CS150 Special Topic: Advanced Programming Languages

Guannan Wei
guannan.wei@tufts.edu

Sept 2, 2025

Tufts University

About Me

- Assistant Professor, Department of Computer Science
- Postdoc, INRIA/ENS-PSL, France
- PhD in Computer Science, Purdue University
- Research: programming languages, formal methods, software engineering
- Web: https://continuation.passing.style

Why CS 150 Advanced Programming Languages?

Why CS 150 Advanced Programming Languages?

- Get an idea of what PL research looks like
  - Acquire necessary skills to understand PL research results
  - Prepare for PhD study, or just for fun

Why CS 150 Advanced Programming Languages?

- Get an idea of what PL research looks like
  - Acquire necessary skills to understand PL research results
  - Prepare for PhD study, or just for fun
- Work on a serious research project
  - Develop something new and interesting
  - Could be open ended, or potentially lead to a publication

Why CS 150 Advanced Programming Languages?

- Get an idea of what PL research looks like
  - Acquire necessary skills to understand PL research results
  - Prepare for PhD study, or just for fun
- Work on a serious research project
  - Develop something new and interesting
  - Could be open ended, or potentially lead to a publication
- Communicate scientific progress/results
  - Identify exciting ideas, and understand their importance and contribution
  - Explain your ideas and findings through talks and technical writing
  - Get feedback

Format of this course

This is a research-oriented course.

- Lecture
- Paper discussion
- Project
- No assignment and no exams

What is research?

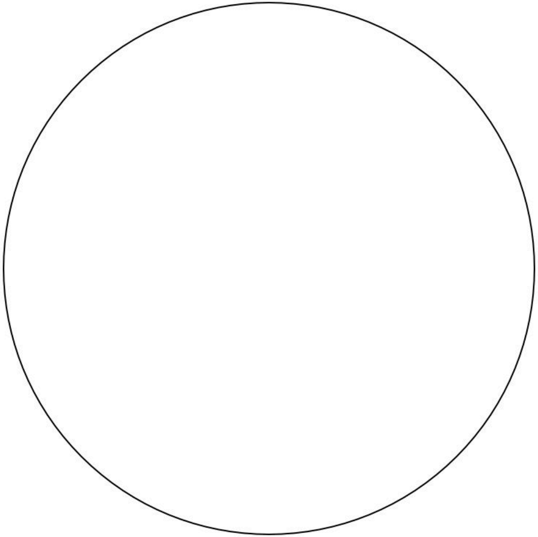
What is research?

- From my Master advisor Matt Might

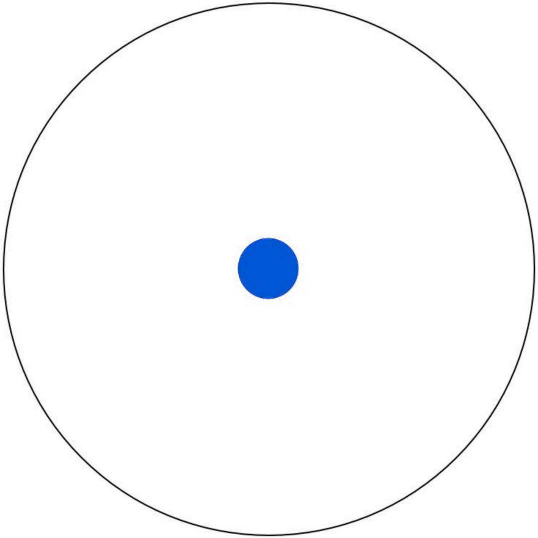
  The illustrated guide to a Ph.D.

  https://matt.might.net/articles/phd-school-in-pictures/

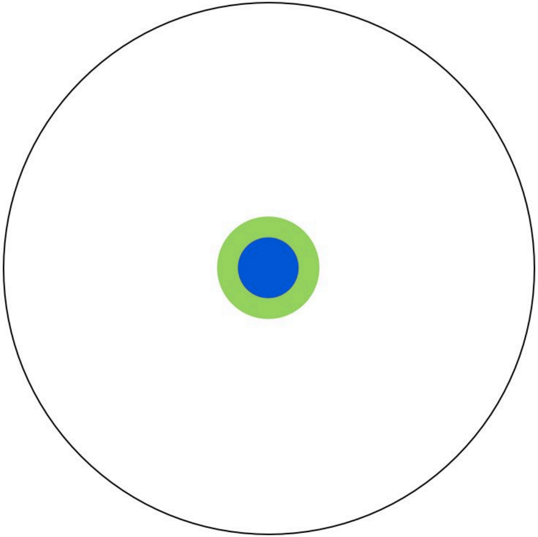
Imagine a circle that contains all of human knowledge:



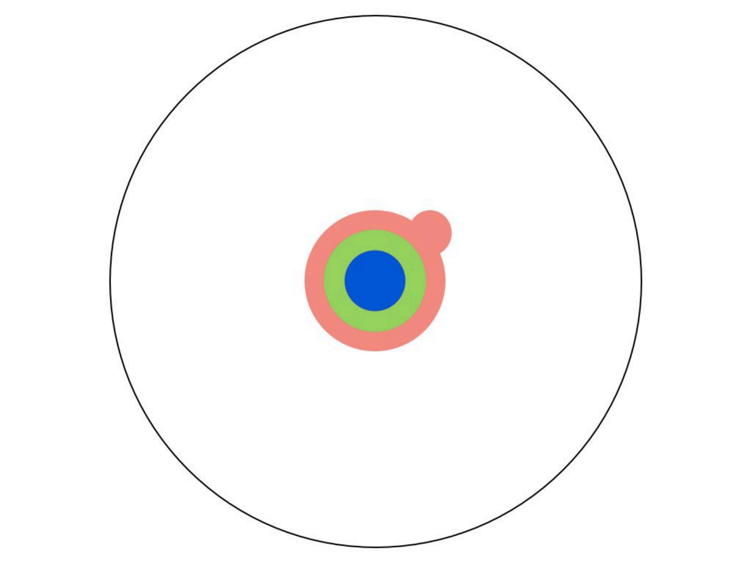
By the time you finish elementary school, you know a little:



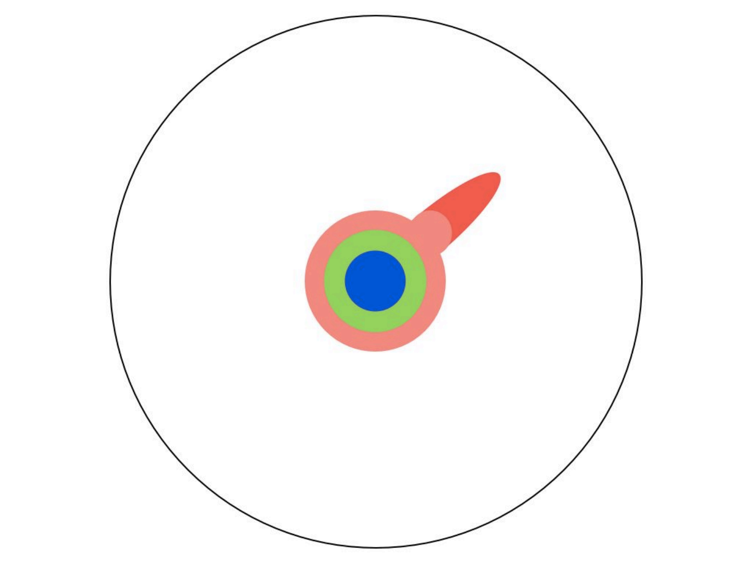
By the time you finish high school, you know a bit more:



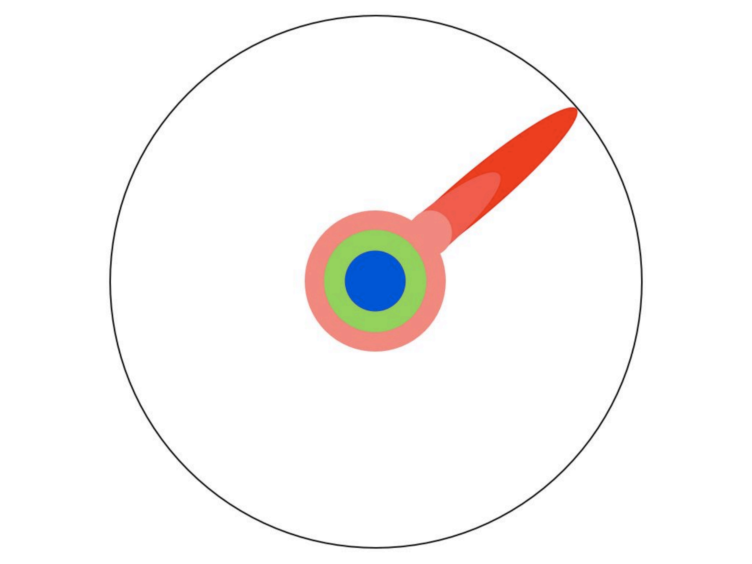
With a bachelor's degree, you gain a specialty:



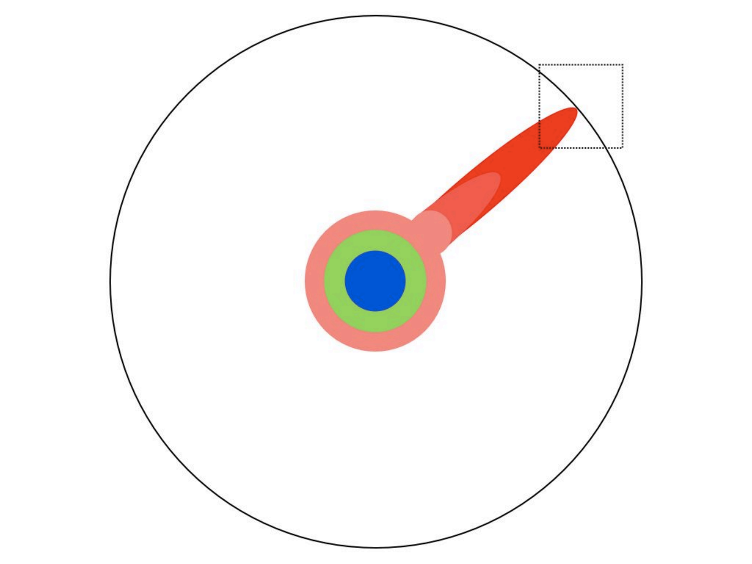
A master's degree deepens that specialty:



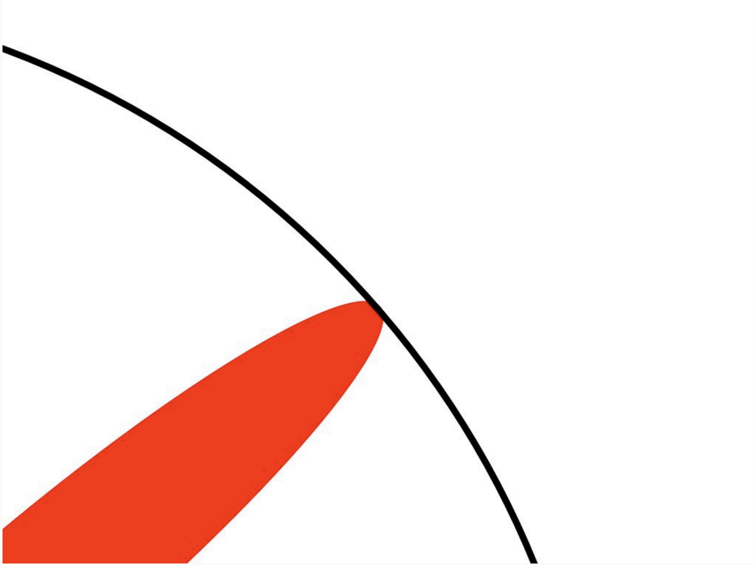
Reading research papers takes you to the edge of human knowledge:



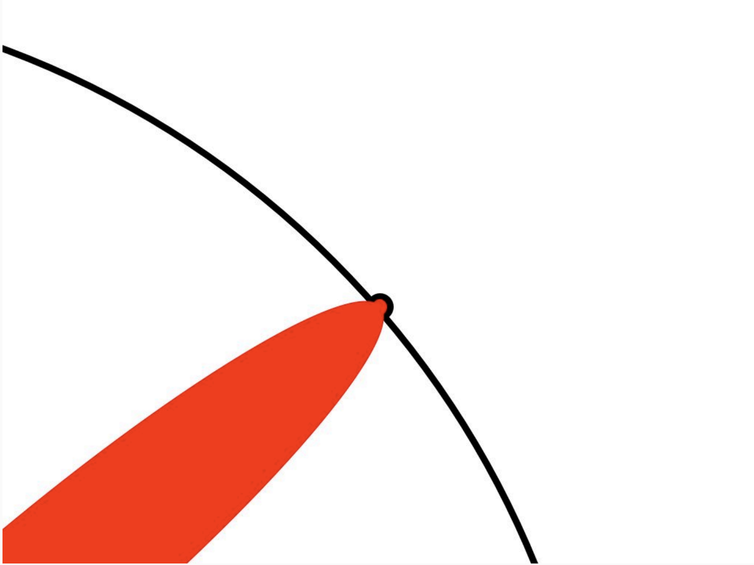
Once you're at the boundary, you focus:



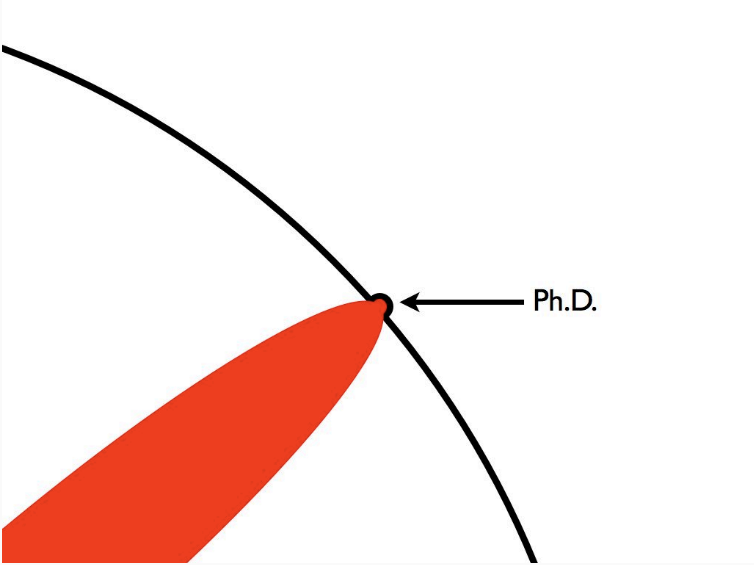
You push at the boundary for a few years:



Until one day, the boundary gives way:

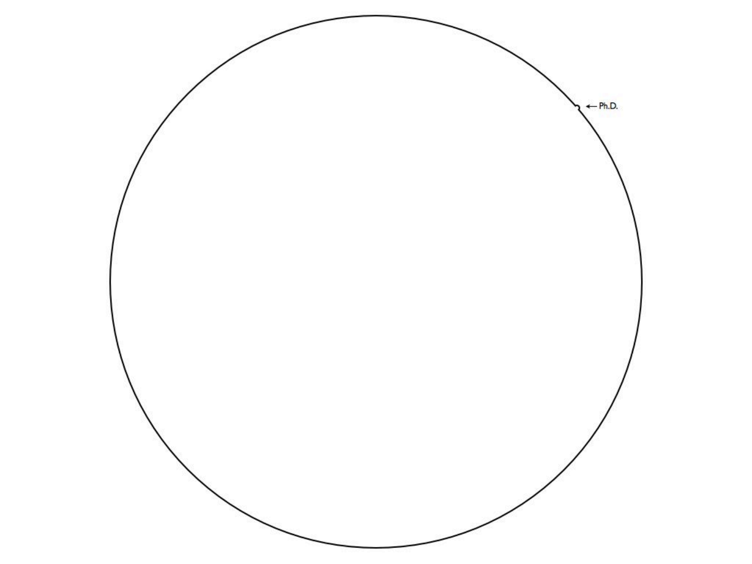


And, that dent you've made is called a Ph.D.:



Of course, the world looks different to you now:

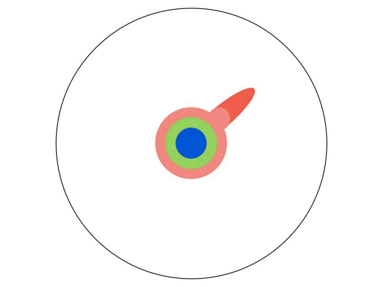
Don't forget the big picture:



Format of this course

Now let's go back to the format of this course:

- Lecture
- Paper discussion
- Project



Logistics - Lectures

- Review some important and fundamental topics
- Goal: fill the gap between CS105 and research papers you would read
- Topics covered:
  - Operational semantics
  - Type/effect systems
  - Metatheory
  - Transformation and optimization
  - Formal methods
  - ...

Logistics - Paper discussion

- Each student is expected to present 2-3 papers and lead the discussion
  - What is the problem that motivates this work?
  - What is the key idea of the paper?
  - What are the important technical details?
  - How can you/others improve this work?
  - How can you use it in your own projects?
  - Demo if possible, and other important related works

Logistics - Paper discussion

- Each student is expected to present 2-3 papers and lead the discussion
  - What is the problem that motivates this work?
  - What is the key idea of the paper?
  - What are the important technical details?
  - How can you/others improve this work?
  - How can you use it in your own projects?
  - Demo if possible, and other important related works
- For audiences:
  - Read the paper and write a summary (half page) before the discussion
  - Summarize contribution, strengths, and weaknesses

Logistics - Paper discussion

- Presenter chooses the paper (at least) 1 week before
- A list of papers (that I find interesting) on the course website

Logistics - Paper discussion

- Presenter chooses the paper (at least) 1 week before
- A list of papers (that I find interesting) on the course website
- Where to find more papers?
  - **SIGPLAN conferences/PACMPL journal**: POPL, PLDI, ICFP, OOPSLA
  - **Journals**:
    - Transactions on Programming Languages and Systems (TOPLAS)
    - Journal of Functional Programming (JFP)
  - Adjacent fields:
    - Logics, verification, semantics (SIGLOG): LICS, CAV, ICALP, FSCD, etc.
    - Software engineering (SIGSOFT): ICSE, FSE, etc.
  - Some symposiums/workshops are good too

Logistics - Project

- You learn the most by building something, and programming is fun!
- Project ideas
  - Design and implement a tiny language with some new/interesting feature
  - Implement an optimization
  - Build a static analysis tool
  - Domain specific language
  - Explore the intersection of PL and another field (e.g., AI, security)
  - Talk to me :)

Logistics - Project

- Proposal
  - 1-page proposal
  - Week 5: proposal presentation (15 min)
  - Others give feedback
- Final report
  - 4-page report (acmart double-column format)
  - Week 14: presentation (25 min)
- Artifact
  - Code, tests, proofs, document, etc.

Logistics - Grading

- Participation: 10%
- Paper discussion: 30%
- Project: 60%
- There is no exam.

Logistics - Misc

- Office hour: by appointment
- Course website: https://continuation.passing.style/teaching/cs150-fall25/
  - Tentative schedule
  - Google Sheet to sign up for presentation slots
  - More resources on writing/presenting papers
- Homework
  - Start looking for interesting papers you'd like to present
  - Start thinking about your project ideas

Questions?

What is PL research?

What is PL research?

- Design and build a programming language
- Ensure that programs meet their specifications
- Make programs faster
- Build tools that improve programmer productivity
- ...

How to define a programming language?

Let's go back to the fundamentals:

- Syntax
  - Concrete syntax
  - Abstract syntax
- Semantics
  - **Dynamic semantics**: what can we say about the program's behavior at run-time
  - Static semantics: what can we say about the program's behavior at compile-time

λ-Calculus

- Syntax

$$
\begin{array}{rcll}
n & \in & \mathbb{N} & \text{natural numbers} \\
t & ::= & n & \text{numbers} \\
 & | & x & \text{variables} \\
 & | & \lambda x.t & \text{abstraction} \\
 & | & t_1\, t_2 & \text{application}
\end{array}
$$

Different ways to defined its semantics

- **Operational semantics**: the meaning of the program is defined by its execution.
  - Structural operational semantics (i.e. small-step semantics)
  - Contextual reduction semantics
  - Abstract machines
  - Natural semantics (i.e. big-step semantics)
  - Evaluators
- Denotational semantics
- Axiomatic semantics

Structural operational semantics (SOS)

$$\begin{array}{lcll} n & \in & \mathbb{N} & \\ t & ::= & n \mid x \mid \lambda x.t \mid t_1\, t_2 & \textbf{terms} \\ v & ::= & n \mid \lambda x.t & \textbf{values} \end{array}$$

Call-by-value (CBV)

$$\frac{}{(\lambda x.t)\, v \to t[x := v]}\ \beta_v \qquad \frac{t_1 \to t_1'}{t_1\, t_2 \to t_1'\, t_2}\ \textsc{App1} \qquad \frac{t_2 \to t_2'}{v\, t_2 \to v\, t_2'}\ \textsc{App2}$$

Structural operational semantics (SOS)

- CBV example

  $(\lambda f.\lambda x.f\ x)((\lambda x.x)(\lambda y.y))$

Structural operational semantics (SOS)

Call-by-value (CBV)

$$\frac{}{(\lambda x.t)\, v \to t[x := v]}\ \beta_v \qquad \frac{t_1 \to t_1'}{t_1\, t_2 \to t_1'\, t_2}\ \textsc{App1} \qquad \frac{t_2 \to t_2'}{v\, t_2 \to v\, t_2'}\ \textsc{App2}$$

Call-by-name (CBN)

$$\frac{}{(\lambda x.t_1)\, t_2 \to t_1[x := t_2]}\ \beta \qquad \frac{t_1 \to t_1'}{t_1\, t_2 \to t_1'\, t_2}\ \textsc{App}$$

Structural operational semantics (SOS)

- What about call-by-need (e.g. lazy evaluation in Haskell)?
  - Call-by-need = call-by-name + sharing
  - *A call-by-need lambda calculus*. Ariola et al. POPL '95

From SOS to contextual reduction semantics

Call-by-value (CBV)

$$\frac{}{(\lambda x.t)\, v \to t[x := v]}\ \beta_v \qquad \frac{t_1 \to t_1'}{t_1\, t_2 \to t_1'\, t_2}\ \textsc{App1} \qquad \frac{t_2 \to t_2'}{v\, t_2 \to v\, t_2'}\ \textsc{App2}$$

- Some properties:
  - Evaluates from left to right
  - Deterministic
- Observe that App1 and App2 are structural congruence rules
  - There are something not changed before/after the step
  - Can we make it more compact?

Contextual reduction semantics

- An alternative to structural operational semantics (Felleisen and Hieb, 1989; Wright and Felleisen, 1992)
    - Define reduction contexts

      Intuition: specify where and when a reduction could happen; context = surrounding invariant terms
    - Define the head reduction rule

      Intuition: the actual computation (e.g. beta)

Contextual reduction semantics

- An alternative to structural operational semantics (Felleisen and Hieb, 1989; Wright and Felleisen, 1992)
  - Define reduction contexts

    Intuition: specify where and when a reduction could happen; context = surrounding invariant terms
  - Define the head reduction rule

    Intuition: the actual computation (e.g. beta)
- Side note:

  "This Felleisen stuff is all syntax, not semantics." – Albert Meyer, 1988
  https://www.cs.cmu.edu/~popl-interviews/felleisen.html

Contextual reduction semantics

$$\begin{array}{rcll} t & ::= & n \mid x \mid \lambda x.t \mid t_1\, t_2 & \textbf{terms} \\ v & ::= & n \mid \lambda x.t & \textbf{values} \end{array}$$

Call-by-value (CBV)

$$E \;::=\; \square \mid v\,E \mid E\,t \quad \textbf{reduction contexts}$$

$$\frac{}{(\lambda x.t)\,v \to t[x := v]}\;\beta_v \qquad\qquad \frac{t_1 \to t_1'}{E[t_1] \to E[t_1']}\;\textsc{Ctx}$$

- E specifies left-to-right evaluation order

Contextual reduction semantics

- A term is *decomposed* to a reduction context E and a redex t_1:

  $t = E[t_1]$

- Focus on t_1, which reduces to t_2:

  $t_1 \to t_2$

- Plug in t_2 back to context E:

  $E[t_1] \to E[t_2]$

Contextual reduction semantics

- CBV example

$((\lambda x.x)(\lambda y.y))(\lambda f.\lambda x.f\ x)$

Contextual reduction semantics

$$\begin{array}{lcll} t & ::= & n \mid x \mid \lambda x.t \mid t_1\, t_2 & \textbf{terms} \\ v & ::= & n \mid \lambda x.t & \textbf{values} \end{array}$$

Call-by-name (CBN)

- Question: define the evaluation context for CBN.