DistAI: Data-Driven Automated Invariant Learning for Distributed Protocols

Jianan Yao, Runzhou Tao, Ronghui Gu, Jason Nieh, Suman Jana, Gabriel Ryan

Columbia University

Why learn invariants for distributed protocols?

- Distributed systems are hard to implement correctly
  - lost or corrupt packets
  - node failures
  - ...

Why learn invariants for distributed protocols?

- Distributed systems are hard to implement correctly
  - lost or corrupt packets
  - node failures
  - ...


THE VERGE TECH REVIEWS SCIENCE CREATORS ENTERTAINMENT VIDEO MORE

TECH AMAZON

Prolonged AWS outage takes down a big chunk of the internet

AWS has been experiencing an outage for hours

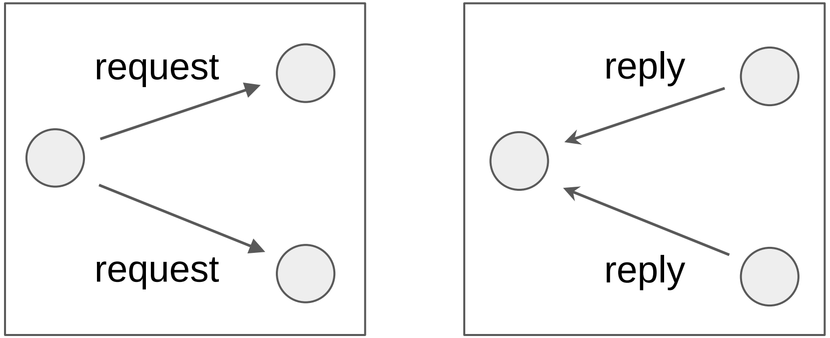
By Jay Peters | @jaypeters | Updated Nov 25, 2020, 5:39pm EST

Why learn invariants for distributed protocols?

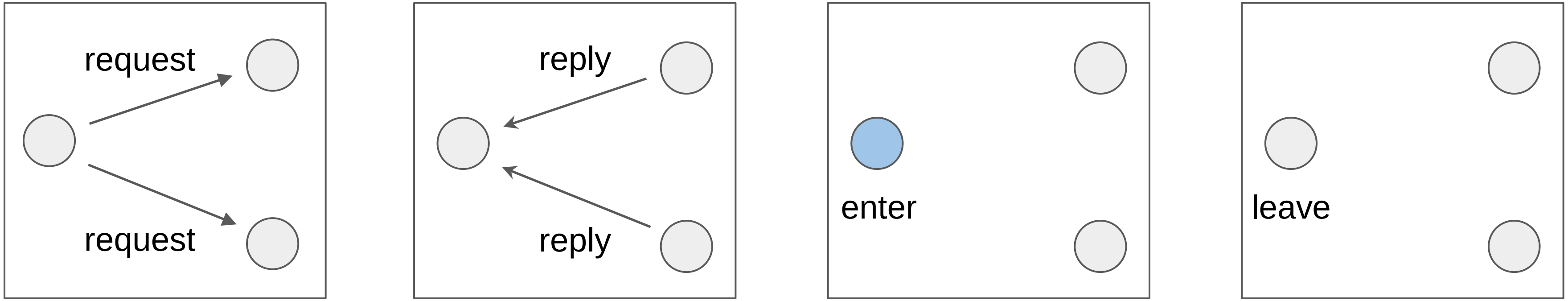
- Distributed systems are hard to implement correctly
- To prove the desired correctness property holds

Find an **inductive invariant**

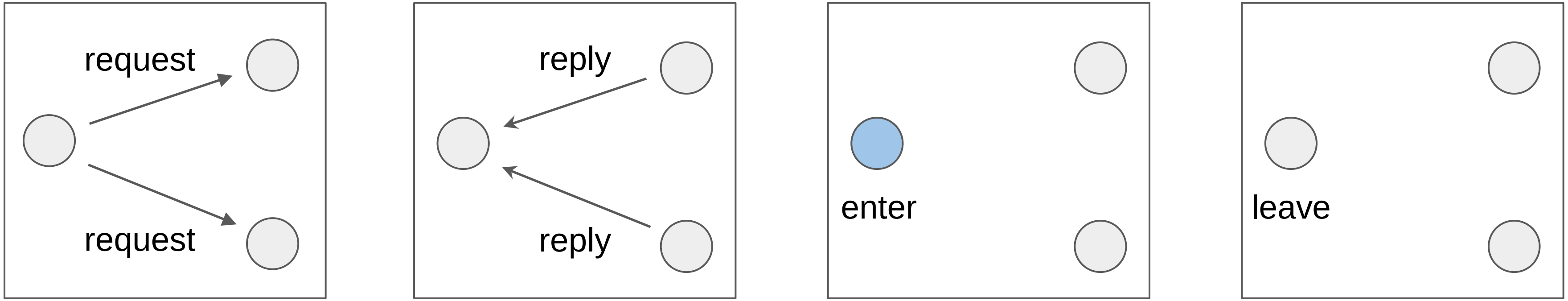
Example: mutual exclusion protocol



Example: mutual exclusion protocol

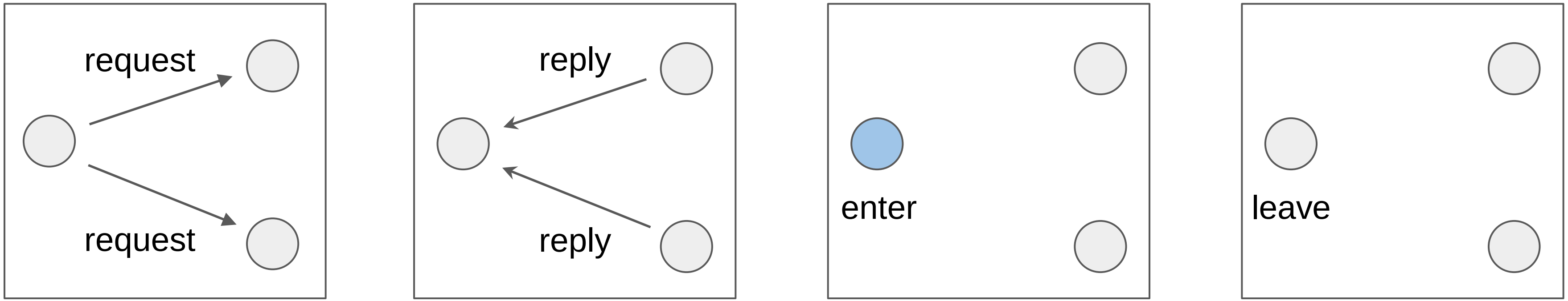


Example: mutual exclusion protocol



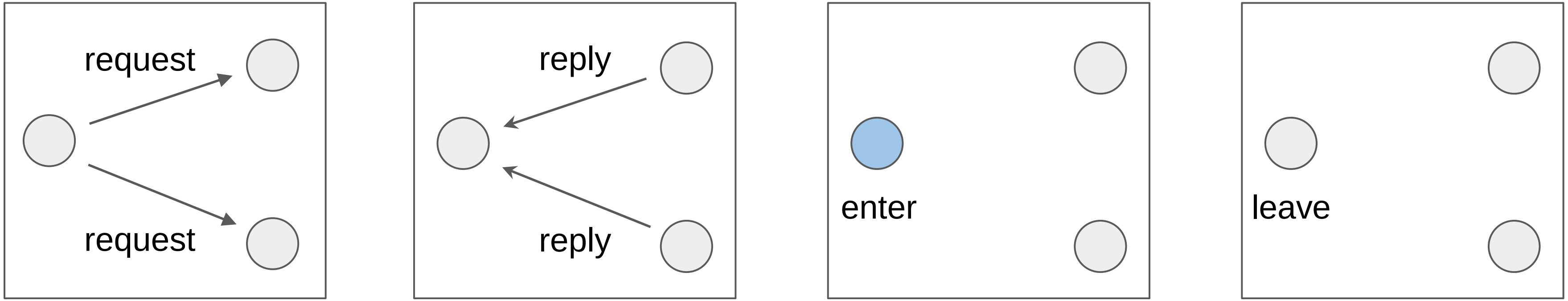
Correctness property: $\forall\, N1\, N2.\, holds(N1) \,\wedge\, holds(N2) \,\rightarrow\, N1 = N2$

Example: mutual exclusion protocol



Correctness property: $\forall N1\, N2.\, holds(N1) \wedge holds(N2) \rightarrow N1 = N2$ ✖ inductive

Example: mutual exclusion protocol



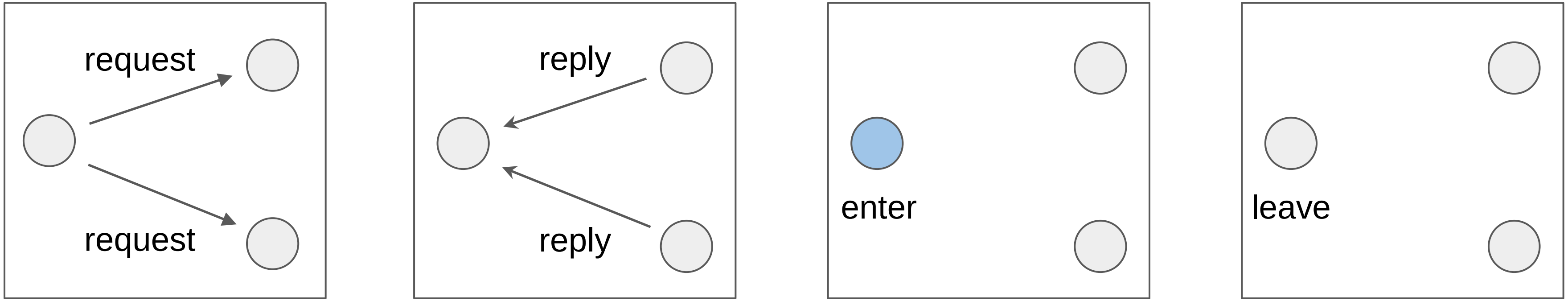
Correctness property: $\forall N1\, N2.\, holds(N1) \wedge holds(N2) \rightarrow N1 = N2$

Invariants:

$$\forall N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

Example: mutual exclusion protocol



Correctness property: $\forall\, N1\, N2.\, holds(N1) \land holds(N2) \rightarrow N1 = N2$

Invariants:

$\forall\, N1\, N2.\, \neg(replied(N1, N2) \land replied(N2, N1))$

$\forall\, N1\, N2.\, holds(N1) \land N1 \neq N2 \rightarrow replied(N1, N2)$

✔ inductive

Related work

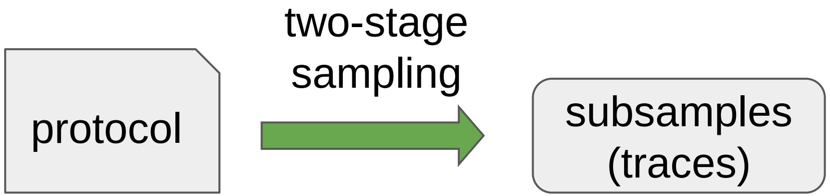
- IVy (Padon et al., PLDI ’16)
  - Cannot find invariants
- I4 (Ma et al., SOSP ’19)
  - Not guaranteed to find invariants
- FOL-IC3 (Koenig et al., PLDI ’20)
  - Slow in practice

Our contribution

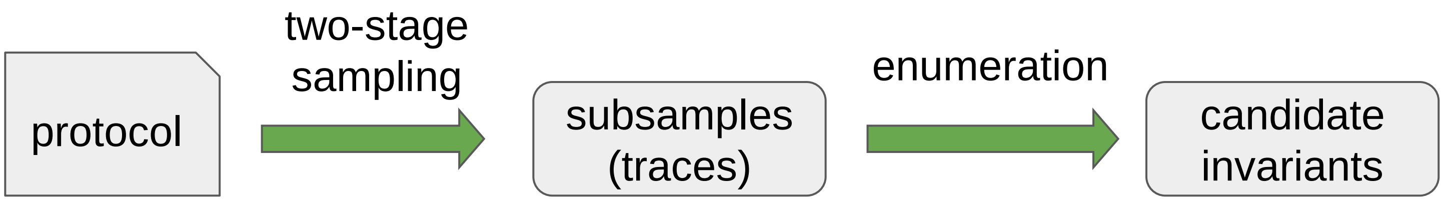
- DistAI, a data-driven method to learn inductive invariants for distributed protocols.
    - Fully automated
    - Guaranteed to succeed
    - Fast

DistAI workflow

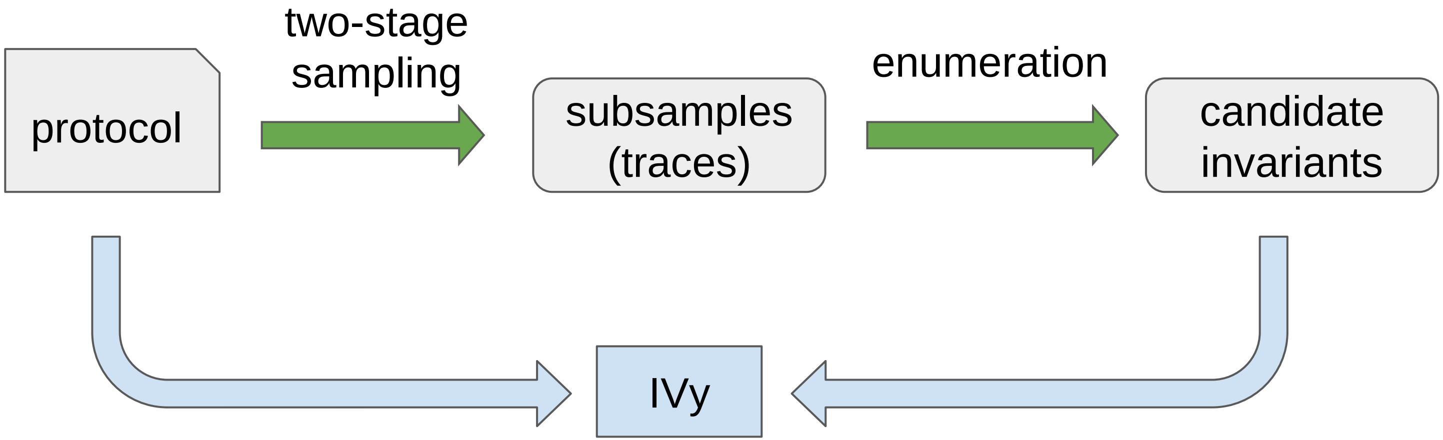
DistAI workflow



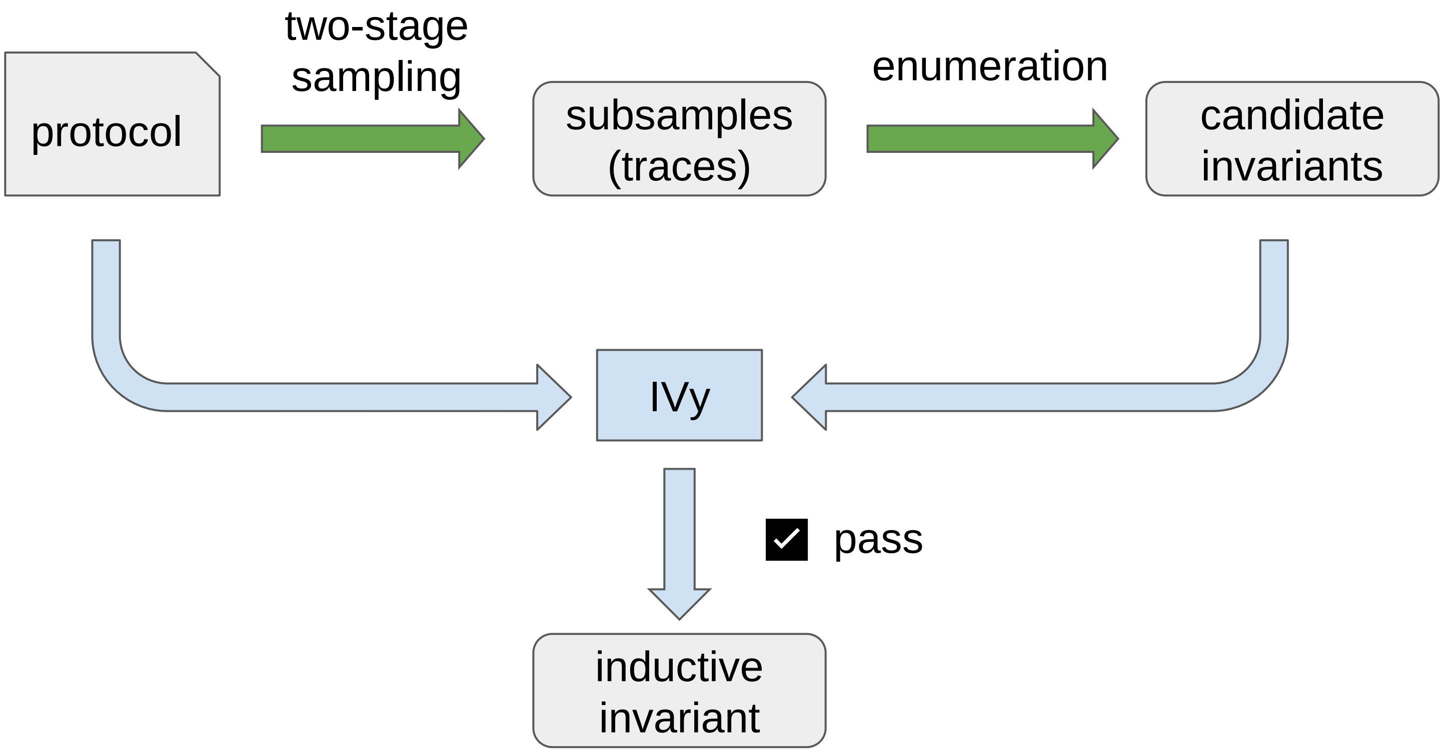
DistAI workflow



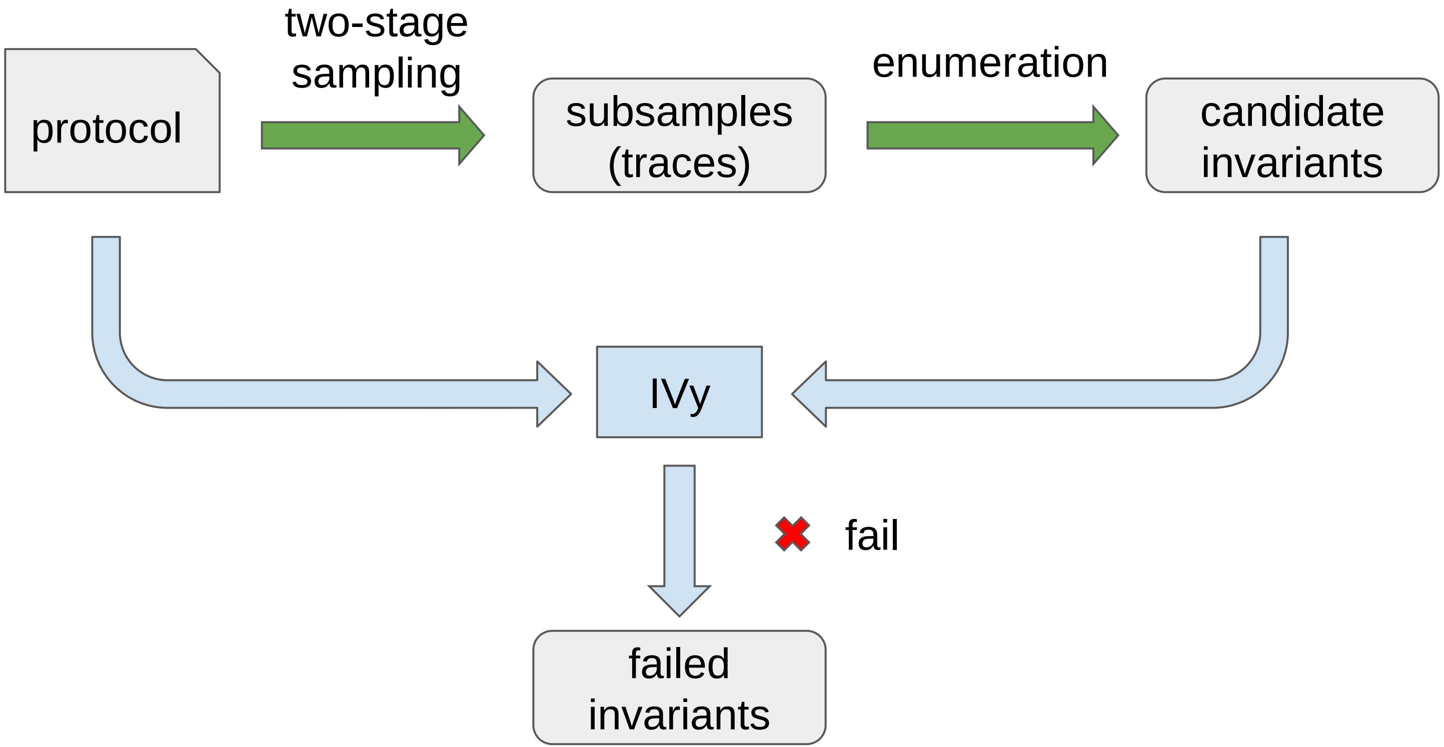
DistAI workflow



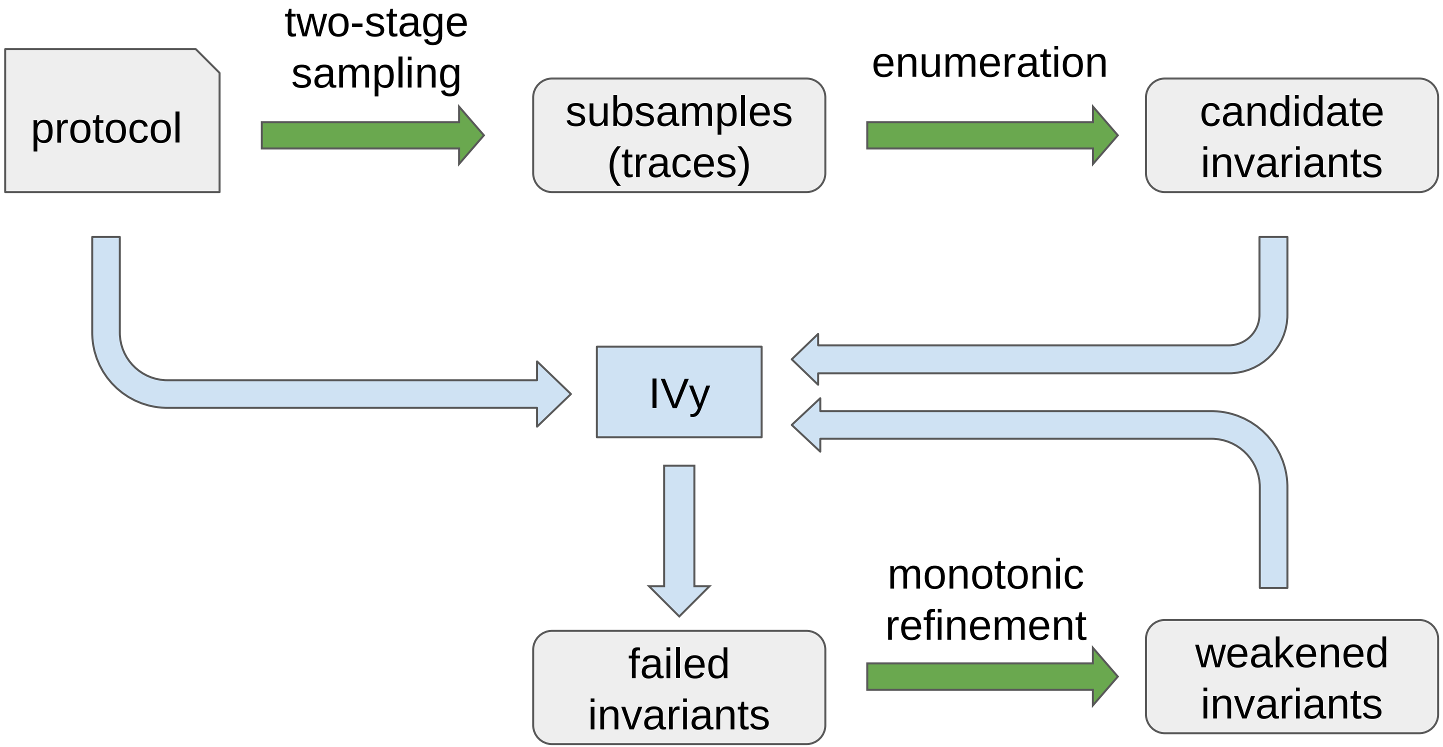
DistAI workflow



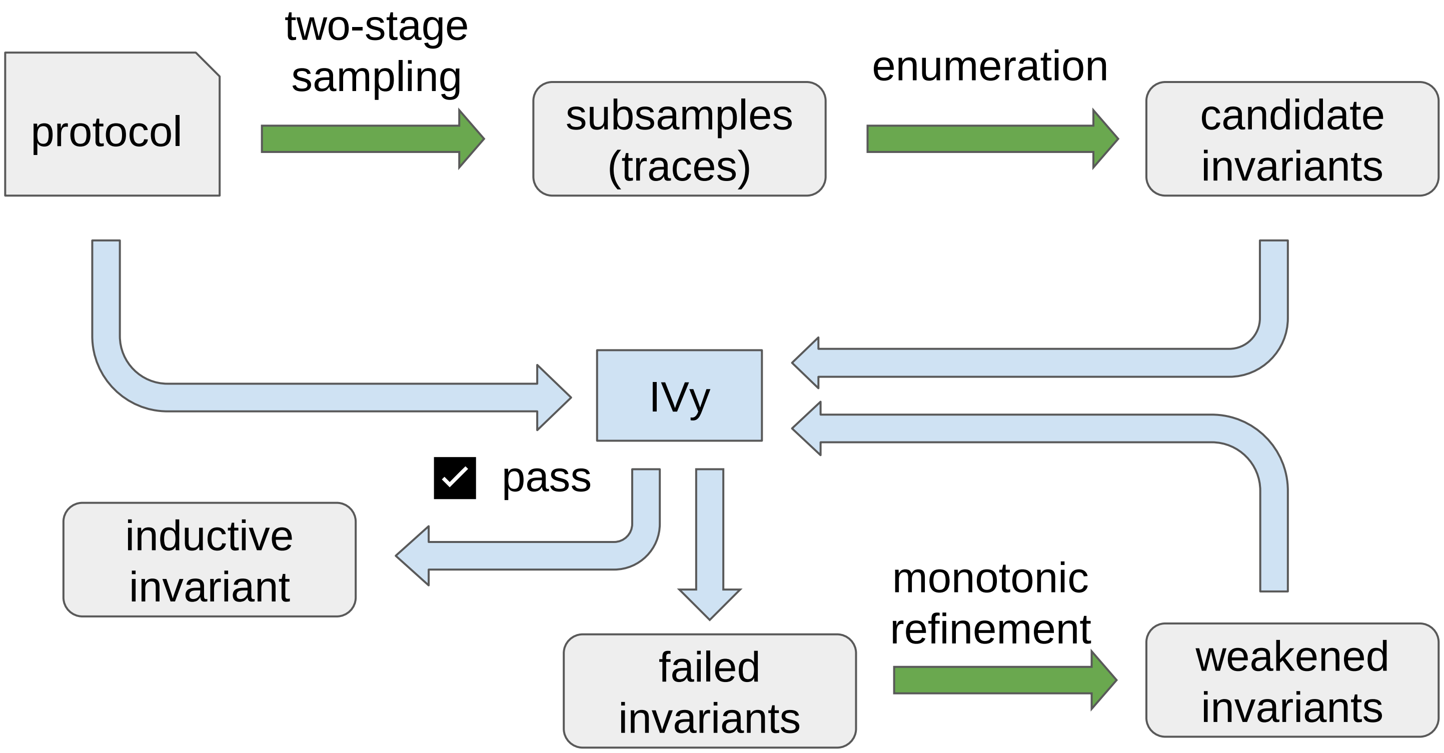
DistAI workflow



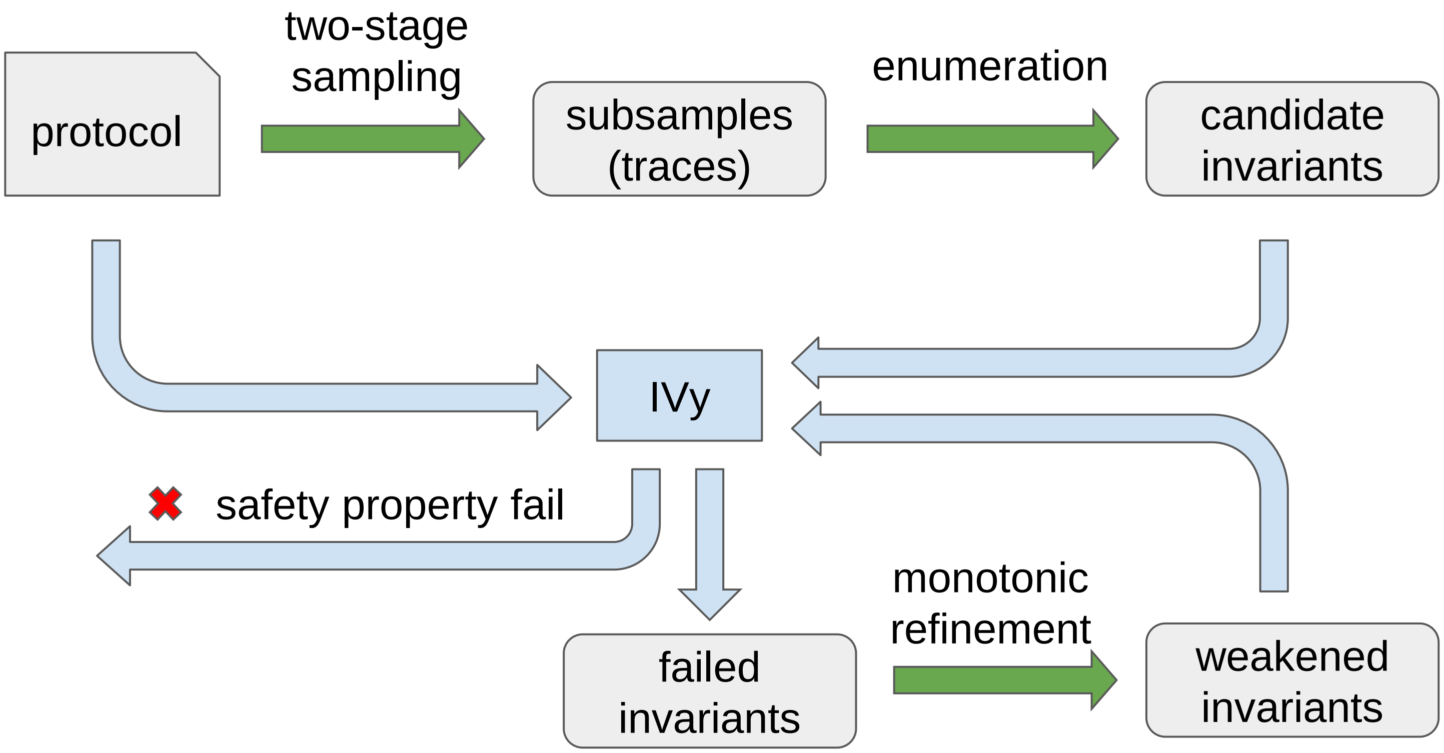
DistAI workflow



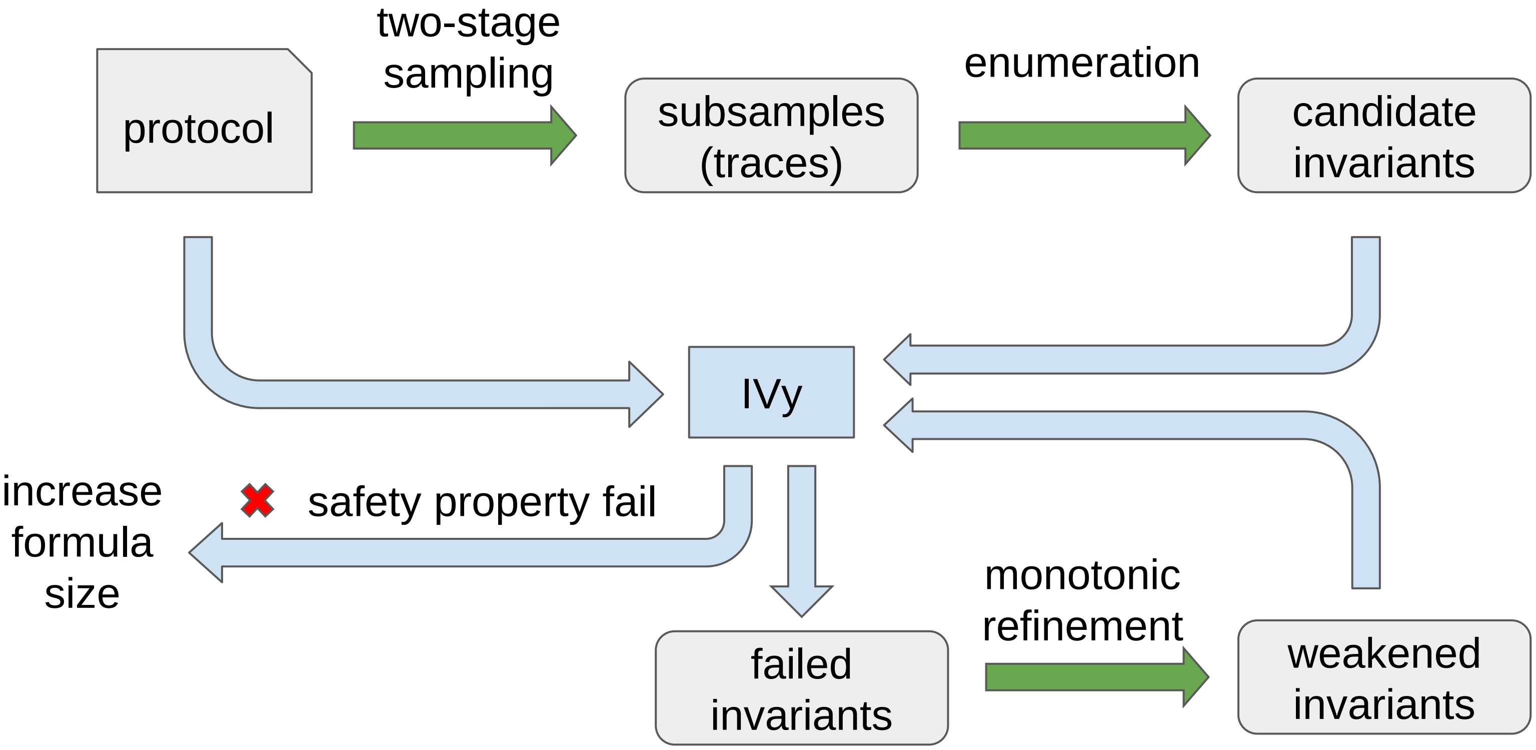
DistAI workflow



DistAI workflow



DistAI workflow

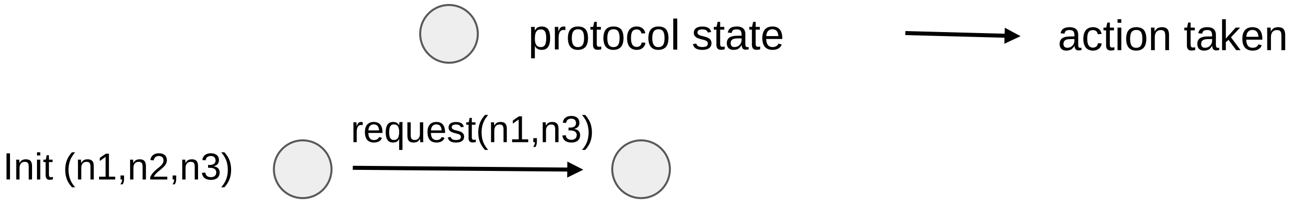


Sampling

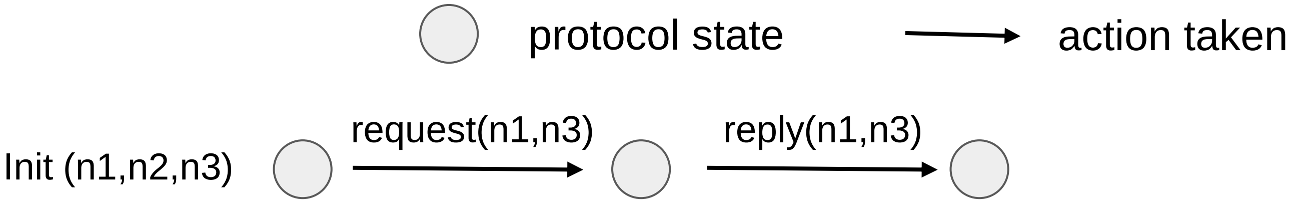
protocol state → action taken

Init (n1,n2,n3)

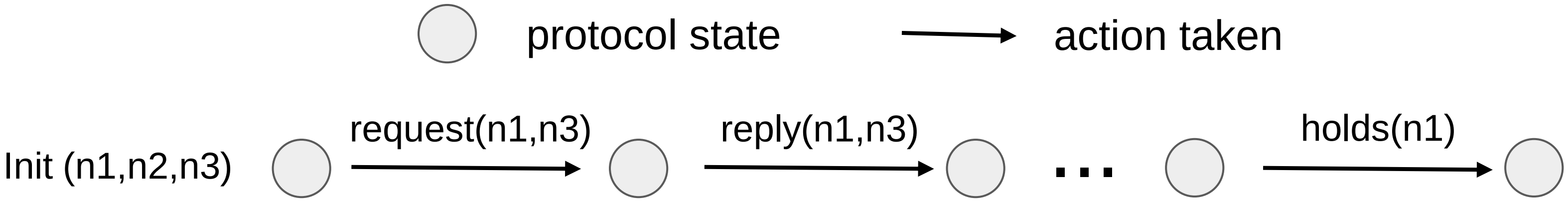
Sampling



Sampling

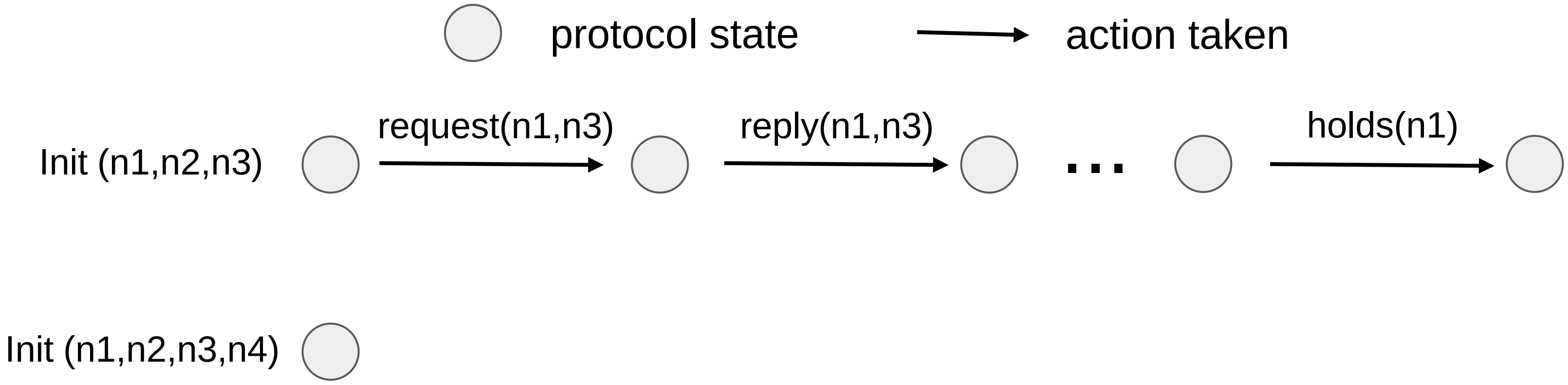

protocol state
action taken
request(n1,n3)
reply(n1,n3)
Init (n1,n2,n3)

Sampling



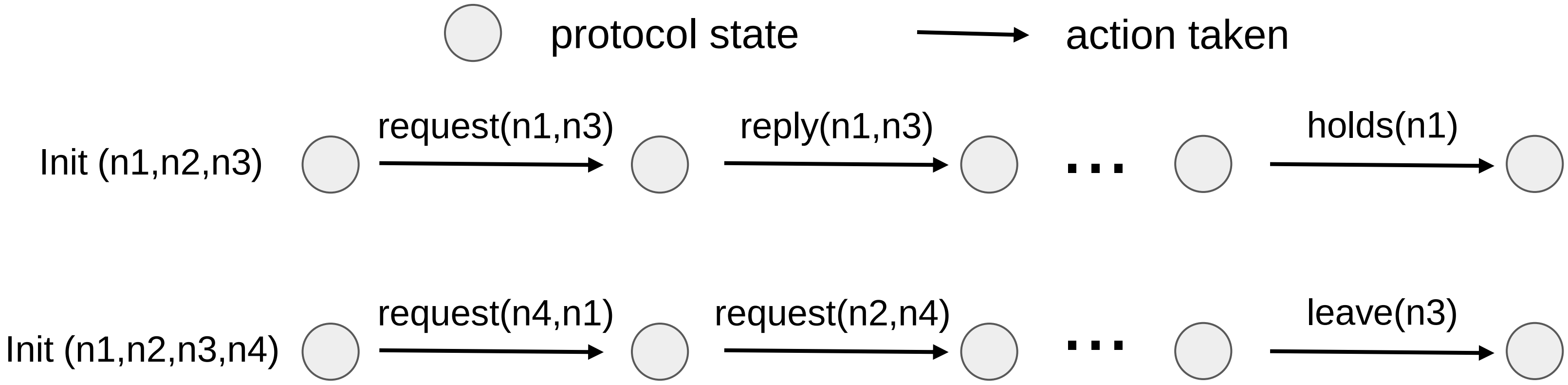
protocol state
action taken
Init (n1,n2,n3)
request(n1,n3)
reply(n1,n3)
...
holds(n1)

Sampling

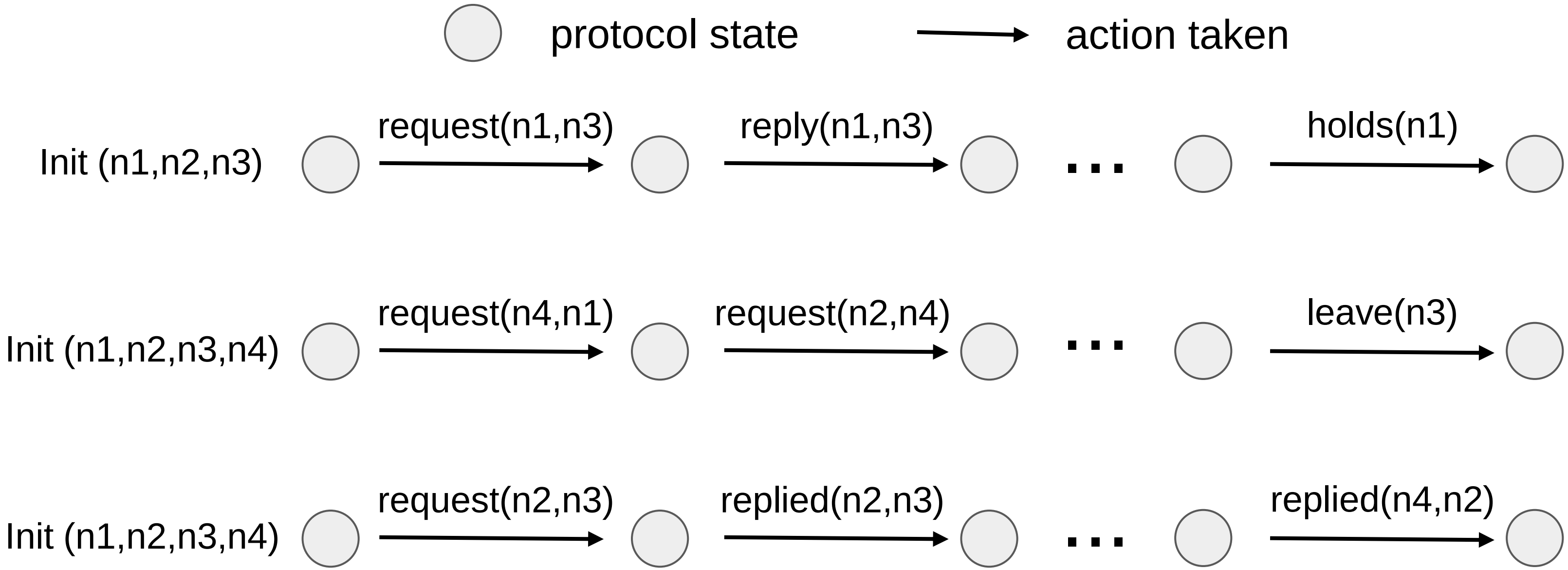


protocol state
action taken
request(n1,n3)
reply(n1,n3)
holds(n1)
Init (n1,n2,n3)
...
Init (n1,n2,n3,n4)

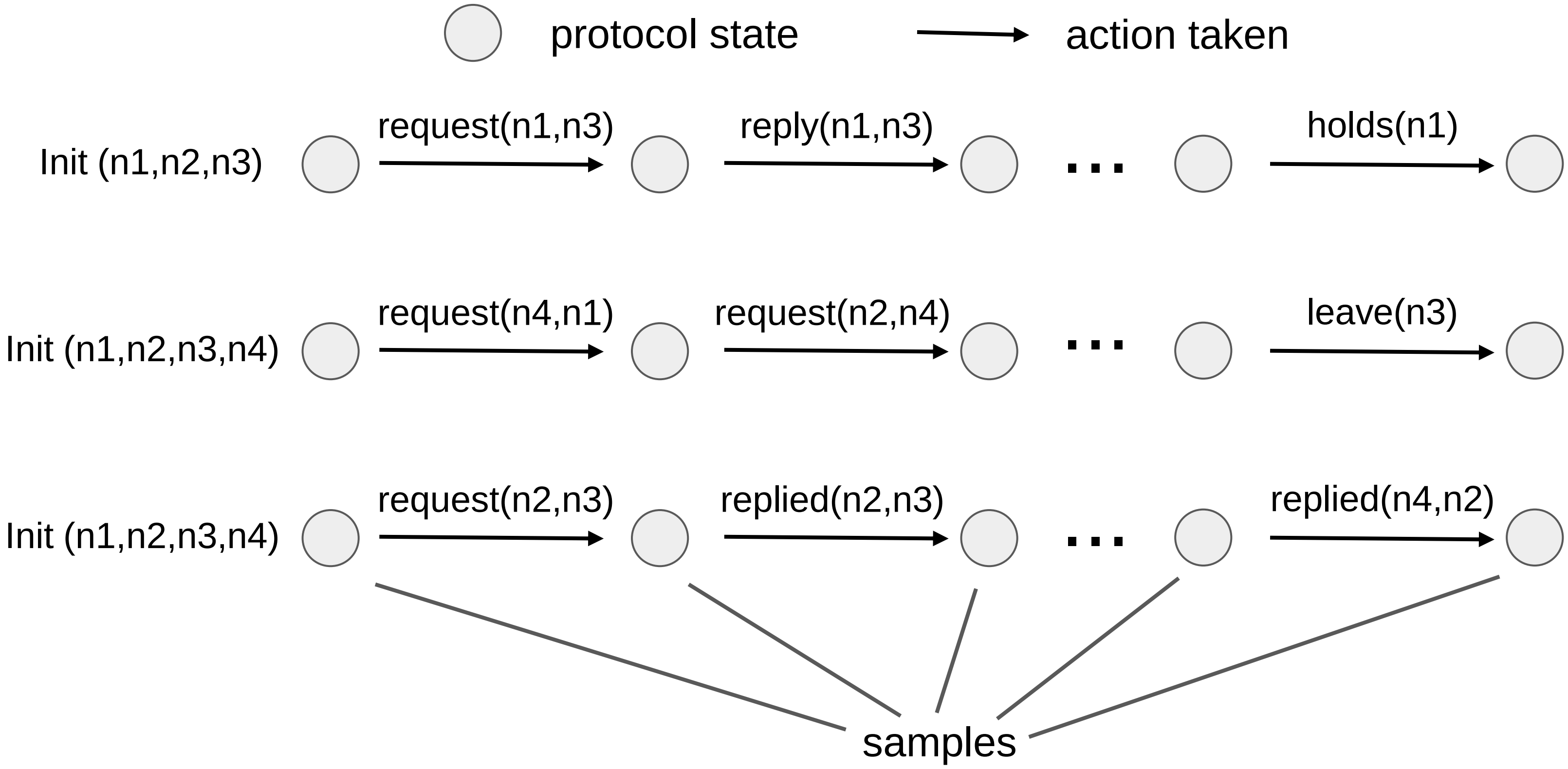
Sampling


protocol state
action taken
Init (n1,n2,n3)
request(n1,n3)
reply(n1,n3)
...
holds(n1)
Init (n1,n2,n3,n4)
request(n4,n1)
request(n2,n4)
...
leave(n3)

Sampling


protocol state
action taken
Init (n1,n2,n3)
request(n1,n3)
reply(n1,n3)
...
holds(n1)
Init (n1,n2,n3,n4)
request(n4,n1)
request(n2,n4)
...
leave(n3)
Init (n1,n2,n3,n4)
request(n2,n3)
replied(n2,n3)
...
replied(n4,n2)

Sampling



protocol state
action taken
Init (n1,n2,n3)
request(n1,n3)
reply(n1,n3)
...
holds(n1)
Init (n1,n2,n3,n4)
request(n4,n1)
request(n2,n4)
...
leave(n3)
Init (n1,n2,n3,n4)
request(n2,n3)
replied(n2,n3)
...
replied(n4,n2)
samples

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

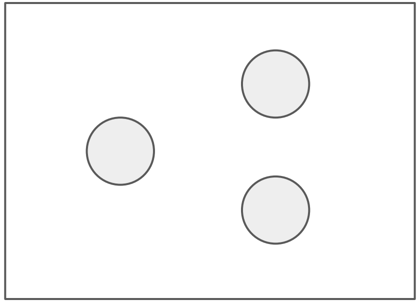
$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

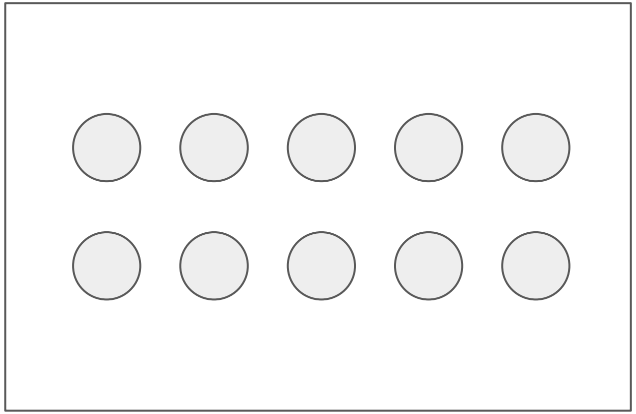
Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$



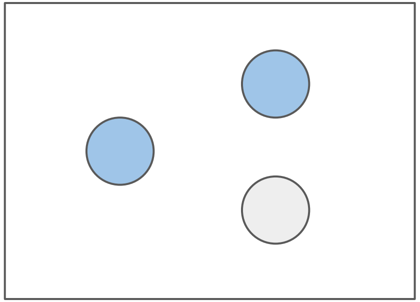


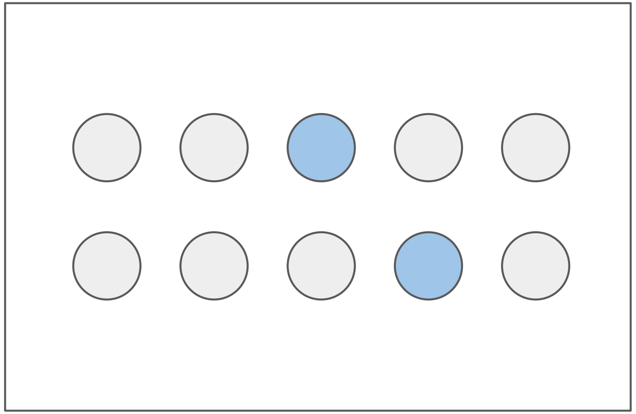
Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$





Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \land replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \land N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \land replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \land N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

requested(n2,n4)

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

$$N1 \leftarrow n2$$
$$N2 \leftarrow n3$$

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

$$N1 \leftarrow n2$$
$$N2 \leftarrow n3$$

requested(X,Y)

X\Y	N1	N2
N1	0	1
N2	1	0

Subsampling

- Real invariants: small number of quantified variables

$$\forall\, N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall\, N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall\, N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

$$N1 \leftarrow n2$$
$$N2 \leftarrow n3$$

requested(X,Y)

X\Y	N1	N2
N1	0	1
N2	1	0

subsample

Subsampling

- Real invariants: small number of quantified variables

$$\forall N1\, N2.\, \neg(replied(N1, N2) \wedge replied(N2, N1))$$

$$\forall N1\, N2.\, holds(N1) \wedge N1 \neq N2 \rightarrow replied(N1, N2)$$

- Given a template (e.g., $\forall N1\, N2$), project samples using variable mappings

requested(X,Y)

X\Y	n1	n2	n3	n4
n1	0	1	1	1
n2	1	0	1	0
n3	1	1	0	0
n4	1	1	1	1

$$N1 \leftarrow n3$$
$$N2 \leftarrow n4$$

requested(X,Y)

X\Y	N1	N2
N1	0	0
N2	1	1

subsample

Candidate invariant enumeration

- An invariant must hold on every subsample.

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall\, N1\, N2$) & max-literal (e.g., 3).

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall N1\, N2$) & max-literal (e.g., 3).

$$\forall N1\, N2.\, p(N1, N2) \ \vee\ q(N1, N2) \ \vee\ r(N1) \ \vee\ r(N2)$$

literal = 4

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall N1\, N2$) & max-literal (e.g., 3).
- Check the formula against the subsamples.

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall N1\, N2$) & max-literal (e.g., 3).
- Check the formula against the subsamples.
- If implied by an existing invariant, skip the check.

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall N1\, N2$) & max-literal (e.g., 3).
- Check the formula against the subsamples.
- If implied by an existing invariant, skip the check.

$$\forall N1.\, \neg replied(N1, N1)$$

$$\forall N1.\, \neg replied(N1, N1) \;\vee\; holds(N1)$$

$$\forall N1 \neq N2.\, \neg replied(N1, N1)$$

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall\, N1\, N2$) & max-literal (e.g., 3).
- Check the formula against the subsamples.
- If implied by an existing invariant, skip the check.

$\forall\, N1.\, \neg replied(N1, N1)$	☑
$\forall\, N1.\, \neg replied(N1, N1)\ \vee\ holds(N1)$	skip
$\forall\, N1 \neq N2.\, \neg replied(N1, N1)$	skip

Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., $\forall\, N1\, N2$) & max-literal (e.g., 3).
- Check the formula against the subsamples.
- If implied by an existing invariant, skip the check.

$\forall\, N1.\, \neg replied(N1, N1)$	✖
$\forall\, N1.\, \neg replied(N1, N1) \lor holds(N1)$	check
$\forall\, N1 \neq N2.\, \neg replied(N1, N1)$	check

Candidate invariant enumeration

- The **strongest** possible invariants w.r.t. the subsamples.

Candidate invariant enumeration

- The **strongest** possible invariants w.r.t. the subsamples.

I enumerated candidate invariants

I' any invariants that holds on the subsamples

Candidate invariant enumeration

- The **strongest** possible invariants w.r.t. the subsamples.

I enumerated candidate invariants

I' any invariants that holds on the subsamples

$$I \implies I'$$

Candidate invariant enumeration

- The **strongest** possible invariants w.r.t. the subsamples

I enumerated candidate invariants

I' any invariants that holds on the subsamples

$$I \implies I'$$

- Feed candidate invariants to IVy

Monotonic refinement

$$(enumerated)\ I \implies I^*\ (correct)$$

Monotonic refinement

$$(enumerated)\ I \implies I^*\ (correct)$$

- An invariant must hold on every subsample. ✅
- A formula that holds on every subsample is an invariant. ❌

Monotonic refinement

$$(enumerated)\ I \implies I^*\ (correct)$$

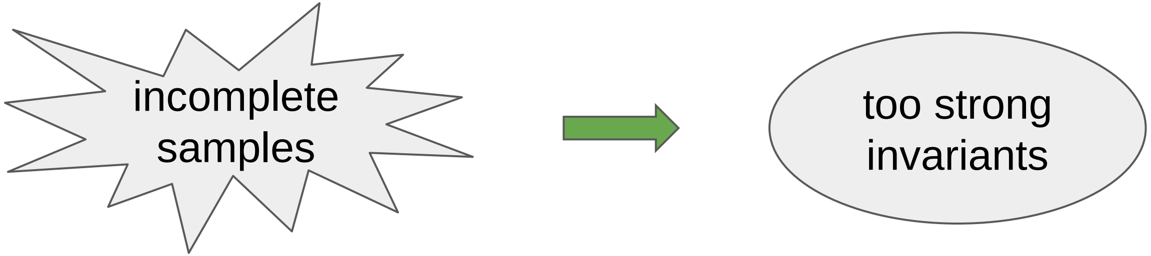
- An invariant must hold on every subsample. ✓
- A formula that holds on every subsample is an invariant. ✖

incomplete samples

Monotonic refinement

$$(enumerated)\ I \implies I^*\ (correct)$$

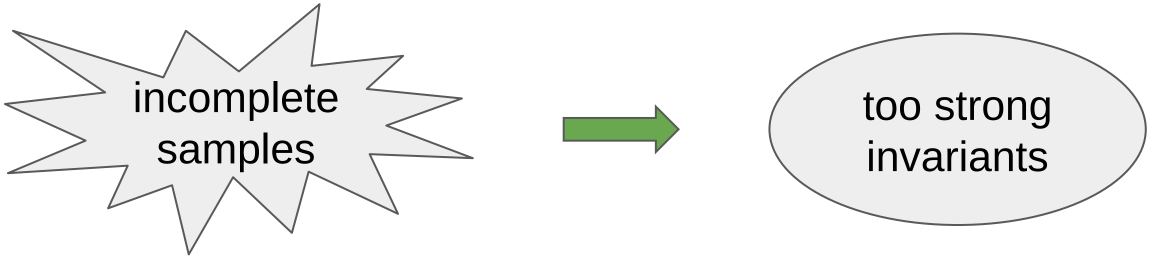
- An invariant must hold on every subsample. ✅
- A formula that holds on every subsample is an invariant. ❌



Monotonic refinement

$$(enumerated)\ I \implies I^*\ (correct)$$

- An invariant must hold on every subsample. ✓
- A formula that holds on every subsample is an invariant. ✗



weaken the candidate invariants

Monotonic refinement

- **Minimum** weakening

$$\forall\, N1.\, \neg replied(N1, N1)$$ ✖

Monotonic refinement

- **Minimum** weakening

$$\forall N1.\, \neg replied(N1, N1)$$ ✖

- Add all its weakened variants to the candidate set

Monotonic refinement

- **Minimum** weakening

$$\forall\, N1.\, \neg replied(N1, N1)$$ ✖

- Add all its weakened variants to the candidate set

$$\forall\, N1.\, \neg replied(N1, N1)\ \vee\ holds(N1)$$

$$\forall\, N1 \neq N2.\, \neg replied(N1, N1)$$

Monotonic refinement

- **Minimum** weakening

$$\forall\, N1.\, \neg replied(N1, N1)$$

- Add all its weakened variants to the candidate set

$$\forall\, N1.\, \neg replied(N1, N1) \;\vee\; holds(N1)$$

$$\forall\, N1 \neq N2.\, \neg replied(N1, N1)$$

- Never “bypass” the correct invariants

Convergence

- When safety property fails, increase template or maximum literal and retry.

Convergence

- When safety property fails, increase template or maximum literal and retry.

Theorem 3. *If the safety property of a protocol is provable with a ∃-free invariant set, then DistAI will terminate with one such invariant set in finite time.*

Convergence

- When safety property fails, increase template or maximum literal and retry.

Theorem 3. *If the safety property of a protocol is provable with a ∃-free invariant set, then DistAI will terminate with one such invariant set in finite time.*

strongest invariants + **minimum** weakening

Evaluation

- Evaluated on 14 distributed protocols (12 from prior work, 2 newly introduced)
- Compared with I4 and FOL-IC3

Evaluation

Protocol		**Protocol**	
asynchronous lock server		chord ring maintenance	
database chain replication		decentralized lock	
distributed lock		hashed sharding	
leader election		learning switch	
lock server		Paxos	
permissioned blockchain		Ricart-Agrawala	
simple consensus		two-phase commit	

Evaluation

■ I4

Protocol		Protocol	
asynchronous lock server		chord ring maintenance	■
database chain replication	■	decentralized lock	
distributed lock	■	hashed sharding	
leader election	■	learning switch	■
lock server	■	Paxos	
permissioned blockchain		Ricart-Agrawala	■
simple consensus	■	two-phase commit	■

Evaluation

■ I4 ■ FOL-IC3

Protocol		Protocol	
asynchronous lock server	FOL-IC3	chord ring maintenance	I4
database chain replication	I4	decentralized lock	FOL-IC3
distributed lock	I4, FOL-IC3	hashed sharding	FOL-IC3
leader election	I4, FOL-IC3	learning switch	I4
lock server	I4, FOL-IC3	Paxos	
permissioned blockchain	FOL-IC3	Ricart-Agrawala	I4, FOL-IC3
simple consensus	I4	two-phase commit	I4, FOL-IC3

Evaluation

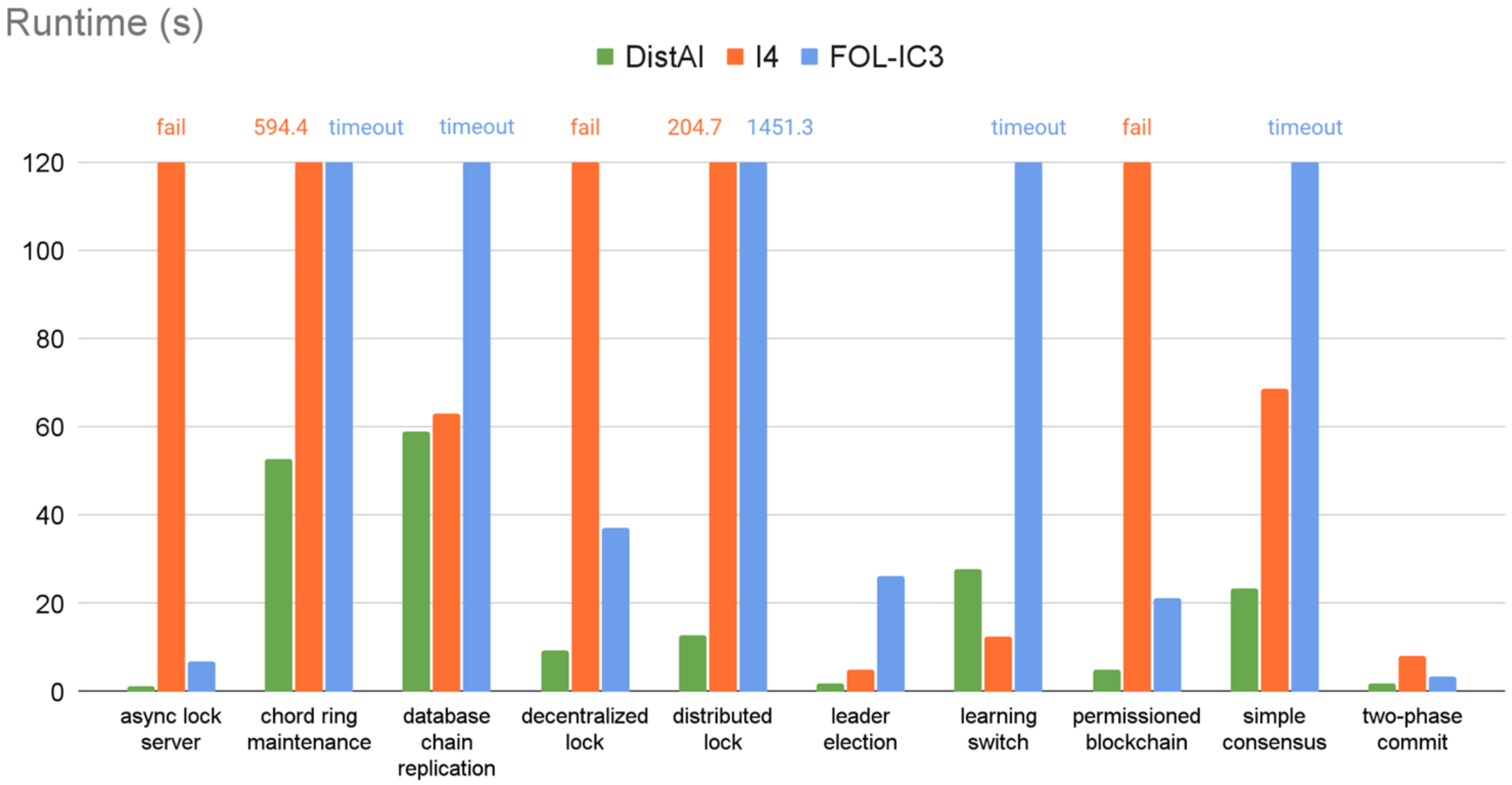
■ I4 ■ FOL-IC3 ■ **DistAI**

Protocol				**Protocol**			
asynchronous lock server		FOL-IC3	DistAI	chord ring maintenance	I4		DistAI
database chain replication	I4		DistAI	decentralized lock		FOL-IC3	DistAI
distributed lock	I4	FOL-IC3	DistAI	hashed sharding		FOL-IC3	DistAI
leader election	I4	FOL-IC3	DistAI	learning switch	I4		DistAI
lock server	I4	FOL-IC3	DistAI	Paxos			DistAI
permissioned blockchain		FOL-IC3	DistAI	Ricart-Agrawala	I4	FOL-IC3	DistAI
simple consensus	I4		DistAI	two-phase commit	I4	FOL-IC3	DistAI

Evaluation

■ I4 ■ FOL-IC3 ■ **DistAI**

Protocol				**Protocol**			
asynchronous lock server		■	■	chord ring maintenance	■		■
database chain replication	■		■	decentralized lock		■	■
distributed lock	■	■	■	hashed sharding		■	■
leader election	■	■	■	learning switch	■		■
lock server	■	■	■	Paxos			□
permissioned blockchain		■	■	Ricart-Agrawala	■	■	■
simple consensus	■		■	two-phase commit	■	■	■



Runtime (s)
DistAI
I4
FOL-IC3
fail
594.4
timeout
timeout
fail
204.7
1451.3
timeout
fail
timeout
120
100
80
60
40
20
0
async lock server
chord ring maintenance
database chain replication
decentralized lock
distributed lock
leader election
learning switch
permissioned blockchain
simple consensus
two-phase commit

Conclusion

- We present DistAI, a data-driven automated invariant learning system
  - Two-stage sampling
  - Candidate invariant enumeration
  - Monotonic refinement
- Compared with alternative methods, DistAI
  - Fully automated
  - Guarantee to succeed
  - Much faster

Thank you

- Feel free to contact us if you have any questions
    - Jianan Yao: jianan@cs.columbia.edu
    - Runzhou Tao: runzhou.tao@columbia.edu