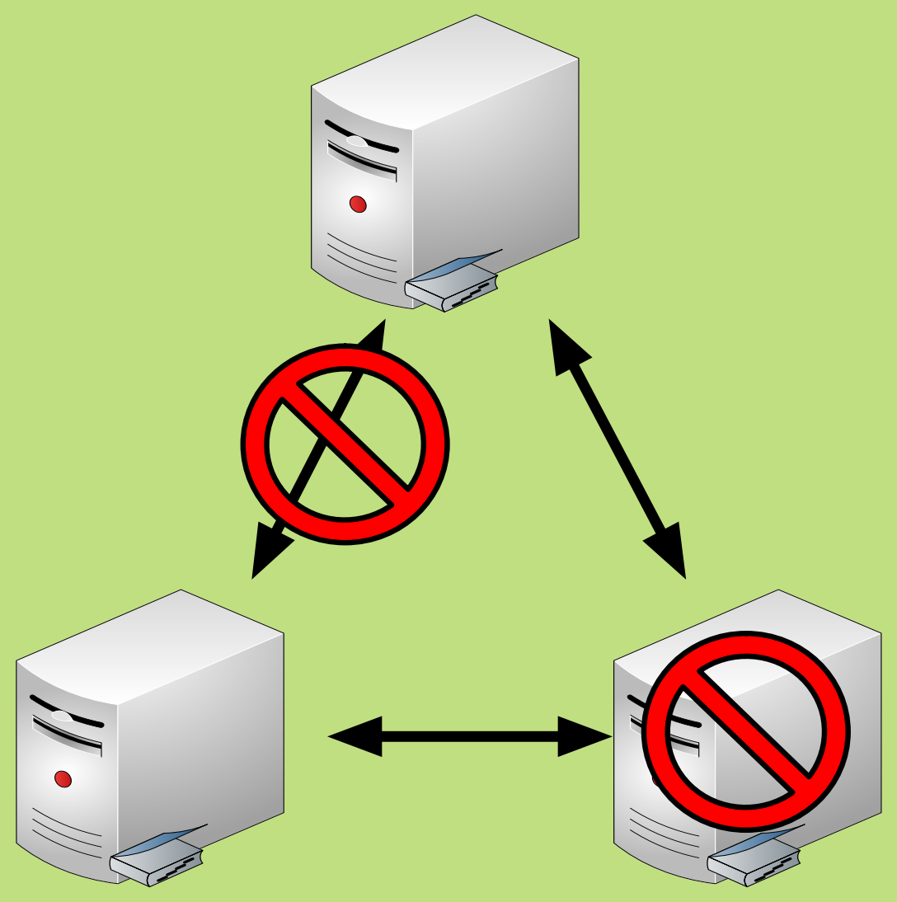
Distributed Test Case Generation using Model Inference with Dara

Checking the correctness of distributed systems is HARD!



Developer Difficulties:

- **Massive state space:** low testing coverage
- **Non-Deterministic Bugs:** inexplicable crashes
- **Partial Failures:** complex corner cases

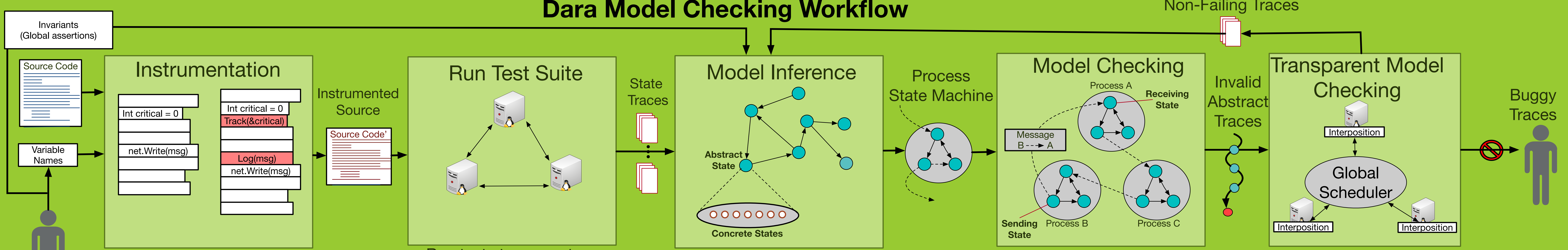
Two approaches to model checking systems

1) **Abstract**: Check abstract system model e.g. SPIN
Fast & **Deep** state expiration, implementation can admit bugs.

2) **Concrete**: Check system implementation directly e.g. MODIST
Sound, but **Shallow** state space exploration, some deep bugs go undetected.

Dara : Combine both approaches, infer model from program traces, check abstract bugs with **SPIN**. Validate abstract bugs by **replaying** trace with transparent model checker. Iteratively refine abstract model using new traces discovered during replay.

Dara Model Checking Workflow



Traverse program AST's and **inject logging** to capture **messages**, and specified **variables**

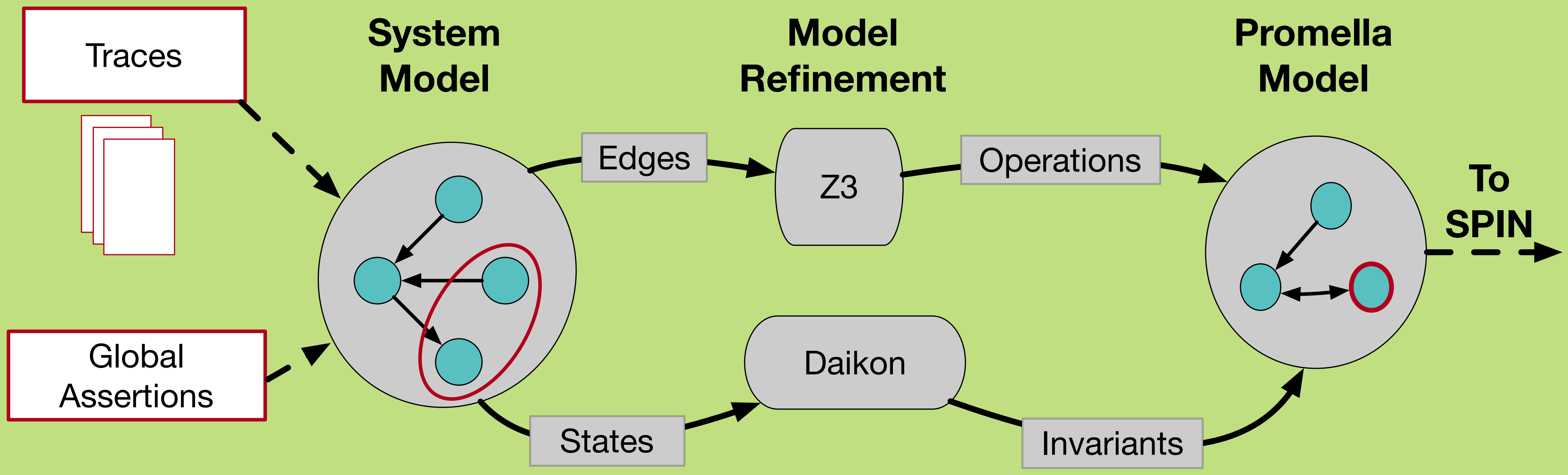
Run tests to generate a **bootstrapping** set of logs for the model inference engine.

Compose **state traces** into a finite state machine (**FSM**). Inject **invariants** into FSM.

Model check FSM with **SPIN**. Convert **fail-stop**, **deadlock**, and **invariant-violating** traces to schedules.

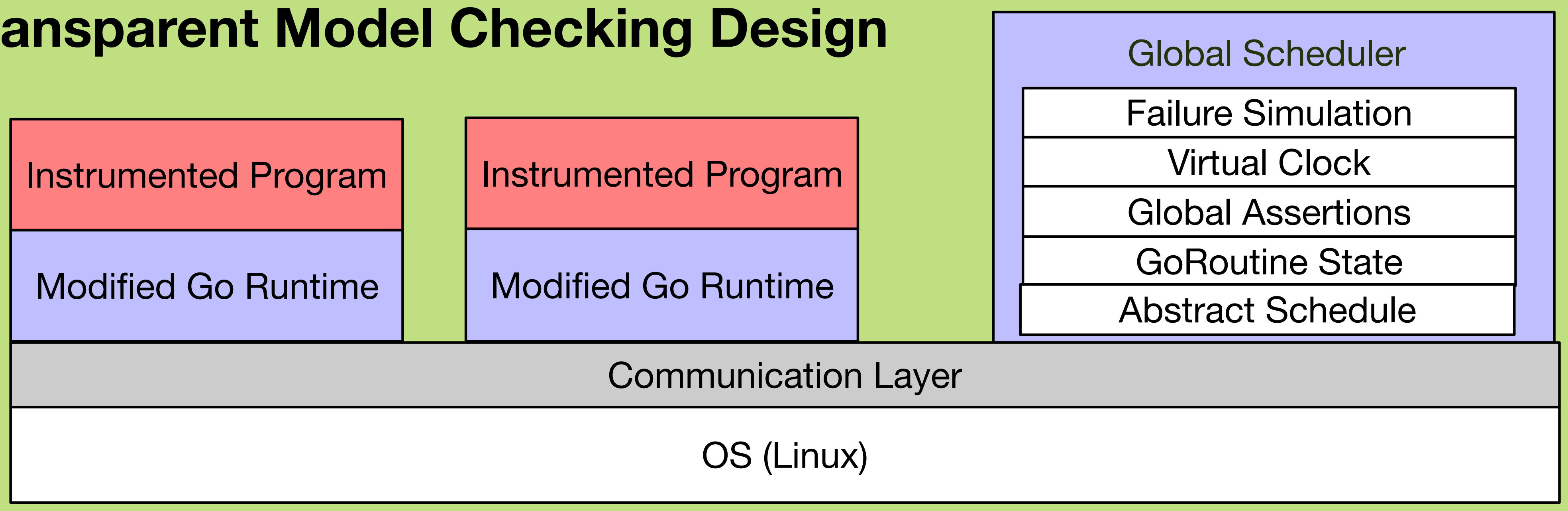
Replay schedules, and control IO. Report replicated **bugs**. Send **non-failing** traces back to model inference.

Model Inference



- Program state traces are converted into an **FSM**
- Nodes are **unique instances** of variable values
- Edges are events: messages & syscalls
- Z3 **collapses** FSM by inferring operations over edges
- Daikon invariants: **heuristic** to measure abstract-concrete traces gap

Transparent Model Checking Design



Instrumented Program | Instrumented Program

Modified Go Runtime | Modified Go Runtime

Global Scheduler
Failure Simulation
Virtual Clock
Global Assertions
GoRoutine State
Abstract Schedule

Communication Layer

OS (Linux)

- Model checker schedules threads via **global scheduler**
- Go Runtimes acts as an **interposition layer** between programs and the OS
- **Non-determinism** is captured e.g., Messaging & system calls
- Centralized scheduler **deterministically replays** abstract schedules
- Infeasible schedules generate new behaviour and refine abstract model

Related Work

- **Testing**: Unit, Integration, Stress
- **Modeling Languages**: TLA+[TOPLAS'94], SPIN [ECBS'05], COQ [INRIA'04]
- **Verification**: IronFleet [SOSP'15], Verdi [PLDI'15], Chapar [POPL'16])
- **Transparent Model Checkers**: MODIST [NSDI'09],[SOSP'11], CHESS [OSDI'08]

Ongoing Work

- Model Check etcd (raft KV store), btcd (BTC miner)
- Evaluate state space exploration vs MODIST
- Automatic detection of state variables
- Extend Model checking to temporal invariants

Open source repository

github.com/wantonsolutions/dara

Stewart Grant
Ivan Beschastnikh

THE UNIVERSITY OF BRITISH COLUMBIA