Hiding Backtracking Operations in Software Model Checking from the Environment

Cyrille Artho, Yoshinori Tanabe, Etsuya Shibayama
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Watcharin Leungwattanakit, Masami Hagiya
University of Tokyo, Tokyo, Japan

Most non-trivial applications use some form of input/output (I/O), such as network communication. When model checking such an application, a simple state space exploration scheme is not applicable: Backtracking during the state space search causes states to be revisited, and I/O operations to be repeated. Because I/O operations are visible by the environment, software model checking needs to encapsulate such operations in a caching layer that hides such actions. In order to mediate between the model checker and the environment, the cache layer has to pair request and response messages correctly. It also has to distinguish between complete and partial messages. Finally, operations that open or close communication channels require special treatment as well. —

Software model checkers [3] cannot handle networked programs, which limits their applicability. Program transformations allow networked applications to be model checked on a single-process model checker [1]. However, the large number of thread interleavings limits scalability. A different approach consists of mediating between backtracking state space exploration of the model checker, and the linear time line of its environment [2]. This approach caches any operations that have an externally visible impact, in particular, network communication.

Previous work has assigned a mapping of each communication state to a previously seen communication trace. Each operation is mapped to a history of known operations, extending that history ("cache") if new states are explored. Whenever a mismatch between states is encountered after backtracking, network communication inside the program depends on its thread schedule. Such programs cannot be verified with our approach; they are also often faulty.

Our approach is applicable to any program where the result of a client request does not depend on actions of other client processes. This includes most Internet services, such as time servers, echo servers, FTP, and HTTP servers. Compared to other approaches [1], our caching approach is orders of magnitudes faster, because communication serialization inherently comprises an efficient partial-order reduction.

Recent work has shown that mapping states to traces is not sufficient. For more complex interactive protocols, requests also have to be mapped to their response. Our implementation achieves this, and also allows for requests and responses to span several messages. Furthermore, we also cache actions that manipulate communication channels themselves (opening or closing them). This hides backtracking of externally visible actions effectively from programs running outside the model checker, and makes model checking of programs that interact with their environment feasible in a scalable way.

References

[1] C. Artho and P. Garoche. Accurate centralization for applying model checking on networked applications. In *Proc. ASE 2006*, Tokyo, Japan, 2006.

[2] C. Artho, B. Zweimüller, A. Biere, E. Shibayama, and S. Honiden. Efficient model checking of applications with input/output. *Post-proceedings of Eurocast 2007*, 2007. To be published.

[3] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda. Model checking programs. *Automated Software Engineering Journal*, 10(2):203–232, 2003.