Brief Announcement: Efficient Model Checking of Fault-tolerant Distributed Protocols Using Symmetry Reduction*

Péter Bokor, Marco Serafini, Neeraj Suri, and Helmut Veith

Technische Universität Darmstadt, Germany
{pbokor,marco,suri,veith}@cs.tu-darmstadt.de

Motivation Fault-tolerant (FT) distributed protocols represent fundamental building blocks behind many practical systems. A rigorous design of these protocols is desired given the complexity of manual proofs. The application of *model checking* (MC) [2] for protocol verification is attractive with its full automation and rich property language. However, being an exhaustive exploration method, its scalability is limited by the number of different system states. Although *FT distributed protocols* usually display a high degree of symmetry which stems from permuting different processes, MC efforts targeting their automated verification often disregard this symmetry. Therefore, we propose to leverage the framework of symmetry reduction [6] and improve on existing applications of it. Our secondary contribution is to define a high-level description language (called FTDP) to ease the symmetry-aware specification of FT distributed protocols.

Preliminary: Symmetry Reduction with Scalarsets Formally, *symmetry* is a permutation π acting on all reachable system states satisfying that for every state s and its successor s' it holds that $\pi(s')$ is a successor of $\pi(s)$ [6]. For example, a state s of a distributed system where two processes assume different local states is symmetric with another state $\pi(s)$ where these two local states are swapped. Symmetry reduction eases model checking by exploring a single (or some) representative states within each set of symmetric states. This method preserves CTL* temporal logic properties [2] if the property under verification does not distinguish symmetric states. Unfortunately, the detection of symmetries is in general prohibitively complex. Therefore, we take the approach of creating symmetric models *by construction*. In order to indicate symmetries in the model the designer uses a special data type called *scalarset* with a set of restricted operations [3]. The restrictions guarantee that any permutation of scalarset values results in symmetric states.

Role-based Symmetry Reduction FT distributed protocols can be often defined through r many *roles*, i.e., different types of independent processes having non-intersecting states whose state transitions are activated by non-intersecting

* Research supported in part by Microsoft Research, IBM Faculty Award

sets of incoming messages or internal events. Assume that n_i is the number of process *instances* in role i. In real implementations a protocol is executed by n physical nodes. Every node is a parallel composition of role instances with at most one role instance per node. We observe that it is unnecessary to explicitly model nodes if the properties of the protocol specify roles rather than nodes. Therefore, in our *role-based* approach we define each role as a new scalarset of size n_i. This enables us to permute role instances of the same role even if they are physically located on different nodes.

Role-based symmetries differ from node-based ones (commonly used for systems with replicated components) where a single scalarset of size n is used. The node-based approach only allows the permutation of entire nodes together with all hosted role instances. To compare the maximum achievable reduction assume that $n_i = n$ for all i. The reduction in the number of states of the state graph using role-based symmetries compared to the unreduced graph can be a factor of up to $(n!)^r$. This reduction is exponentially higher than the best case benefit of the node-based approach, which is up to $n!$.

The FTDP Language The FTDP language supports automatic verification of finite FT distributed protocols. First of all, FTDP allows the definition of roles and forces (using the scalarset approach) that symmetry is not violated. Furthermore, FTDP syntactically enriches low-level specification languages. FTDP differs from existing algorithm description languages such as +CAL [5] in providing built-in abstractions for (a) synchronous and asynchronous message-passing and (b) a variety of fault types. FTDP supports the specification of safety and liveness properties (specified in CTL*). In fact, FTDP specifications closely resemble the pseudocode of common distributed protocols.

Evaluation We have used our approach to analyze (debug and verify) real protocols, e.g., Paxos [4]. The Paxos protocol uses three roles: leader, acceptor, and learner. Our prototype FTDP implementation extends the SS language of the Murφ symmetry reduction model checker [3]. The experiments reveal that (1) the state graph obtained through role-based reduction contains up to twenty times less states than the unreduced model, (2) the benefit of this reduction approaches the theoretical maximum of $(n!)^r$, and (3) the node-based approach visits up to ten times more states than the role-based one. More details about the role-based approach and the FTDP language can be found in [1].

References

1. P. Bokor, M. Serafini, N. Suri, H. Veith. Practical Symmetry Reduction of FT Distributed Protocols. TR-TUD-DEEDS-04-04-2009, 2009.
2. E. Clarke, O. Grumberg, D. Peled. *Model Checking.* MIT Press, 2000.
3. C. N. Ip, D. L. Dill. Better Verif. Through Symmetry. *FMSD*, 9(1-2):41–75, 1996.
4. L. Lamport. The Part-Time Parliament. *ACM TOCS*, 16(2):133–169, 1998.
5. L. Lamport. Checking a Multithreaded Alg. with +CAL. *DISC*, p. 151–163, 2006.
6. A. Miller et al. Symmetry in Temporal Logic MC. *ACM C. Surv.*, 38(3):8, 2006.