Programming Languages
Topic of Ultimate Mastery

Wes Weimer

CS 615

http://www.cs.virginia.edu/~weimer/2007-615

(note: CS 615 == CS 655)

Reasonable Initial Skepticism

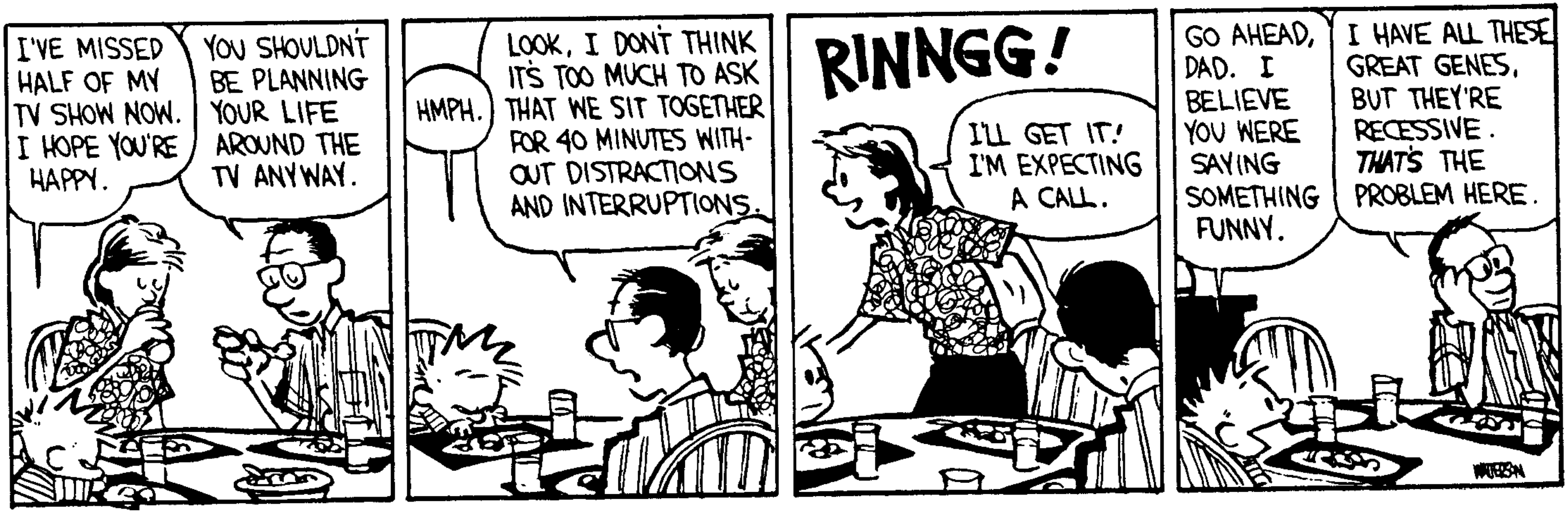
Today’s Class

- Vague Historical Context
- Goals For This Course
- Requirements and Grading
- Course Summary

- Convince you that PL is useful

Meta-Level Information

- Please interrupt at any time!
- Completely cromulent queries:
  - I don’t understand: please say it another way.
  - Slow down, you talk too fast!
  - Wait, I want to read that!
  - I didn’t get joke X, please explain.


I'VE MISSED HALF OF MY TV SHOW NOW. I HOPE YOU'RE HAPPY.
YOU SHOULDN'T BE PLANNING YOUR LIFE AROUND THE TV ANYWAY.
HMPH.
LOOK, I DON'T THINK IT'S TOO MUCH TO ASK THAT WE SIT TOGETHER FOR 40 MINUTES WITHOUT DISTRACTIONS AND INTERRUPTIONS.
RINNGG!
I'LL GET IT! I'M EXPECTING A CALL.
GO AHEAD, DAD. I BELIEVE YOU WERE SAYING SOMETHING FUNNY.
I HAVE ALL THESE GREAT GENES, BUT THEY'RE RECESSIVE. THAT'S THE PROBLEM HERE.
WATTERSON

What Have You Done For Us Lately?

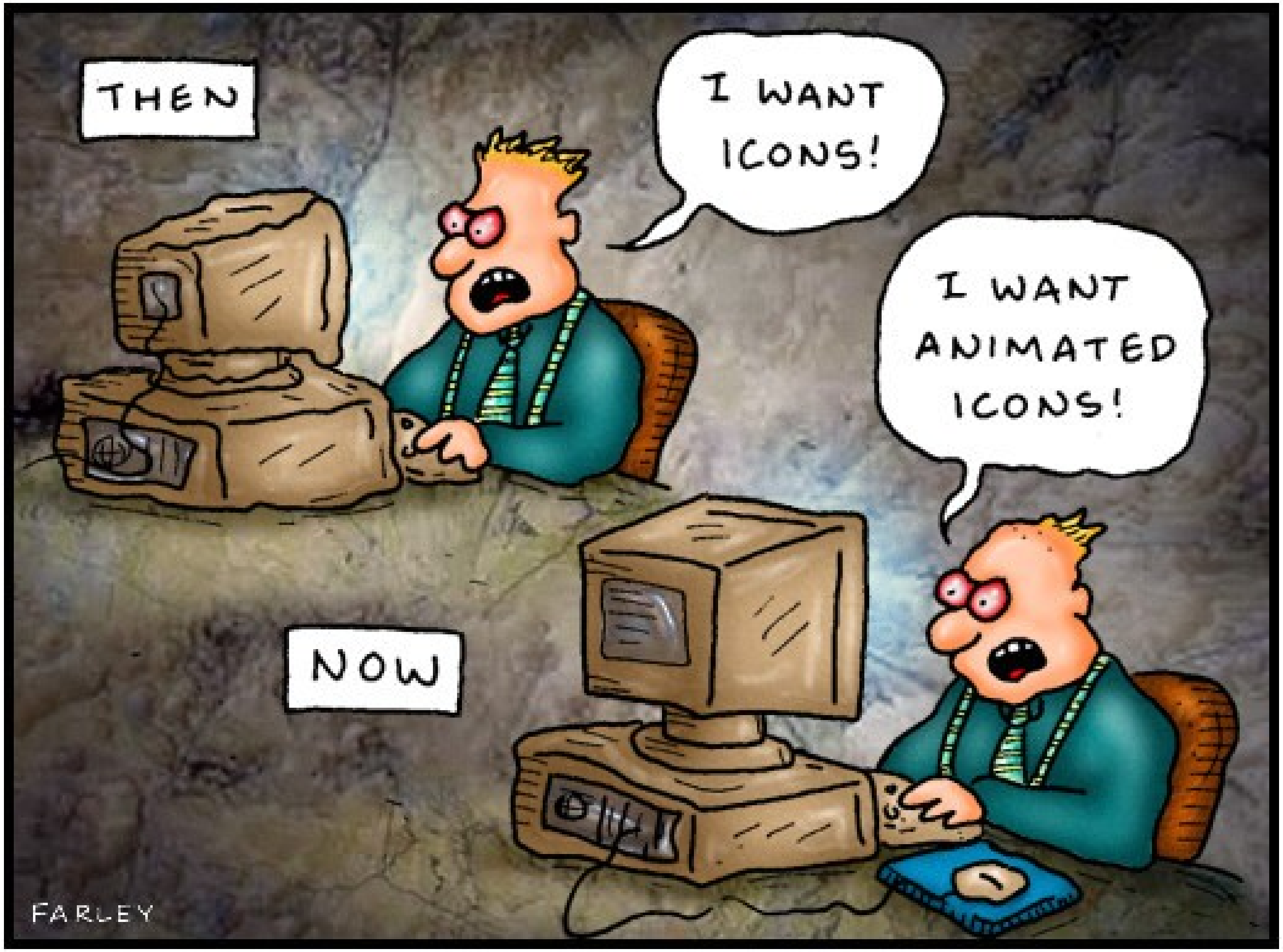
- Isn’t PL a solved problem?
  - PL is an old field within Computer Science
  - 1920’s: “computer” = “person”
  - 1936: Church’s Lambda Calculus (= PL!)
  - 1937: Shannon’s digital circuit design
  - 1940’s: first digital computers
  - 1950’s: FORTRAN (= PL!)
  - 1958: LISP (= PL!)
  - 1960’s: Unix
  - 1972: C Programming Language
  - 1981: TCP/IP
  - 1985: Microsoft Windows
  - 1992: Ultima Underworld / Wolfenstein 3D

> “… a prestigious line of work with a long and glorious tradition.” - Vizzini

Don’t We Already Have Compilers?

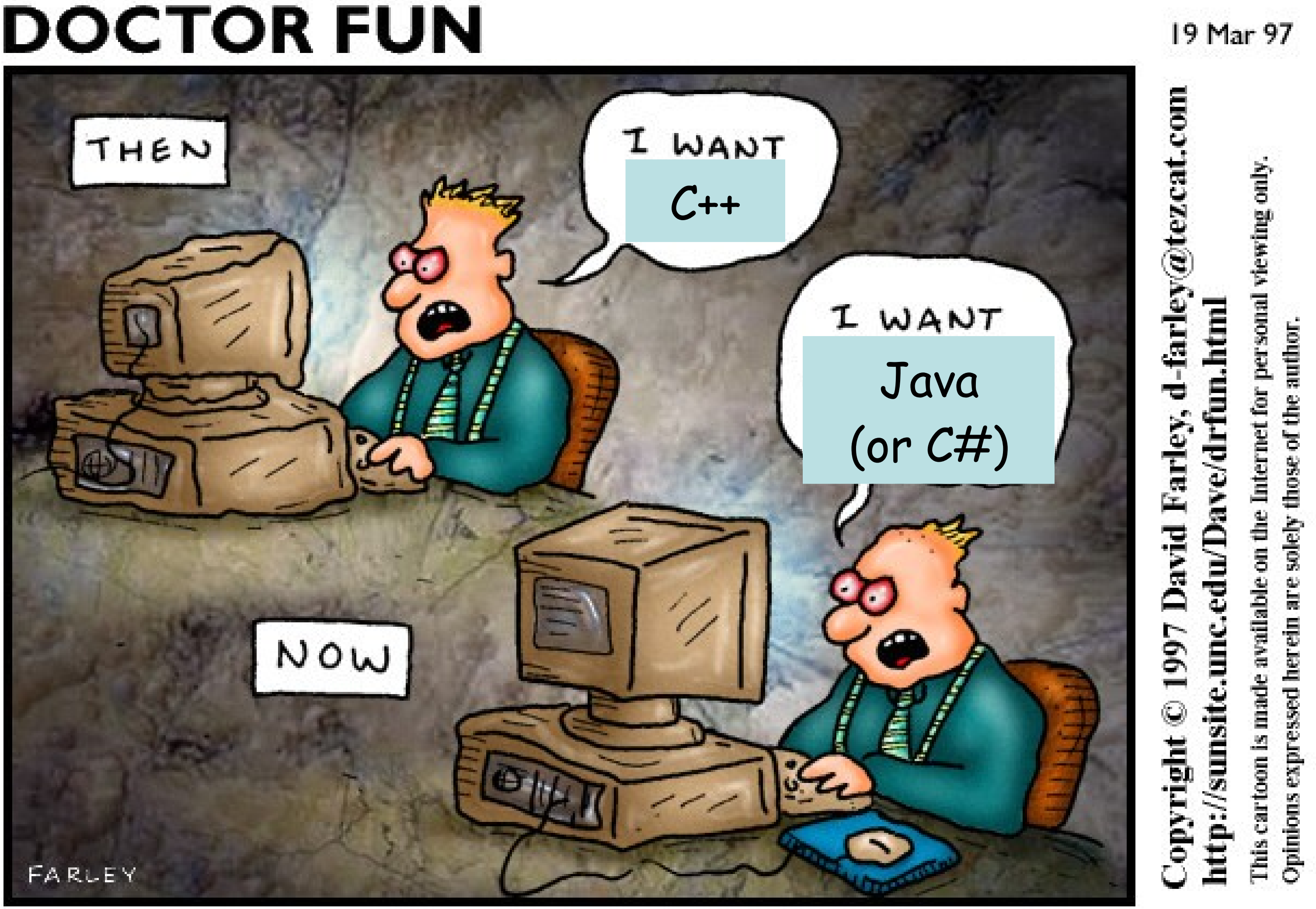
DOCTOR FUN

19 Mar 97



Progress

Dismal View Of PL Research


DOCTOR FUN
19 Mar 97
THEN
I WANT C++
NOW
I WANT Java (or C#)
FARLEY
Copyright © 1997 David Farley, d-farley@tezcat.com
http://sunsite.unc.edu/Dave/drfun.html
This cartoon is made available on the Internet for personal viewing only.
Opinions expressed herein are solely those of the author.

Progress

Parts of Computer Science

- CS = (Math × Logic) + Engineering
  - Science (from Latin *scientia* - knowledge) refers to a system of acquiring knowledge - based on empiricism, experimentation, and methodological naturalism - aimed at finding out the truth.
- We rarely actually do this in CS
  - "CS theory" = Math (logic)
  - "Systems" = Engineering (bridge building)

Programming Languages

- Best of both worlds: Theory and Practice!
  - Only pure CS theory is more primal
- Touches most other CS areas
  - Theory: DFAs, PDAs, TMs, language theory (e.g., LALR)
  - Systems: system calls, assembler, memory management
  - Arch: compiler targets, optimizations, stack frames
  - Numerics: FORTRAN, IEEE FP, Matlab
  - AI: theorem proving, ML, search
  - DB: SQL, persistent objects, modern linkers
  - Networking: packet filters, protocols, even Ruby on Rails
  - Graphics: OpenGL, LaTeX, PostScript, even Logo (= LISP)
  - Security: buffer overruns, .net, bytecode, PCC, …
  - Software Engineering: obvious

Overarching Theme

- I assert (and shall convince you) that

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

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

Goal #1

- Learn to **use** advanced PL techniques

Useful Complex Knowledge

- A proof of the fundamental theorem of calculus
- A proof of the max-flow min-cut theorem
- Nifty Tree node insertion (e.g., B-Trees, AVL, Red-Black)
- The code for the Fast Fourier Transform
- And so on …

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’ll 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) is a functional language with polymorphism and monomorphic references (i.e. pointers)
- Standard ML ('85) innovates by adding polymorphic reference
- 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 → int (takes an argument of type "list of α", returns an integer, for any type α)

  head : α list → α

- 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: ptr τ
- Expressions:

    alloc : $\tau \to$ ptr τ (allocate a cell to store a τ)

    *e : τ when e : ptr τ (read through a pointer)

    *e := e’ with e : ptr τ and e’ : τ

    (write through a pointer)

- Works just as you might expect

Polymorphic References: A Major Pain

Consider the following program fragment:

Code	Type inference	
fun id(x) = x	id : $\alpha \to \alpha$	(for any α)
val c = alloc id	c : ptr ($\alpha \to \alpha$)	(for any α)
fun inc(x) = x + 1	inc : int → int	
*c := inc	Ok, since c : ptr (int → int)	
(*c) (“hi”)	Ok, c : ptr (string → string)	

Reconciling Polymorphism and References

- Type system fails to prevent a type error!
- Common solution:
  - value restriction: generalize only the type of values!
    - easy to use, simple proof of soundness
- X Features ⇒ X^2 Complication
- To see what went wrong we needed 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
    - [In 1994], the language was renamed "**Java**" after a trademark search revealed that the name "Oak" was used by a manufacturer of video adapter cards.

Recall Goal #2

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

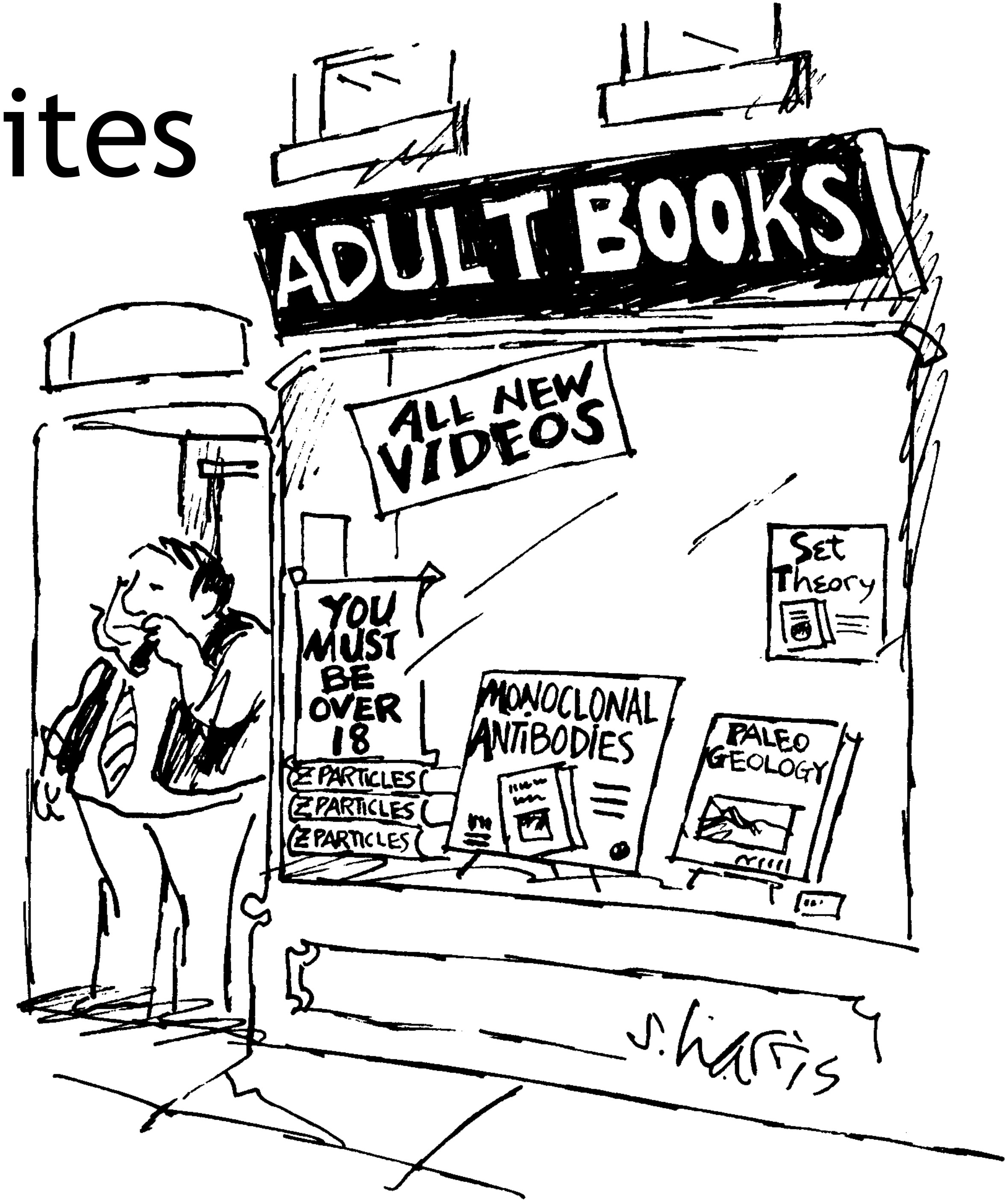
Goal #3

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

Final Goal: Fun

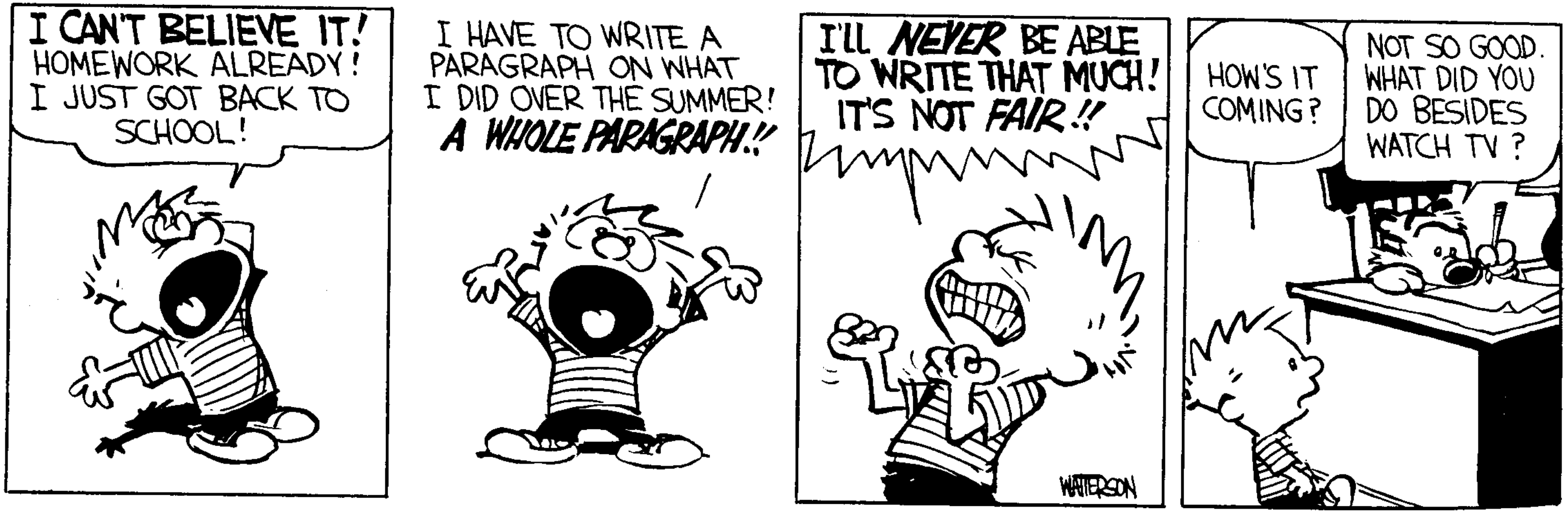
Prerequisites

- Undergraduate compilers course
  - Not always
- “Mathematical maturity”



Assignments

- Short Homework Assignments (4)
- Long Homework Assignment (1)
- Daily Reading (~2 papers per class)
- **Final Project**


I CAN'T BELIEVE IT! HOMEWORK ALREADY! I JUST GOT BACK TO SCHOOL!
I HAVE TO WRITE A PARAGRAPH ON WHAT I DID OVER THE SUMMER! A WHOLE PARAGRAPH!!
I'LL NEVER BE ABLE TO WRITE THAT MUCH! IT'S NOT FAIR!!
WATTERSON
HOW'S IT COMING?
NOT SO GOOD. WHAT DID YOU DO BESIDES WATCH TV?

Homework Problem Sets

- Some material can be “mathy”
- Much like Calculus, practice is handy
- Short: ~3 theory + 1 coding per HW
- You have one week to do each one
- Long: analysis of real C programs
- NB: I will offer suggestions and comments on your English prose.

Final Project

- Literature survey, implementation project, or research project
- Write a 10-page paper (a la PLDI)
- Give a 10-15 minute presentation
- On the topic of your choice
  - I will help you find a topic (many examples)
  - Best: integrate PL with your current research

How Hard Is This Class?

This Shall Be Avoided

In 1930, the Republican-controlled House of Representatives, in an effort to alleviate the effects of the... Anyone? Anyone? ... the Great Depression, passed the ... Anyone? Anyone? The tariff bill? The Hawley-Smoot Tariff Act? Which, anyone? Raised or lowered? ... raised tariffs, in an effort to collect more revenue for the federal government. Did it work? Anyone? Anyone know the effects?

Key Features of PL

Now that the NDA is up, we're seeing a lot of people out there unhappy with the D&D Online experience we've delivered.
Let's take a look at our key features list, see if we didn't miss something.
iLOV
NERF THI
Alright, dungeons. Where are we at?
Dungeons & Dragons Online key Features:
-Dungeons
-Dragons
They're in, sir.
Good. Glad to hear it. And dragons?
Dragons? Come on, guys. Don't tell me we forgot *dragons*.
I think... *Jim* was on Dragons.
No, no — I was on Dungeons.
©2006 Mike Krahulik and Jerry Holkins

Programs and Languages

- Programs
  - What are they trying to do?
  - Are they doing it?
  - Are they making some other mistake?
  - Were they hard to write?
  - Could we make it easier?
  - Should you run them?
  - How should you run them?
  - How can I run them faster?

Programs and Languages

- Languages
  - Why are they annoying?
  - How could we make them better?
  - What tasks can they make easier?
  - What cool features might we add?
  - Can we stop mistakes before they happen?
  - Do we need new paradigms?
  - How can we help out *My Favorite Domain*?

Common PL Research Tasks

- Design a new language feature
- Design a new type system / checker
- Design a new program analysis
- Find bugs in programs
- (Help people to) Fix bugs in programs
- Transform programs (source or assembly)
- Interpret and execute programs
- Prove things about programs
- Optimize programs

Grand Unified Theory

- Design a new type system
- Your type-checker becomes a bug-finder
- No type errors ⇒ proof program is safe
- Design a new language feature
- To prevent the sort of mistakes you found
- Write a source-to-source transform
- Your new feature works on existing code

CS 615 - Core Topics

- Operational semantics
- Type theory
- Verification conditions
- Abstract interpretation
- Lambda Calculus
- Type systems

"SO, BY A VOTE OF 8 TO 2 WE HAVE DECIDED TO SKIP THE INDUSTRIAL REVOLUTION COMPLETELY, AND GO RIGHT INTO THE ELECTRONIC AGE."

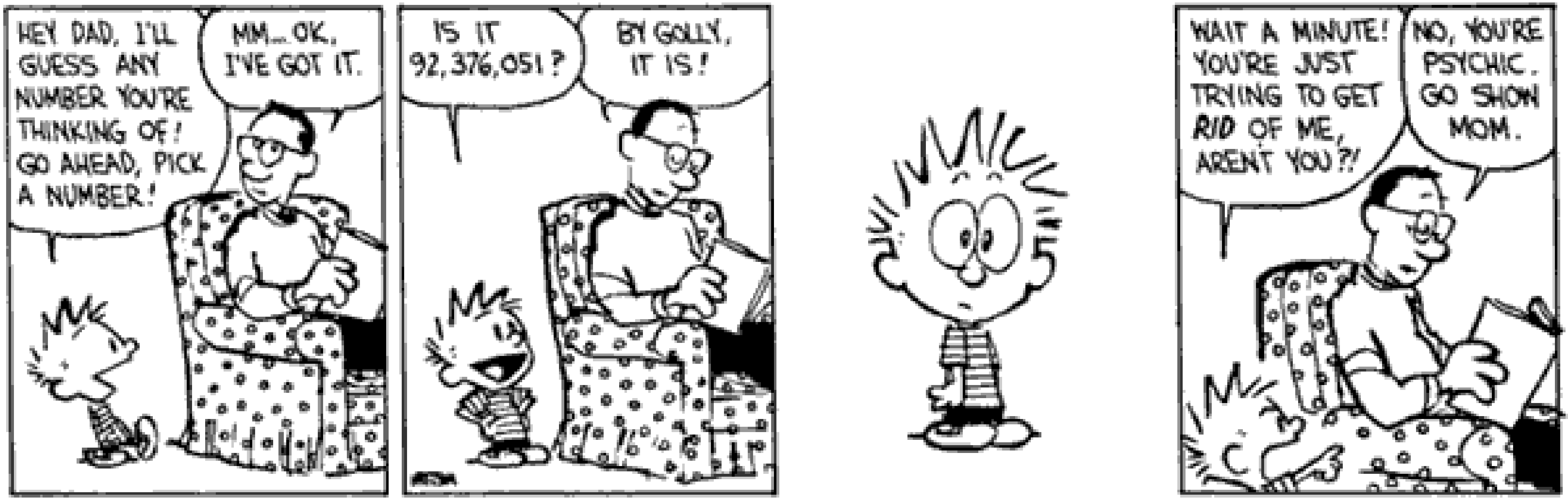
Special Topics

- Object-Oriented Languages
- Software Model Checking
- Type Systems for Resource Management
- Automated Deduction / Theorem Proving

- What do you want to hear about?

First Topic: Model Checking

- Verify critical properties of software or find bugs
- Take an important program (e.g., a device driver)
- Merge it with a property (e.g., no deadlocks, asynchronous IRP handling, BSD sockets, database transactions, …)
- Transform the result into a *boolean program*
  - Same control flow, but only boolean variables
- Use a model checker to explore the resulting *state space*
  - Result 1: program provably satisfies property
  - Result 2: program violates property *right here on line 92,376!*


HEY DAD, I'LL GUESS ANY NUMBER YOU'RE THINKING OF! GO AHEAD, PICK A NUMBER!
MM... OK, I'VE GOT IT.
IS IT 92,376,051?
BY GOLLY, IT IS!
WAIT A MINUTE! YOU'RE JUST TRYING TO GET RID OF ME, AREN'T YOU?!
NO, YOU'RE PSYCHIC. GO SHOW MOM.

Example Program

```
Example ( ) {
    do{
        lock();
        old = new;
        q = q->next;
        if (q != NULL){
            q->data = new;
            unlock();
            new ++;
        }
    } while(new != old);
    unlock();
    return;
}
```

Is this program correct?

Example Program

```
Example ( ) {
    do{
        lock();
        old = new;
        q = q->next;
        if (q != NULL){
            q->data = new;
            unlock();
            new ++;
        }
    } while(new != old);
    unlock();
    return;
}
```

Is this program correct?

What does correct mean?

- Doing no evil?
- Doing some good?

How do we determine if a program is correct?

Verification by Model Checking

```
Example ( ) {
1: do{
         lock();
         old = new;
         q = q->next;
2:       if (q != NULL){
3:            q->data = new;
              unlock();
              new ++;
         }
4: } while(new != old);
5:    unlock();
      return;
}
```

1. (Finite State) Program
2. State Transition Graph
3. Reachability

- Pgm → Finite state model
- State explosion

+ State Exploration
+ Counterexamples

Precise [SPIN, SMV, Bandera, JPF]

For Our Next Exciting Episode

- See webpage under “Lectures”
- Read the two articles
- Peruse the optional readings

THE WB
STAR WARS
Fresh tonight at 8/7 c
Katya, we're *rebels.* I won't have you dating a junior officer!
You're just a *clone* of my dad – I don't have to do what you say! And Brian and I are *going* to prom.
On Coruscant, capital of the Empire!
Katya... I just got off the comlink with Medical Frigate X7-T. Brian was killed in a speeder bike accident!
That can't be... I'm pregnant with his Imperial child!
The WB TUESDAY
Next week, New Found Glory guest stars on a very special Star Wars...
©2005 Mike Krahulik and Jerry Holkins