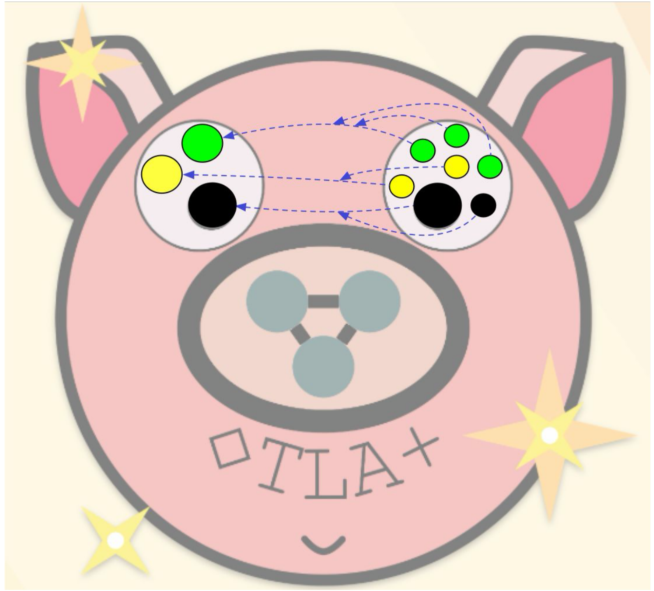
TraceLinking Implementations with their Verified Designs



Finn Hackett and Ivan Beschastnikh
University of British Columbia

Building and Running Distributed Systems is Notoriously Error-prone

👉 TLA+ modeling helps

🤯 Concurrency

🤯 Partial Failure

🤯 Networks

How Does TLA+ Help?

Abstract modeling tool, can model check / verify
→ represent distributed algorithm using sets + state machines

Has helped industrial projects, some big systems have specs.

… but no natural link to e.g., 100k lines of C++/Java/etc impl code

Trace Validation [VLDB’20, SEFM‘24, NSDI’25, ATC’25] (Some industrial successes so far!)

💡 Correspond implementation logs with formal specification (in TLA+)

Formal relationship between log contents and spec meaning

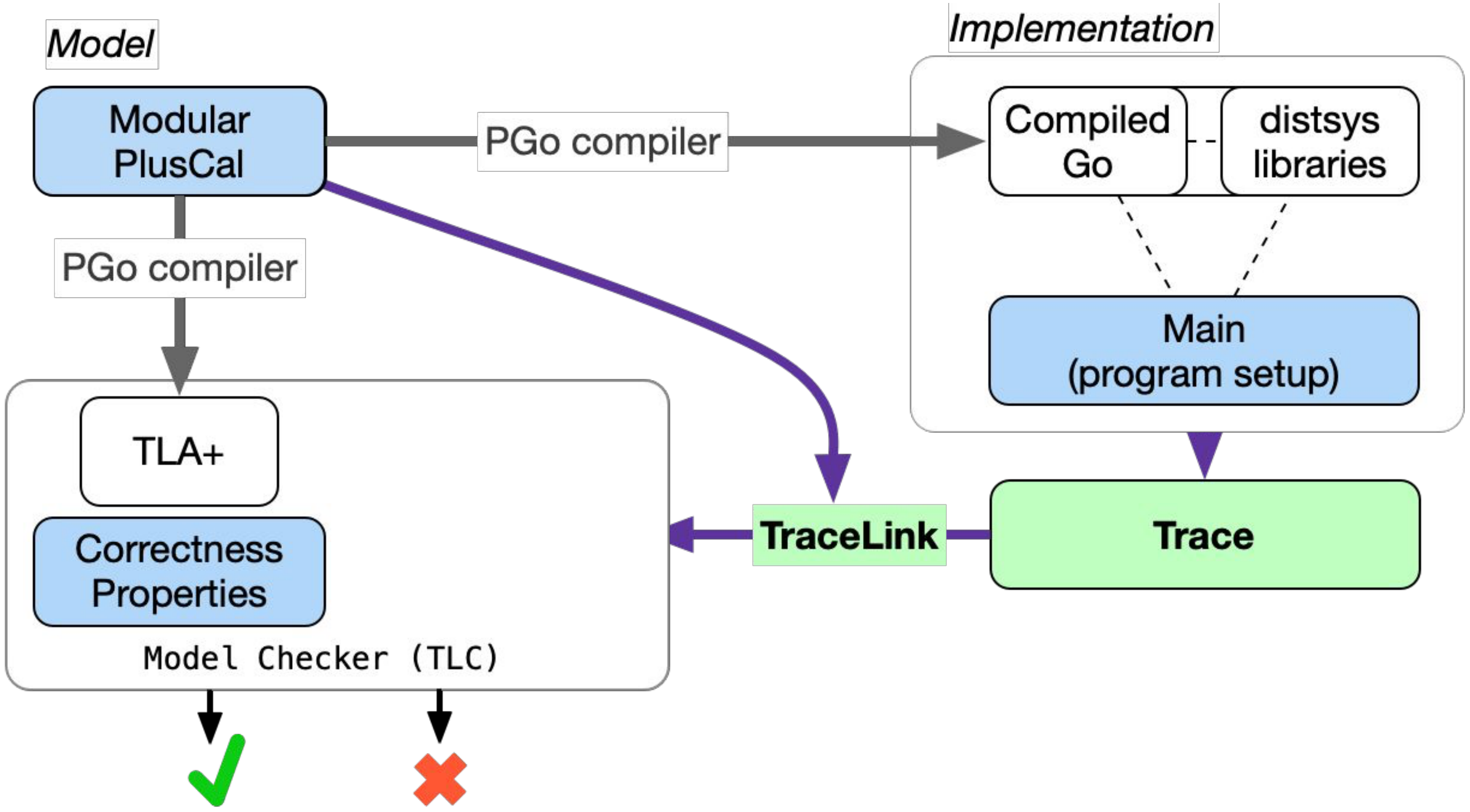
💡 Use TLA+ model checker to solve for ambiguities in log

🤔 Existing work does all setup and instrumentation by hand, low detail logs…

🚀 We have PGo [1], a specification compiler. Use it to add detailed statement level instrumentation, and generate all the setup!

[1] Compiling Distributed System Models with PGo. Hackett, Hosseini, Costa, Do, Beschastnikh. ASPLOS 2023.

TraceLink: Push-button Validation of PGo Systems



Imagine a (Very Simple) Distributed System

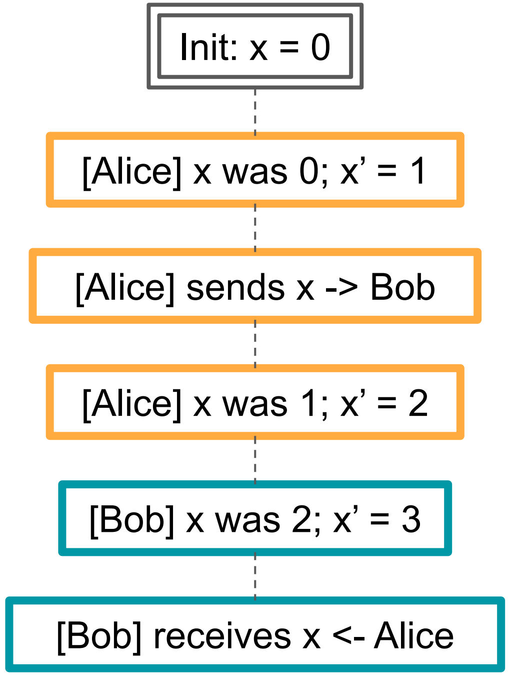
We are implementing a global variable x, accessed by Alice and Bob.

Alice and Bob can increment x.

To share x, Alice and Bob exchange messages (e.g., UDP messages).

Alice + Bob have local copies of x.

Prior Work: Validating Pre-sorted Traces [VLDB'20, SEFM'24, NSDI'25, ATC'25]



Implementing a global variable x, accessed by Alice and Bob.

Alice and Bob can increment x.

To share x, Alice and Bob exchange messages (e.g., UDP).

Alice + Bob have local copies of x.

- ☐ X increments by 1
- ☐ X doesn't go down
- ☐ Alice sends x to Bob

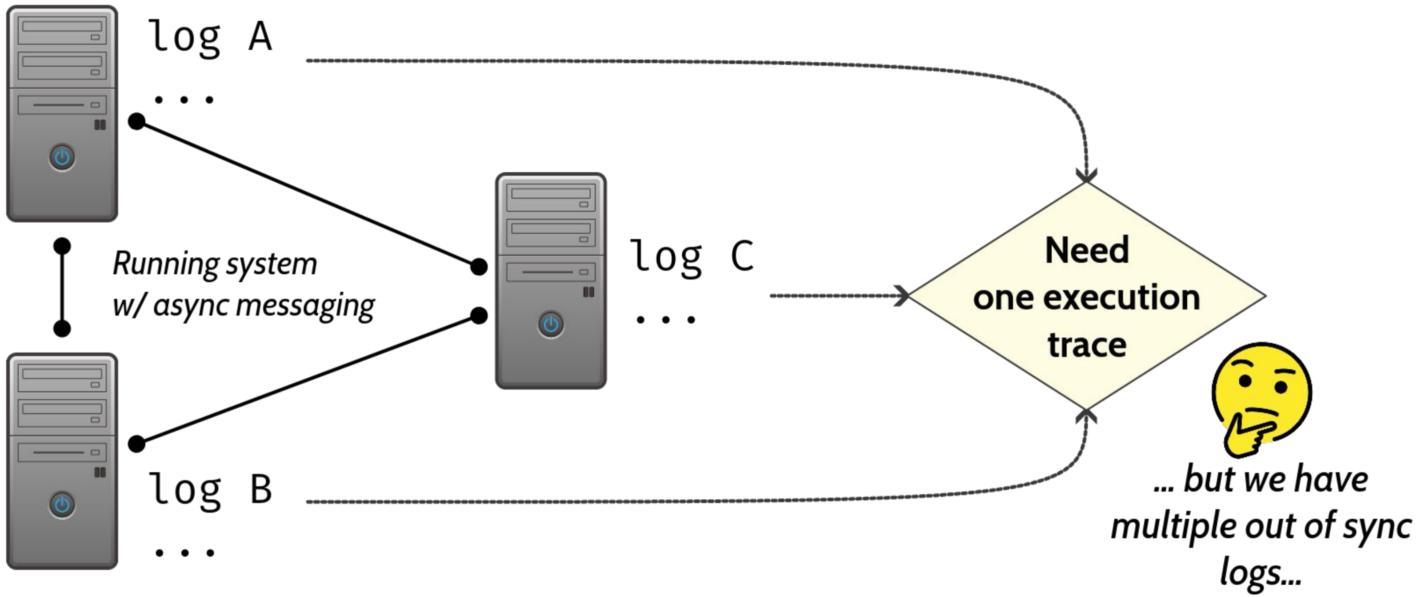
…

⚠️ Is Bob psychic?!

Spec we're using

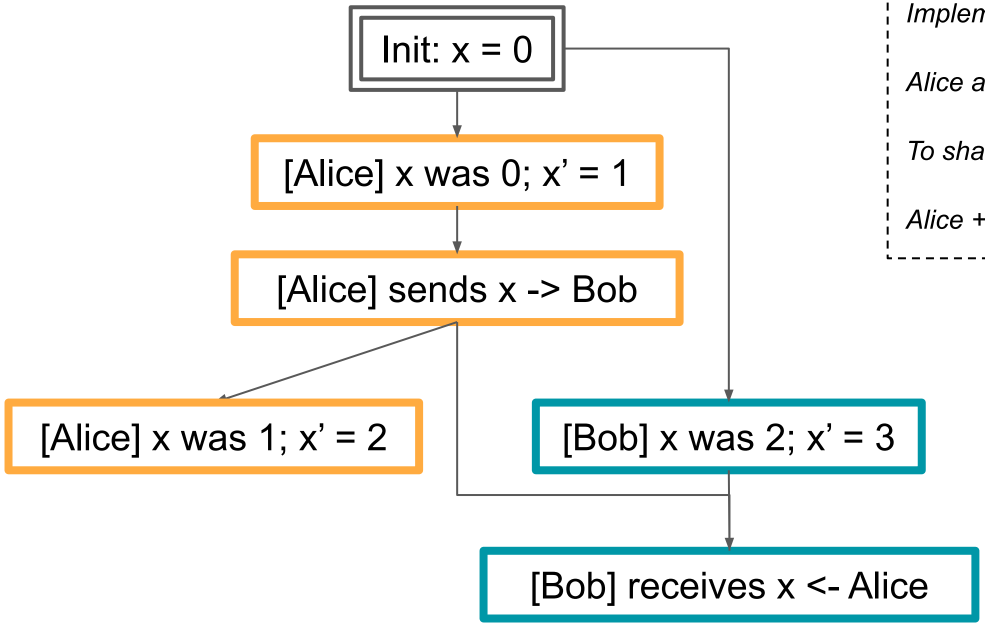
**assuming model of x and model of sync are not internally correlated. Omitting such things is a feature of TLA+. Several bugs we found were omissions like this.*

Partial Order w/ Vector Clocks



👉 Track causality with vector clocks, get partial order

Causality-aware Trace Validation



Implementing a global variable x, accessed by Alice and Bob.

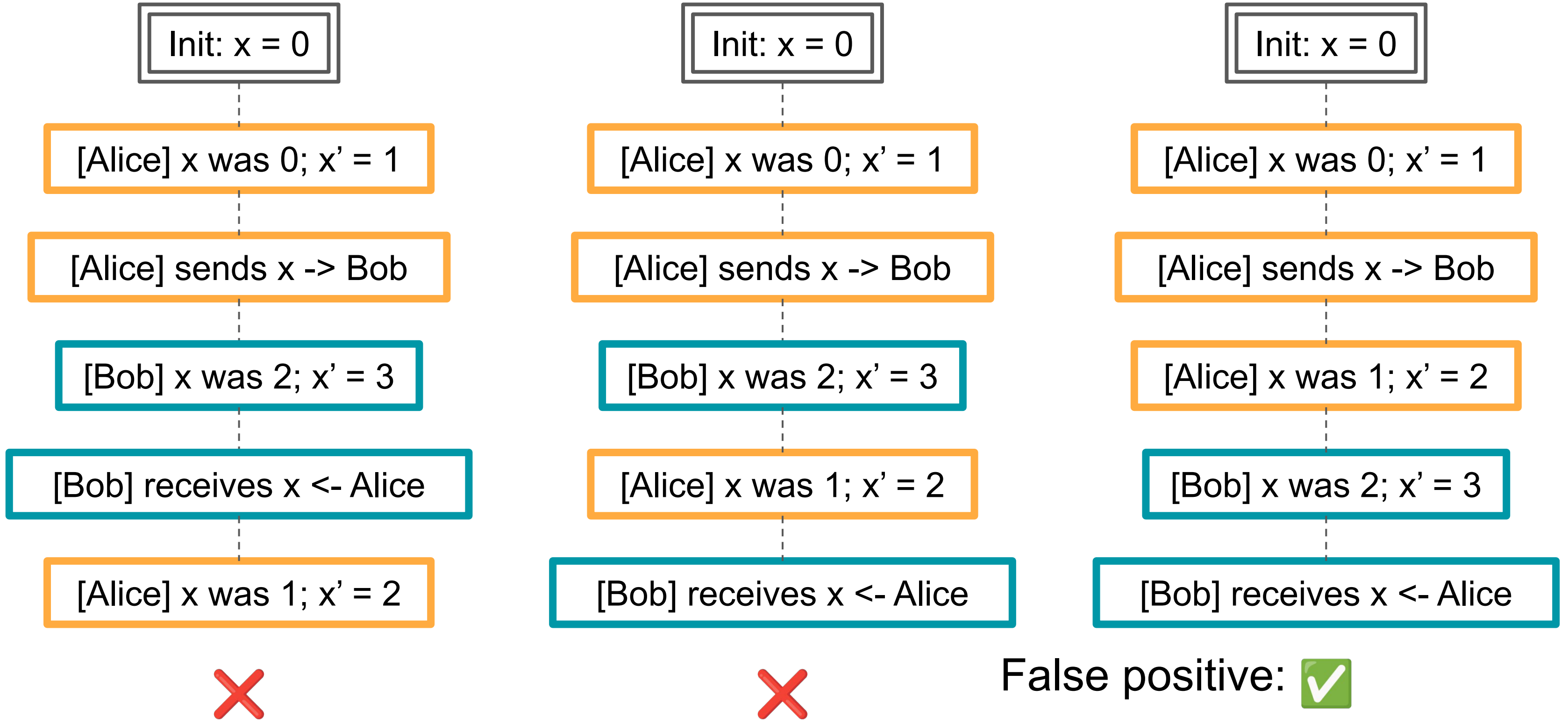
Alice and Bob can increment x.

To share x, Alice and Bob exchange messages (e.g., UDP).

Alice + Bob have local copies of x.

💡 Missing causal link visible

Multiple Interpretations – Some are False Positives



Selecting Paths During Model Checking (Explicit-State)

🤔

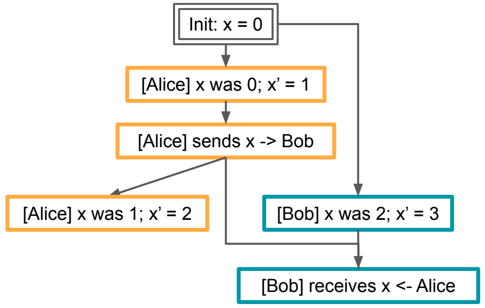
Depth-First Search

~ any one path

🤔

Breadth-First Search

~ pick all paths

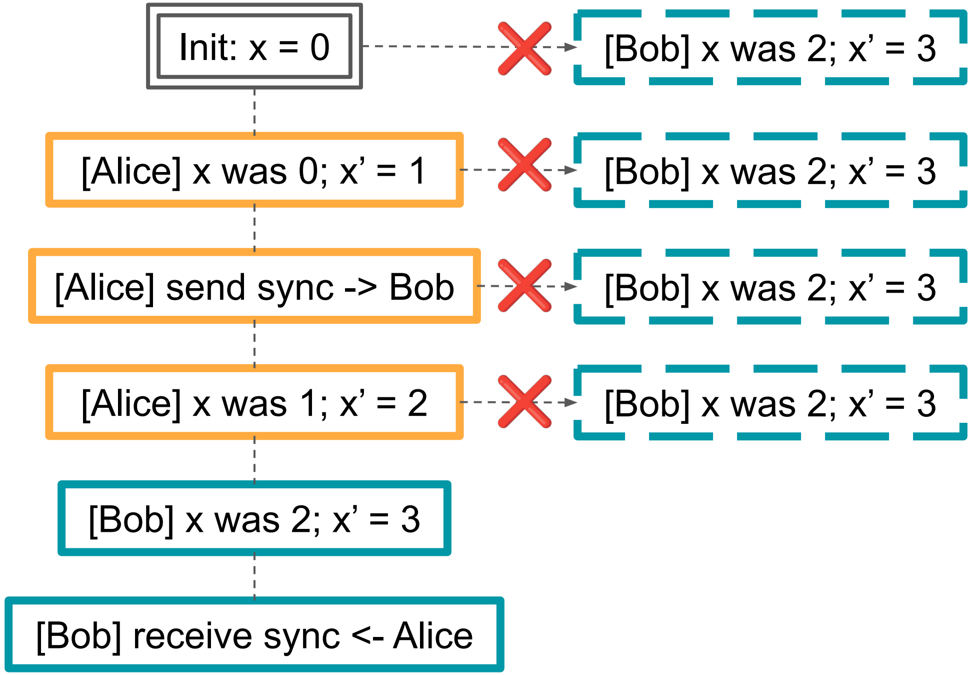


I Don't Want to Check Every Path!

… but some paths don't show the bug.

Exponential search space, we'll be here all day (literally).

The “Sidestep” Strategy



- 💡 Check every alt path for 1 step
- 🎉 Same complexity as single path
- 🎉 Finds problem at earliest point*

**as long as problem can be found in 1 alt step*

Many more details in our paper

- 📜 How we used the PGo spec compiler, how instrumentation works
- 📜 Semantics for log to TLA+ translation
- 📜 9 bugs we found (in already verified systems)
  - *Specification assumptions (I/O device behavior mostly)*
  - *Latent PGo compiler bug*
  - *Own instrumentation (we don't trust logging)*
- 📜 Compression for generated TLA+, 1-100x efficiency (better for bigger inputs)

Evaluation: Systems we Tested

All test systems compiled with PGo

- **dqueue:** basic producer-consumer model. Good smoke test.
- **locksvc:** distributed lock service. Has concurrency + invariants.
- **raftkvs:** full-scale Raft-based key-value store, PGo’s main evaluation target.

Most bugs found at scale in raftkvs.

Log sizes up to **100k events**, across up to **26 processes**.

Some **counter-examples >10k states deep**.

List of Bugs TraceLink Found

- 🐛 2x network assumption 👈
- 🐛 1x PGo miscompilation
- 🐛 2x instrumentation error
- 🐛 2x timeout model
- 🐛 1x failure detector model
- 🐛 1x model abstraction

Bug Type: Modular PlusCal Env Assumptions

TCP send-receive order *between different connections*

- Send 2 messages to same recipient over different connections
- We assume receive order ⇔ send order, which is incorrect

- True for <u>same connection</u>, accidentally assumed it for <u>all messages to same recipient</u>
- Subtle modeling error, can affect correctness

Credit to Horatiu Cirstea for initially showing this possibility.

Sidestep Strategy Effectiveness

+4% rate of finding known bugs (70% → 74%) 🤷

-32% length of counterexample – when bug found (diff min = 841, max = 11021)

> *Sidestep finds “earliest” problem point, which is a lot easier to triage*
>
> *% change is in length of irrelevant info a human has to ignore*

Sidestep often more efficient, except for models with many processes.

> *Then slower (13min vs 10min), but advantage above still applies.*

Contributions

🎉 Push-button validation for PGo systems

🎉 Causal-aware validation with “sidestep”

🎉 Found 9 bugs in PGo context

Ongoing work

Dropping reliance on PGo, keep partial automation?

Apply causal validation to hand written code

👋 Graduating soon, will be looking for research work.

github.com/distCompiler/pgo