉   11.08.05

Last month automaker Toyota announced a recall of 160,000 of its Prius hybrid vehicles following reports of vehicle warning lights illuminating for no reason, and cars' gasoline engines stalling unexpectedly. But unlike the large-scale auto recalls of years past, the root of the Prius issue wasn't a hardware problem -- it was a programming error in the smart car's embedded code. The Prius had a software bug.

With that recall, the Prius joined the ranks of the buggy computer -- a club that began in 1945 when engineers found a moth in Panel F, Relay #70 of the Harvard Mark II system.The computer was running a test of its multiplier and adder when the engineers noticed something was wrong. The moth was trapped, removed and taped into the computer's logbook with the words: "first actual case of a bug being found."

Sixty years later, computer bugs are still with us, and show no sign of going extinct. As the line between software and hardware blurs, coding errors are increasingly playing tricks on our daily lives. Bugs don't just inhabit our operating systems and applications -- today they lurk within our cell phones and our pacemakers, our power plants and medical equipment. And now, in our cars.

But which are the worst?

It's all too easy to come up with a list of bugs that have wreaked havoc. It's harder to rate their

Examples of Terrible Bugs

▶ Therac-25 (1985-1987)
  - 6 patients were severely injured
▶ Ariane 5 (1996)
  - A space rocket exploded

▶ Tokyo Stack Exchange (2006)
  - Mizuho lost 41.6 billion yen
  - TSE was ordered to pay 10.7 billion yen

Goal

▸ How to avoid bugs in softwares?
  - by skilled programmers?
  - by sophisticated software development processes?
  - by a lot of testing cycles?

  - **by mathematically-founded ways** for **bug-free** softwares

Topics

▸ How to model software and programs
▸ How to state bug-freeness
▸ How to prove bug-freeness

Main Topics of My Part

‣ λ calculus
  - a model of computation
  - a model of functional programs

‣ Curry-Howard correspondence
  - correspondence between
    - programs
    - proofs

(Untyped) λ-calculus

▸ One of the simplest functional
programming languages
  - only functions and applications

```
M ::= x
    |  M₁ M₂
    |  λx.M
```

"fn x => M" in SML

▸ A model of computation
  - can represent all computable functions

23

Simply-Typed λ-Calculus

▸ One of the simplest typed functional programming language

▸ Type safety
  • No (a certain sort of) bugs
  • Proved by subject reduction + progress

▸ Strong normalization
  • Every typed program terminates

Curry-Howard Correspondence

▸ A program has a type P

  ≡ P holds in a certain logic

- Writing proofs is programming!
  - Simply-typed λ calculus
    for (intuitionistic) propositional logic
  - Coq (based on CoC)
  - Agda (based on Martin-Löf Type Theory)
- Well-typed proofs are valid
- Strong normalization is a key to consistency

Polls