Automated

I4: ~~Incremental Inference of Inductive Invariants for~~ Verification of Distributed Protocols

Haojun Ma, Aman Goel, Jean-Baptiste Jeannin
Manos Kapritsos, Baris Kasikci, Karem A. Sakallah

University of Michigan

Artifacts Evaluated Functional acm
Artifacts Available acm
Results Replicated acm

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

Distributed Systems Are Subtle

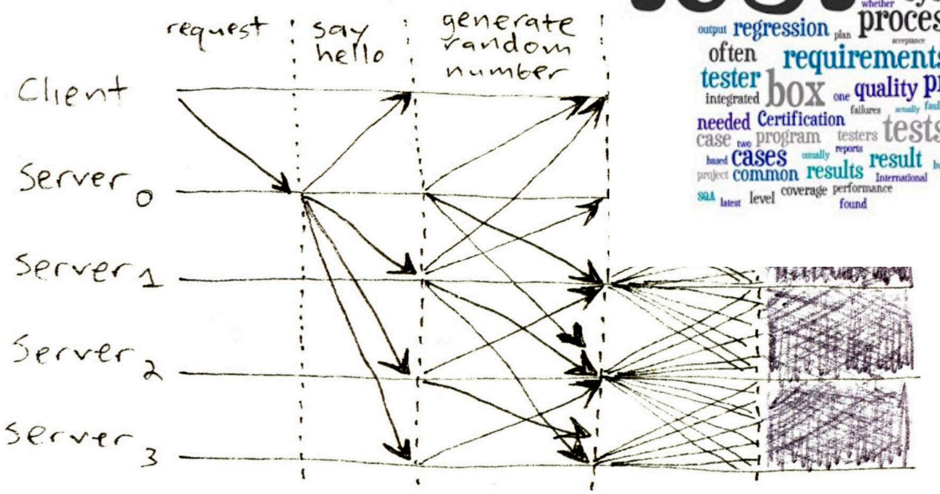
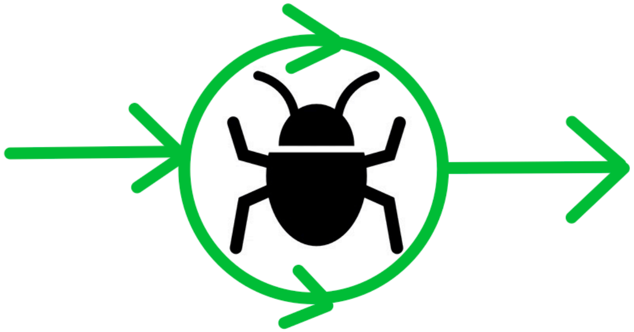


Figure 1: Typical Figure 2 from Byzantine fault paper: Our network protocol



[Mickens 2013]

The Alternative: Formal Verification

Existing Verification Approaches

Verdi(Coq)
[PLDI'15]

IronFleet(Dafny)
[SOSP'15]

Ivy
[PLDI'16]

I4

Person-years Person-months Person-days Person-hours Automated

Manual Effort

All existing approaches require the human to find an **inductive invariant**

We want to automatically find inductive invariants …
… by **combining the power of Ivy and model checking**

Preview of Results

Protocol	Traditional approach	Ivy	I4
Lock server	500 lines (Verdi)	<1 hour	Automated
Distributed lock	A few days (IronFleet)	A few hours	< 5 min

Numbers come from Ivy [PLDI 2016]

Outline

Motivation

Verification of distributed systems

I4: a new approach

Design of I4

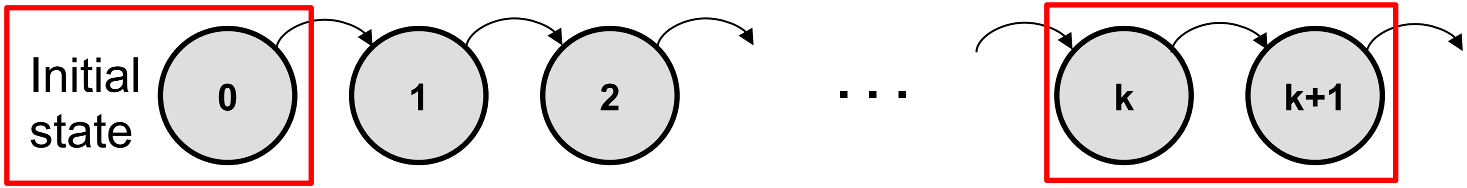
Evaluation

Conclusion

Induction on Distributed Protocol

Goal: prove that the safety property **always** holds

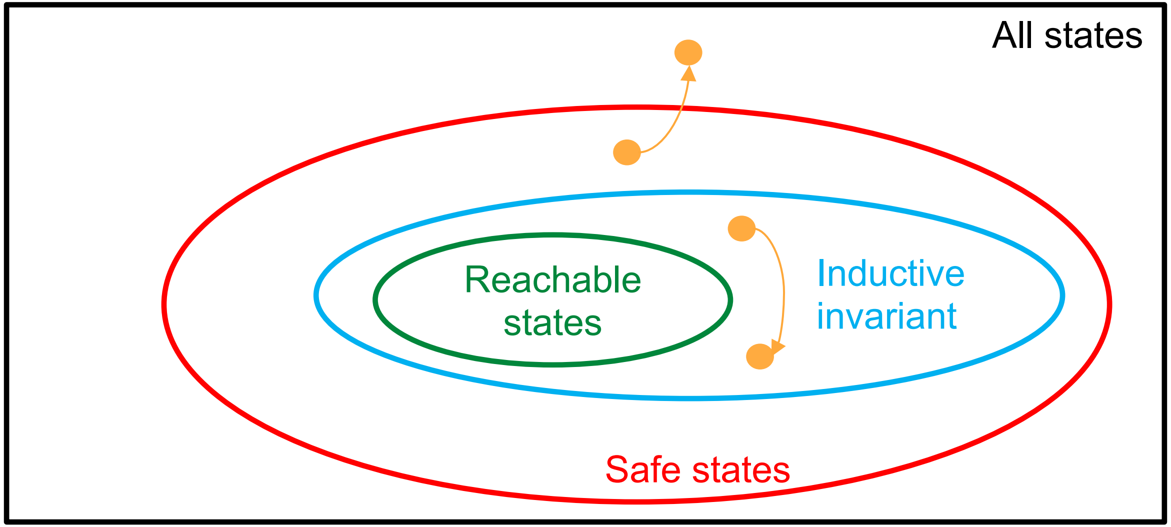
An execution:



Inductive proof

- Base case: prove initial state is safe
- Inductive step: if state **k** is safe, prove state **k+1** is safe

Safety Property vs. Inductive Invariant



Inductive Invariants Are Complex

$$\forall\ N_1, N_2 : node,\ \ E : epoch.$$
$$locked(E, N_1) \wedge locked(E, N_2) \implies N_1 = N_2$$

I.e. No tw… …e.

Existing approaches rely on manual effort and human intuition

$\wedge\ \ \forall\ N_1, N_2, E.\ held(N_1) \wedge trans(E, N_2) \implies le(E, ep(N_1))$

$\wedge\ \ \forall\ N_1, N_2, E.\ trans(E, N_1) \wedge \neg le(E, ep(N_1)) \implies$ …

$\wedge\ \ \forall\ N_1, N_2, E_1, E_2.\ (trans(E_1, N_1) \wedge \neg le(E_1, ep($ …

Strengthening Assertion

Outline

Motivation

Verification of distributed systems

I4: a new approach

Design of I4

Evaluation

Conclusion

I4: a new approach

Goal: Find an inductive invariant ***without*** relying on human intuition.

Insight: Distributed protocols exhibit ***regularity***.

- Behavior doesn’t fundamentally change as the size increases
- E.g. distributed lock, Chord DHT ring, …

Implication: We can use inductive invariants from small instances to infer a ***generalized*** inductive invariant that holds for all instances.

Leveraging Model Checking

☺ Fully automated

☹ Doesn't scale to distributed systems

I4 applies model checking to small, finite instances …

… and then generalizes the result to all instances.

Outline

Motivation

Verification of distributed systems

I4: a new approach

Design of I4

Evaluation

Conclusion

Design of I4


Protocol.ivy
Invariant generation on a **finite** instance
Invariant generalization
Increase Size


Protocol.ivy
Invariant generation on a **finite** instance
Invariant generalization
Increase Size

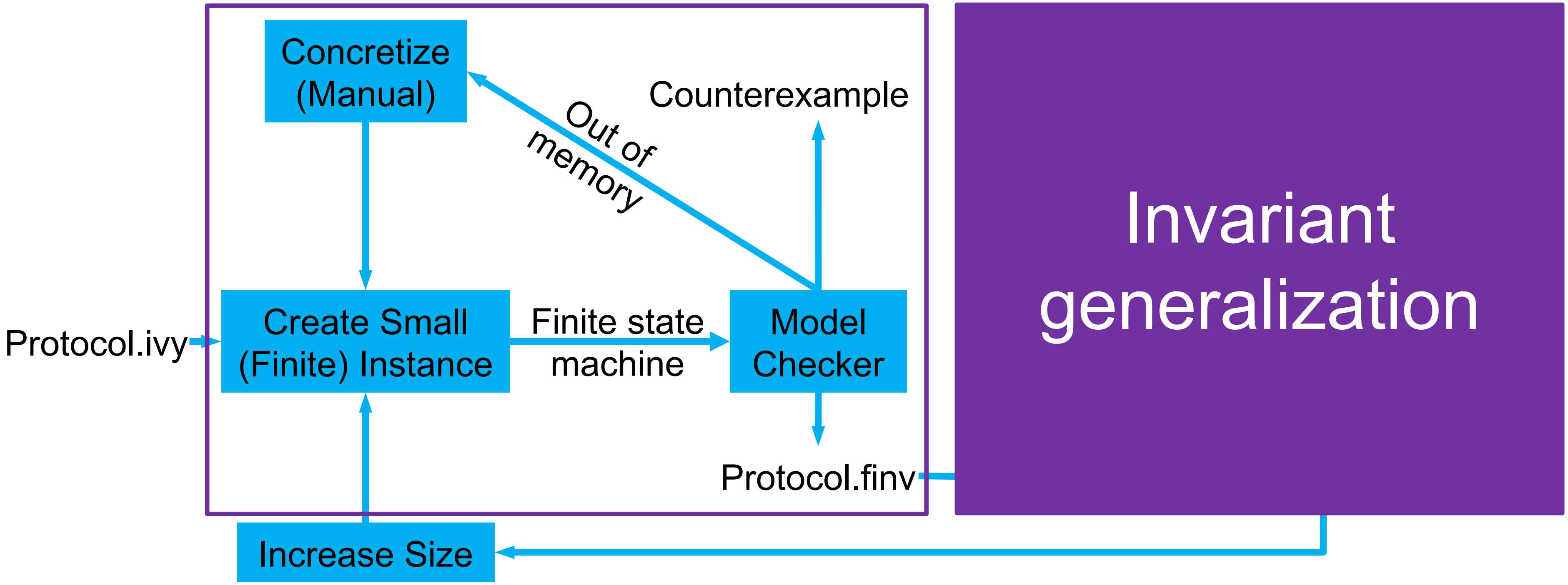
Making The Model Checking Problem Easier

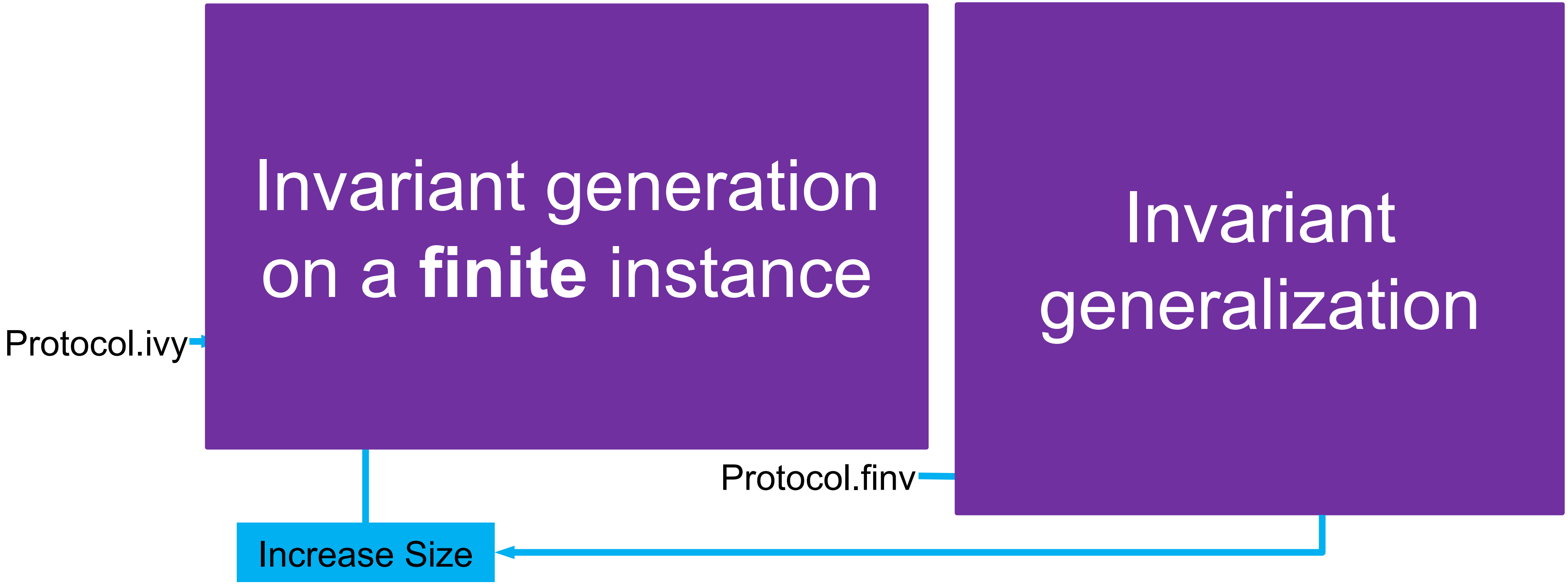
Symmetry

E.g. FIRST is the node that sends the first message

FIRST = Node1

Invariant Generation on a Finite Instance




Invariant generation on a **finite** instance
Invariant generalization
Protocol.ivy
Protocol.finv
Increase Size

Generalizing The Inductive Invariant

$P(N_1, N_2)$

$$\forall N_1, N_2. N_1 \neq N_2 \implies P(N_1, N_2)$$

Invariant generation on a finite instance

$P(N_1, N_2)$ $\quad N_1 = first$

Protocol.ivy

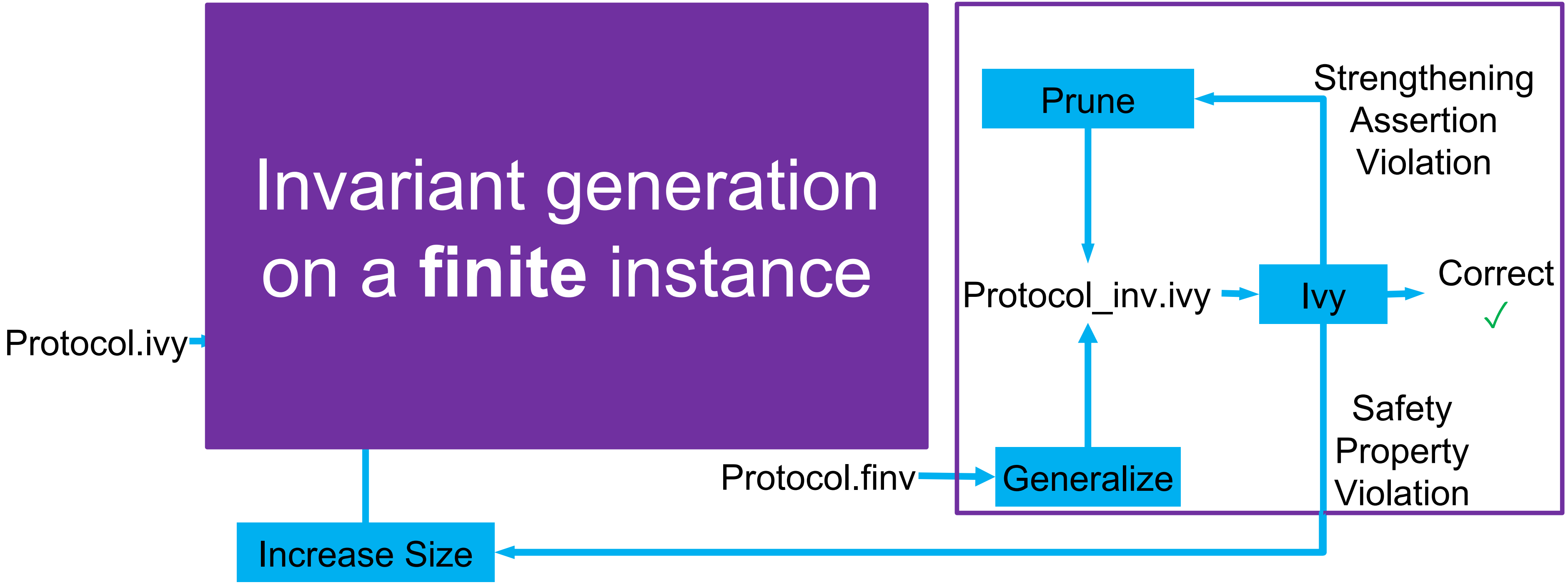
$$\forall N_1, N_2. (N_1 \neq N_2) \wedge (N_1 = first) \wedge (N_2 \neq first) \implies P(N_1, N_2)$$

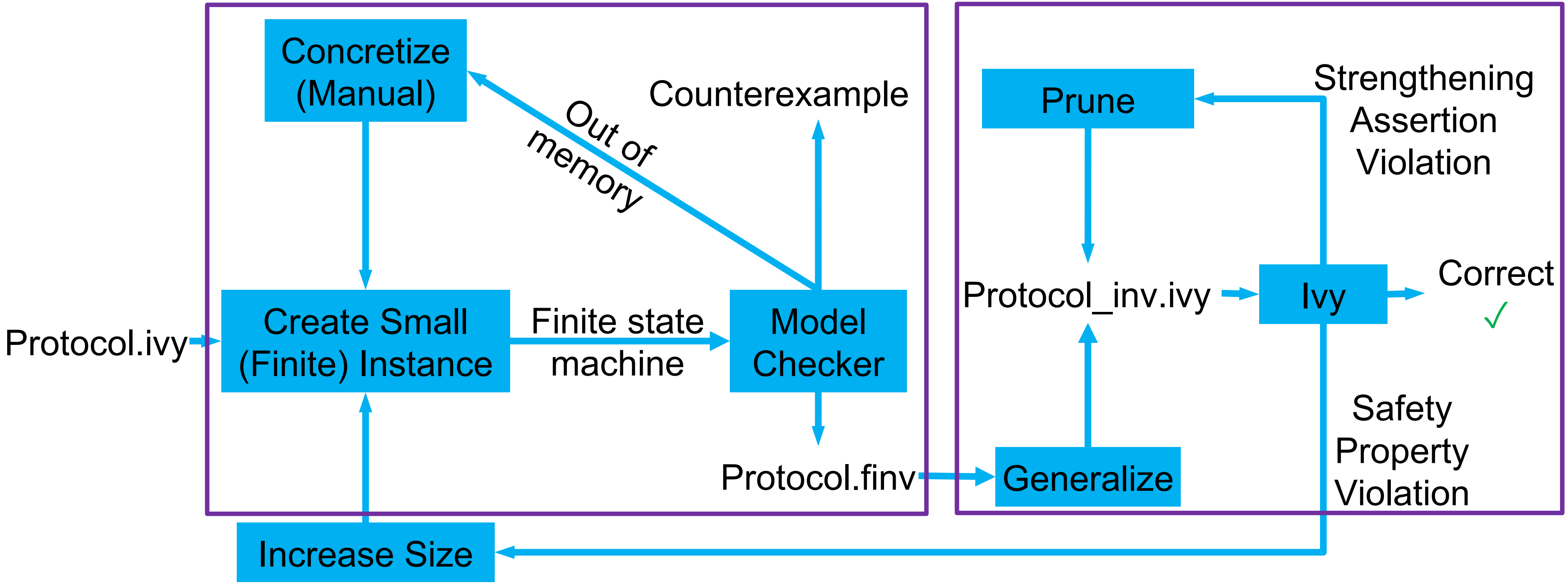
Protocol.finv

Generalize

Increase Size

Invariant Generalization




Concretize
(Manual)
Out of
memory
Counterexample
Protocol.ivy
Create Small
(Finite) Instance
Finite state
machine
Model
Checker
Protocol.finv
Increase Size
Prune
Strengthening
Assertion
Violation
Protocol_inv.ivy
Ivy
Correct
✓
Generalize
Safety
Property
Violation

Outline

Motivation

Verification of distributed systems

I4: a new approach

Design of I4

Evaluation

Conclusion

Evaluation

Lock Server
Leader Election
Distributed lock

Blind Tests {
Chord Ring
Learning Switch
Database Chain Consistency
Two-Phase Commit

Result Summary

Protocol	Manual Effort	Total time (sec)	Minimal instance size
Lock server	None	0.9	2 clients, 1 server
Leader election in ring	<5min	6.2	3 nodes, 3 ids
Distributed lock	<5min	159.6	2 nodes, 4 epochs
Chord ring	<5min	628.9	4 nodes
Learning switch	None	10.7	3 nodes, 1 packets
Database chain Consistency	None	12.6	3 transactions, 3 operations, 1 key, 2 node
Two-Phase Commit	None	4.3	6 nodes

Outline

Motivation

Verification of distributed systems

I4: a new approach

Design of I4

Evaluation

Conclusion

Conclusion

Thanks

Regularity of distributed protocols makes it possible to automatically infer inductive invariants of distributed protocols from small instances.

By combining the power of **model checking** and **Ivy**, I4 can verify a number of interesting protocols with little to no manual effort.

https://github.com/GLaDOS-Michigan/I4

Artifacts Evaluated Functional acm
Artifacts Available acm
Results Replicated acm

I’m looking for a research intern for next summer. If you’re interested, just contact me.

```
type node
type epoch

relation le(E:epoch, E:epoch)
relation locked(E:epoch, N:node)
relation transfer(E:epoch, N:node)
relation held(N:node)

individual zero : epoch
individual e : epoch
function ep(N:node) : epoch
individual first : node
```

```
after init {
        held(X) := X:node = first;
        ep(N) := zero;
        ep(first) := e;
        transfer(E,N) := false;
        locked(E,N) := false
}
```