EECS 483: Compiler Construction

Lecture 25:
Parsing in Practice, Verified Compilation

April 16
Winter Semester 2025

Announcements

- Reminder: Assignment 5 Due Sunday
- Exam review on Monday
- Course Evaluations are now open. Please fill them out!

Debugging parser conflicts.
Disambiguating grammars.

LALRPOP DEMO

Poll

Is implementing a compiler difficult?

Is implementing a bug-free compiler difficult?

Is implementing a bug-free compiler important?

When Do Bugs Matter

"move fast and break things"

devtools

social media

weather app

hardware design

banking

medical software

aerospace software

smart contracts

bugs are cheap

bugs are expensive

The more expensive bugs are,
the more effort is justified in eliminating them

Expensive Bugs of History: Ariane 5

- The first launch of the European Space Agency's Ariane 5 rocket in 1996 ended with an explosion 37 seconds after launch

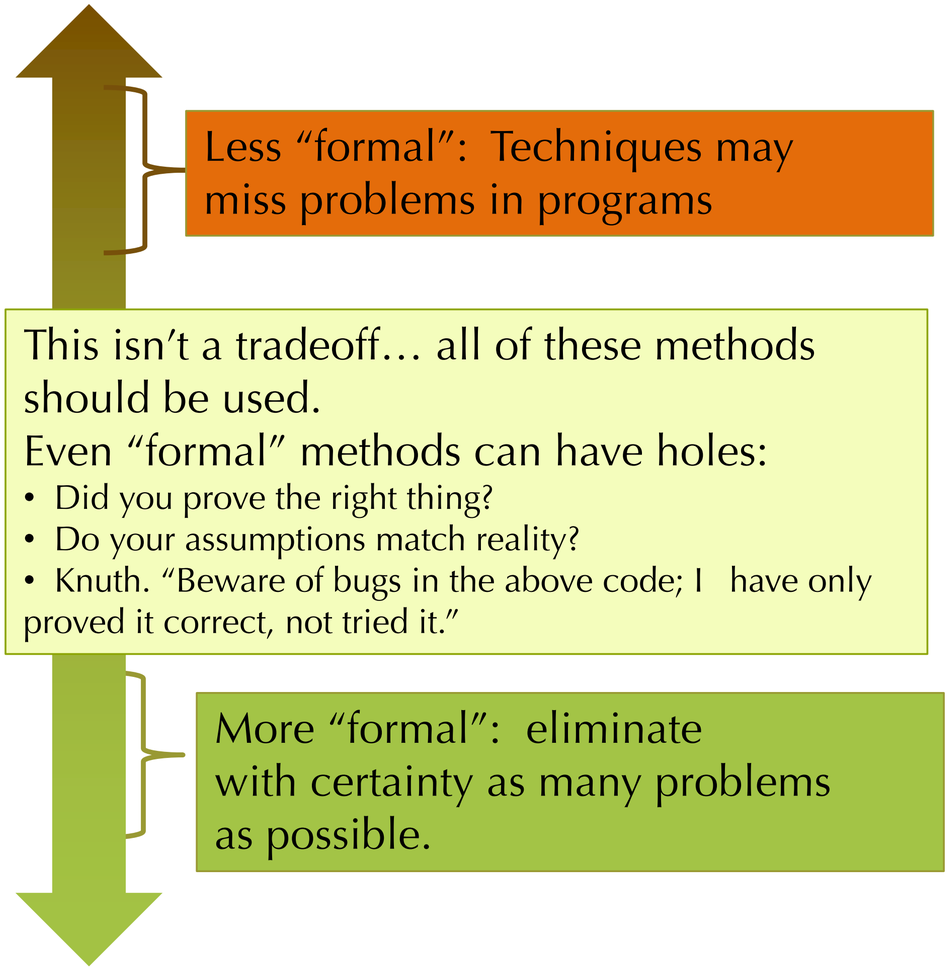
- Root cause: a conversion from a 64-bit float to a 16 bit int overflowed and the resulting exception was not handled. Led to junk data that triggered a self-destruct.

- Estimated cost: ~$750 million (inflation adjusted)

Details: https://dl.acm.org/doi/10.1145/251880.251992

Approaches to Software Reliability

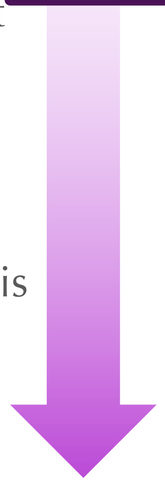
- Social
  - Code reviews
  - Extreme/Pair programming
- Methodological
  - Design patterns
  - Test-driven development
  - Version control
  - Bug tracking
- Technological
  - “lint” tools, static analysis
  - Fuzzers, random testing
- Mathematical
  - Sound programming languages tools
  - “Formal” verification



Goal: Verified Software Correctness

- Social
  - Code reviews
  - Extreme/Pair programmin
- Methodological
  - Design patterns
  - Test-driven development
  - Version control
  - Bug tracking
- Technological
  - “lint” tools, static analysis
  - Fuzzers, random testing
- Mathematical
  - Sound programming languages tools
  - “Formal” verification

Q: How can we move the needle towards mathematical software correctness properties?



Taking advantage of advances in computer science:

- Moore's law
- improved programming languages & theoretical understanding
- better tools: interactive theorem provers

Compilers are Essential Infrastructure

If formal verification methods are applied only to **source code** in e.g., C, then those guarantees are only valid if the compiler is bug-free.

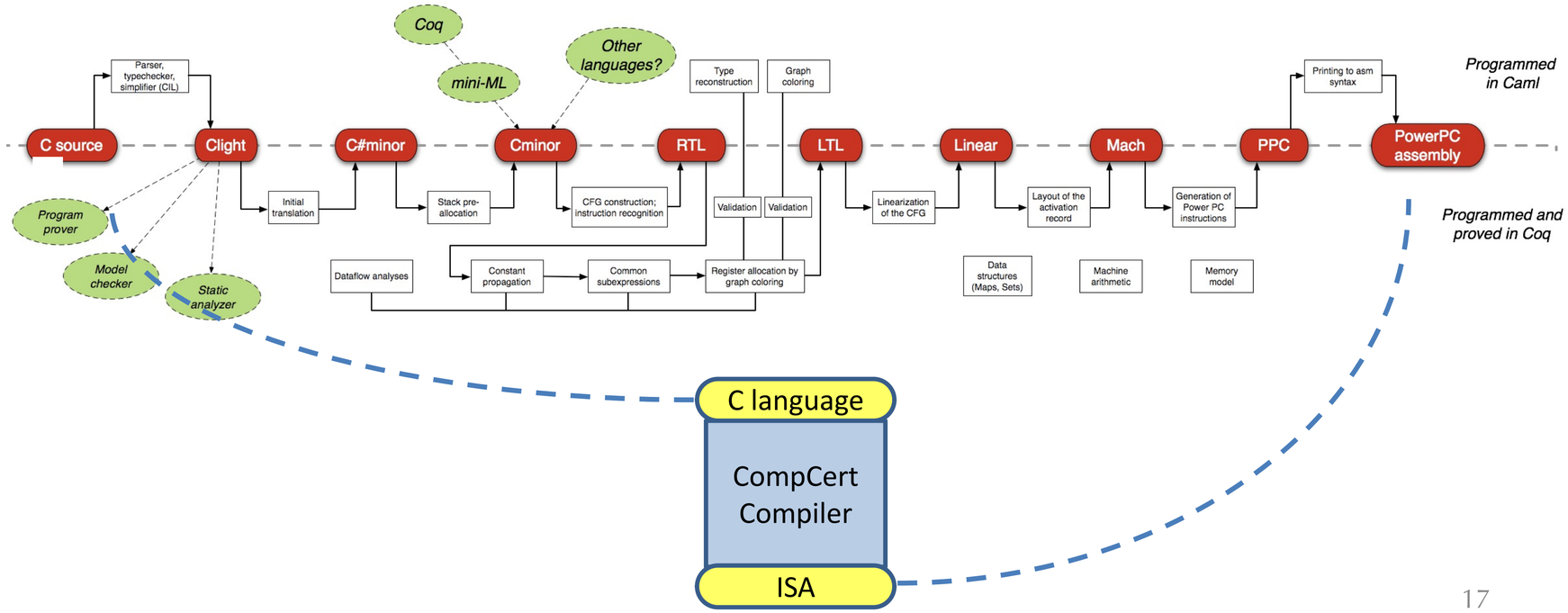
Best practice when using unverified compilers is to do analysis/auditing of the actual assembly code

Verifying the compiler: high impact, as it enables other verified software

CompCert – A Verified C Compiler

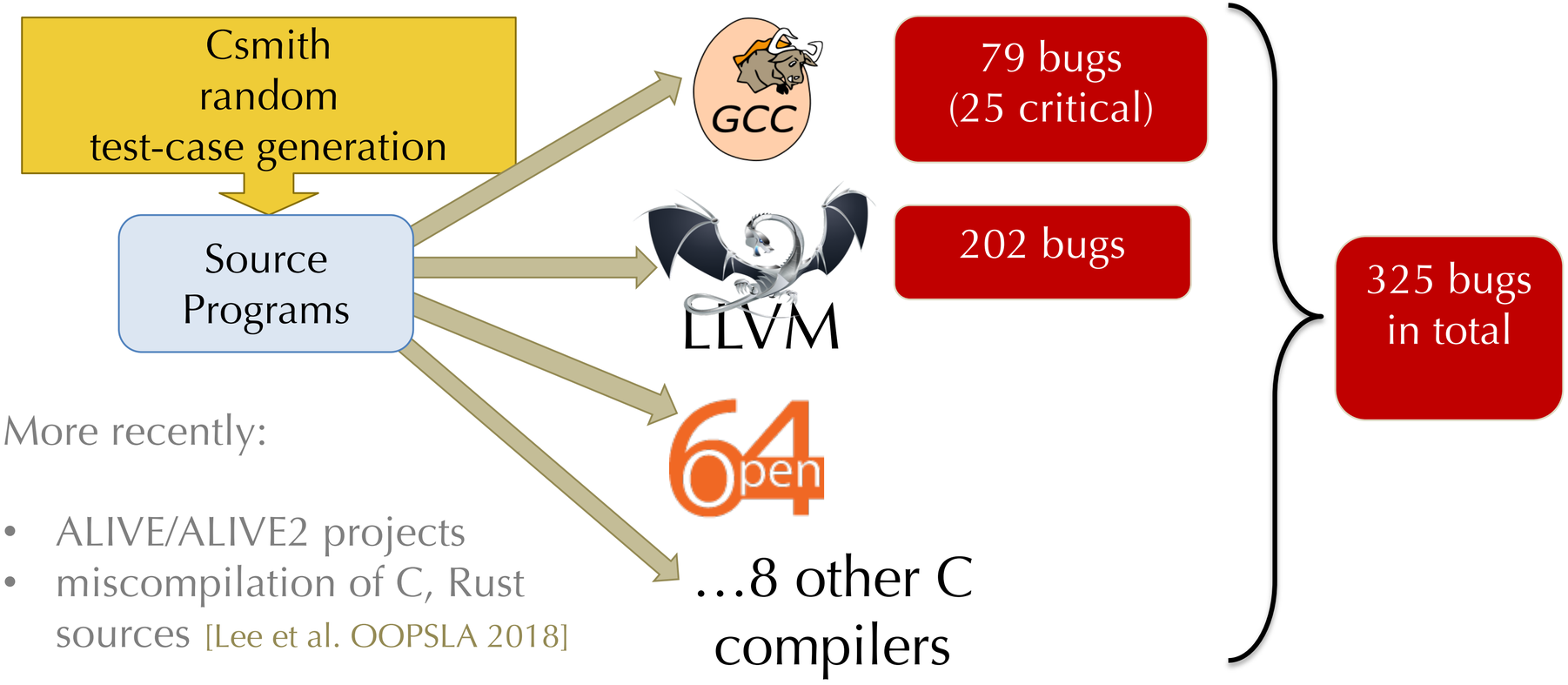
Xavier Leroy
INRIA

Optimizing C Compiler,
proved correct end-to-end
with machine-checked proof in Rocq



Compiler Bugs

[Regehr's group: Yang et al. PLDI 2011]



More recently:

- ALIVE/ALIVE2 projects
- miscompilation of C, Rust sources [Lee et al. OOPSLA 2018]

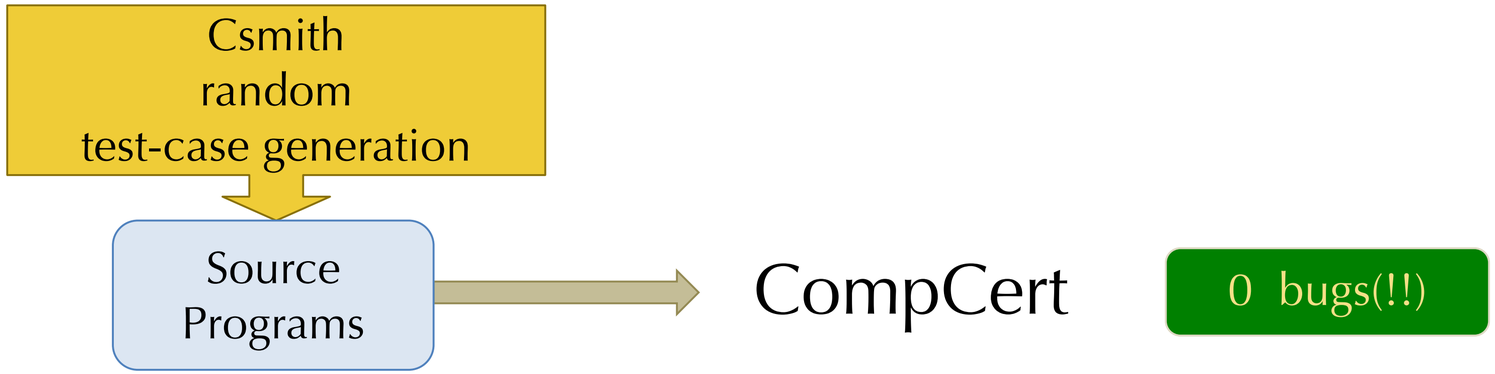
Finding and Understanding Bugs in C Compilers

Yang, Chen, Eide and Regehr, PLDI 2011

https://dl.acm.org/doi/10.1145/1993316.1993532

Csmith on CompCert?

[Yang et al. PLDI 2011]



Verification Works!

"The striking thing about our CompCert results is that the middle-end bugs we found in all other compilers are absent. As of early 2011, the under-development version of CompCert is the only compiler we have tested ***for which Csmith cannot find wrong-code errors***. This is not for lack of trying: we have devoted about six CPU-years to the task. ***The apparent unbreakability of CompCert supports a strong argument that developing compiler optimizations within a proof framework, where safety checks are explicit and machine-checked, has tangible benefits for compiler users.***"

– Regehr et. al 2011

Can it Scale?

- Use of theorem proving to verify “real” software is still considered to be the bleeding edge
of research.

Verified
Software
Toolchain

- **CompCert** – fully verified C compiler
  Leroy, INRIA
- **Vellvm** – formalized LLVM IR
  Zdancewic, Penn
- **Ynot** – verified DBMS, web services
  Morrisett, Harvard
- **Verified Software Toolchain**
  Appel, Princeton
- **Bedrock** – web programming, packet filters
  Chlipala, MIT
- **CertiKOS** – certified OS kernel
  Shao & Ford, Yale
- **CakeML** – certified compiler
- **SEL4** – certified secure OS microkernel
- **Kami** – verified RISCV architecture
- **DaisyNSF** – verified NFS file system
- …

Vellvm
verified
LLVM

CAKEML
A Verified Implementation of ML

AIRBUS

Microsoft

NASA

aws

Google

Meta

BEDROCK
Systems Inc

| galois |

Verified Compilers: How it's Made

- Compcert is implemented in the interactive theorem prover Rocq (https://rocq-prover.org/)
  - Rocq, and similar systems Lean, Agda, Idris are all based on **dependent type theory**, essentially they are "just" functional programming languages with an extremely fancy type system
  - The type system is so powerful that you can use it to write full functional correctness. Based on the "Curry-Howard correspondence" that unifies logical reasoning and programming

Demo:
verifying simple compilers in Agda