Finding Invariants of Distributed Systems: It’s a Small (Enough) World After All

Travis Hance
Marijn Heule
Ruben Martins
Bryan Parno

Carnegie Mellon University
Distributed systems are hard to get right.
Solution: Verification?

• **Formal verification** can rule out bugs in:
  • Abstract protocol descriptions
  • Implementation
  • Liveness

• **Problem**: Verification is hard
  • Hand-crafted system invariants
  • Invariants must be checked for inductiveness
Solution: *Automated* Verification

- Less painstaking manual proof work
- **Problem:** Automated Verification is also hard, often undecidable
- Prior work
  - Automate invariant checking (IVy)
  - Automate invariant *finding*
    - I4 (Ma et al.) (SOSP, 2019)
    - Separators algorithm (Padon et al.) based on IC3 (PLDI, 2020)
- Many useful protocols (e.g., Paxos) still out of reach of fully-automated solutions
SWISS Contributions

• System to automatically prove safety for distributed protocols
• Scales to automate the verification of Paxos
• Handles universal & existential quantifiers
• Can accept additional user guidance—otherwise fully automated
• Produces partial invariants even when it doesn’t complete
Consensus protocol

Abstract protocol description

Safety property

Nodes agree on result

SWISS

Find invariants

Invariants & proof of safety condition

Partial invariants
SWISS Overview: Invariants

• An **invariant** is a statement about the system which holds true at every point in the execution

• We need an invariant which is ...
  • **Useful** – it can be used to prove the safety condition
  • **Inductive** – it is itself strong enough to prove that it remains true
An Example Invariant (Paxos)

\[
\begin{align*}
(\forall r, v_1, v_2. \text{proposal}(r, v_1) \land \text{proposal}(r, v_2) \implies v_1 = v_2) \\
(\forall n, r, v. \text{vote}(n, r, v) \implies \text{proposal}(r, v)) \\
(\forall r, v. (\exists n. \text{decision}(n, r, v)) \implies \exists q. \forall n. \text{member}(n, q) \implies \text{vote}(n, r, v)) \\
(\forall n, v. \neg \text{vote}(n, \text{none}, v)) \\
(\forall n, r, r_{\text{max}}, v. \text{onebmaxvote}(n, r, r_{\text{max}}, v) \implies \text{oneb}(n, r)) \\
(\forall n, r, r_1, r_2. \text{oneb}(n, r_2) \land r_2 > r_1 \implies \text{leftrnd}(n, r_1)) \\
(\forall n, r_1, r_2, v_1, v_2. \text{onebmaxvote}(n, r_2, \text{none}, v_1) \land r_2 > r_1 \implies \neg \text{vote}(n, r_1, v_2)) \\
(\forall n, r, r_{\text{max}}, v. \text{onebmaxvote}(n, r, r_{\text{max}}, v) \land r_{\text{max}} \neq \text{none} \implies r > r_{\text{max}} \land \text{vote}(n, r_{\text{max}}, v)) \\
(\forall n, r, r', r_{\text{max}}, v. \text{onebmaxvote}(n, r, r_{\text{max}}, v) \land r > r' \land r' > r_{\text{max}} \implies \neg \text{vote}(n, r', v')) \\
(\forall r_1, r_2, v_1, v_2. q. r_2 > r_1 \land \text{proposal}(r_2, v_2) \land v_1 \neq v_2 \implies \exists n. \text{member}(n, q) \land \text{leftrnd}(n, r_1) \land \neg \text{vote}(n, r_1, v_1))
\end{align*}
\]

PADON, O., LOSA, G., SAGIV, M., AND SHOHAM, S.

Paxos made EPR: Decidable reasoning about distributed protocols.

SWISS Overview

Abstract protocol description

Breadth
- Cast a “wide net”
- Find any invariant
- Many small invariants $I_1, I_2, I_3, \ldots, I_n$

Finisher
- Find invariant to complete proof
- One big invariant $I_{\text{last}}$

Invariants & proof of safety condition

SWISS

Safety property

Partial invariants
### SWISS Invariant Search

Exploring the space of candidate invariant predicates for Paxos

<table>
<thead>
<tr>
<th></th>
<th>Candidate invariant space</th>
<th>Number of candidate invariants</th>
<th>Symmetries</th>
<th>Counter-example filters</th>
<th>Removing redundant invariants</th>
<th>Invariant predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finisher</td>
<td>6 terms</td>
<td>~ 99,000,000,000,000</td>
<td>~ 200,000,000,000</td>
<td>155</td>
<td>155</td>
<td>5</td>
</tr>
<tr>
<td>Breadth</td>
<td>3 terms</td>
<td>~ 820,000,000</td>
<td>~ 3,000,000</td>
<td>~ 900,000</td>
<td>2,250</td>
<td>801</td>
</tr>
</tbody>
</table>

100 ms on average
Brute force is not feasible

**Counterexample-guided synthesis:**
When one predicate fails to be inductive, use it to narrow your search space.

Finisher—which is directed by the desired safety property—is more effective at filtering a large space of candidate invariants.
Evaluation

- Benchmark synthesis on 27 protocols, including 6 Paxos variants
  - SWISS solves 18 / 27 each within 6 hours
  - Includes Paxos and variant Flexible Paxos
  - Also solves Multi Paxos if given additional guidance on input search space
Evaluation

I4 (2019) is usually the fastest, but doesn’t handle existential quantifiers.

---

I4: incremental inference of inductive invariants for verification of distributed protocols.


MA, H., GOEL, A., JEANNIN, J., KAPRITSOS, M., KASIKCI, B., AND SAKALLAH, K. A.
Padon et al.’s Separators algorithm (2020) does not scale to Paxos, but is often faster on other benchmarks.

Still out of reach
Fast Paxos
Vertical Paxos
Stoppable Paxos

KOENIG, J. R., PADON, O., IMMERMAN, N., AND AIKEN, A.
First-order quantified separators.
Further Evaluation in Paper

• Analysis of the sizes of invariants we expect harder protocols to require
• Benchmarks of individual optimizations
  • Optimizations that didn’t help
• Parallelizability
• SMT bottlenecks
• Performance on restricted search spaces
Conclusion

• SWISS scales invariant synthesis to protocols not tackled previously
• SWISS has differing strengths relative to prior approaches—suggests there are still ideas that can be combined and improved

Finding Invariants of Distributed Systems: It’s a Small (Enough) World After All
Travis Hance
thance@andrew.cmu.edu