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
  - ...

Distributed systems are hard to implement correctly due to:
- 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
  - ...

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

\[
\forall N1 \ N2. \ holds(N1) \land holds(N2) \rightarrow N1 = N2
\]
Example: mutual exclusion protocol

Correctness property: \( \forall N1, N2. holds(N1) \land holds(N2) \rightarrow N1 = N2 \)
Example: mutual exclusion protocol

Correctness property: \( \forall N1, N2. \text{holds}(N1) \land \text{holds}(N2) \rightarrow N1 = N2 \)

Invariants:

\( \forall N1, N2. \neg(\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \)

\( \forall N1, N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \)
Example: mutual exclusion protocol

Correctness property: \[ \forall N1 \, N2. \, \text{holds}(N1) \land \text{holds}(N2) \rightarrow N1 = N2 \]

Invariants:
\[ \forall N1 \, N2. \, \neg(\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \]
\[ \forall N1 \, N2. \, \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \]
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

protocol
DistAI workflow

- protocol
- two-stage sampling
- subsamples (traces)
DistAI workflow

protocol → two-stage sampling → subsamples (traces) → enumeration → candidate invariants
DistAI workflow

- protocol
- two-stage sampling
- subsamples (traces)
- enumeration
- candidate invariants

IVy
DistAI workflow

protocol \rightarrow \text{two-stage sampling} \rightarrow \text{subsamples (traces)} \rightarrow \text{enumeration} \rightarrow \text{candidate invariants}

\text{IVy} \leftrightarrow \text{inductive invariant}

\checkmark \text{ pass}
DistAI workflow

- protocol
- two-stage sampling
- subsamples (traces)
- enumeration
- candidate invariants
- IVy
- fail
- failed invariants
DistAI workflow

- Protocol
- Two-stage sampling
- Subsamples (traces)
- Enumeration
- Candidate invariants
- IVy
- Failed invariants
- Monotonic refinement
- Weakened invariants
DistAI workflow

1. Protocol
2. Two-stage sampling
3. Subsamples (traces)
4. Enumeration
5. Candidate invariants

- IVy
  - Inductive invariant
  - Passed
  - Failed invariants
  - Monotonic refinement
  - Weakened invariants
DistAI workflow

- **protocol**
- two-stage sampling
- subsamples (traces)
- enumeration
- candidate invariants

**IVy**
- safety property fail
- failed invariants
- weakened invariants

- monotonic refinement
DistAI workflow

- **protocol** → **two-stage sampling** → **subsamples (traces)** → **enumeration** → **candidate invariants**

- **increase formula size** → **safety property fail** → **failed invariants**

- **IVy**

- **monotonic refinement** → **weakened invariants**
Sampling

Init \( (n_1, n_2, n_3) \)

protocol state \( \rightarrow \) action taken
Sampling

Init (n1, n2, n3) \[\rightarrow\] request(n1, n3) \[\rightarrow\] protocol state \[\rightarrow\] action taken
Sampling

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

Init (n1,n2,n3)  \(\rightarrow\) request(n1,n3)  \(\rightarrow\) reply(n1,n3)  \(\rightarrow\) ...  \(\rightarrow\) holds(n1)
Sampling

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

Init (n1,n2,n3,n4)
Sampling

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

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

- 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

\[
\forall N1 \, N2. \, \neg (replied(N1, N2) \land replied(N2, N1))
\]

\[
\forall N1 \, N2. \, holds(N1) \land N1 \neq N2 \rightarrow replied(N1, N2)
\]
Subsampling

- Real invariants: small number of quantified variables

\[ \forall N1 \ N2. \neg (\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \]
\[ \forall N1 \ N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \]
Subsampling

- Real invariants: small number of quantified variables

\[
\forall N1\ N2. \neg (\text{replied}(N1, N2) \land \text{replied}(N2, N1))
\]

\[
\forall N1\ N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2)
\]
Subsampling

- Real invariants: small number of quantified variables

  \[ \forall N1 N2. \neg (\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \]

  \[ \forall N1 N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \]

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

- Real invariants: small number of quantified variables
  \[ \forall N1 \, N2. \neg (\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \]
  \[ \forall N1 \, N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \]

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

<table>
<thead>
<tr>
<th>X\Y</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>( X \setminus Y )</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>X\Y</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Subsampling

- Real invariants: small number of quantified variables
  \[ \forall N_1 N_2. \neg (\text{replied}(N_1, N_2) \land \text{replied}(N_2, N_1)) \]
  \[ \forall N_1 N_2. \text{holds}(N_1) \land N_1 \neq N_2 \rightarrow \text{replied}(N_1, N_2) \]

- Given a template (e.g., \( \forall N_1 N_2 \)), project samples using variable mappings

\[
\begin{array}{c|cccc}
\text{X}/\text{Y} & n_1 & n_2 & n_3 & n_4 \\
\hline
n_1 & 0 & 1 & 1 & 1 \\
n_2 & 1 & 0 & 1 & 0 \\
n_3 & 1 & 1 & 0 & 0 \\
n_4 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{X}/\text{Y} & N_1 & N_2 \\
\hline
N_1 & 0 & 1 \\
N_2 & 1 & 0 \\
\end{array}
\]
Subsampling

- Real invariants: small number of quantified variables
  \[ \forall N1 \, N2. \neg (replied(N1, N2) \land replied(N2, N1)) \]
  \[ \forall N1 \, N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow replied(N1, N2) \]
- Given a template (e.g., \( \forall N1 \, N2 \)), project samples using variable mappings

<table>
<thead>
<tr>
<th>X\Y</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Subsampling

- Real invariants: small number of quantified variables
  \[ \forall N1 N2. \neg (\text{replied}(N1, N2) \land \text{replied}(N2, N1)) \]
  \[ \forall N1 N2. \text{holds}(N1) \land N1 \neq N2 \rightarrow \text{replied}(N1, N2) \]

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

<table>
<thead>
<tr>
<th>(X\backslash Y)</th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \text{requested}(X,Y) \)

\( N1 \leftarrow n3 \)
\( N2 \leftarrow n4 \)

\( \text{subsample} \)

\( \text{requested}(X,Y) \)

<table>
<thead>
<tr>
<th>(X\backslash Y)</th>
<th>N1</th>
<th>N2</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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., $\forall N_1 N_2$) & max-literal (e.g., 3).
Candidate invariant enumeration

- An invariant must hold on every subsample.
- Enumerate formulas under a template (e.g., \( \forall N_1 \, N_2 \)) & max-literal (e.g., 3).

\[
\forall N_1 \, N_2. \, p(N_1, N_2) \lor q(N_1, N_2) \lor r(N_1) \lor r(N_2) \quad \# \text{ literal } = 4
\]
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.

\[
\forall N1. \neg replied(N1, N1)
\]

\[
\forall N1. \neg replied(N1, N1) \lor holds(N1)
\]

\[
\forall N1 \neq N2. \neg replied(N1, N1)
\]
Candidate invariant enumeration

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

$$\forall N1. \neg \text{replied}(N1, N1)$$ skip

$$\forall N1. \neg \text{replied}(N1, N1) \lor \text{holds}(N1)$$ skip

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

\[
\forall N1. \neg\text{replied}(N1, N1) \quad \text{x}
\]

\[
\forall N1. \neg\text{replied}(N1, N1) \lor \text{holds}(N1) \quad \text{check}
\]

\[
\forall N1 \neq N2. \neg\text{replied}(N1, N1) \quad \text{check}
\]
Candidate invariant enumeration

- The strongest possible invariants w.r.t. the subsamples.
Candidate invariant enumeration

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

\[
I \quad \text{enumerated candidate invariants}
\]

\[
I' \quad \text{any invariants that holds on the subsamples}
\]
Candidate invariant enumeration

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

\[ I \quad \text{enumerated candidate invariants} \]

\[ I' \quad \text{any invariants that holds on the subsamples} \]

\[ I \implies I' \]
Candidate invariant enumeration

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

\[ I \quad \text{enumerated candidate invariants} \]

\[ I' \quad \text{any invariants that holds on the subsamples} \]

\[ I \implies I' \]

- Feed candidate invariants to IVy
Monotonic refinement

\[(\text{enumerated}) \  I \implies I^* \ (\text{correct})\]
Monotonic refinement

\[(\text{enumerated}) \ I \implies I^* \ (\text{correct})\]

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

$$(\text{enumerated}) \ I \implies I^* \ (\text{correct})$$

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

\[(\text{enumerated}) \ I \implies I^* \ (\text{correct})\]

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

\[(\text{enumerated}) \ I \implies I^* \ (\text{correct})\]

- 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

\[ \forall N1. \neg \text{replied}(N1, N1) \]
Monotonic refinement

- **Minimum** weakening

  \[ \forall N1. \neg \text{replied}(N1, N1) \]

- Add all its weakened variants to the candidate set
Monotonic refinement

- **Minimum** weakening

  \[ \forall N1. \neg \text{replied}(N1, N1) \quad \times \]

- Add all its weakened variants to the candidate set

  \[ \forall N1. \neg \text{replied}(N1, N1) \lor \text{holds}(N1) \]

  \[ \forall N1 \neq N2. \neg \text{replied}(N1, N1) \]
Monotonic refinement

- **Minimum** weakening
  \[ \forall N1. \neg \text{replied}(N1, N1) \]

- Add all its weakened variants to the candidate set
  \[ \forall N1. \neg \text{replied}(N1, N1) \lor \text{holds}(N1) \]
  \[ \forall N1 \neq N2. \neg \text{replied}(N1, N1) \]

- 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 $\exists$-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 $\exists$-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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>asynchronous lock server</td>
<td>chord ring maintenance</td>
</tr>
<tr>
<td>database chain replication</td>
<td>decentralized lock</td>
</tr>
<tr>
<td>distributed lock</td>
<td>hashed sharding</td>
</tr>
<tr>
<td>leader election</td>
<td>learning switch</td>
</tr>
<tr>
<td>lock server</td>
<td>Paxos</td>
</tr>
<tr>
<td>permissioned blockchain</td>
<td>Ricart-Agrawala</td>
</tr>
<tr>
<td>simple consensus</td>
<td>two-phase commit</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>asynchronous lock server</td>
<td>chord ring maintenance</td>
</tr>
<tr>
<td>database chain replication</td>
<td>decentralized lock</td>
</tr>
<tr>
<td>distributed lock</td>
<td>hashed sharding</td>
</tr>
<tr>
<td>leader election</td>
<td>learning switch</td>
</tr>
<tr>
<td>lock server</td>
<td>Paxos</td>
</tr>
<tr>
<td>permissioned blockchain</td>
<td>Ricart-Agrawala</td>
</tr>
<tr>
<td>simple consensus</td>
<td>two-phase commit</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>I4</th>
<th>FOL-IC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>asynchronous lock server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chord ring maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>database chain replication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>decentralized lock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distributed lock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hashed sharding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>leader election</td>
<td></td>
<td></td>
</tr>
<tr>
<td>learning switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lock server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paxos</td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissioned blockchain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ricart-Agrawala</td>
<td></td>
<td></td>
</tr>
<tr>
<td>simple consensus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>two-phase commit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>I4</th>
<th>FOL-IC3</th>
<th>DistAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>asynchronous lock server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>database chain replication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>distributed lock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>leader election</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lock server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissioned blockchain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>simple consensus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chord ring maintenance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decentralized lock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hashed sharding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>learning switch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paxos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ricart-Agrawala</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>two-phase commit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>asynchronous lock server</td>
<td>chord ring maintenance</td>
</tr>
<tr>
<td>database chain replication</td>
<td>decentralized lock</td>
</tr>
<tr>
<td>distributed lock</td>
<td>hashed sharding</td>
</tr>
<tr>
<td>leader election</td>
<td>learning switch</td>
</tr>
<tr>
<td>lock server</td>
<td>Paxos</td>
</tr>
<tr>
<td>permissioned blockchain</td>
<td>Ricart-Agrawala</td>
</tr>
<tr>
<td>simple consensus</td>
<td>two-phase commit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I4</th>
<th>FOL-IC3</th>
<th>DistAI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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