Verus: program verification for practical engineering

Andrea Lattuada

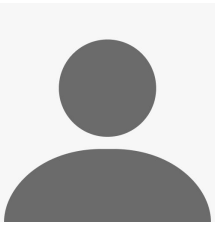
Travis Hance

Jay Bosamiya

Matthias Brun

Chanhee Cho

Hayley LeBlanc



Pranav Srinivasan

Reto Achermann

Tej Chajed

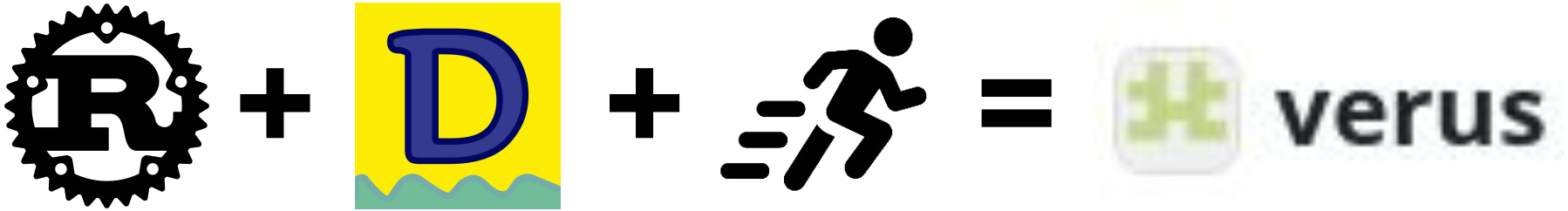
Chris Hawblitzel

Jon Howell

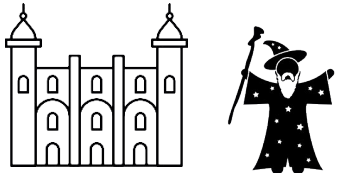
Jay Lorch

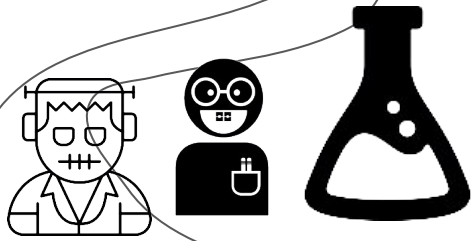
Oded Padon

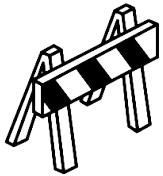
Bryan Parno


verus

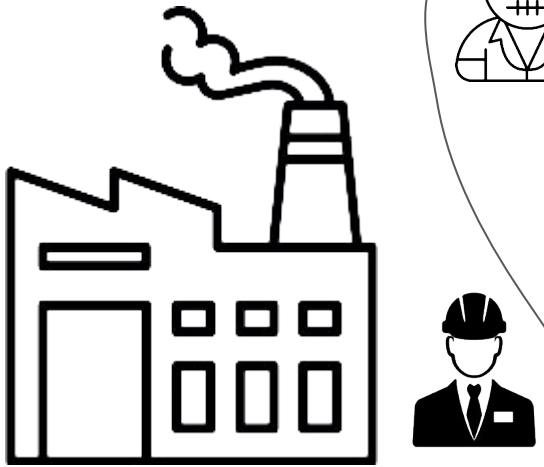
Why a new verifier?

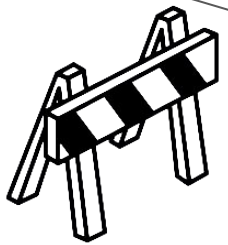






barriers to further research

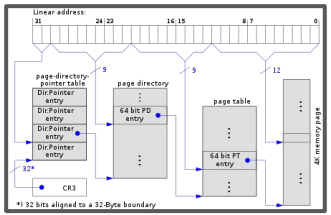




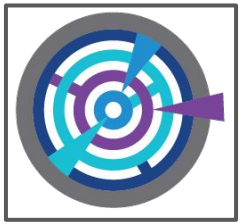
barriers to productive use

Design choices driven by observed burdens

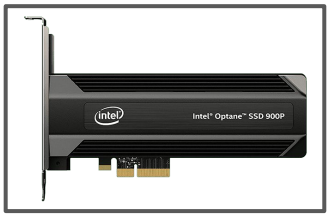
OS Page Table



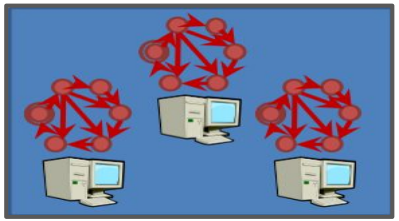
Write-Optimized Key-Value Store



Persistent-Memory Log

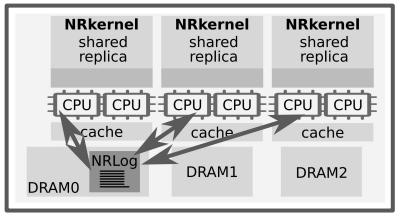


Distributed Key-Value Store



Concurrent Memory Allocator

NUMA-aware Replica Library



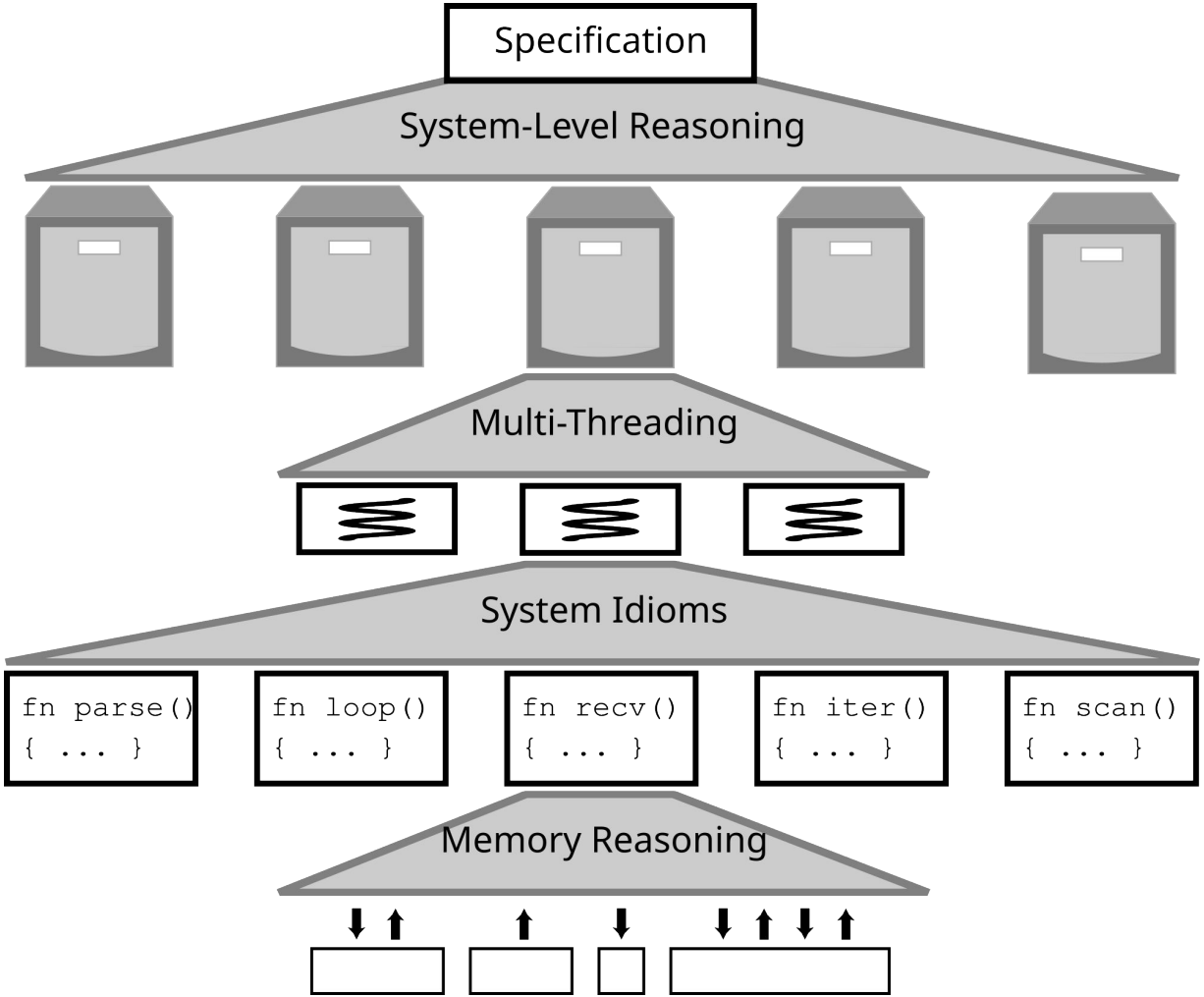
Automation: *faster* proofs

context: general SMT-based automation by default

problem: sluggish SMT queries

solution: efficiency-optimized verification condition generation

- easy wins: Rust ownership, aggressive context pruning
- tradeoffs: conservative triggers, total spec fn


Specification
System-Level Reasoning
Multi-Threading
System Idioms
fn parse() { ... }
fn loop() { ... }
fn recv() { ... }
fn iter() { ... }
fn scan() { ... }
Memory Reasoning

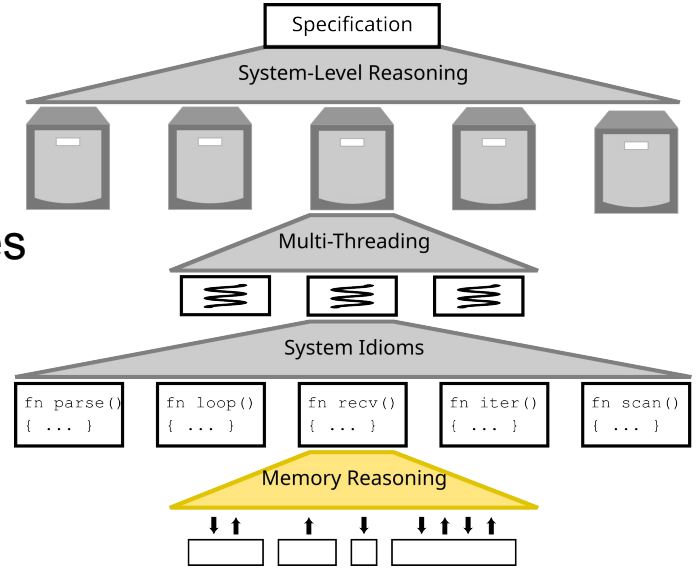
Memory reasoning

context: performant code mutates rich data structures

problem: framing-based reasoning
is objectively awful [OOPSLA22]

solution:

- Rust ownership by default
- Sophisticated borrow checker:
elegant syntax for common patterns



Systems-specific proof automation

context: performant systems code
relies on %, >>, constants

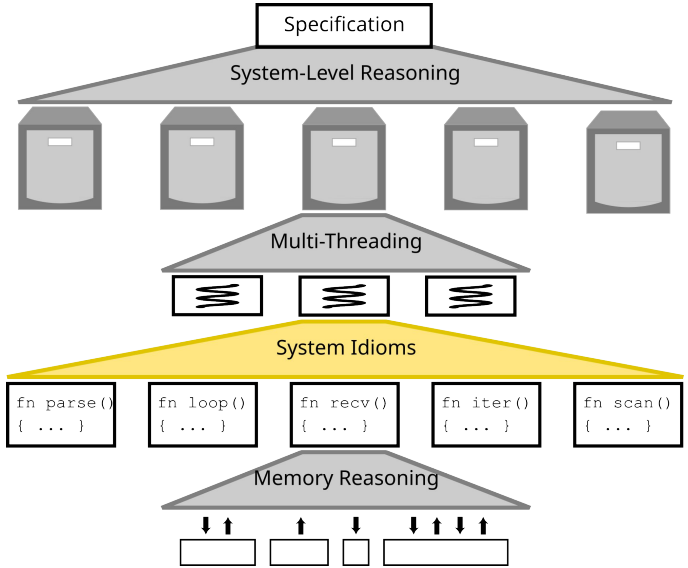
problem: mixed theories grieve Z3

solution: Local `assert_by` modes for

nonlinear (%),

bitvector (>>), and

assert-by-computation

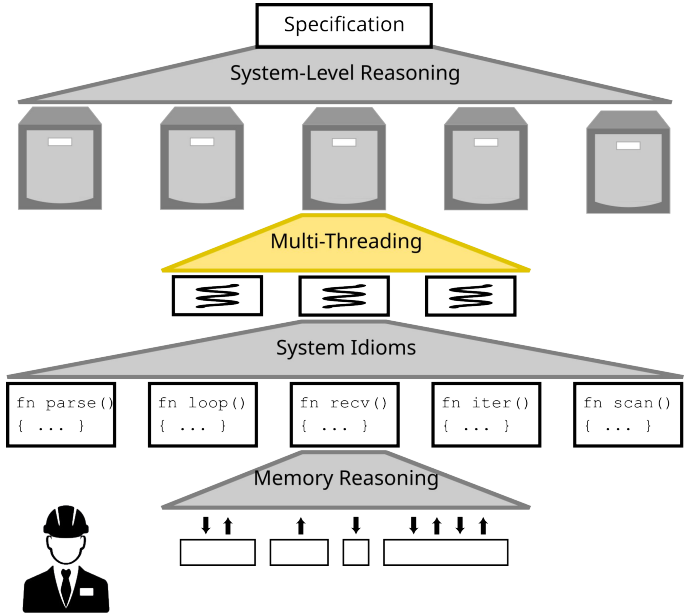


Thread concurrency

context: performant code exploits shared-memory threading

problem: separation logic is low-level

solution: VerusSync high-level "sharded state machine" expresses concurrency discipline

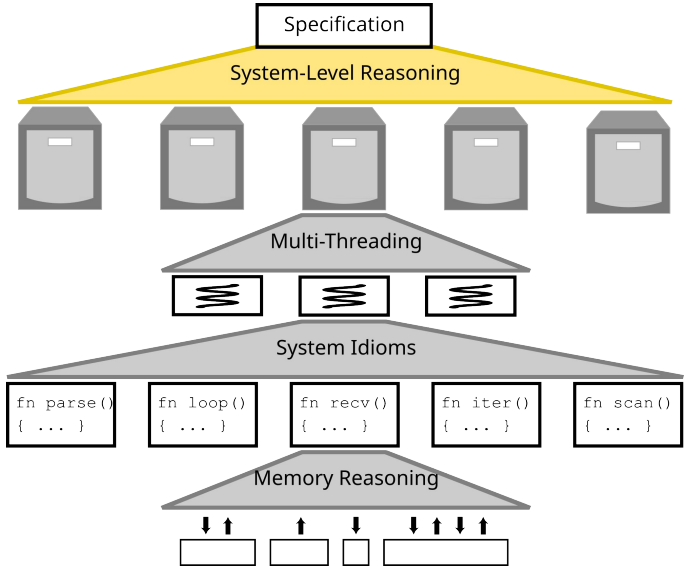


Systems-level reasoning

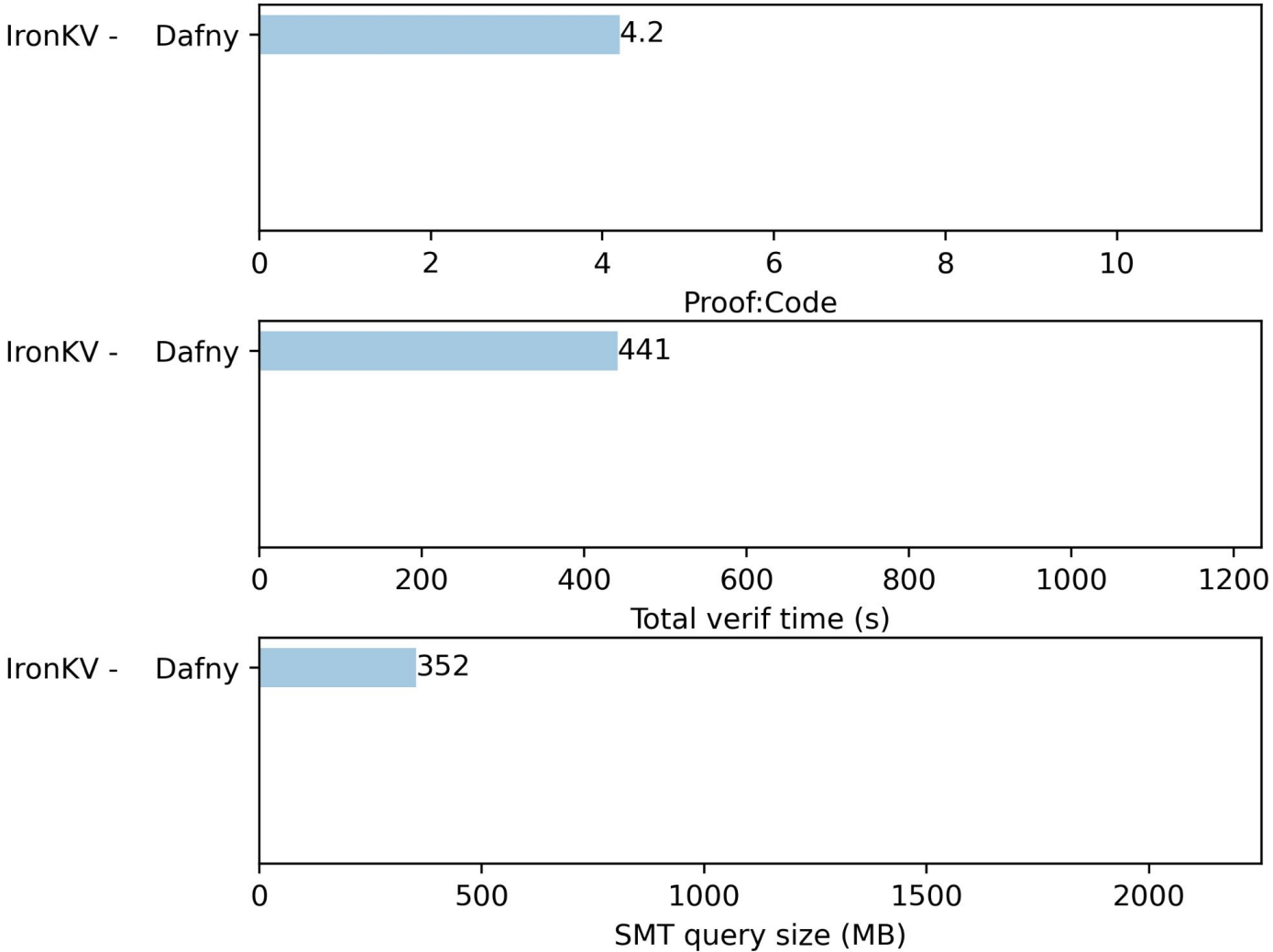
context: IronFleet style models *systems* as atomic state machines

problem: embedded TLA+ makes ugly boilerplate

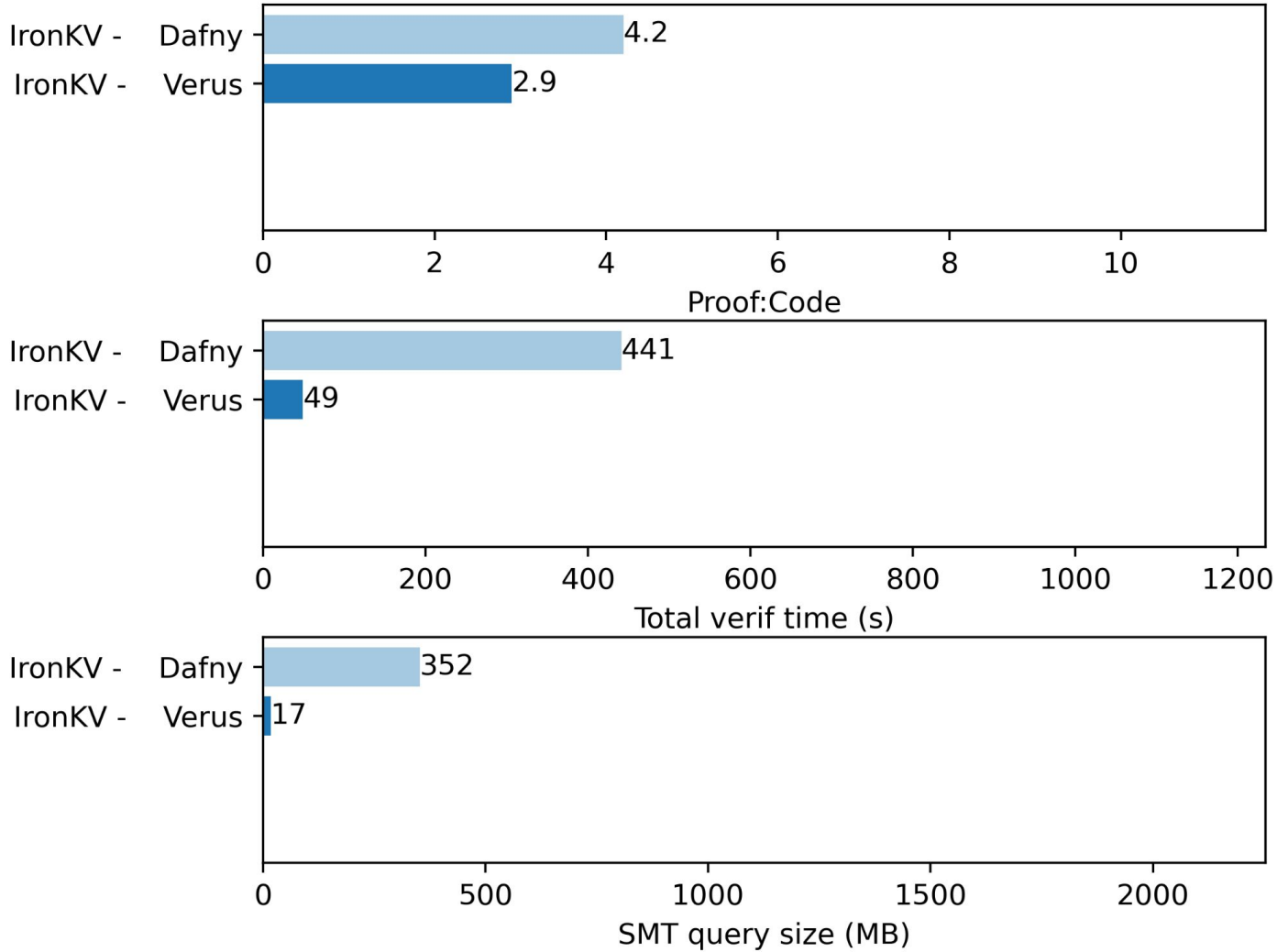
solution: native atomic state machine syntax



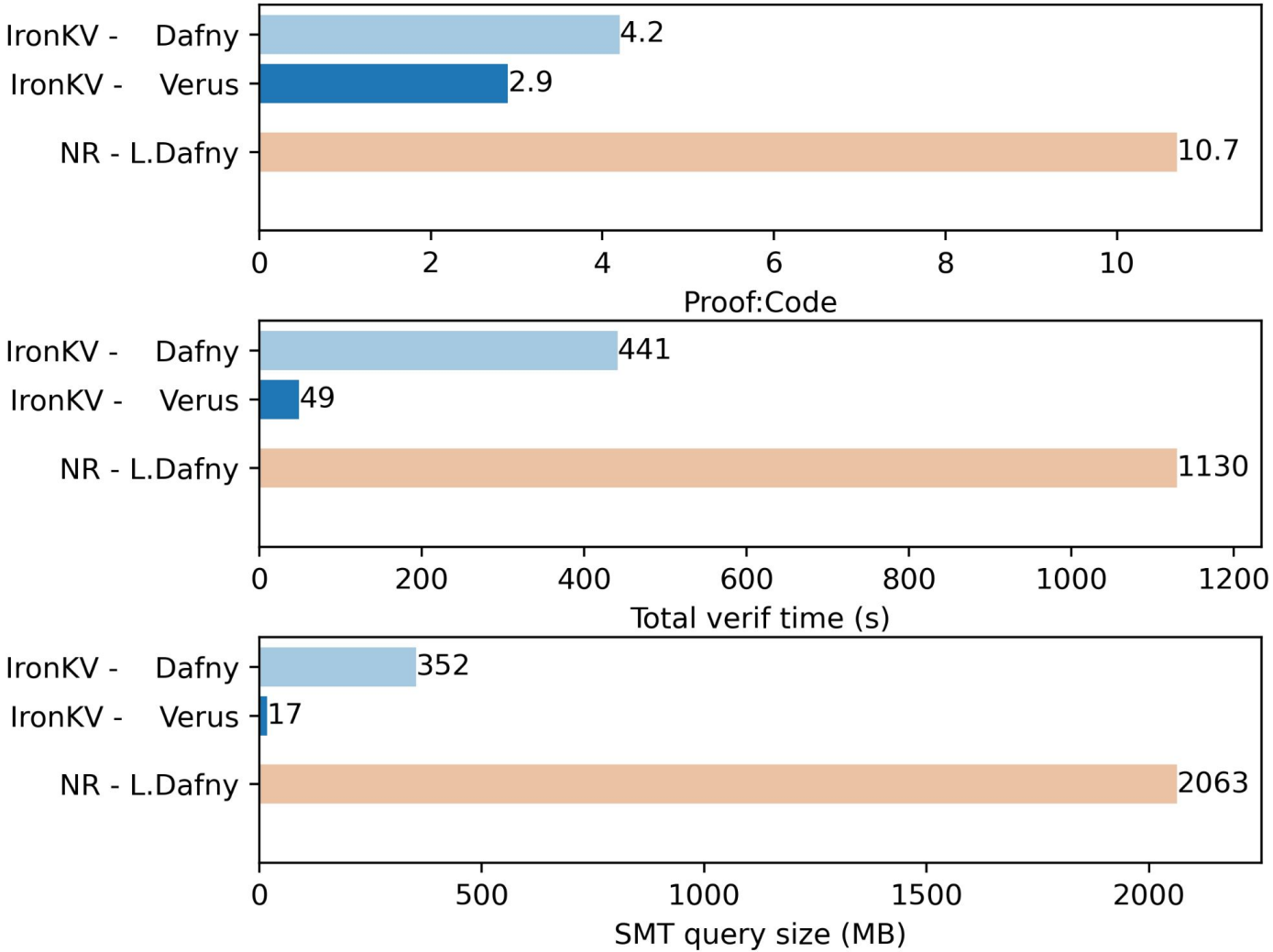
Macrobenchmarks



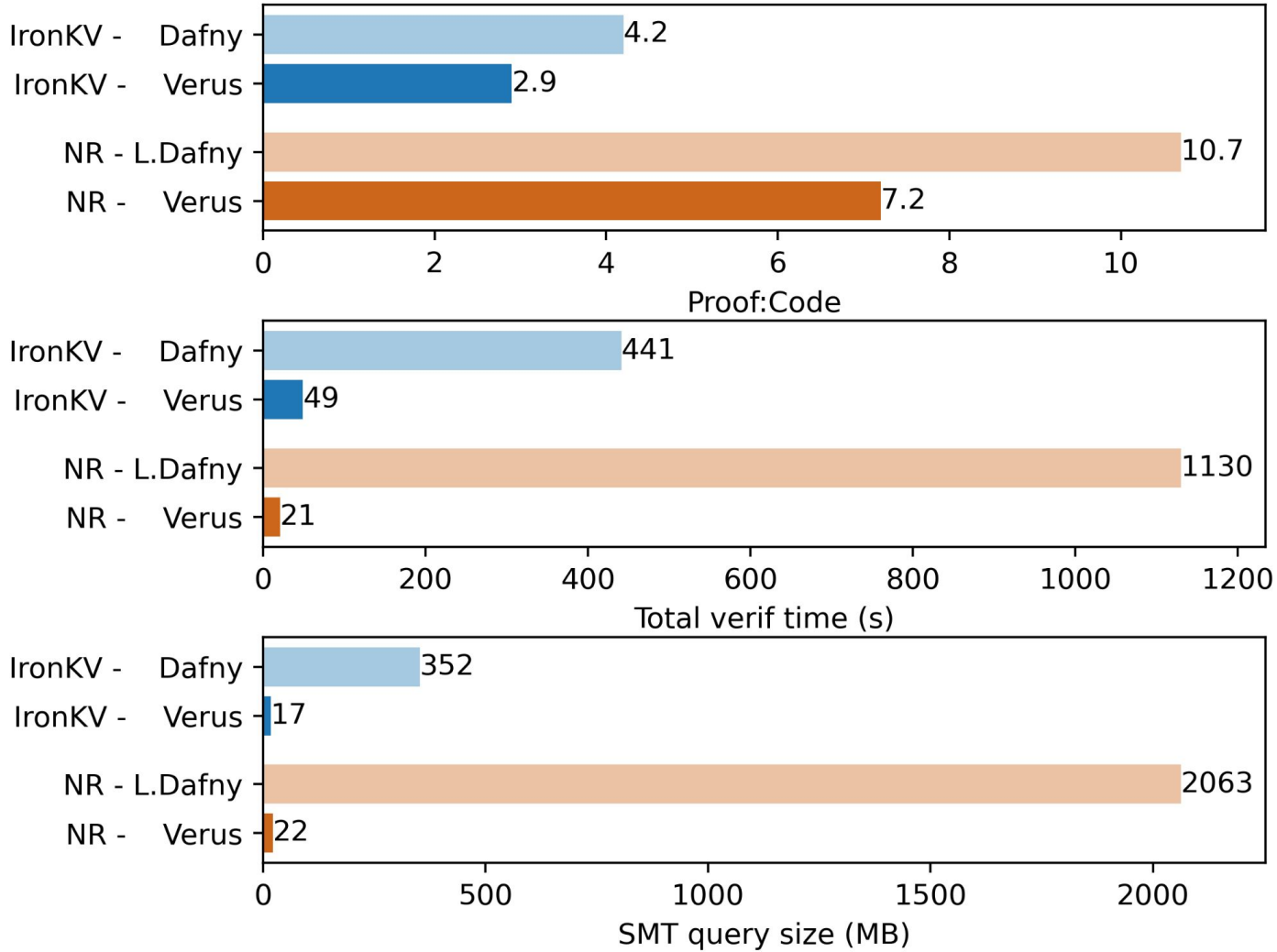
Macrobenchmarks



Macrobenchmarks



Macrobenchmarks



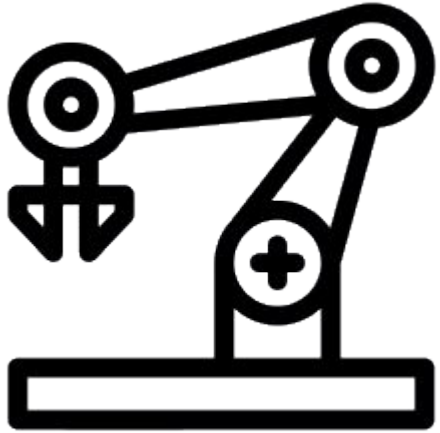
Conclusion

Verus is a new verifier aimed at practical use + new research.

Try it out:	`play.verus-lang.org`
Install it:	`github.com/verus-lang/verus`
Ask users:	`verus-lang.zulipchat.com`

backup

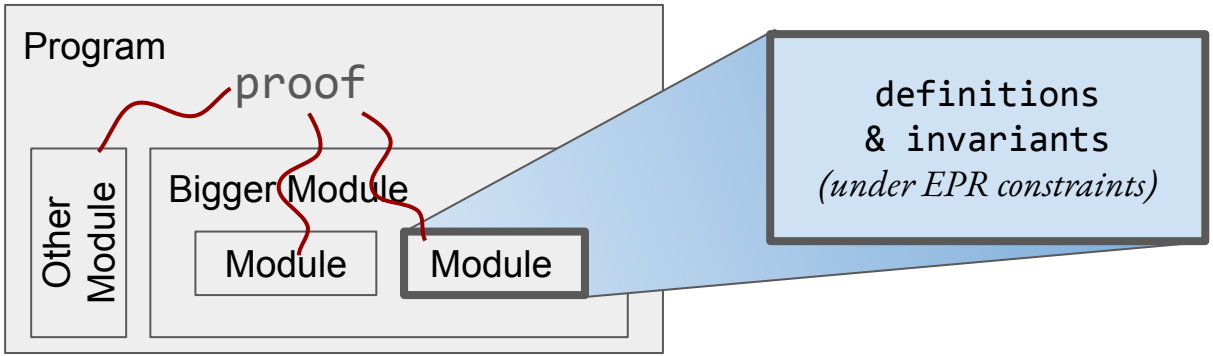
Automation: *fully automatic* proofs



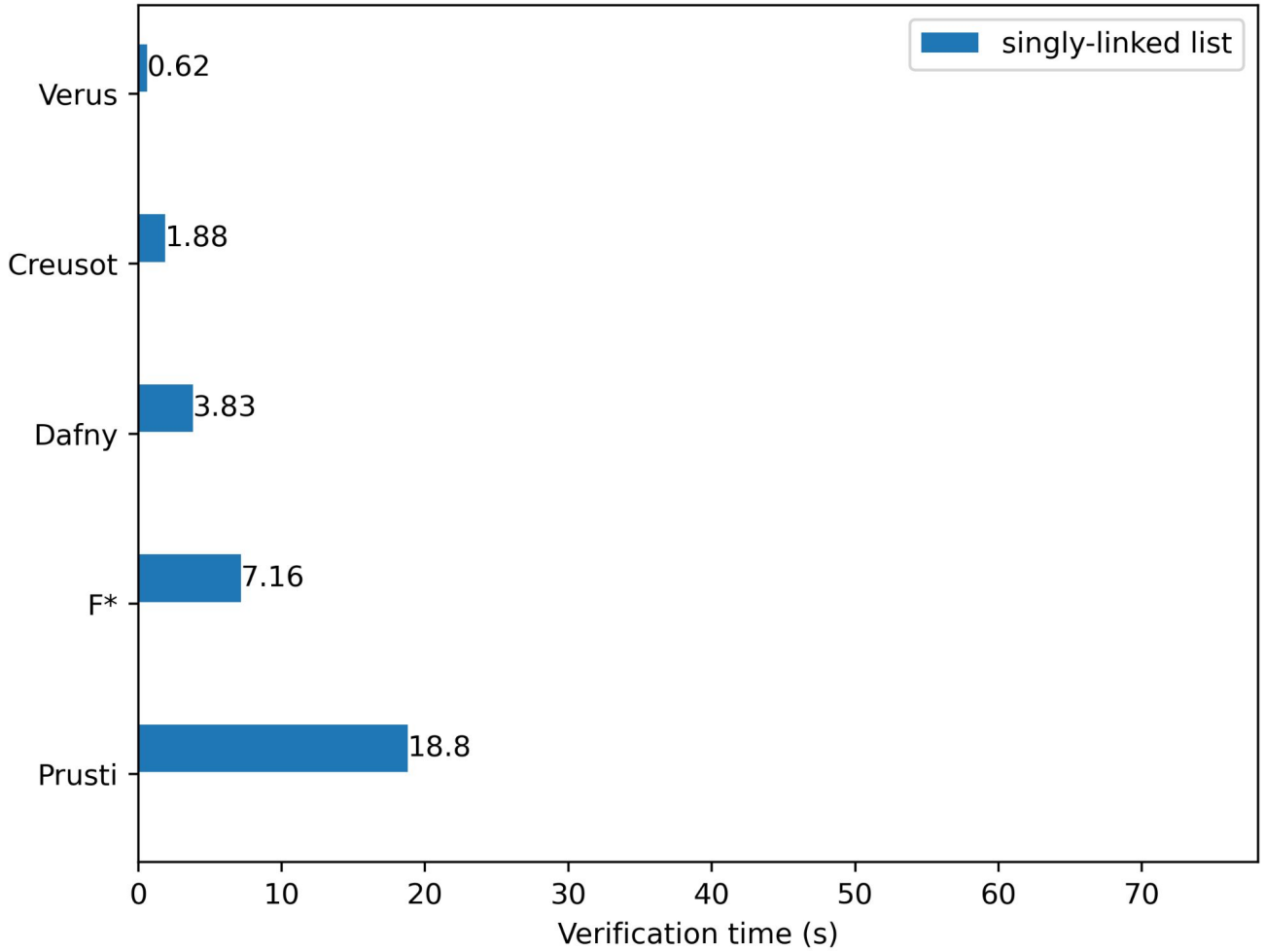
context: exploit full automation (ivy, mypyvy, pushbutton)

problem: all-or-nothing

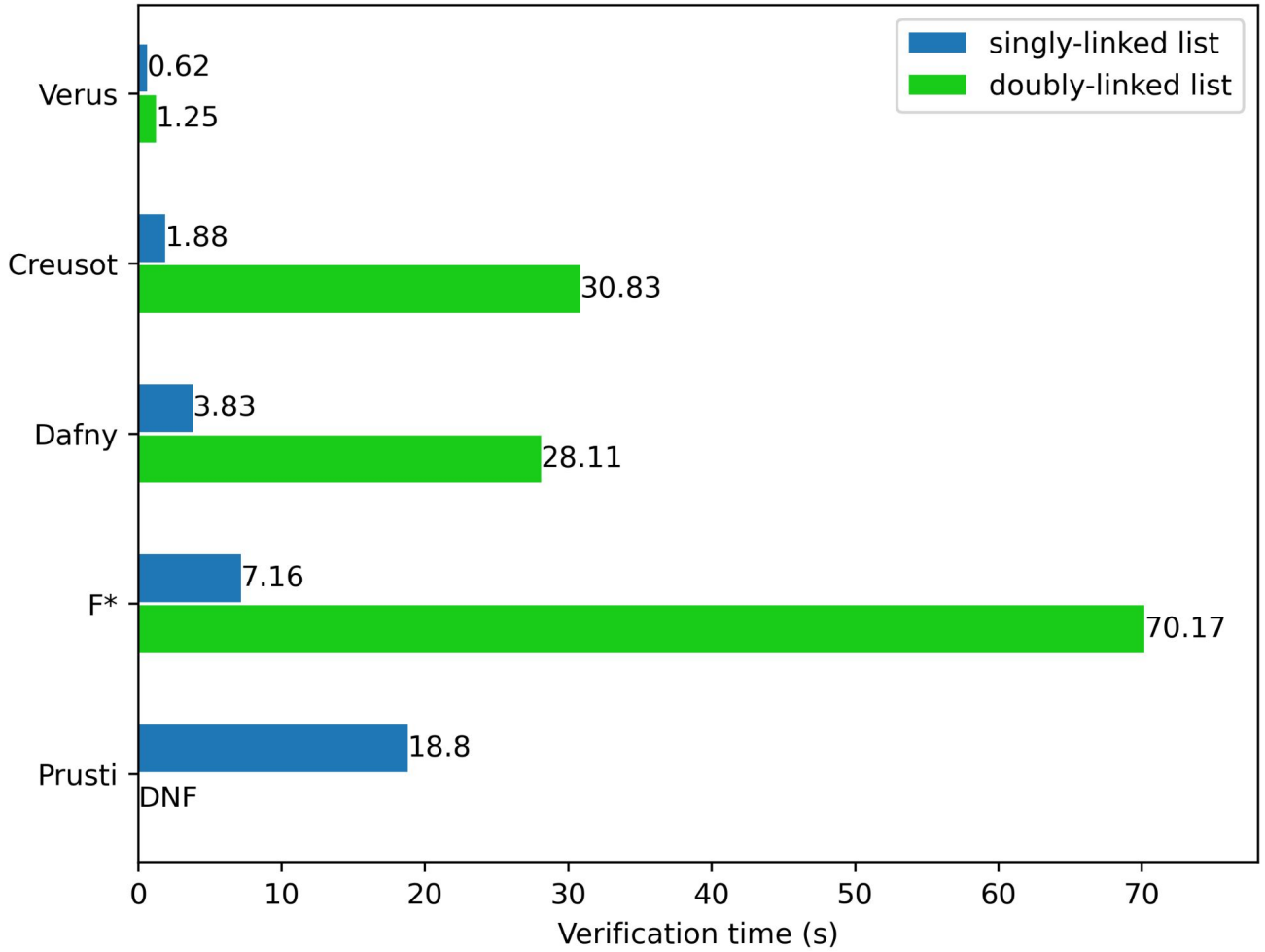
solution: opt-in integration



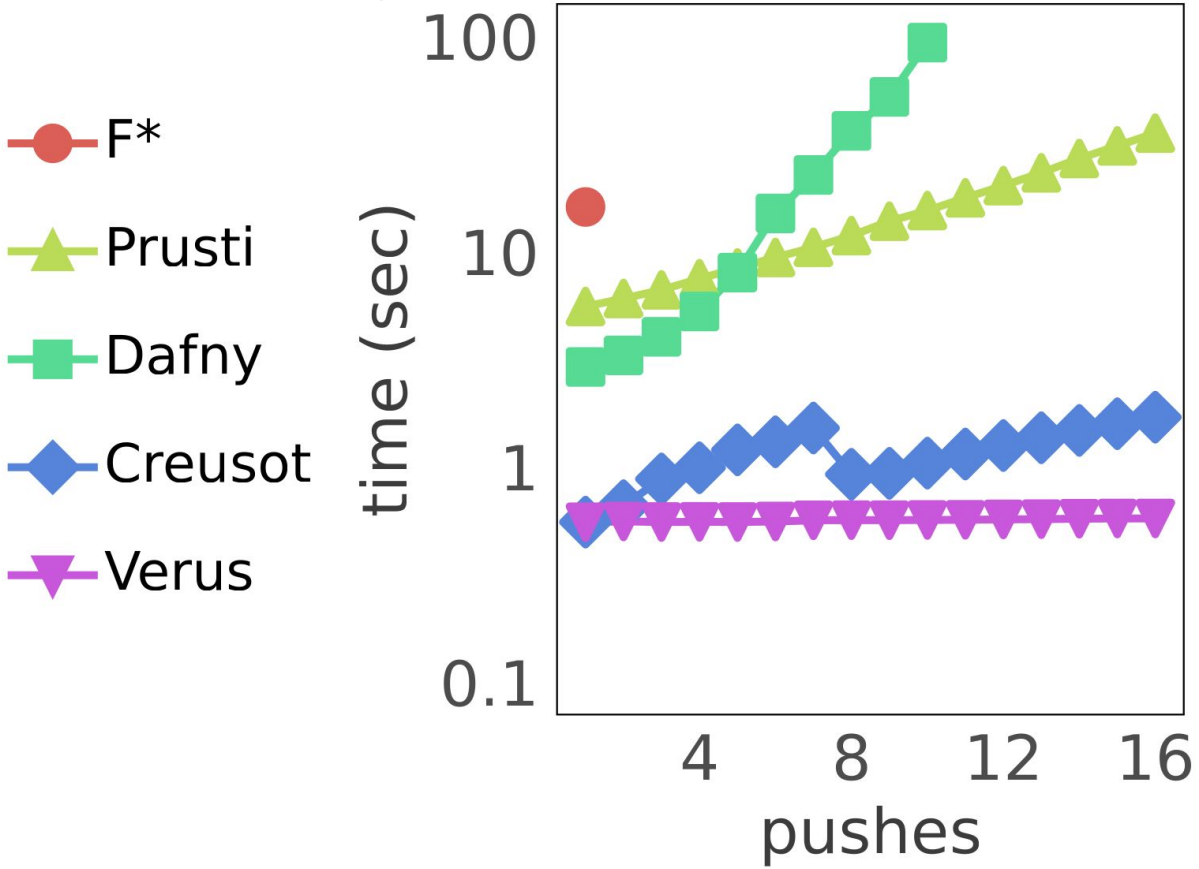
Millibenchmarks



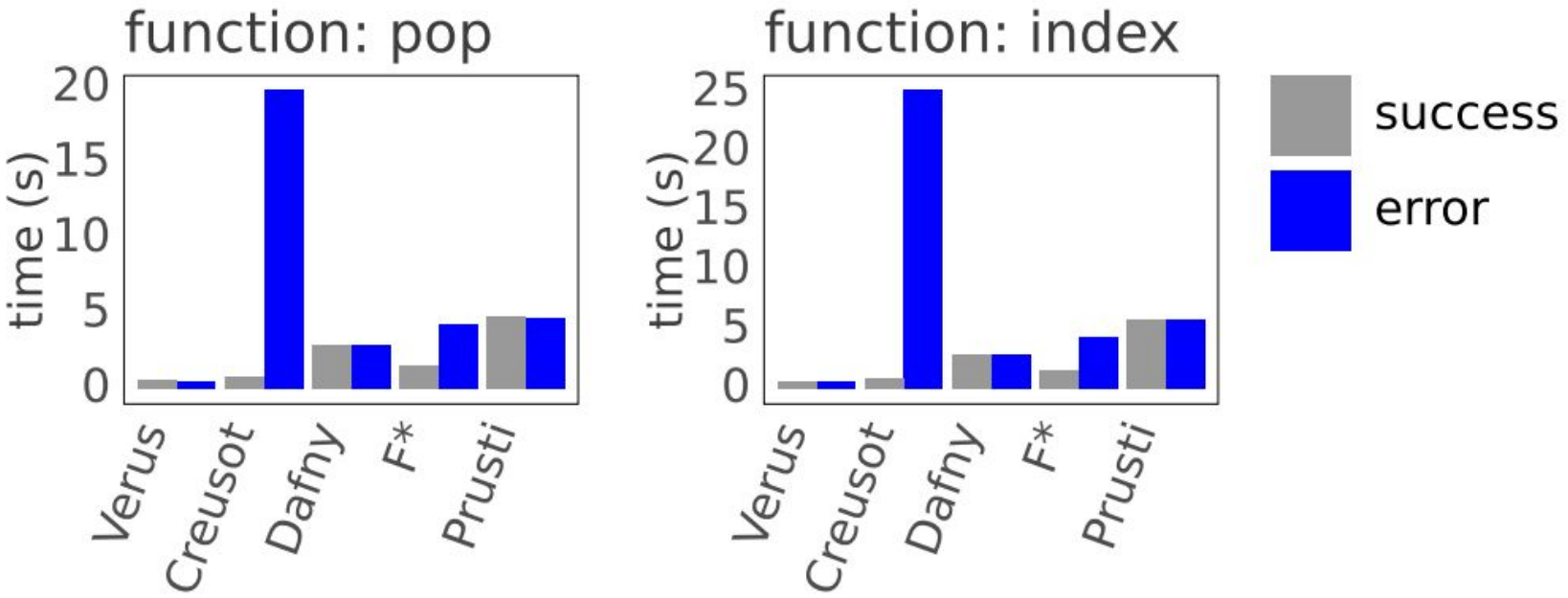
Millibenchmarks



Verus scales with fn complexity



Verifier is fast on failures



Future Directions

Improving Rust compatibility / feature coverage => more practically usable

Verify Rust standard library

Proof stability improvements

Experiments with other backends