Inferring Phase Invariants from Phase Structures

Yotam Feldman  James R. Wilcox  Sharon Shoham  Mooly Sagiv

@yotamfe, @wilcoxjay, @SagivMooly
Safety of Infinite-State Systems
Inductive Invariants

inductiveness

initial

not bad
Distributed Protocols in EPR

EPR: A decidable fragment of first order logic

Used for modelling distributed protocols
[Padon et al. PLDI’16, OOPSLA’17, POPL’18, Taube et al. PLDI’18, Berkovits et al. CAV’19]

Our focus: universally quantified invariants for EPR distributed protocols
Proving with Inductive Invariants

**Deductive verification** – **manually** specify inductive invariant

Labor intensive

**Invariant inference** – **automatically** search for invariant

Limited and fragile
Our Approach

User-guided invariant inference – manually specify high-level intuition, automatically find full proof

1. Guide invariant inference using phase structures
2. Apply to inference of universally quantified invariants on challenging distributed protocols modelled in EPR
Example: Sharded Key-Value Store

State: modeled over global relations

- local state
- network

{n_1

{k_3 : ..., k_4 : 1, k_51 : ...}

{n_2

{k_1 : ..., k_7 : ..., k_95 : ...}

{k_6 : ..., k_32 : ..., k_65 : ...}

Example: Sharded Key-Value Store

change local table:
\[
\text{table}(n_1, k_4) := v
\]

\{k_3 : \ldots, \\
k_4 : \forall, \\
k_51 : \ldots\}\n
\{k_1 : \ldots, \\
k_7 : \ldots, \\
k_95 : \ldots\}\n
n_1 \quad n_2

\{k_6 : \ldots, \\
k_{32} : \ldots, \\
k_{65} : \ldots\}\n
Example: Sharded Key-Value Store

```
reshard:
  table(n₁, k₄) := ⊥
  transfer_msg(n₁, n₂, k₄, v, s₄₁) := true
```

Example: Sharded Key-Value Store

Example: Sharded Key-Value Store

\[
\text{drop transfer message:}
\]

\[
\text{transfer}_\text{msg}(n_1, n_2, k_4, v, s_{41}) := \text{false}
\]

\[
\{k_3 : \ldots, k_4 \times v, k_{51} : \ldots\}
\]

\[
\{k_1 : \ldots, k_7 : \ldots, k_{95} : \ldots\}
\]

\[
\{k_6 : \ldots, k_{32} : \ldots, k_{65} : \ldots\}
\]

Example: Sharded Key-Value Store

retransmit:

\[ \text{transfer}_{\text{msg}}(n_1, n_2, k_4, v, s_{41}) := \text{true} \]

recv transfer message:
\[
table(n_2, k_4) := v; \quad seq_{\text{recv}}(n_2, n_1, s_{41}) := true
\]
\[
\text{transfer}_{\text{msg}}(n_1, n_2, k_4, v, s_{41}) := false
\]

Example: Sharded Key-Value Store

Example: Sharded Key-Value Store

retransmit:

\[ \text{transfer}_{\text{msg}}(n_1, n_2, k_4, v, s_{41}) := \text{true} \]

\[ \{ k_3 : \ldots, k_4 \times v, k_{51} : \ldots \} \]

ignored according to seq num

\[ (k_4 : v, s_{41}) \]

\[ \{ k_1 : \ldots, k_4 : v, k_7 : \ldots, k_{95} : \ldots \} \]

\[ \{ k_6 : \ldots, k_{32} : \ldots, k_{65} : \ldots \} \]
Example: Sharded Key-Value Store

Safety property:
\[ \forall n_1, n_2, k, v_1, v_2. \quad \text{table}(n_1, k, v_1) \land \text{table}(n_2, k, v_2) \rightarrow v_1 = v_2 \]

Example: Sharded Key-Value Store

Deductive Verification for Sharded KV

\[\text{invariant } \forall k, n_1, n_2, v_1, v_2. \; \text{table}(n_1, k, v_1) \land \text{table}(n_2, k, v_2) \implies n_1 = n_2 \land v_1 = v_2\]

\[\text{invariant } \forall k, n_1, n_2. \; \text{owner}(n_1, k) \land \text{owner}(n_2, k) \implies n_1 = n_2\]

\[\text{invariant } \forall k, n, v. \; \text{table}(n, k, v) \implies \text{owner}(n, k)\]

\[\text{invariant } \forall k, \text{src}, \text{dst}, v, s, n. \; \neg (\text{transfer_msg}(\text{src}, \text{dst}, k, v, s) \land \neg \text{seqnum_recd}(\text{dst}, \text{src}, s) \land \text{owner}(n, k))\]

\[\text{invariant } \forall k, \text{src}, \text{dst}, v, s, n. \; \neg (\text{unacked}(\text{src}, \text{dst}, k, v, s) \land \neg \text{seqnum_recd}(\text{dst}, \text{src}, s) \land \text{owner}(n, k))\]

\[\text{invariant } \forall k, \text{src}_1, \text{src}_2, \text{dst}_1, \text{dst}_2, v_1, v_2, s_1, s_2. \; \text{transfer_msg}(\text{src}_1, \text{dst}_1, k, v_1, s_1) \land \neg \text{seqnum_recd}(\text{dst}_1, \text{src}_1, s_1) \land \text{transfer_msg}(\text{src}_2, \text{dst}_2, k, v_2, s_2) \land \neg \text{seqnum_recd}(\text{dst}_2, \text{src}_2, s_2) \implies \text{src}_1 = \text{src}_2 \land \text{dst}_1 = \text{dst}_2 \land v_1 = v_2 \land s_1 = s_2\]

\[\text{invariant } \forall k, \text{src}_1, \text{src}_2, \text{dst}_1, \text{dst}_2, v_1, v_2, s_1, s_2. \text{transfer_msg}(\text{src}_1, \text{dst}_1, k, v_1, s_1) \land \neg \text{seqnum_recd}(\text{dst}_1, \text{src}_1, s_1) \land \text{unacked}(\text{src}_2, \text{dst}_2, k, v_2, s_2) \land \neg \text{seqnum_recd}(\text{dst}_2, \text{src}_2, s_2) \implies \text{src}_1 = \text{src}_2 \land \text{dst}_1 = \text{dst}_2 \land v_1 = v_2 \land s_1 = s_2\]

\[\text{invariant } \forall \text{src}_1, \text{src}_2, \text{dst}_1, \text{dst}_2, v_1, v_2, s_1, s_2. \text{unacked}(\text{src}_1, \text{dst}_1, k, v_1, s_1) \land \neg \text{seqnum_recd}(\text{dst}_1, \text{src}_1, s_1) \land \text{unacked}(\text{src}_2, \text{dst}_2, k, v_2, s_2) \land \neg \text{seqnum_recd}(\text{dst}_2, \text{src}_2, s_2) \implies \text{src}_1 = \text{src}_2 \land \text{dst}_1 = \text{dst}_2 \land v_1 = v_2 \land s_1 = s_2\]

Labor intensive
## Invariant Inference for Sharded KV

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Inductive Invariant Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Sharded KV</td>
<td>failed to converge in 1 hour in 13/16 Z3 seeds</td>
</tr>
</tbody>
</table>

**Solution:** guide using phase structure

---

Phase Structure of Distributed KV’s Proof

∀k.

Key k is **owned**

Key k is **transferring**

- drop_msg (*, k)
- retransmit (*, k)
- ...(*, k)
- actions(*, k')
- **reshard**(*,*, k,*)
- **recv_transfer_msg**(*,*, k,*)
- drop_msg (*, k)
- retransmit (*, k)
- ...(*, k)
- actions(*, k')
Phase Structure of Distributed KV’s Proof

∀k.

Key \( k \) is **owned**
(by one node)

Key \( k \) is **transferring**
(to one node)

- **reshard\((\ast, \ast, k, \ast)\)**
- **recv_transfer_msg\((\ast, \ast, k, \ast)\)**
- **drop_msg \((\ast, k)\)**
- **retransmit \((\ast, k)\)**
- ...\((\ast, k)\)
- actions \((\ast, k')\)
∀\(k\).

Key \(k\) is **owned**

\(\varphi_1\)

Key \(k\) is **transferring**

\(\varphi_2\)

reshard\((*,*,k,*)\)

*Phase characterizations*

\(\forall n_1, n_2, v, s.\)
\(\neg(transfer\_msg(n_1, n_2, k, v, s)\)
\(\wedge \neg seq\_recvd(n_1, n_2, k, v, s))\)

... (actions)

\(\forall n,v. \neg table(n,k,v)\)

...
∀\(k\).

Key \(k\) is owned

\(\varphi_1\)

drop_msg (\(*, k\))
retransmit (\(*, k\))
\(\ldots (\*, k)\)
actions (\(*, k')\)

Key \(k\) is transferring

\(\varphi_2\)

recv_transfer_msg (\(*,*, k, *\))

Phase characterizations

reshard (\(*,*, k, *\))

Goal: automatically find inductive phase characterizations
Inductive Phase Invariants

\[ \forall k. \]

\[ \varphi_1 \xrightarrow{\text{drop}_\text{msg}} \text{retransmit} \]

\[ \xrightarrow{\text{recv}_\text{transfer}_\text{msg}} \]

\[ \varphi_2 \]

\[ \xrightarrow{\text{drop}_\text{msg}} \text{retransmit} \]
Inductive Phase Invariants

Init $\Rightarrow \varphi_1$

\[ \forall k. \]

Diagram:

- $\varphi_1$
  - incoming edges: reshard, drop_msg, retransmit, ...
  - outgoing edges: $\varphi_2$
- $\varphi_2$
  - incoming edges: recv_transfer_msg, drop_msg, retransmit, ...
  - outgoing edges: null
Inductive Phase Invariants

\[ \forall k. \varphi_1 \]\n
\[ \varphi_1 \Rightarrow \text{Safety, ...} \]
Inductive Phase Invariants

\[ \forall k. \quad \varphi_1 \Rightarrow \varphi_1 \]

\[ \varphi_1 \Rightarrow \text{Safety, ...} \]

\[ \varphi_1 \land TR_{\text{reshard}} \Rightarrow \varphi_2', ..., \]
Inductive Phase Invariants

\[ \forall k. \quad \varphi_1 \implies \varphi_1 \wedge TR_{reshard} \implies \varphi_2', \ldots \]

\[ \varphi_1 \wedge TR \implies TR_{reshard} \vee TR_{drop_msg} \vee ..., \ldots \]

\[ Init \implies \varphi_1 \]

\[ \varphi_1 \implies Safety, \ldots \]
Inference Using Phase Structures

Phase structure

Protocol

Safety

Inductive phase invariant

∀

∀

∀
Inference Using Phase Structures

Phase structure

∀k

安全协议

counterexample trace: no inductive phase invariant
Inferring Inductive Phase Invariants

\[ \text{Init} \implies \varphi_1 \]

\[ \varphi_1 \implies \text{Safety, ...} \]

\[ \varphi_1 \land TR_{\text{reshard}} \implies \varphi_2', ..., \]

\[ \varphi_1 \land TR \implies TR_{\text{reshard}} \lor TR_{\text{drop_msg}} \lor \ldots, ... \]

System of linear Constrained Horn Clauses (CHC) over unknown predicates \( \varphi_1, \varphi_2 \)!

\[ \forall k. \]

\[ \varphi_1 \]

\[ \varphi_2 \]

\text{reshard}

\text{drop_msg}

\text{retransmit}

\text{recv\_transfer\_msg}
Phases Guide Inference

∀\(k\).

\[\text{reshard}(*, *, k, *)\]

\[\text{recv}(*, *, k, *)\]

\[\ldots(*, k)\]

\[\ldots(*, k)\]
Phases Guide Inference

- Semantic disjunctive template
Phases Guide Inference

• Semantic disjunctive template
• Phase decomposition and incremental construction
Phases Guide Inference

• Semantic disjunctive template
• Phase decomposition and incremental construction
• Impossible transitions
Implementation

• **mypyvy**: a tool inspired by Ivy, over Z3
  • Statically-typed Python

Invariant inference:

• Standard **PDR**\(^\forall\) for standard inductive invariants
  • Standard PDR adaptation **Phase-PDR**\(^\forall\) for phase invariants

## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Inductive Invariant [seconds]</th>
<th>Phase Structure [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock server (single lock)</td>
<td>2.21</td>
<td>0.67</td>
</tr>
<tr>
<td>Lock server (multiple locks)</td>
<td>2.73</td>
<td>1.06</td>
</tr>
<tr>
<td>Simple consensus</td>
<td>60.54</td>
<td>1355*</td>
</tr>
<tr>
<td>Ring leader election</td>
<td>152.44</td>
<td>2.53</td>
</tr>
<tr>
<td>Sharded KV (basic)</td>
<td>1.79</td>
<td>1.59</td>
</tr>
<tr>
<td>Sharded KV</td>
<td>2070*</td>
<td>372.5</td>
</tr>
<tr>
<td>MESI cache coherence</td>
<td>-</td>
<td>90.1</td>
</tr>
</tbody>
</table>

* not all runs terminated in 1 hour
- no runs terminated in 1 hour
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Inductive Invariant [seconds]</th>
<th>Phase Structure [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock server (single lock)</td>
<td>2.21</td>
<td>0.67</td>
</tr>
<tr>
<td>Lock server (multiple locks)</td>
<td>2.73</td>
<td>1.06</td>
</tr>
<tr>
<td>Simple consensus</td>
<td>60.54</td>
<td>1355*</td>
</tr>
<tr>
<td>Ring leader election</td>
<td>152.44</td>
<td>2.53</td>
</tr>
<tr>
<td>Sharded KV (basic)</td>
<td>1.79</td>
<td>1.59</td>
</tr>
<tr>
<td>Sharded KV</td>
<td>2070*</td>
<td>372.5</td>
</tr>
<tr>
<td>MESI cache coherence</td>
<td>-</td>
<td>90.1</td>
</tr>
</tbody>
</table>

* not all runs terminated in 1 hour
- no runs terminated in 1 hour
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Inductive Invariant [seconds]</th>
<th>Phase Structure [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock server (single lock)</td>
<td>2.21</td>
<td>0.67</td>
</tr>
<tr>
<td>Lock server (multiple locks)</td>
<td>2.73</td>
<td>1.06</td>
</tr>
<tr>
<td>Simple consensus</td>
<td>60.54</td>
<td>1355*</td>
</tr>
<tr>
<td>Ring leader election</td>
<td>152.44</td>
<td>2.53</td>
</tr>
<tr>
<td>Sharded KV (basic)</td>
<td>1.79</td>
<td>1.59</td>
</tr>
<tr>
<td>Sharded KV</td>
<td>2070*</td>
<td>372.5</td>
</tr>
<tr>
<td>MESI cache coherence</td>
<td>-</td>
<td>90.1</td>
</tr>
</tbody>
</table>

* not all runs terminated in 1 hour
- no runs terminated in 1 hour
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Inductive Invariant [seconds]</th>
<th>Phase Structure [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock server (single lock)</td>
<td>2.21</td>
<td>0.67</td>
</tr>
<tr>
<td>Lock server (multiple locks)</td>
<td>2.73</td>
<td>1.06</td>
</tr>
<tr>
<td>Simple consensus</td>
<td>60.54</td>
<td>1355*</td>
</tr>
<tr>
<td>Ring leader election</td>
<td>152.44</td>
<td>2.53</td>
</tr>
<tr>
<td>Sharded KV (basic)</td>
<td>1.79</td>
<td>1.59</td>
</tr>
<tr>
<td>Sharded KV</td>
<td>2070*</td>
<td>372.5</td>
</tr>
<tr>
<td>MESI cache coherence</td>
<td>-</td>
<td>90.1</td>
</tr>
</tbody>
</table>

* not all runs terminated in 1 hour
- no runs terminated in 1 hour
Summary

User-guided invariant inference by **phase structures**

- Convey high-level intuition
- Direct proof search effectively
  - Semantic disjunctive template
  - Incrementality between phases
  - Disabled transitions
- Facilitate inference beyond the state of the art
- Faster convergence

- **Sketching correctness** of infinite-state systems