CSE 230:
Principles of Programming Languages

SLIDES BY ME

… What are we studying in this course?

This course is NOT about…

Rust vs. Python: Which is better?

How to make your own new programming language

What language you should implement your next project in

… although all of these are related!

I need a program that gets student information from the school DB and sorts them by grades

```rust
#[derive(Debug, Serialize, Deserialize, sqlx::FromRow)]
struct Student {
    id: i32,
    name: String,
    grade: f64,
}

#[tokio::main]
async fn main() -> Result<(), sqlx::Error> {
    // 1. Establish the connection pool
    let pool = PgPoolOptions::new()
        .max_connections(5)
        .connect("postgres://username:password@localhost/school_db")
        .await?;
```

Allocate stack, push variable onto stack, route network request…

I need a program that gets student information from the school DB and sorts them by grades

≠

Allocate stack, push variable onto stack, route network request…

How do I make sure this guy is doing the right thing?

≠

Allocate stack, push variable onto stack, route network request…

Actually, how do I understand what this guy is doing at all?

≠

Allocate stack, push variable onto stack, route network request…

First allocate a stack, then push, then I need to route…

≠

Allocate stack, push variable onto stack, route network request…

≠

Allocate stack, push variable onto stack, route network request…

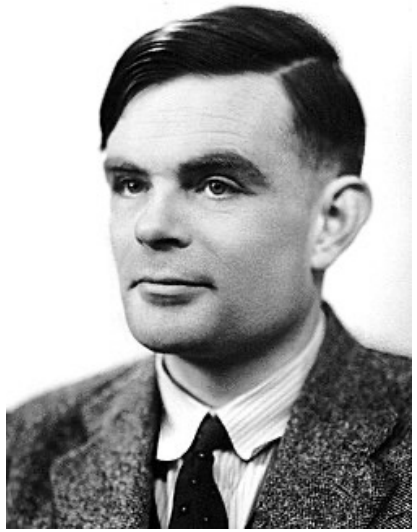
I need a program that gets student information from the school DB and sorts them by grades

Need something to bridge this gap

Allocate stack, push variable onto stack, route network request…

Kurt Godel
1931: Incompleteness theorems

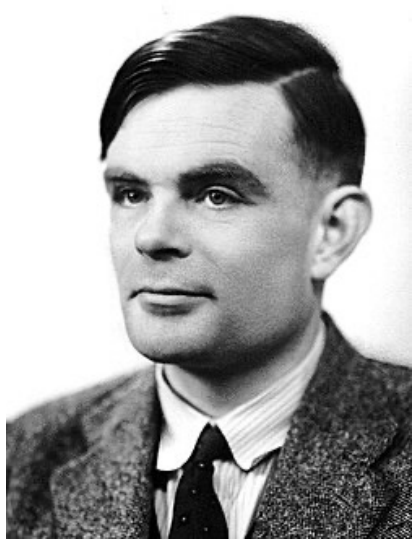
Alonzo Church
1936: λ-Calculus



Alan Turing
1936: Turing Machines

Kurt Godel
1931: Incompleteness theorems

Alonzo Church
1936: λ-Calculus
“What kinds of things can we compute with functions?”

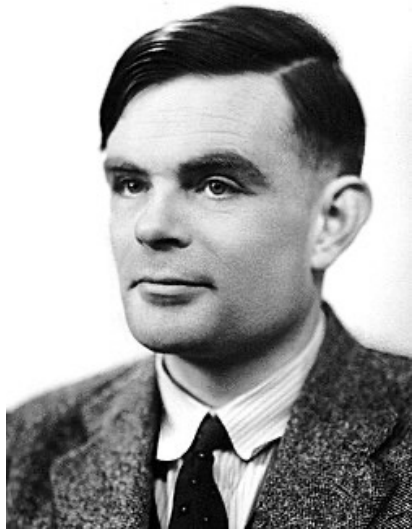


Alan Turing
1936: Turing Machines
“What kinds of things can we compute with a CPU and memory?”

Kurt Godel
1931: Incompleteness theorems

Alonzo Church
1936: λ-Calculus
“What kinds of things can we compute with functions?”

=



Alan Turing
1936: Turing Machines
“What kinds of things can we compute with a CPU and memory?”

Haskell Curry

Curry-Howard Isomorphism

William Howard

Mathematical		**Computer**
Propositions	⇔	Types
Proofs	⇔	Programs
Simplification	⇔	Computation

Program analysis

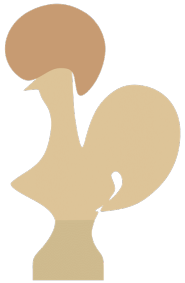
Program verification

Model checking

Program semantics

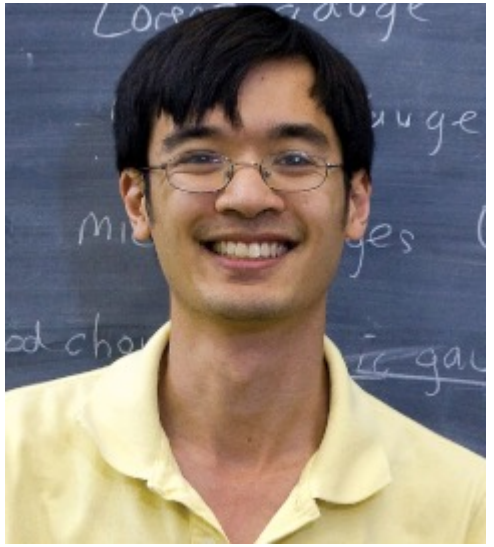
Type theory

Compilers & Optimizations



Mathematical		**Computer**
Propositions	⇒	Types
Proofs	⇒	Programs
Simplification	⇒	Computation

LEAN
THEOREM PROVER



Mathematical		Computer
Propositions	⇐	Types
Proofs	⇐	Programs
Simplification	⇐	Computation

What do we want to learn from all this?

I need a program that gets student information from the school DB and sorts them by grades

Need something to bridge this gap

Allocate stack, push variable onto stack, route network request…

I need a program that gets student information from the school DB and sorts them by grades

Computer behavior
= Math

Allocate stack, push variable onto stack, route network request…

I need a program that gets student information from the school DB and sorts them by grades

Humans = Relatively good at math

Computer behavior = Math

Allocate stack, push variable onto stack, route network request…

I need a program that gets student information from the school DB and sorts them by grades

A little mathematics can bridge this gap

Allocate stack, push variable onto stack, route network request…

Program that gets student information from the school DB

Program Verification

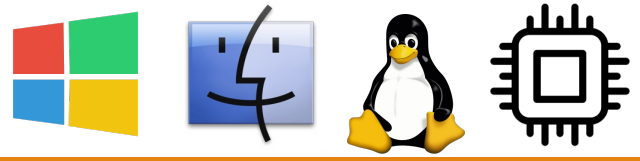
```rust
#[tokio::main]
async fn main() -> Result<(), sqlx::Error> {
    // 1. Establish the connection pool
    let pool = PgPoolOptions::new()
        .max_connections(5)
        .connect("postgres://username:password@localhost/school_db")
        .await?;
```

Compiler Verification

```assembly
sort_students:
    ucomisd %xmm1, %xmm0       # Compare Grade A and Grade B
    ja      done               # If A > B, they are already in order (jump above

    # If B > A, swap the pointers (simplified logic)
    movq    %rax, %rcx         # Move A to temp register
    movq    %rbx, %rax         # Move B to A's position
    movq    %rcx, %rbx         # Move temp (A) to B's position
```

OS / Hardware Verification



Program that gets student information from the school DB

Program Verification

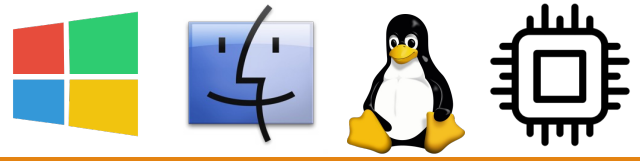
```rust
#[tokio::main]
async fn main() -> Result<(), sqlx::Error> {
    // 1. Establish the connection pool
    let pool = PgPoolOptions::new()
        .max_connections(5)
        .connect("postgres://username:password@localhost/school_db")
        .await?;
```

Compiler Verification

```
sort_students:
    ucomisd %xmm1, %xmm0       # Compare Grade A and Grade B
    ja      done               # If A > B, they are already in order (jump above

    # If B > A, swap the pointers (simplified logic)
    movq    %rax, %rcx         # Move A to temp register
    movq    %rbx, %rax         # Move B to A's position
    movq    %rcx, %rbx         # Move temp (A) to B's position
```

OS / Hardware Verification



Bridging this gap has practical merit too

1996, Ariane 5 rocket flight:

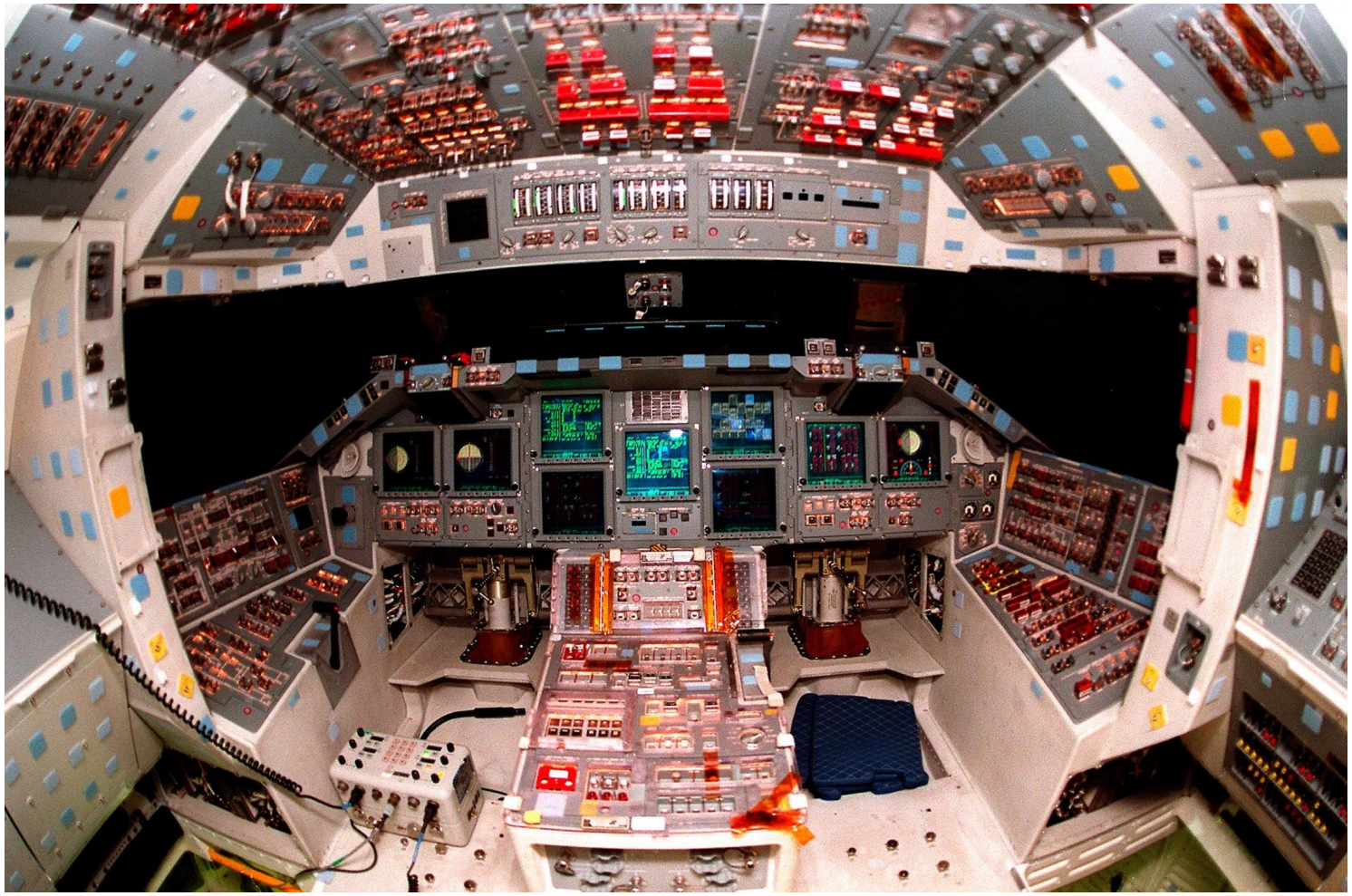
Crashed due to an integer overflow error causing an unhandled exception

To avoid these kinds of errors: you need to understand exactly what your program is doing

Norman H. Cohen (nco...@watson.ibm.com) wrote:
: The only programs I know of with deliberate memory leaks are those whose
: executions are short enough, and whose target machines have enough
: virtual memory space, that running out of memory is not a concern.
: (This class of programs includes many student programming exercises and
: some simple applets and utilities; it includes few if any embedded or
: safety-critical programs.)

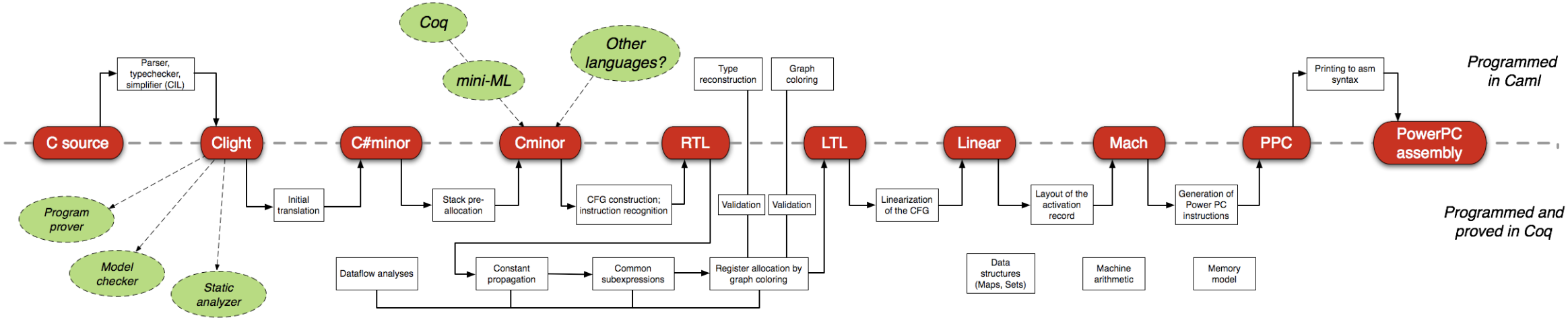
This sparked an interesting memory for me. I was once working with a customer who was producing on-board software for a missile. In my analysis of the code, I pointed out that they had a number of problems with storage leaks. Imagine my surprise when the customers chief software engineer said "Of course it leaks". He went on to point out that they had calculated the amount of memory the application would leak in the total possible flight time for the missile and then doubled that number. They added this much additional memory to the hardware to "support" the leaks. Since the missile will explode when it hits its target or at the end of its flight, the ultimate in garbage collection is performed without programmer intervention.

--
Kent Mitchell | One possible reason that things aren't
Technical Consultant | going according to plan is .....
Rational Software Corporation | that there never *was* a plan!

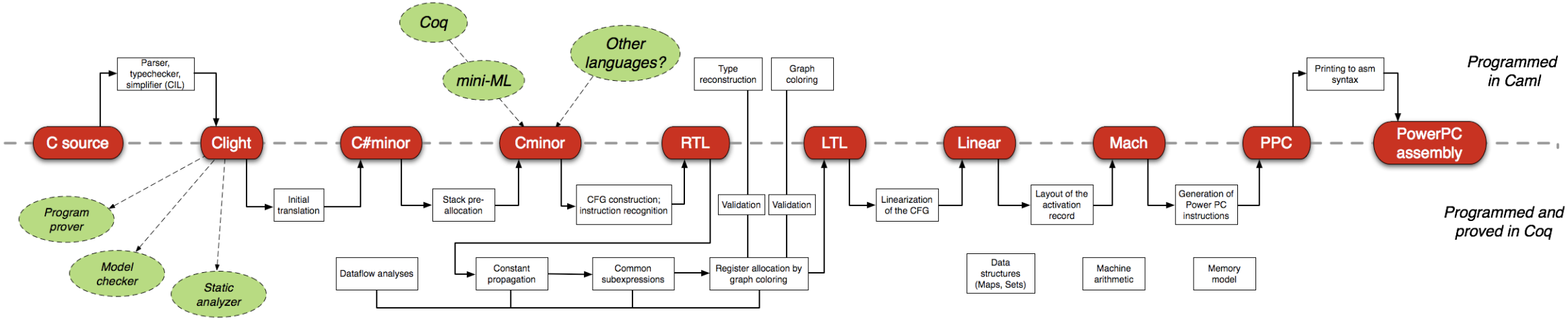


People are now verifying safety-critical software:
such as airplane control software, based on these ideas

All have internal teams working on the same ideas
to ensure their products are consistent



Open-Source example: CompCert, a fully verified compilation chain for C



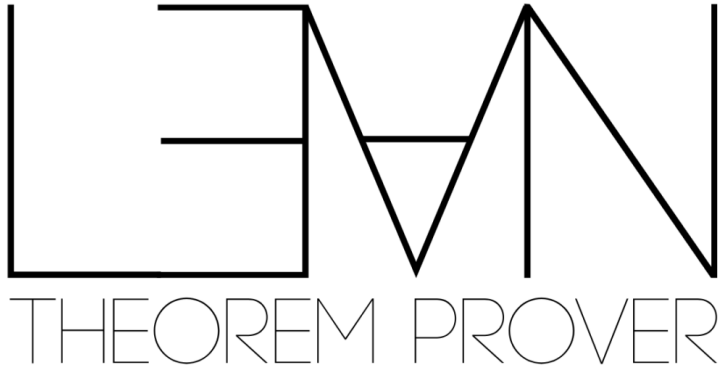
: < 10 bugs in the then-unverified portion (early 2011)

:79 bugs
(early 2011)

:202 bugs
(early 2011)

Open-Source example: CompCert, a fully verified compilation chain for C



Slated to become even more important

… so what are we learning in this course?

Mathematical abstractions! That …

- Allow you to express exactly what you want…
- Allow you to derive what the program is exactly doing…
- Prove that your intent and what program behavior are equivalent!

… so what are we learning in this course?

Mathematical abstractions! That …

- Allow you to express exactly what you want…
- Allow you to derive what the program is exactly doing…
- Prove that your intent and what program behavior are equivalent!

Medium of Instruction: Rocq (Coq)

Rocq is a *mechanized* proof assistant

- If you give it a proof, it can automatically check whether that proof is true or not
- Properties & Relations can be expressed as Rocq expressions & Theorems
- Theorems can be proved via tactics
- We will be using Rocq as the means for expressing properties and proofs

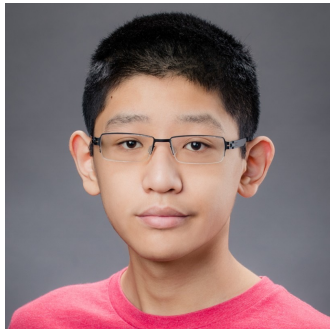
Course Logistics

Staff & Office Hours

Loris D’Antoni (Instructor)

- Office Hours: Thu, 12:30 PM ~ 1:20 PM (CSE 3214)

Timothy Zhou (TA)



- Office Hours: Mon / Wed, 1:00 PM ~ 2:00 PM (CSE 3260)

Administrivia

Lectures: Tue & Thu, 11:00 AM ~ 12:20 PM (HSS 1330)

Midterms: **04/28 (Tue)**, 11:00AM ~ 12:20 PM (HSS 1330)

Finals: **6/9 (Tue)**, 11:30 AM ~ 2:29 PM (HS 1330)

Course Website & Piazza

https://ucsd-cse230-loris.github.io/

https://piazza.com/ucsd/spring2026/cse230/home

Github ID Form

https://forms.gle/kfhm8aS4DTyTZEBT7

Grading

Programming Assignments (40 %)

- 5 assignments, 6 late days
- Your first homework will be assigned next week

Midterm (30%) and Final (30%)

- Paper exams

More info on course website

…without further ado, let’s dive into Rocq!