論文の内容の要旨

Distributed Software Model Checking Using I/O Cache and Process Checkpointing
(入出力キャッシュとチェックポインティングによる分散ソフトウェアのモデル検査)

氏名　レオンワタナキット ワチャリン

Distributed software has played a major role in recent decades. Its spread comes with the increased complexity in the development of such software. Concurrent execution and inter-process communication are common in distributed applications. Such programming elements are prone to errors hard to detect. This leads to difficulties in maintaining software quality. Model checking is a powerful technique to find faults in concurrent applications. However, most model checkers concentrate on stand-alone applications and lack the capability to handle distributed applications. This thesis introduces the concepts of I/O cache and process checkpointing. A series of approaches, based on these concepts, are implemented to support a model checker in dealing with distributed applications. All of these approaches concentrate on verifying a single process called system under test (SUT), treating other processes as peers. Verifying a single process has an advantage of scalability. The state space that must be explored is much more smaller than the composite state space of all processes in an application. As a result, the model checker can completely explore the single state space in reasonable time. The I/O cache provides an emulated I/O interface for the SUT. It allows communication between a SUT and peer since the computation results on the peer are stored in the cache, ready to be replayed. However, the I/O cache cannot perfectly imitate peers' non-deterministic behaviors, which render the cache contents imprecise. Accordingly, we apply the idea of process checkpointing and trade verification time for precision. Checkpointing tools allow multiple peer processes to be backtracked in sync with a SUT. They can also be used to suppress non-deterministic behaviors of peers. An implementation of each approach has also been done as an extension for Java PathFinder, a Java model checker. We succeed in verifying an extensive class of distributed applications, e.g. a multi-process HTTP/SSH server, client and a download manager program. The proposed approaches are also compared to one another in term of important properties in software verification, e.g. soundness and completeness. The classes of SUT/peer applications supported by each approach are discussed as well.