A puzzle or a dilemma

Suppose I want to slay crashing programs before they have a chance to run (=static type safety).

Do I let this one go?

```
let c = ref (fun x -> x) in
c := (fun x -> x + 1)
c := (fun x -> not x)
!c true
```

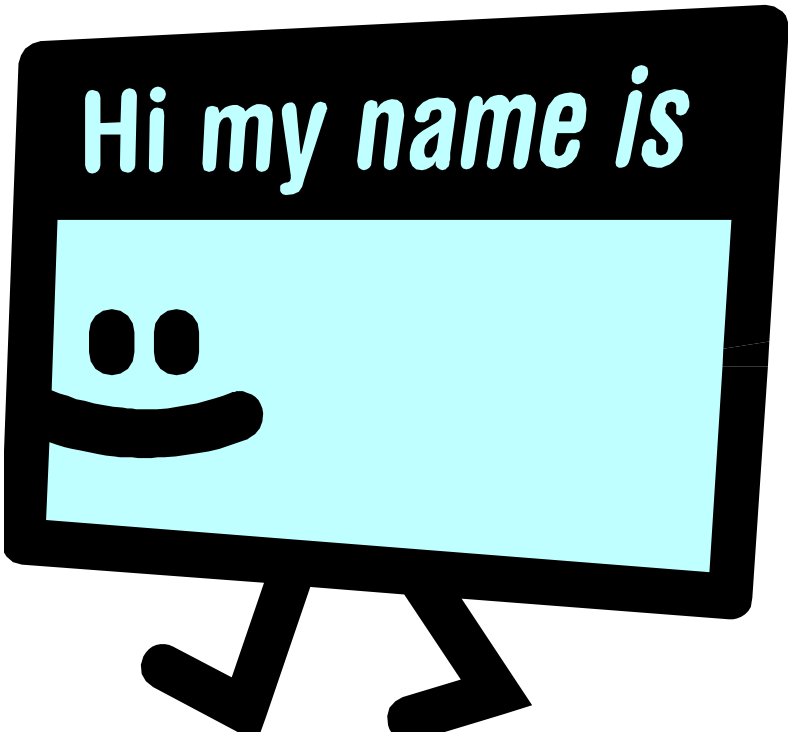
Fundamentals of Programming Languages

Bor-Yuh Evan Chang

Meeting 1: Welcome

CSCI 5535, Fall 2023
csci5535.cs.colorado.edu/f23

Getting to Know You: "I, …, wonder …"



Distraction-Free Classroom

- Let's turn off our cell phones and wi-fi

Distraction-Free Classroom

- If you have a **need** to use a laptop, please discuss with me after class

Introductions: Your guide this semester

- Office Hours: Tue 3:15-4 ECCS 114F, Thu 1-1:45 ECCS 114J

Introductions

- Who am I?
- About you?
  - What do you want to get out of this class?

Introductions

- Introduce yourself to someone you haven't met before. [credit Boulder New Tech Meetup]

- Two minutes!

HW: Post on Piazza

- Introduce yourself. Include one thing your classmates probably don't know about you.

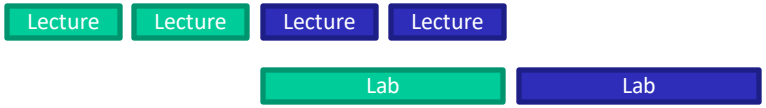
- Background
  - Comfort with functional programming?
  - Comfort with mathematical logic and induction?
  - Experience with building language tools (interpreters, translators)?
  - What do you want out of this class?
  - Can be a separate private note to me

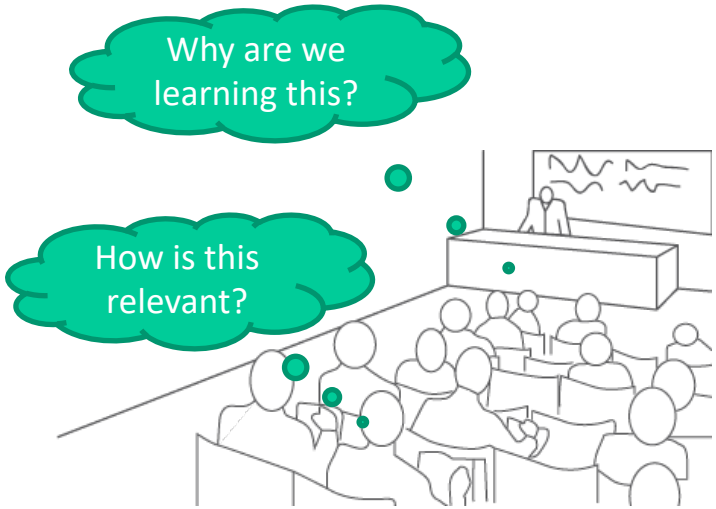
Focusing on guiding towards understanding …

foundational subset of

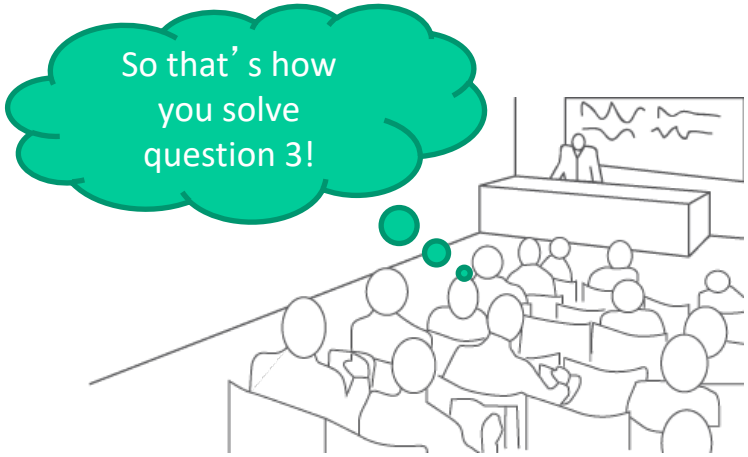
- Project-based course: Formalize a programming language
  - Incrementally in homework assignments
  - Generally: two weeks of discussion towards completing the assignment, driven by you (in class + on Piazza)!
  - Assignments due ~~Friday~~ Saturday
  - No late submissions but generous "redo" policy

Traditional Format





Our Class



Discussion, discussion, discussion

- Discussion, not lecture
  - Only meeting I will use slides
- Please interrupt at any time!
- It's completely ok to say:
  - I don't understand. Please say it another way.
  - Slow down!

- **Course is project-based**
  - Lab assignments prompt the discussion
  - Expectation on you to be active

Oath

Lab Assignment Schedule

- Week 1
  - Sun: Lab released
- Week 2
  - Fri: Lab due at 6pm

Succeeding in 5535

- Engage and be active
  - Week 1 Sun-Mon: Review previous lab and read new chapter immediately
  - Week 1 Wed: Resolve any questions from previous lab or set up for new lab
  - Week 2 Fri: Submit and enjoy the weekend!

- How not to succeed: Start the lab in Week 2

Administrivia

- Public Website: csci5535.cs.colorado.edu/f23
  - notes, resources, etc.
- Course-Specific: Canvas
  - grades, feedback, etc.
  - discussion forum: Piazza
  - lab repositories: GitHub (via GitHub Classroom)

Today

- What this course is and is not
- Tell some stories
- Goals for this course
- Requirements and grading
- Course summary
- Start HW0

- Convince you that PL is cool and useful

"Isn't PL a solved problem?"

- We have lots of programming languages. Go home?
- What do you think this course is about?

Zach: Not just syntax. Address ~~the~~ a problem domain

Karthik: Design patterns. Formalizing.

Aditya: Why do we lots of PL?

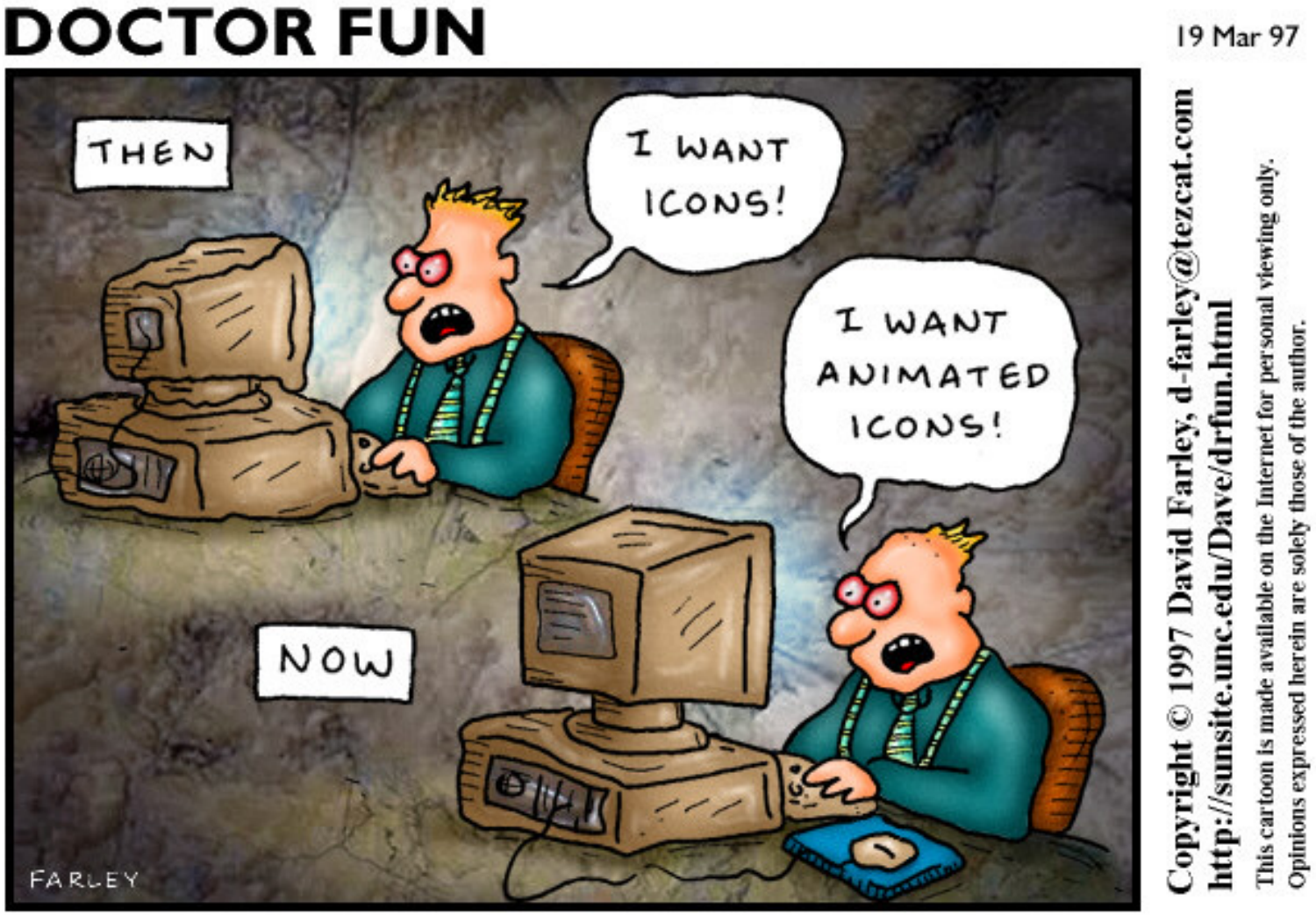
Lawrence: Model your domain. Verification of your program.

"Isn't PL a solved problem?"

"Isn't PL a solved problem?"

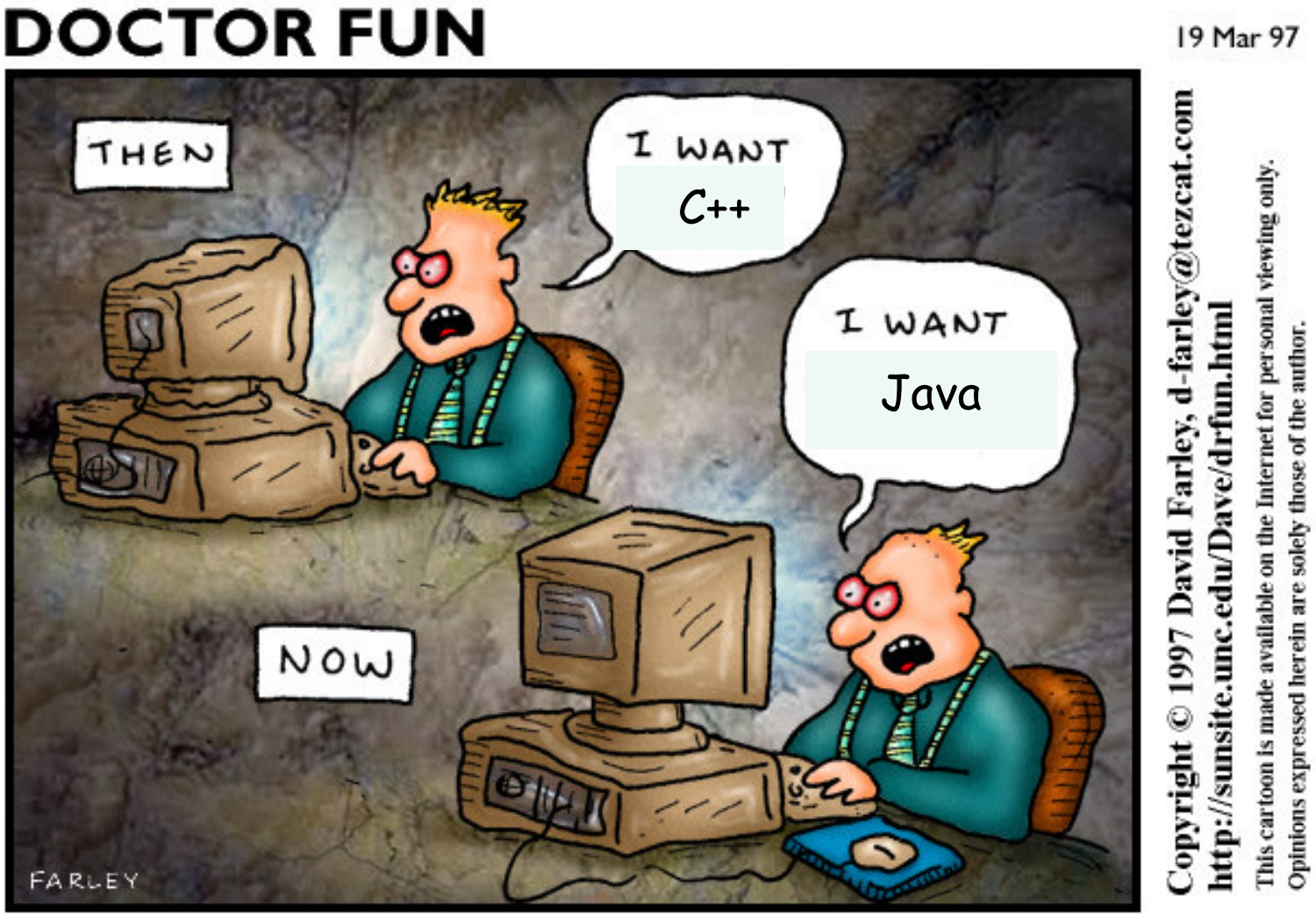
"Isn't PL a solved problem?"

New and Better Compilers?



Progress

A Dismal View of PL Research



Progress

Programming Languages

- Touches most other areas of CS
  - Theory: DFAs, TMs, language theory (e.g., LALR)
  - Systems: system calls, memory management
  - Arch: compiler targets, optimizations, stack frames
  - Numerics: FORTRAN, IEEE FP, Matlab
  - AI: theorem proving, search
  - DB: SQL, transactions
  - Networking: packet filters, protocols
  - Graphics: OpenGL, LaTeX, PostScript
  - Security: buffer overruns, .NET, bytecode, PCC, …
  - Computational Biology: pathway models
  - Software Engineering: software quality, development tools
  - Human Computer Interaction: development tools
- Both theory (math) and practice (engineering)

Overarching Theme

- I assert (and shall convince you) that

- PL is one of the most vibrant and active areas of CS research today
  - It is both theoretical and practical
  - It intersects most other CS areas

- You will be able to use PL techniques in your own projects

Goals

Goal 1

Learn to use advanced PL techniques

mathematical reasoning about computation

No Useless Memorization

- I will not waste your time with useless memorization
- This course will cover complex subjects
- I will teach their details to help you understand them the first time
- But you will never have to memorize anything low-level
- Rather, learn to apply broad concepts

Goal 2

When (not if) you design a language, it will avoid the mistakes of the past, and you will be able to describe it formally

Story: The Clash of Two Features

- Real story about bad programming language design
- Cast includes famous scientists
- ML ('82) functional language with polymorphism and monomorphic references (i.e., pointers)
- Standard ML ('85) innovates by adding polymorphic references
- It took 10 years to fix the "innovation"

Polymorphism (Informal)

- Code that works uniformly on various types of data
- Examples of function signatures:

  length : α list $\rightarrow$ int (takes an argument of type "list of α", returns an integer, for any type α)

  head : α list $\rightarrow \alpha$

- Type inference:
  - generalize all elements of the input type that are not used by the computation

References in Standard ML

- Like "updatable pointers" in C
- Type constructor: τ ref

  x : int ref "x is a pointer to an integer"
- Expressions:

  ref : $\tau \rightarrow \tau$ ref

  (allocate a cell to store a τ, like malloc)

  !e : τ when e : τ ref

  (read through a pointer, like *e)

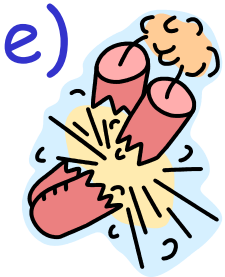
  e := e′ with e : τ ref and e′ : τ

  (write through a pointer, like *e = e′)
- Works just as you might expect

Polymorphic References: A Major Pain

Consider the following program fragment:

Code	Type inference
fun id(x) = x	id : $\alpha \rightarrow \alpha$ (for any α)
val c = ref id	c : $(\alpha \rightarrow \alpha)$ ref (for any α)
fun inc(x) = x + 1	inc : int $\rightarrow$ int
c := inc	Ok, since c : (int $\rightarrow$ int) ref
(!c) (true)	Ok, since c : (bool $\rightarrow$ bool) ref



Reconciling Polymorphism and References

- Type system fails to prevent a type error!
- Commonly accepted solution today:
  - value restriction: generalize only the type of values!
    - easy to use, simple proof of soundness
  - many "failed fixes"
- To see what went wrong we need to understand semantics, type systems, polymorphism and references

Story: Java Bytecode Subroutines

- Java bytecode programs contain subroutines (jsr) that run in the caller's stack frame (why?)
- jsr complicates the formal semantics of bytecodes
  - Several verifier bugs were in code implementing jsr
  - 30% of typing rules, 50% of soundness proof due to jsr
- It is not worth it:
  - In 650K lines of Java code, 230 subroutines, saving 2427 bytes, or 0.02%
  - 13 times more space could be saved by renaming the language back to Oak

Recall Goal 2

When (not if) you design a language, it will avoid the mistakes of the past, and you will be able to describe it formally

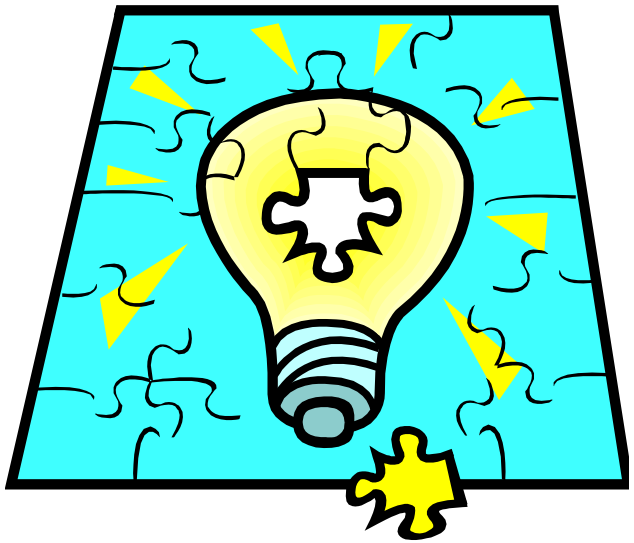
Goal 3

e.g., how do we get "good" programs from AI?

Understand current PL research (POPL, PLDI, OOPSLA, ICFP, TOPLAS, …)

Most Important Goal

Have Lots of Fun!



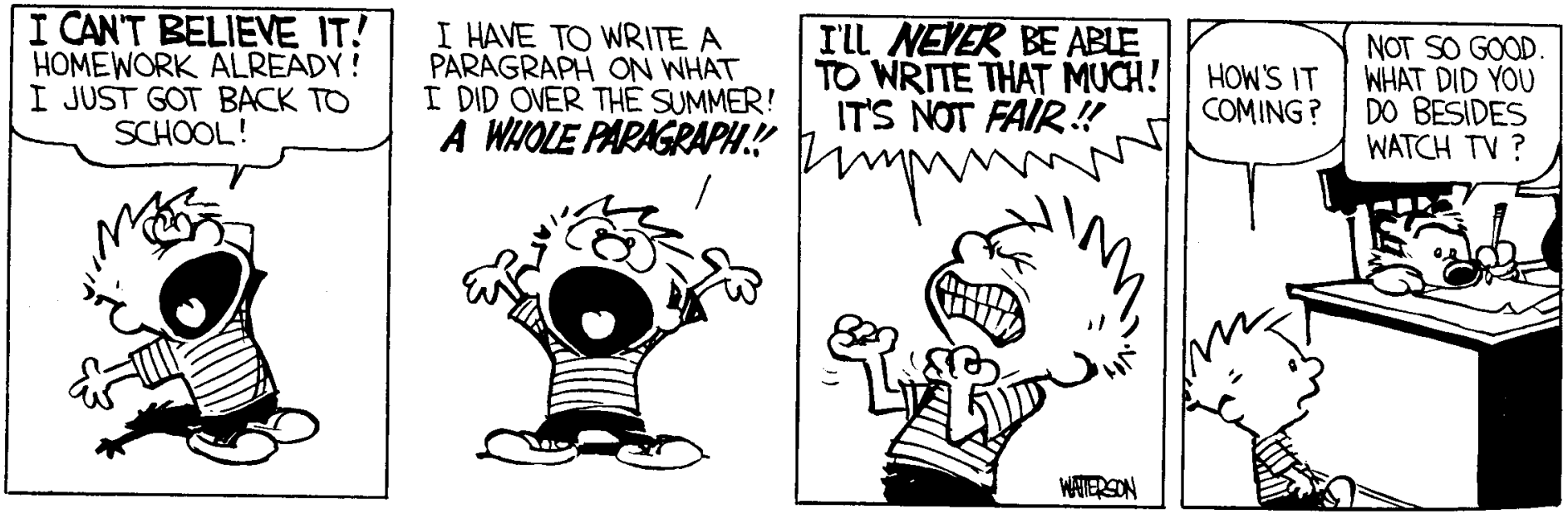
Requirements

Prerequisites

- "Programming experience"
  - exposure to various language constructs (imperative **and** functional) and their meaning/implementation (e.g., CSCI 3155)
  - ideal: undergraduate compilers (e.g., CSCI 4555)
- "Mathematical maturity"
  - we'll use formal notation to describe the meaning of programs
  - expected: familiarity with logic and rigorous proofs
- If you are an undergraduate or from another department, please see me.

Assignments

- Reading and participation (each meeting)
- Homework (for ~half semester)
- Final exam
- Final project



Reading and Participation

- Readings
  - Spark class discussion, post/bring questions
  - Background and context for homeworks

- "A moment's thought" on Piazza
  - Post 1+ substantive comment, question, or answer for each class meeting
  - Due before the next meeting

What is "substantive"?

- May be less than a blog post but more than a tweet.
- Some examples:
  - Questions
  - Thoughtful answers
  - Clarification of some point
  - What you think is the main point in the reading set.
  - An idea of how some work could be improved
- Intent: take a moment to reflect on the day's reading/discussion (*not* to go scour the web)

Homework and Exam

- Homework/Problem Sets
  - Where the "real" learning happens
  - "Math" (logic) + "Programming"
  - Encourage mastery: "redos"
  - Due ~~Fridays~~ Saturdays
  - Collaborate with peers (but acknowledge!)

- Final Exam

Final Project

- Options:
  - Research project
  - Literature survey
  - Implementation project
- Write a ~5-8 page paper (conference-like)
- Give a ~15-20 minute presentation
- On a topic of your choice
  - Ideal: integrate PL with your research
- ~Pair projects

Possible Special Topics

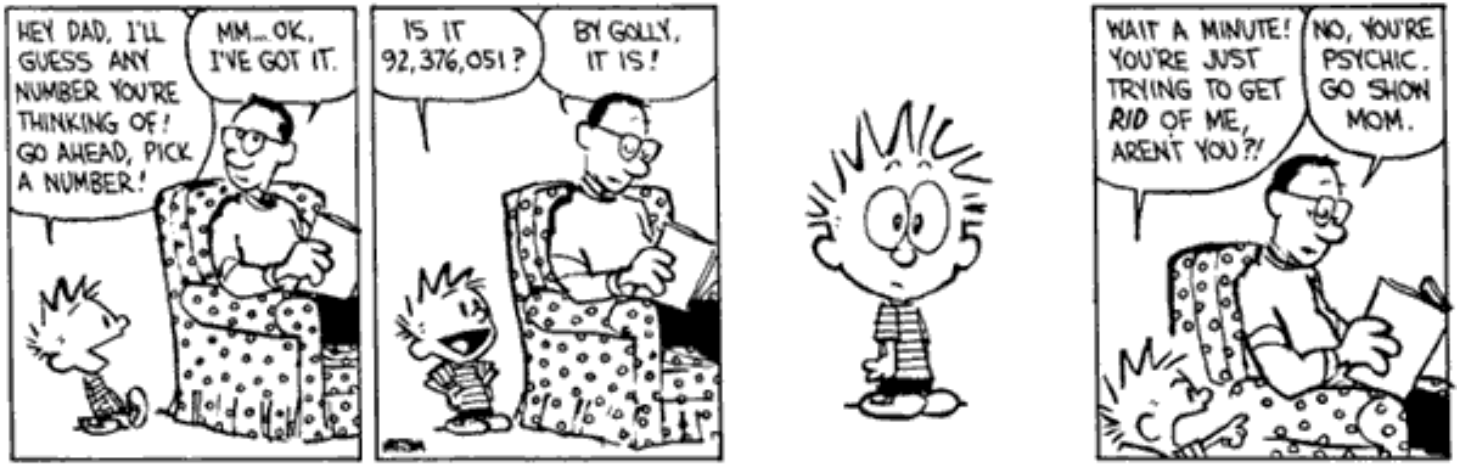
- Types for resource management (Rust)
- Heap reasoning
- Program synthesis
- Neurosymbolic programming

- What do you want to explore?

Example Application: Model Checking

- Verify properties or find bugs in software

- Take an important program (e.g., a device driver)
- Merge it with a property (e.g., no deadlocks)
- Transform the result into a boolean program
- Use a model checker to exhaustively explore the resulting state space
  - Result 1: program provably satisfies property
  - Result 2: program violates property "right here on line 92,376"!



For Next Time

- **Read the course syllabus**
  csci5535.cs.colorado.edu/f23

- Join the course Canvas, Piazza (via Canvas), get on GitHub, upload a profile picture, and introduce yourself