

CS339: Abstractions and Paradigms for Programming

Introduction and Logistics

Manas Thakur
CSE, IIT Bombay



Autumn 2024

Natural Languages

- Name the languages that you can speak/read/write.
- A unique advantage of living in India?
 - *The APP course in no other country can start with this slide!*
- Have you tried learning a new language?
- How different was that language from those you already knew?
- What was your strategy?



Programming Languages

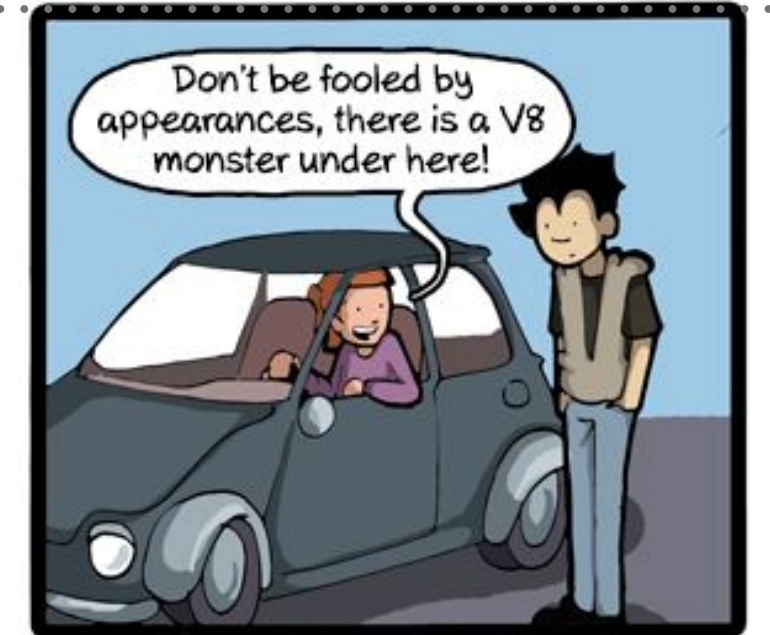
- Name the PLs that you know.
 - What do we mean by *know*?
- Name an English letter that's not a PL.
- Why do we have so many PLs?
- How did so many PLs evolve?
- Or aren't they so many, but just syntactic sugars over a set of fundamental *computational* constructs?

If languages were cabs

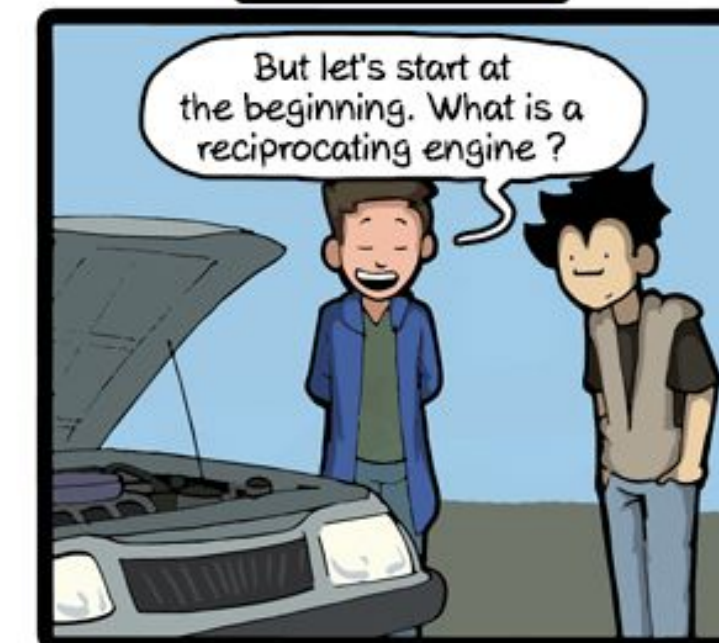
PHP



Javascript



Assembly



JAVA



Objective-C



Open Source

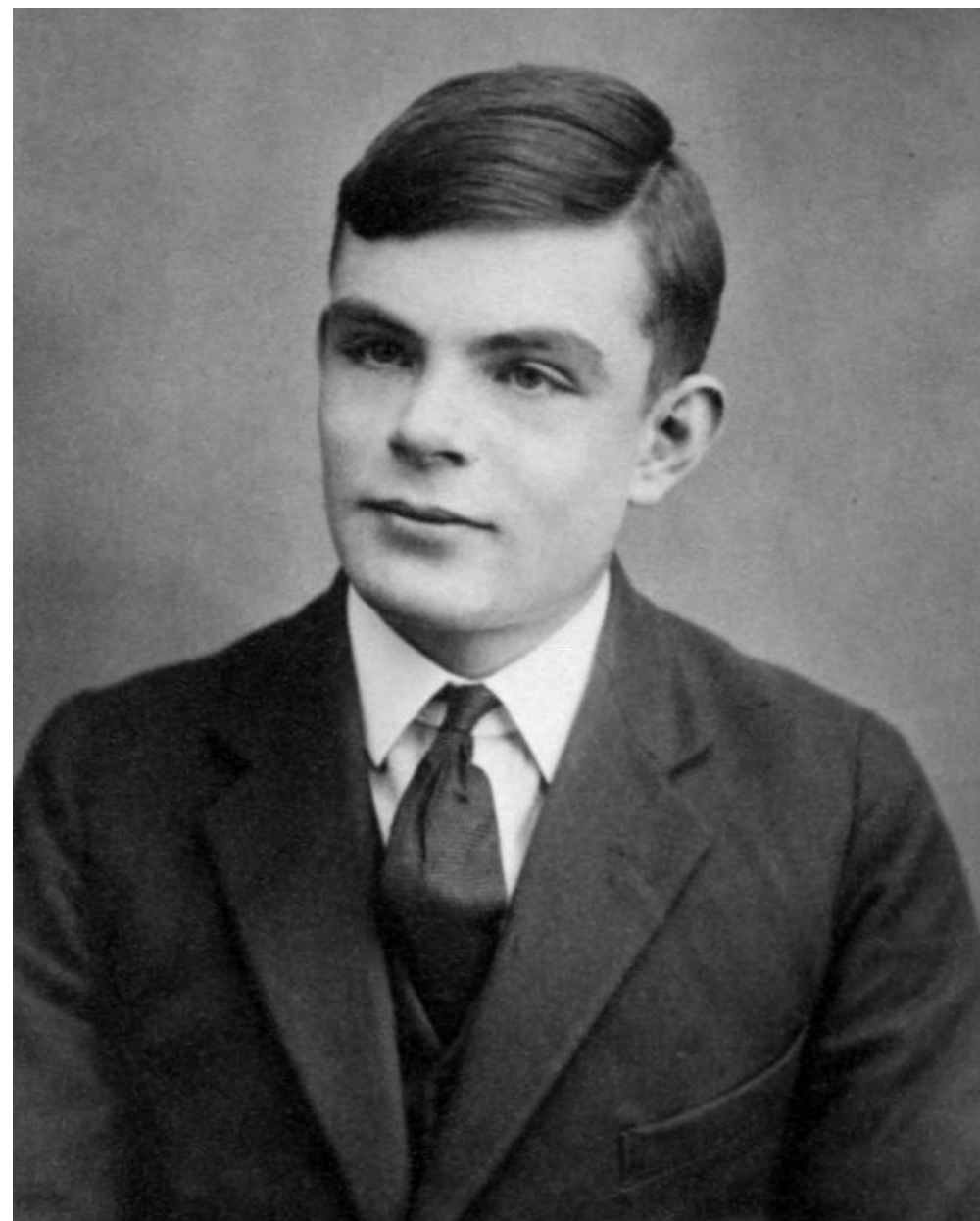


CommitStrip.com



Standing on the shoulders of giants

- Who are these people?



Alan Turing



Alonzo Church

- Hints:
 - Both were computer scientists.
 - The most famous award in CS is named after the LHS.
 - RHS was the PhD advisor of the LHS.

Two models of computation

- The **Turing Machine** (1936) performs computations by:

- Reading input
- Modifying an internal memory using instructions
- Producing output

If you don't remember TOC,
think of a computer that you are used to!!

- The **Lambda Calculus** (1936) performs computations by:

- Evaluating expressions

We would learn more of it in this course.

- **Church-Turing Thesis** (1937): Both the models, and all other reasonable computation models, are equivalent!



Abstractions and Paradigms for Programming

- **WHAT** is the square root of a number X ?

A number Y such that $Y * Y == X$.

- **HOW** to compute the square root of a number X ?

- Make a guess Y .
- Improve the guess Y until its square is near enough to X .

A SQRT “procedure”

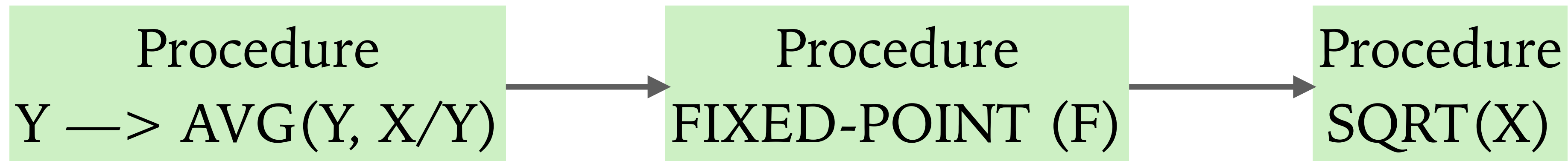


Procedural Abstraction

- Can we reuse the SQRT procedure as a blackbox?

Compute $\text{SQRT}(A) + \text{SQRT}(B)$.

- Can we consume and produce procedures?



Functional
Paradigm

- Functions become real entities in our programs!



Data Abstraction

- Say we have a general procedure to add two things and multiply by another:

Procedure
 $(X + Y) * Z$

- Can X, Y, Z be
 - numbers?
 - vectors?
 - polynomials?
 - signals?

Object-Oriented
Paradigm

Types &
Proofs

Stream
Processing

Language Abstraction

- Can we design a language that gives us higher level primitives?
 - vectors? matrices? tensors?
 - images? videos?
- Can we change an existing language to work differently?
 - lazy evaluation instead of eager?
 - automatic allocation instead of explicit types?
- Can we extend a language to support something more?
 - C to C++?

Logic
Paradigm

Interpreter
Design



Abstractions and Paradigms for Programming

- Computation is much more fundamental (and older) than computers or computer science, and is very much about specifying the right abstractions.
- There are several paradigms (aka *ways*) of specifying computations.
- But not as many as there are programming languages.
- There is no way we can teach you the PL that you can keep using for the next 30-40 years. But if we teach you the ways of specifying and abstracting computations, and equip you with the skills to design your own PLs, you would have learnt enough to become a **successful computing engineer**.



The APP Course

- Empowers us to **choose** the right PL for the task at hand;
- Makes **learning** a new PL much easier;
- Equips us in **designing** a new PL and its interpreter;
- Is huge **fun**!

Side learnings:

- Ability to comprehend large programs
- Language technologies that everyone uses
- Crux of **future** programming languages
- Different *right* ways of thinking about the same thing
- Motivation to learn the art of **cooking** (CS302) and **processing** (CS614, CS618, CS6004) the **food** that everyone (CSXYZ) eats



What do the seniors say?

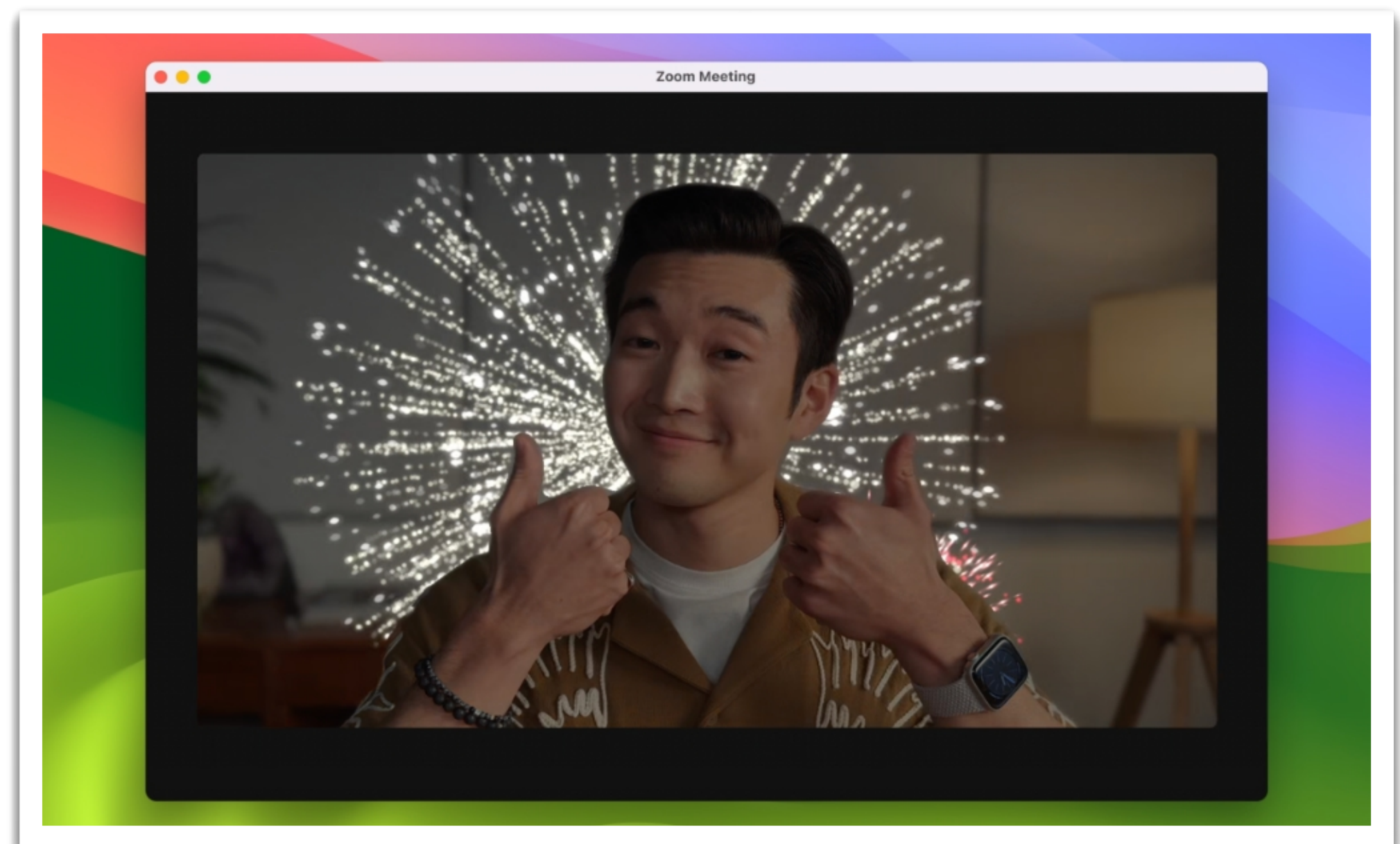
Undoubtedly the best CS course ... Covering the very fundamentals of different paradigms and about programming in general has been extremely helpful as I am currently working on functional and logic programming as a part of my job at Microsoft.

The quality of the lectures was very good. Amazing course, good teaching along with memes and famous quotes from movies make this a fun course. This was the only CS course which I enjoyed as much I enjoyed Data Structures in my second year.

I reflect on my learnings from time to time and felt that I should express it to you as well. I am really glad I studied the course well and scored the best grade during my undergraduate studies. So useful!

Got to learn a lot. The vibe of the lectures is always very engaging. The best professor I have ever met!

This course was simply the best one can ask for.



Logistics (CS339+CS355)

- Three **classes** per week in Slot 2: Mon 9:30am; Tue 10:35am; Thu 11:30am
- One **lab** per week in Slot L1: Mon 2pm, starting next week
- Instructor: Manas Thakur
- TAs: Aditya, Ameya, Anadi, Anand, Arhaan, Dhananjay, Kartik, Meetesh, Preet, Rohit, a few more to come ...
- **Office hour:** Wed 4:30-5:30 pm (CC 308)
- Course webpage (schedule, deadlines): <https://tinyurl.com/app23-iitb-plan>
- Moodle (material, submissions, marks)



Evaluation (CS339)

- Exams
 - Quizzes ($10 * 2$)
 - Midsem (30)
 - Endsem (40)
- Attendance
 - Up to 5 marks; <70 : DX
- Participation
 - Activities, questions, blogs, memes (only constructive) — up to 5 marks



Evaluation (CS355)

- Practice
 - 7 out of 8 labs * 3
- Attendance
 - 7 out of 8 labs * 2
- Exams
 - Midsem: 30
 - Endsem: 35



The Constitution

➤ Rights:

- An in-depth understanding of the various topics
- Acceptable answers to doubts/queries
- Timely and complete evaluations

➤ Duties:

- Sincerity and honesty
- No hesitation in reaching out to the instructor/TAs
- **No plagiarism** in anything

➤ Institute norms:

- Attendance (graded with minimum 70%, but recommended 100%)
- Grading (standard relative grading as per institute policy)



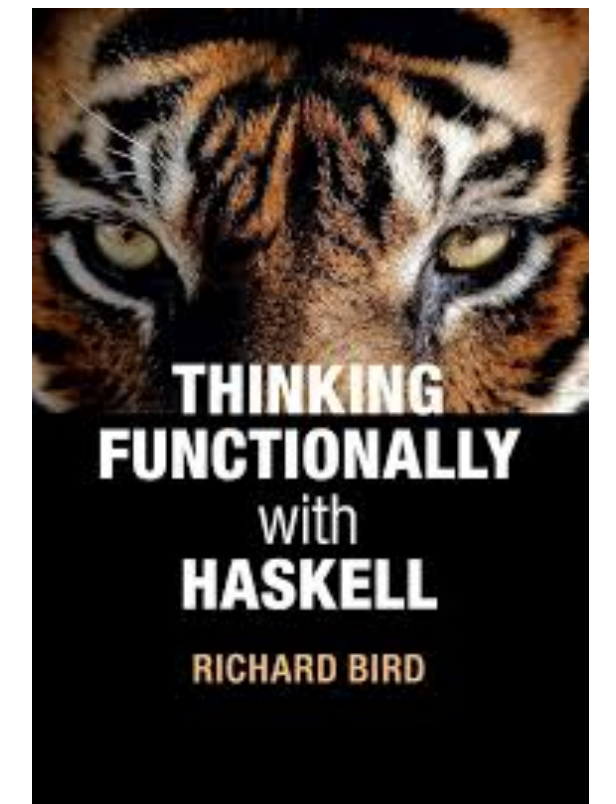
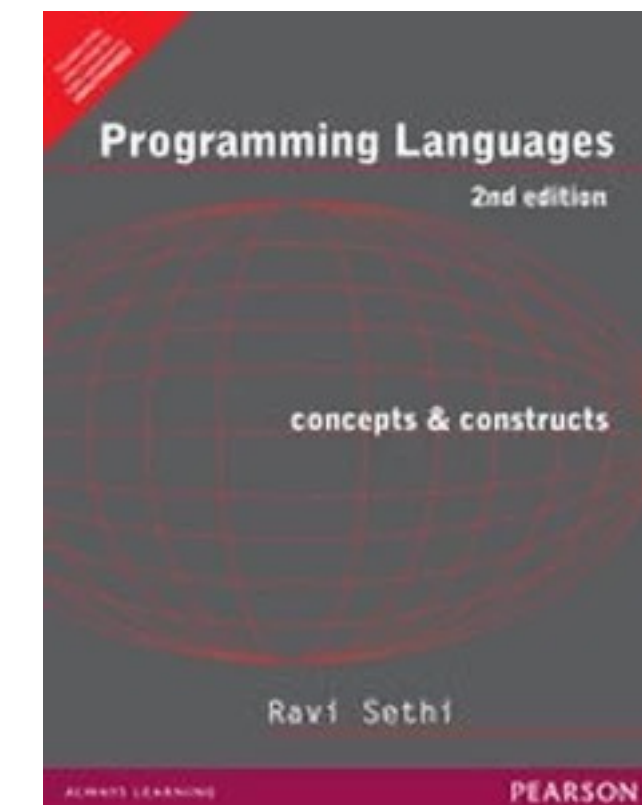
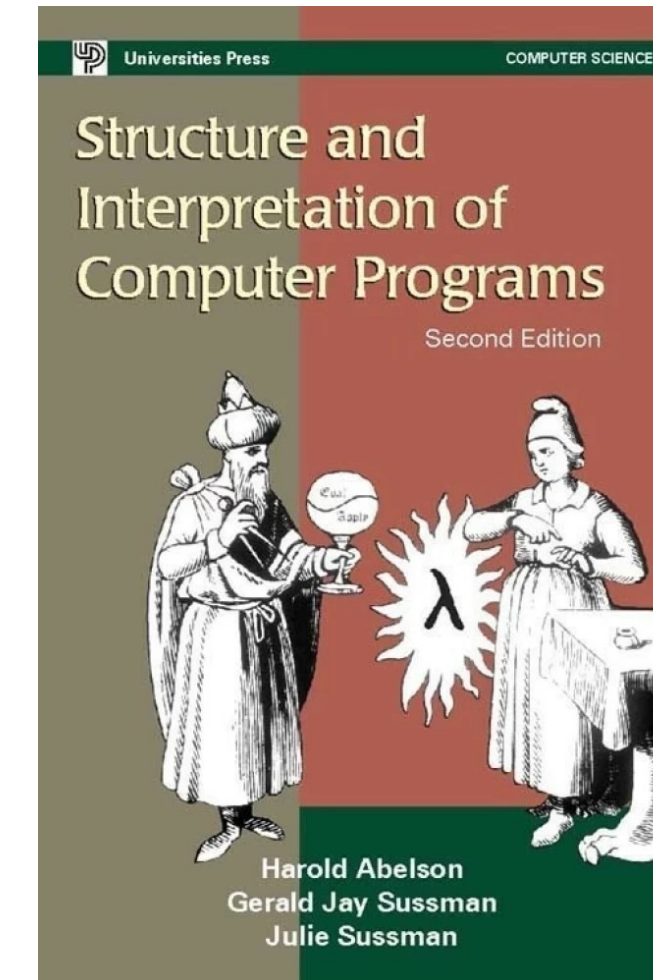
Learning Resources

Most
concepts and
Scheme

Lambda
calculus and
Prolog

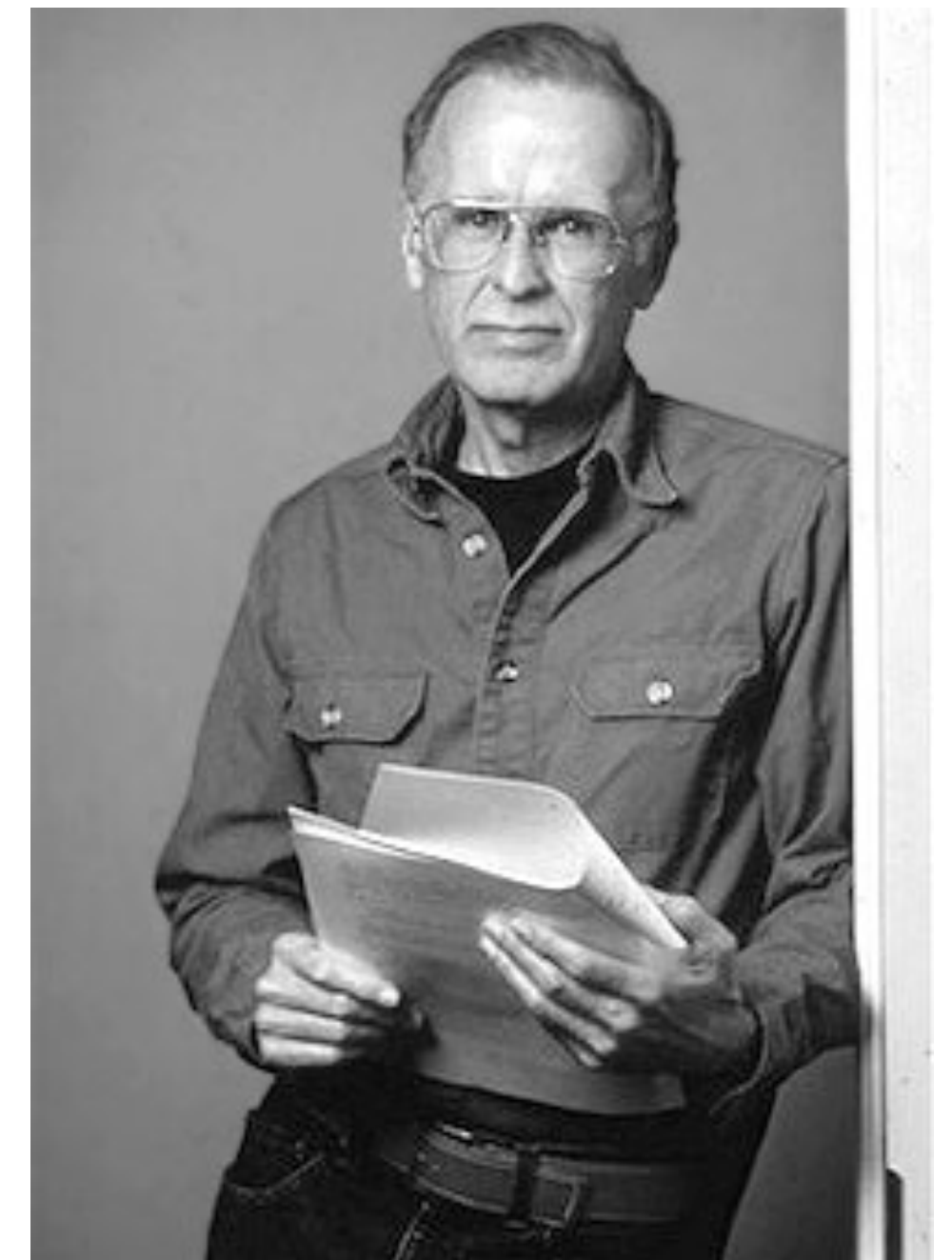
Haskell,
types, proofs

- Textbooks (SICP, PLCC, TFWH)
- Some videos and papers
- Take notes in class (slides won't always be enough/available)
- Practice programs in the lab and at home



First Homework!

- Activity 1:
 - Read first seven pages of John Backus's Turing Award Lecture
 - Try finding out why was that lecture very interesting
 - Write your thoughts by Wednesday 11:55 pm; get **1 PC**.
- Install **DrRacket** with **sicp** package on your computer
 - <https://download.racket-lang.org/>
- Next class: Back to high school!



The *spirit* of this course

Dedication text of *Structure and Interpretation of Computer Programs (SICP)*

Authors: Harold Abelson and Gerald Jay Sussman with Julie Sussman (MIT)

This book [course] is dedicated, in respect and admiration, to the spirit that lives in the computer:

“I think that it’s extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don’t think we are. I think we’re responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun. Above all, I hope we don’t become missionaries. Don’t feel as if you’re Bible salesmen. The world has too many of those already. What you know about computing other people will learn. Don’t feel as if the key to successful computing is only in your hands. What’s in your hands, I think and hope, is intelligence: the ability to see the machine as more than when you were first led up to it, that you can make it more.”

—Alan J. Perlis (April 1, 1922 – February 7, 1990)

