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

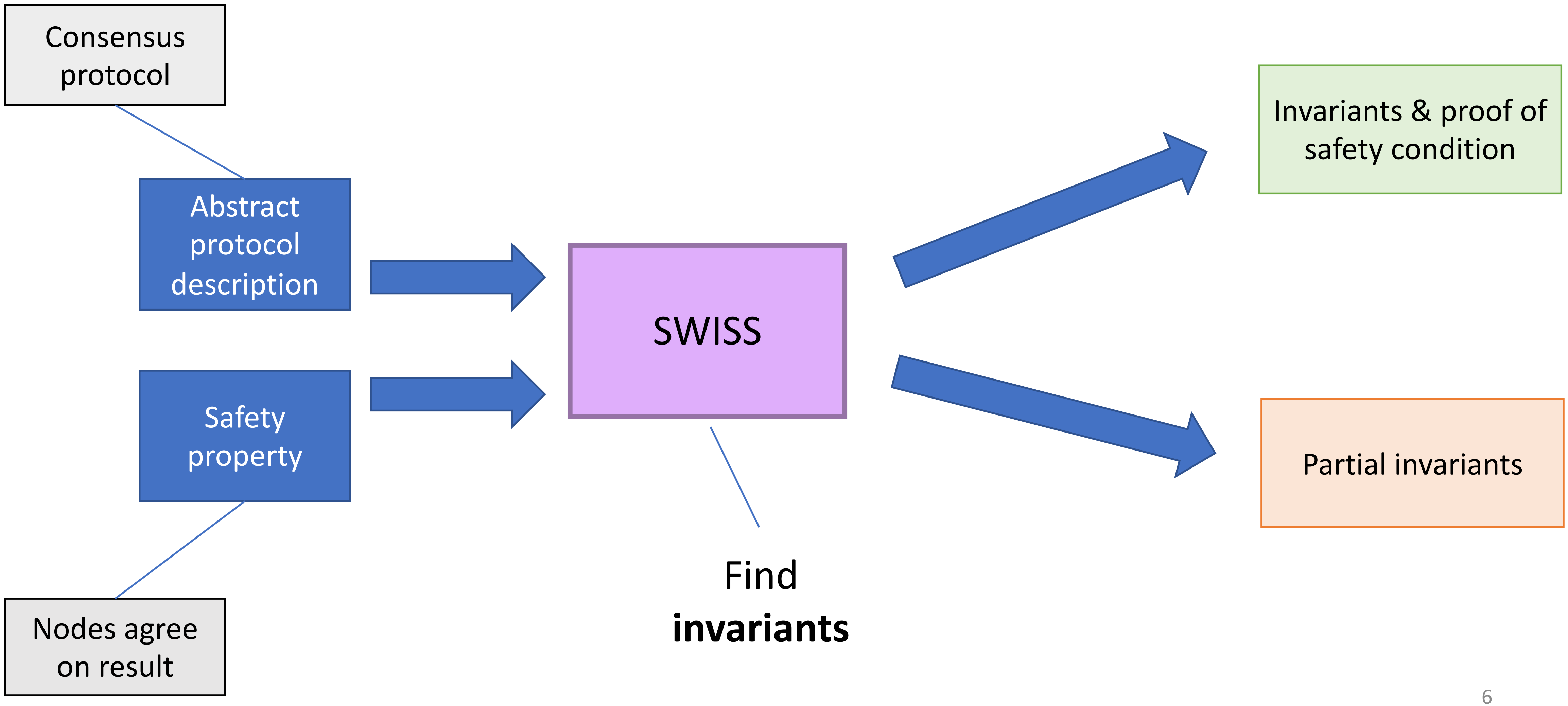
Solution: **Automated** Verification

- 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

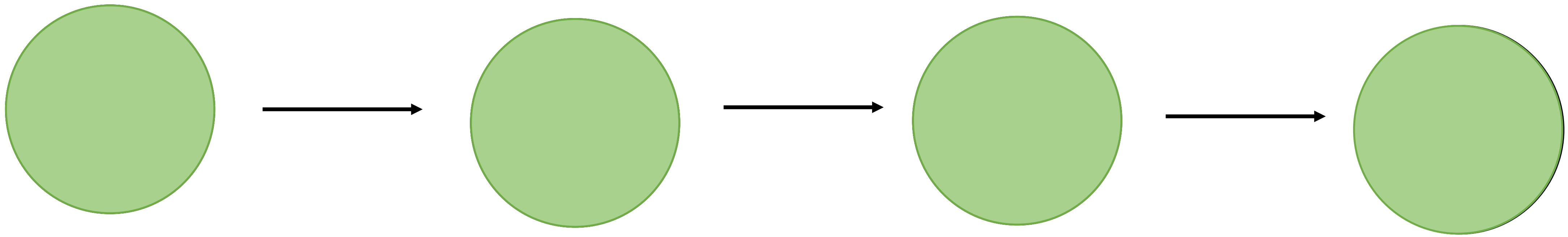
- System to automatically prove safety for distributed protocols
- Scales to automate the verification of Paxos
- Handles universal & existential quantifiers
- Can accept additional user guidance—otherwise fully automated
- Produces partial invariants even when it doesn't complete

SWISS Overview



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

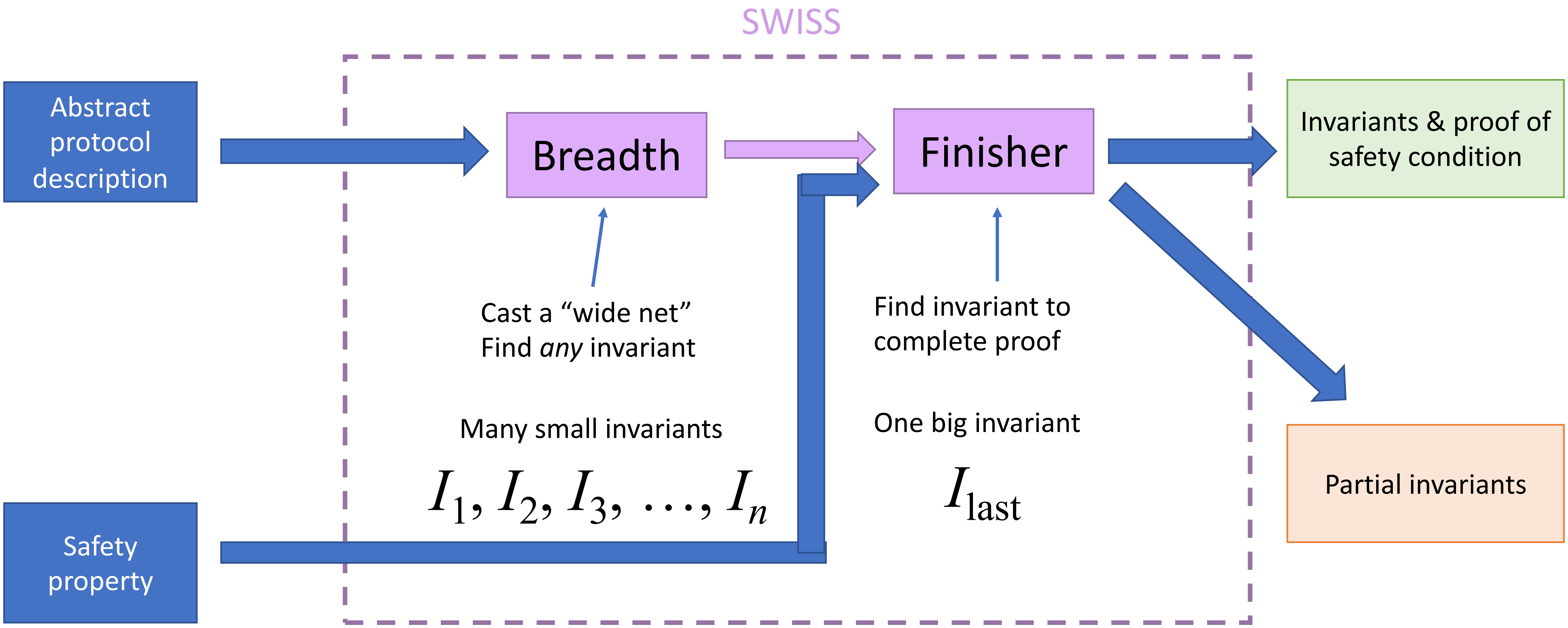


An Example Invariant (Paxos)

$$
\begin{aligned}
&\boxed{(\forall r, v_1, v_2.\ proposal(r, v_1) \wedge proposal(r, v_2) \implies v_1 = v_2)}\\
\wedge &\boxed{(\forall n, r, v.\ vote(n, r, v) \implies proposal(r, v))}\\
\wedge &\boxed{(\forall r, v.\ (\exists n.\ decision(n, r, v)) \implies \exists q.\ \forall n. member(n, q) \implies vote(n, r, v))}\\
\wedge &\boxed{(\forall n, v.\ \neg vote(n, none, v))}\\
\wedge &\boxed{(\forall n, r, r_{max}, v.\ onebmaxvote(n, r, r_{max}, v) \implies oneb(n, r))}\\
\wedge &\boxed{(\forall n, r, r_1, r_2.\ oneb(n, r_2) \wedge r_2 > r_1 \implies leftrnd(n, r_1))}\\
\wedge &\boxed{(\forall n, r_1, r_2, v_1, v_2.\ onebmaxvote(n, r_2, none, v_1) \wedge r_2 > r_1 \implies \neg vote(n, r_1, v_2))}\\
\wedge &\boxed{(\forall n, r, r_{max}, v.\ onebmaxvote(n, r, r_{max}, v) \wedge r_{max} \neq none \implies r > r_{max} \wedge vote(n, r_{max}, v))}\\
\wedge &\boxed{(\forall n, r, r', r_{max}, v, onebmaxvote(n, r, r_{max}, v) \wedge r > r' \wedge r' > r_{max} \implies \neg vote(n, r', v'))}\\
\wedge &\boxed{(\forall r_1, r_2, v_1, v_2, q. r_2 > r_1 \wedge proposal(r_2, v_2) \wedge v_1 \neq v_2 \implies \exists n. member(n, q) \wedge leftrnd(n, r_1) \wedge \neg vote(n, r_1, v_1))}
\end{aligned}
$$

PADON, O., LOSA, G., SAGIV, M., AND SHOHAM, S.
Paxos made EPR: Decidable reasoning about distributed protocols.
Proc. ACM Program. Lang. 1, OOPSLA (Oct. 2017).

SWISS Overview



SWISS Invariant Search

Exploring the space of candidate invariant predicates for Paxos

	Candidate invariant space	Number of candidate invariants	Symmetries	Counter-example filters	Removing redundant invariants	Invariant predicates
Finisher	6 terms	~ 99,000,000,000,000	~ 200,000,000,000	155	155	5
Breadth	3 terms	~ 820,000,000	~ 3,000,000	~ 900,000	2,250	801

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.

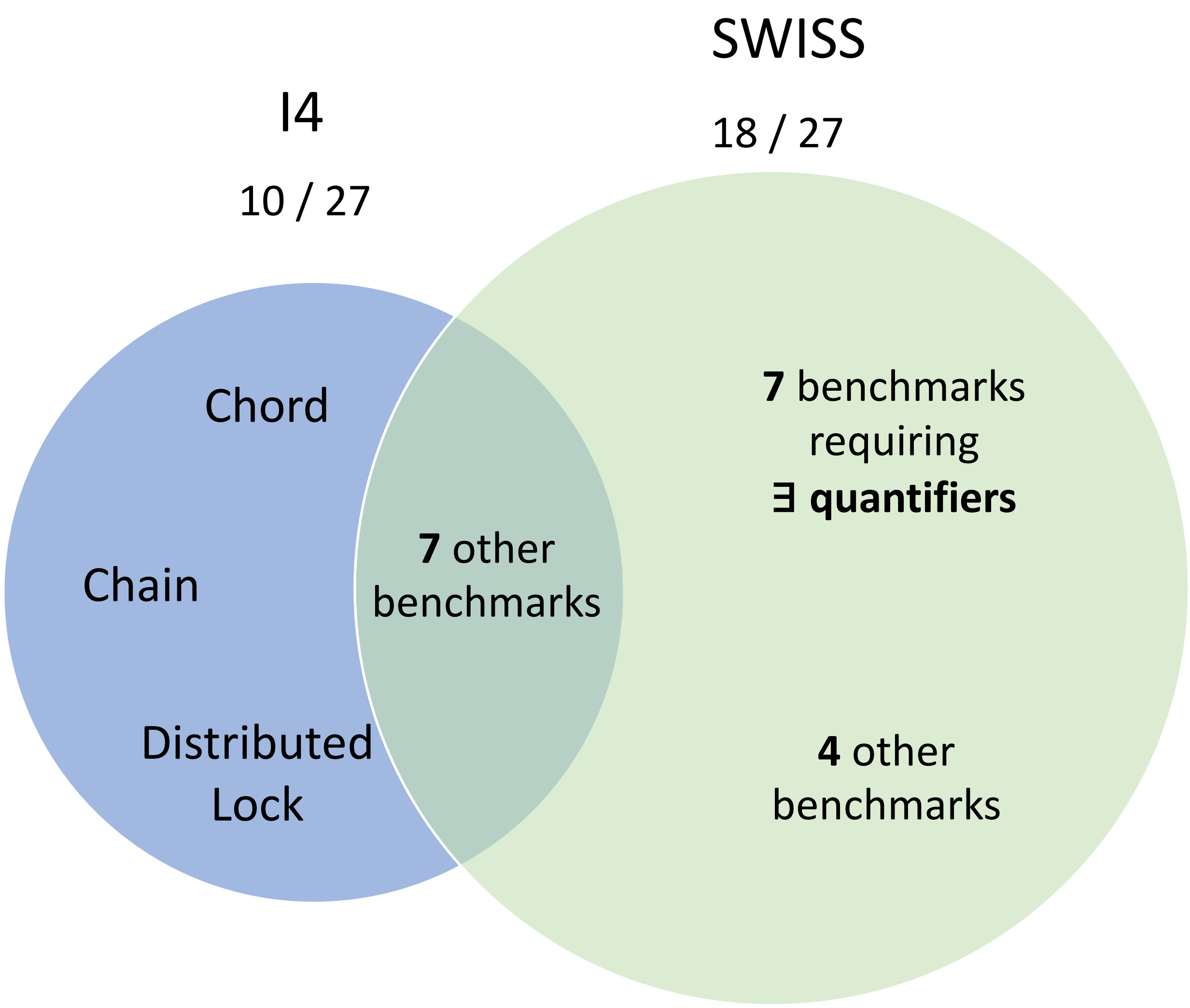
Finisher—which is directed by the desired safety property—is more effective at filtering a large space of candidate invariants.

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.

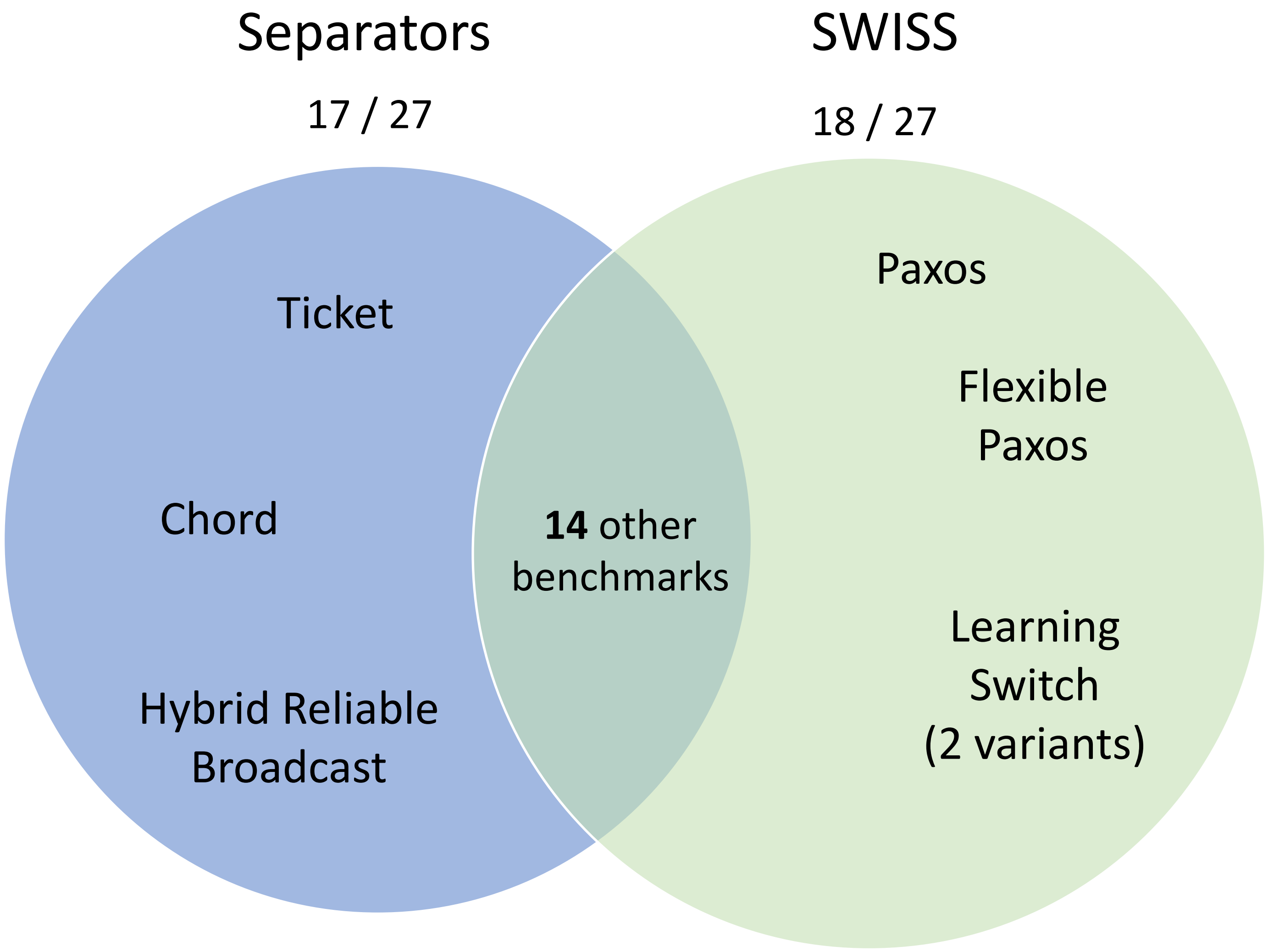


MA, H., GOEL, A., JEANNIN, J., KAPRITSOS, M., KASIKCI, B., AND SAKALLAH, K. A.
I4: incremental inference of inductive invariants for verification of distributed protocols.
In *Proceedings of the ACM Symposium on Operating Systems Principles, (SOSP)* (2019), T. Brecht and C. Williamson, Eds., ACM, pp. 370–384.

Evaluation

Padon et al.'s Separators algorithm (2020) does not scale to Paxos, but is often faster on other benchmarks.

<u>Still out of reach</u>
Fast Paxos
Vertical Paxos
Stoppable Paxos



KOENIG, J. R., PADON, O., IMMERMAN, N., AND AIKEN, A.
First-order quantified separators.
In *Proceedings of the ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)* (2020), A. F. Donaldson and E. Torlak, Eds., ACM, pp. 703–717.

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