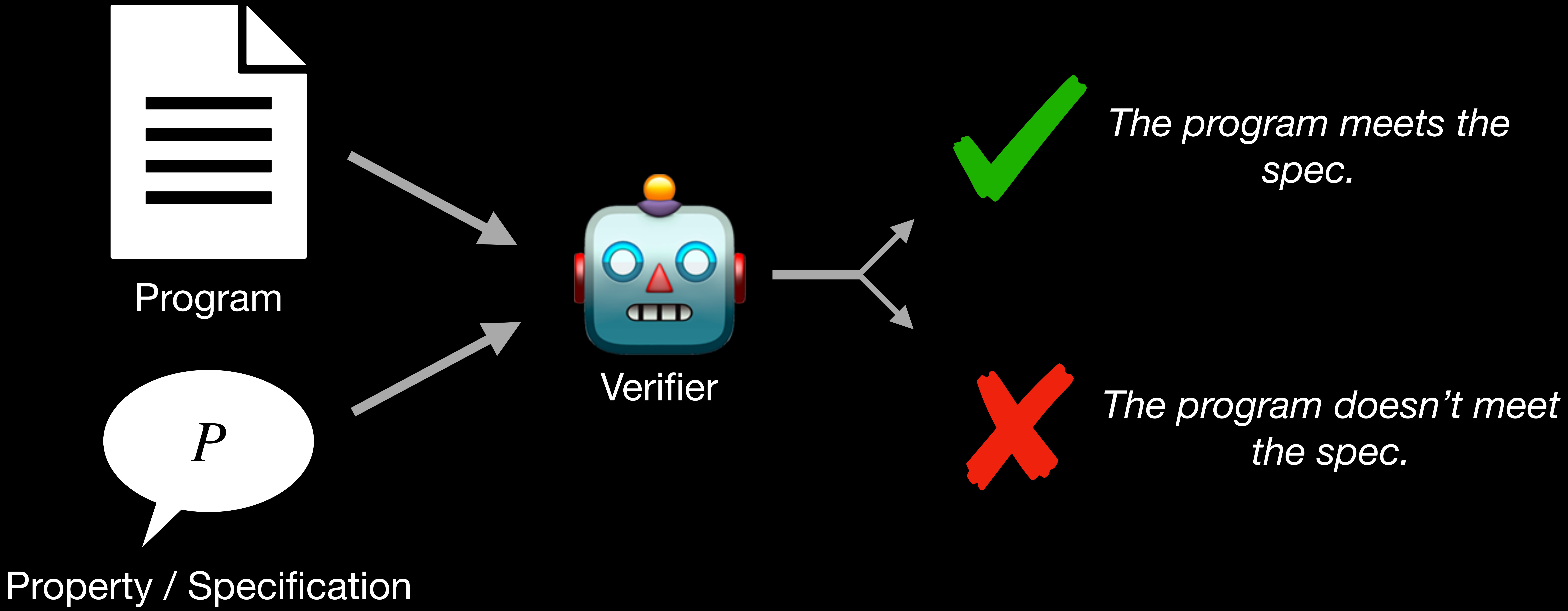
Software Verification

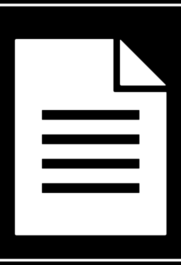
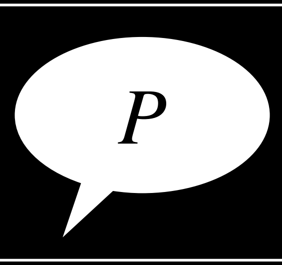
CS162: Programming Languages

Software Verification

Overview



Software Verification
Overview

	 P
Java	Will my program ever dereference a null pointer?
C/C++	Is there going to be memory leaks?
Smart contracts	Is someone else be able to to transfer money out of my account?
Self-driving cars	Will the system crash into pedestrians?
Rocket control	Are there potential integer overflows?

Software Verification

Overview

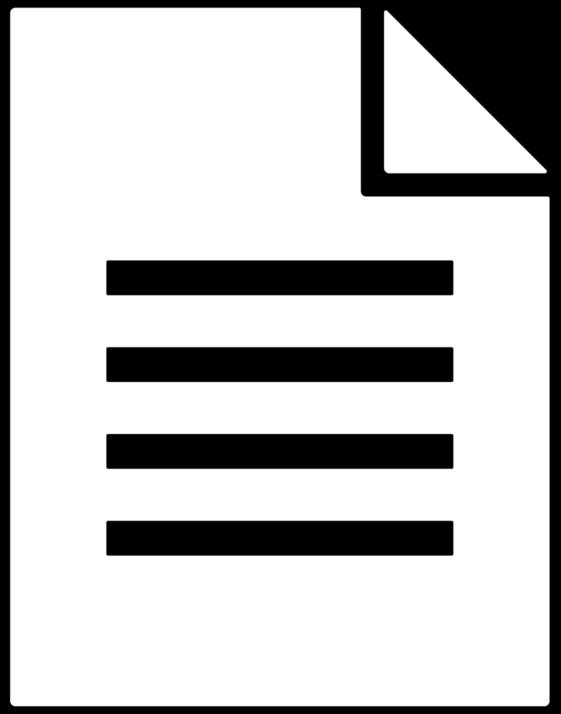
Ariane 5 rocket (June 4, 1996)

Exploded due to an overflow conversion from 64-bit to 16-bit int.

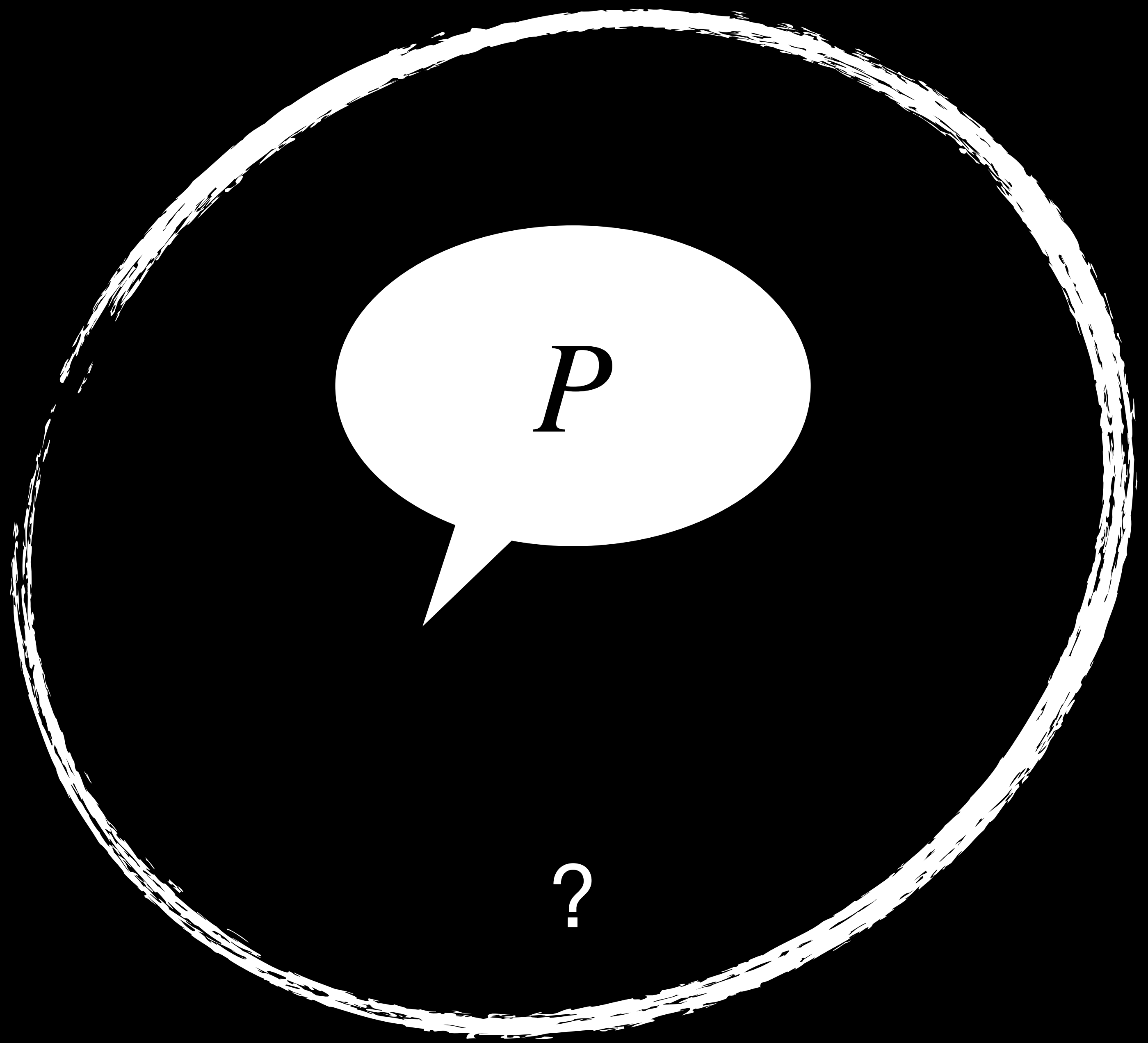
Rocket control	Are there potential integer overflows?

Type Checking

“Featherweight verification”

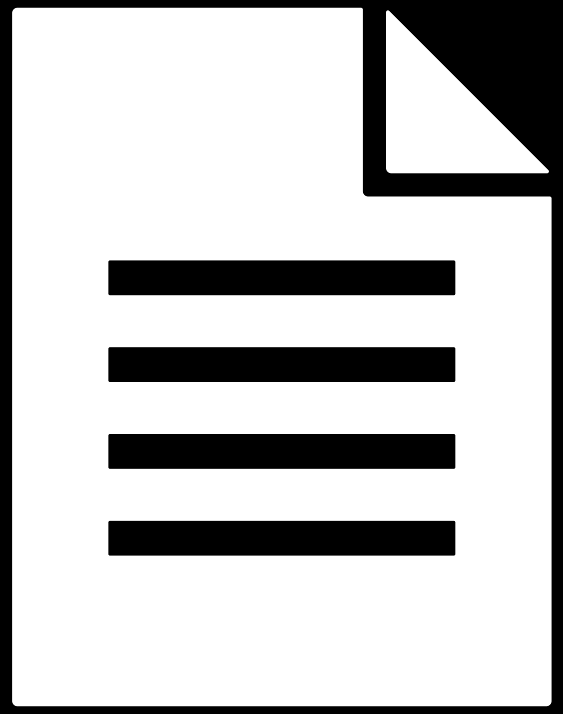


λ^+ program +
type annotations

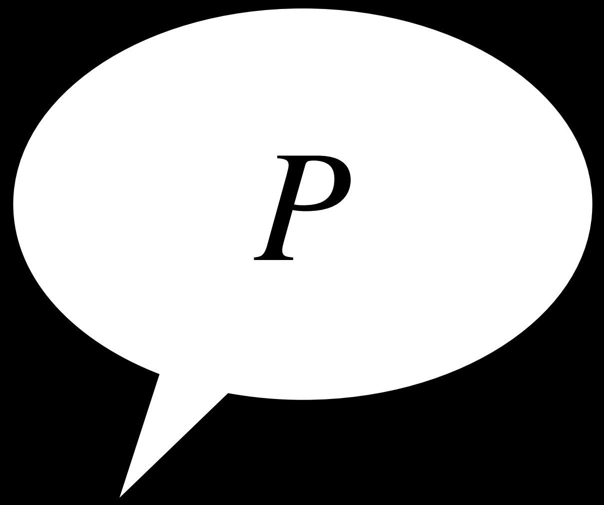


Type Checking

“Featherweight verification”



λ^+ program +
type annotations



Will the program get stuck?

How to verify more interesting properties?

Formal Proofs

“*Heavyweight* verification”

- Treat programs and properties as *mathematical objects*
- *Prove* those properties as you would prove a mathematical theorem.
- Have a program to *check* the proofs for you.

Formal Proofs

The Coq proof assistant

23. Formula for Pythagorean Triples

David Delahaye (in coq-contribs/fermat4):

```
Lemma pytha_thm3 : forall a b c : Z,
  is_pytha a b c -> Zodd a ->
  exists p : Z, exists q : Z, exists m : Z,
    a = m * (q * q - p * p) /\ b = 2 * m * (p * q) /\
    c = m * (p * p + q * q) /\ m >= 0 /\
    p >= 0 /\ q > 0 /\ p <= q /\ (rel_prime p q) /\
    (distinct_parity p q).
```

Formal Proofs

The Coq proof assistant

Coq has been used to certify many *safety-critical* systems:

- **CompCert**: certified C language compiler
- **CertiKOS**: certified concurrent OS kernel
- **Verifiable C**: program logic for C language

Formal Proofs

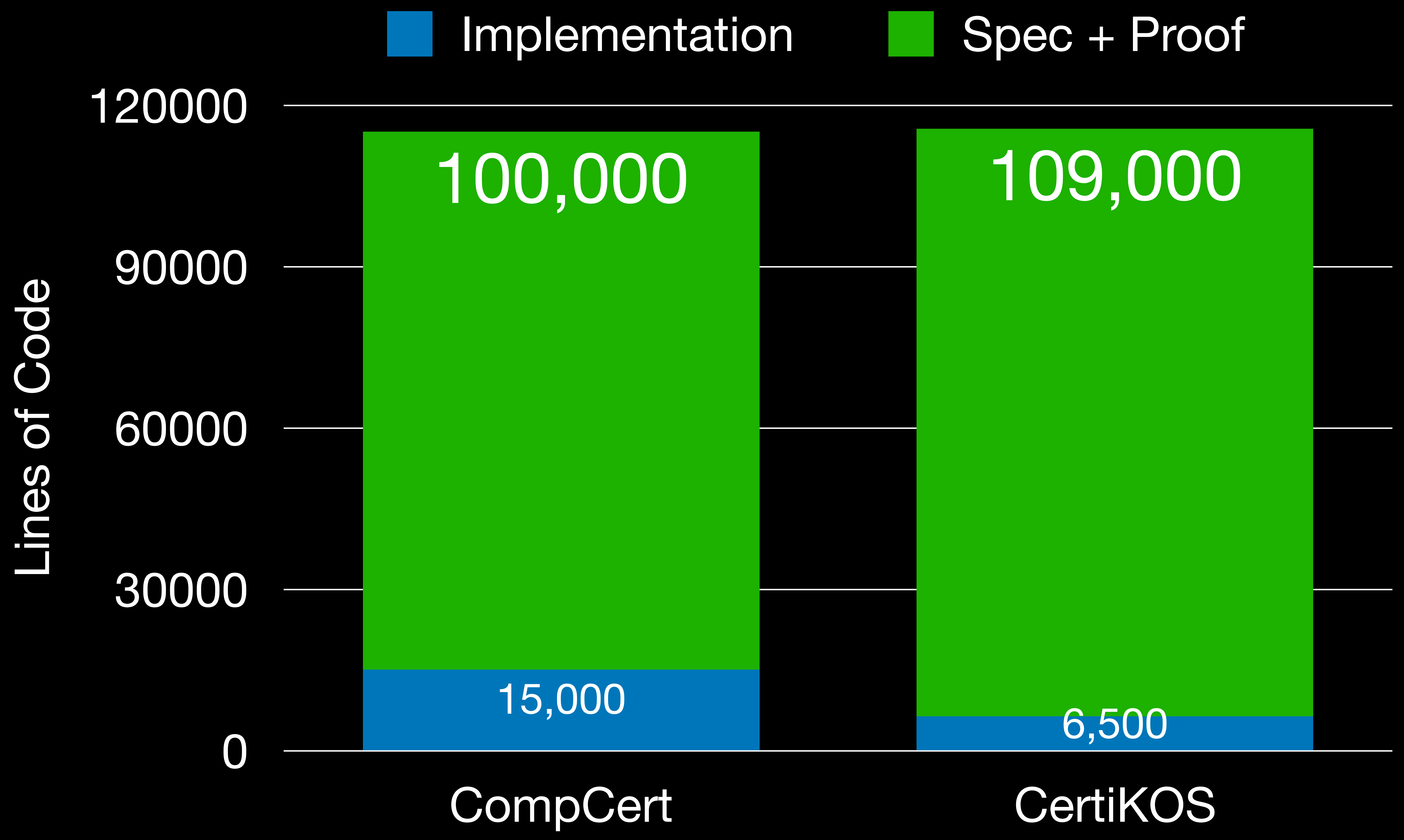
Problem 1: Development effort

Implementation Tests

…Previously

Formal Proofs

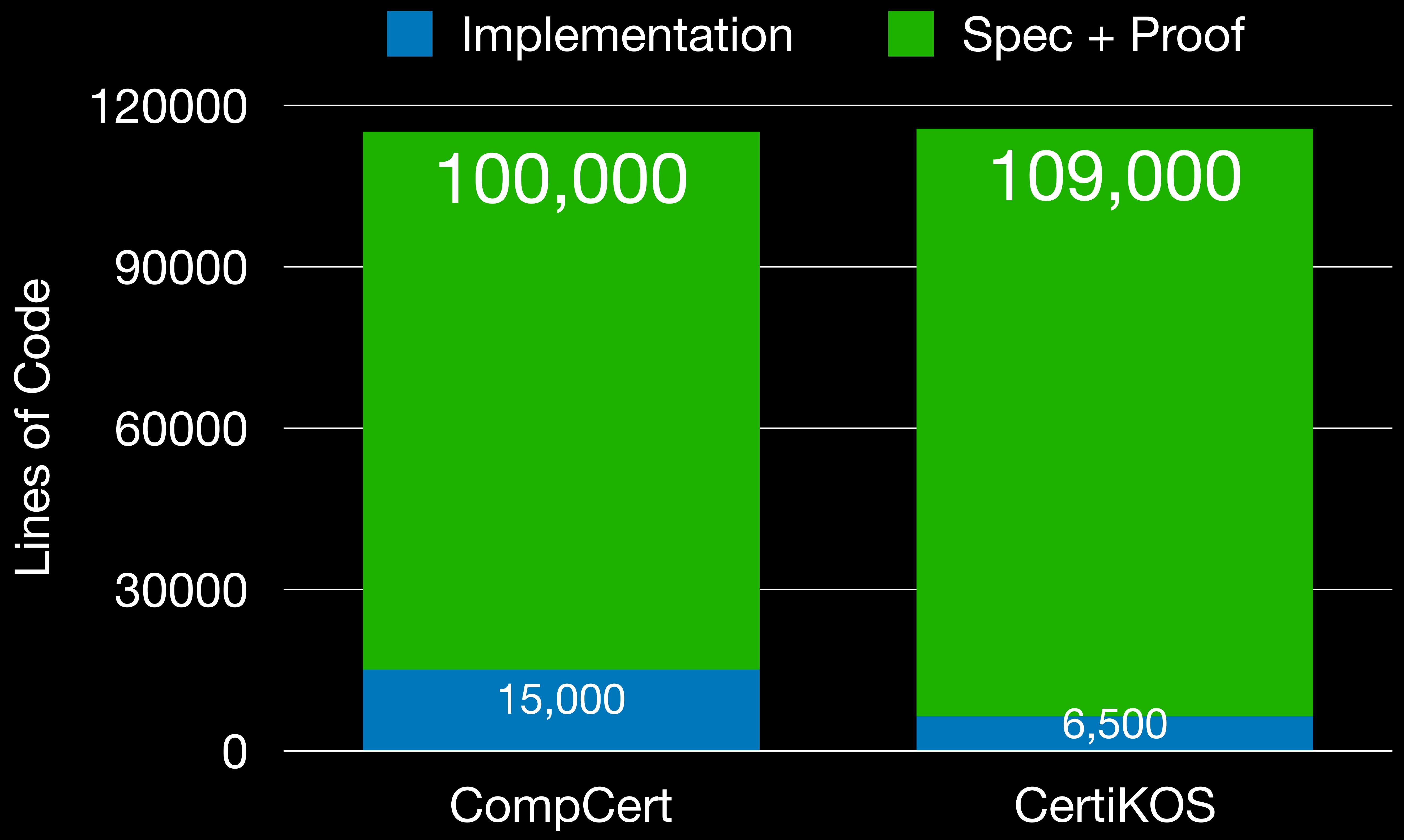
Problem 1: Development effort



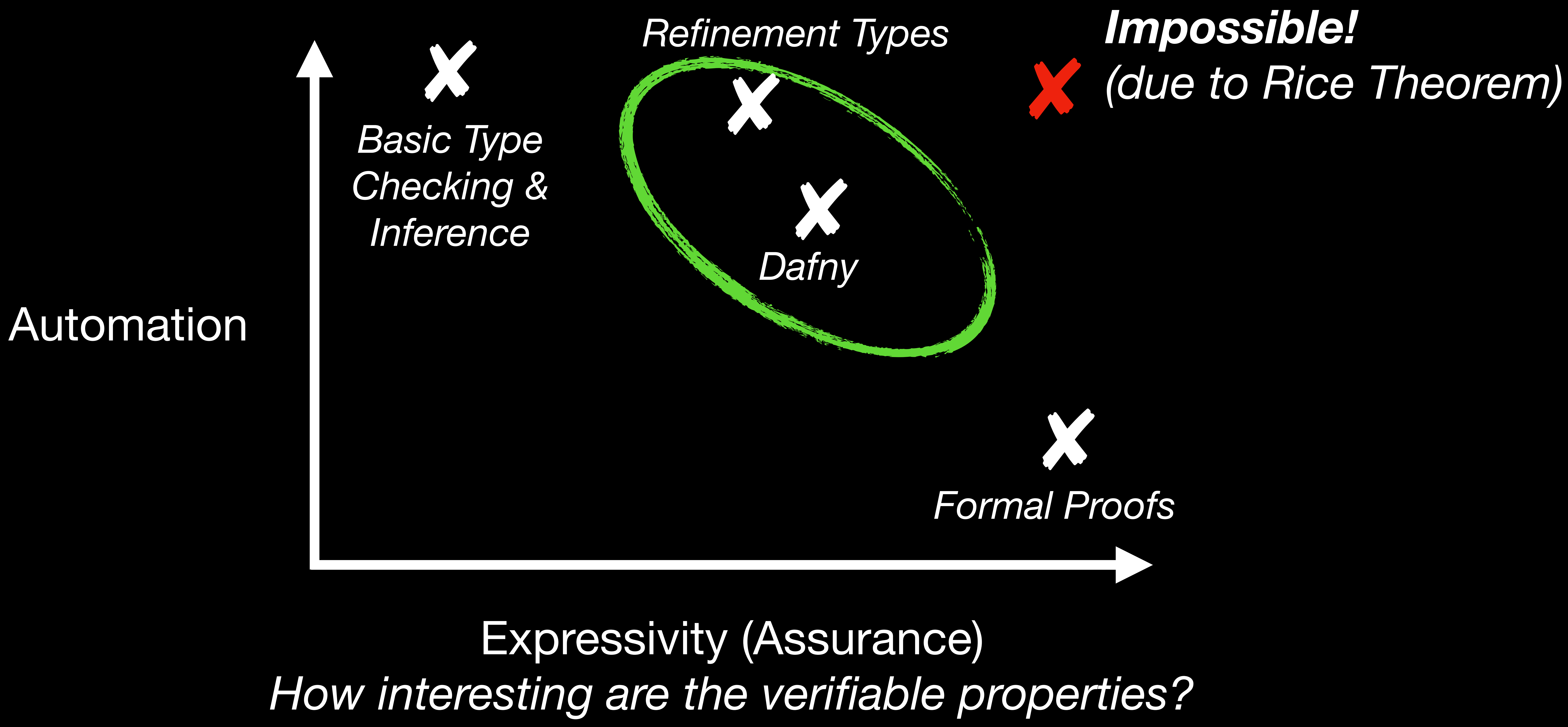
Need to write 10x more code.

Formal Proofs

Problem 2: Maintenance effort



New code breaks proofs!


Refinement Types
Impossible!
(due to Rice Theorem)
Basic Type
Checking &
Inference
Dafny
Automation
Formal Proofs
Expressivity (Assurance)
How interesting are the verifiable properties?

Dafny

Verification via pre- and post-conditions

```
method Abs(x: int) returns (y: int)
ensures 0 <= y
{
   if x < 0
      { return -x; }
   else
      { return x; }
}
```

https://ece.uwaterloo.ca/~agurfink/stqam/rise4fun-Dafny/

Dafny

Hoare logic (aka Floyd–Hoare logic)

Robert W. Floyd

Tony Hoare

Dafny

Hoare logic: The good

- Given a pre-condition P and a statement s, we can infer what effect s has on P, i.e., we infer a post-condition Q.
  - E.g., if P is $x > 0$, after executing $x := x + 1$, we know Q is $x > 1$.
- To check if a function satisfies some P and Q:
  - Propagate P from the beginning to the end of the function, obtaining Q_0.
  - Check if Q_0 implies Q.

Dafny

Hoare logic: The bad

- Caveat: Cannot propagate conditions through a loop!

$$
\begin{array}{l}
P = \{x = 0\} \\
while\ \ < condition > \\
\ \ x := x + 1; \\
Q = \{?\}
\end{array}
$$

- Need to manually provide *loop invariants* — formulas that summarize a loop's behavior.

- Loop invariants can be extremely tricky. ∃ tons of work that try to infer them automatically.

Refinement Types

```
average    :: [Int] -> Int
average xs = sum xs `div` length xs
```

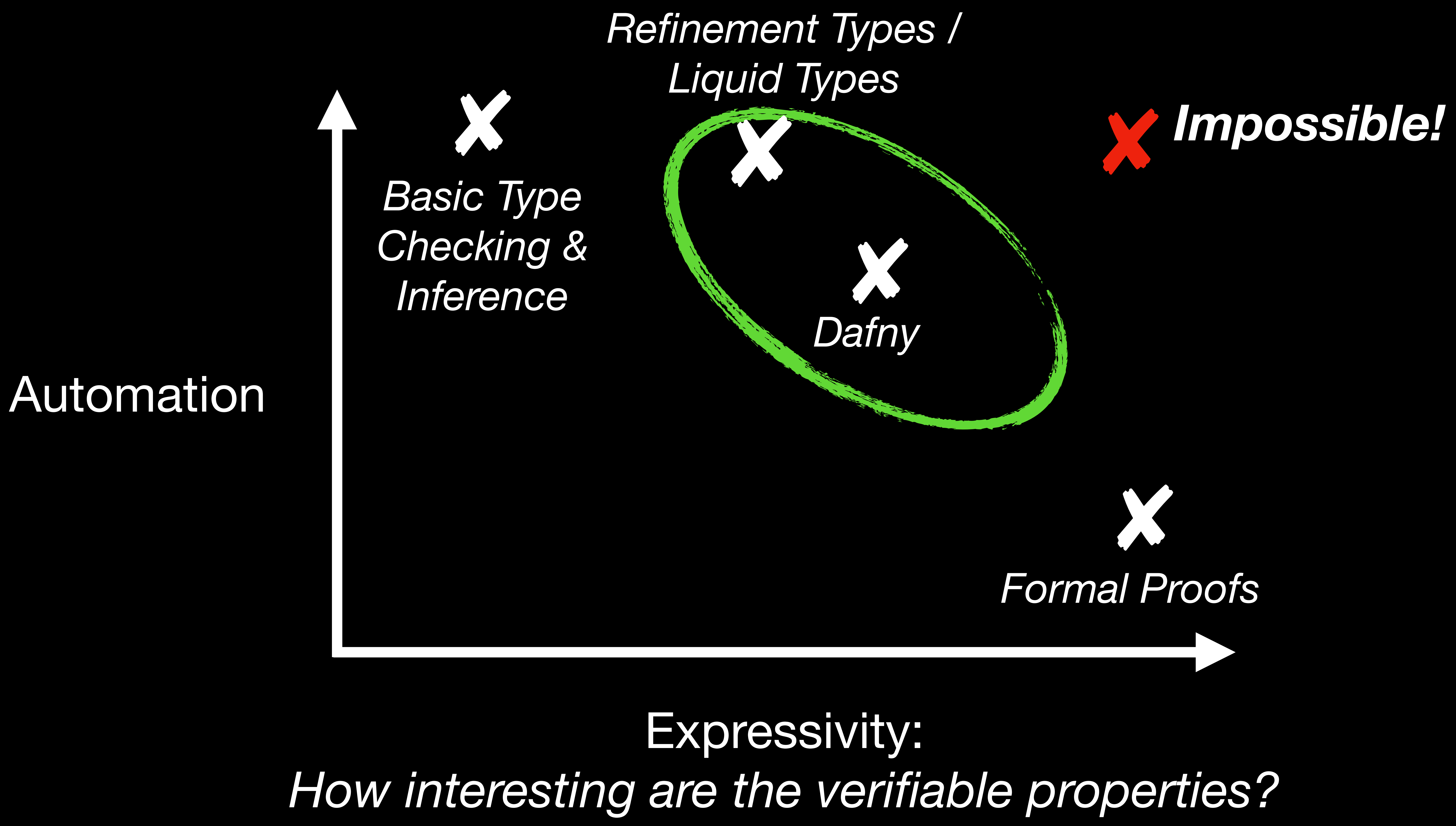
```
average    :: {v:[Int] | length(v) > 0} -> Int
average xs = sum xs `div` length xs
```

https://ucsd-progsys.github.io/liquidhaskell/

Refinement Types

= base type + refinement

- Encode *refined* information in the type system.
- **Expressivity**: Along with datatypes, allow us to express data structures whose invariants are enforced *statically.*
- **Automation**: A variant (called liquid types) admits *decidable* type inference.
- Successful in:
    - Verifying low-level C programs
    - Verifying deep learning models
    - Proving arithmetic overflow safety in smart contracts (from Prof. Feng's group!)
    - …



Refinement Types /
Liquid Types
Impossible!
Basic Type
Checking &
Inference
Dafny
Automation
Formal Proofs
Expressivity:
How interesting are the verifiable properties?

Further Reading

- Formal verification in Coq:
  - The Software Foundations series
- Dafny:
  - Tutorial
- Refinement/liquid types:
  - Patrick Rondon's PhD thesis
  - Tutorial on liquid types
- SMT solvers: Take 292C in spring!