Verification of distributed protocols and systems, where both the number of nodes in the systems and the state-space of each node are unbounded, is a long-standing research goal. In recent years, efforts around the Ivy verification tool [1]–[4] have pushed a strategy of modeling distributed protocols and systems in a new way that enables decidable deductive verification [5]–[8], i.e., given a candidate inductive invariant, it is possible to automatically check if it is inductive, and to produce a finite counterexample to induction in case it is not inductive. Complex protocols require quantifiers in both models and their invariants, including for all-exists quantifier alternations. Still, it is possible to obtain decidability by enforcing a stratification structure on quantifier alternations, often achieved using modular decomposition techniques, which are supported by Ivy. Stratified quantifiers lead not only to theoretical decidability, but to reliably good solver performance in practice, which is in contrast to the typical instability of SMT solvers over formulas with complex quantification.

Reliable automation of invariant checking and finite counterexamples open the path to automating invariant inference [9]. An invariant inference algorithm can propose a candidate invariant, automatically check it, and get a finite counterexample that can be used to inform the next candidate. For a complex protocol, this check would typically be performed thousands of times before an invariant is found, so reliable automation of invariant checking is a critical enabler. Recently, several invariant inference algorithms [9]–[18] have been developed that can find complex quantified invariants for challenging protocols, including Paxos and some of its most intricate variants.

In the tutorial I will provide an overview of Ivy’s principles and techniques for modeling distributed protocols in a decidable fragment of first-order logic. I will then survey several recently developed invariant inference algorithms for quantified invariants, and present one such algorithm in depth: Primal-Dual Houdini [13]. Primal-Dual Houdini is based on a new mathematical duality, and is obtained by deriving the formal dual of the well-known Houdini algorithm. As a result, Primal-Dual Houdini possesses an interesting formal symmetry between the search for proofs and for counterexamples.

REFERENCES


