State space coverage and deterministically testing concurrent software

Piet Cordemans\textsuperscript{1,2}, Jeroen Boydens\textsuperscript{1,2}, and Eric Steegmans\textsuperscript{1}

\textsuperscript{1}iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium.
\{firstname.lastname@cs.kuleuven.be\}

\textsuperscript{2}ReMI-EP, KULab, 8400 Oostende, Belgium.

Abstract

Due to state space explosion [3] of concurrent software, it is hard to determine state space coverage of a particular test. Moreover, non-deterministic execution violates the premise of idempotent behavior across identical test runs. In this respect, the Actor model reduces the size of the state space and prevents data races. However, it does not inhibit race conditions, which might lead to non-deterministic test behavior through multiple test runs. In the context of a single test run, these race conditions relate to states, which might not be reached. In order to deal with this coverage problem, we developed a tool [1] to run tests on concurrent software, intercepting messages sent between Actors. Consequently it deterministically resends each message, which allows to explore a single path in the state space. By reiterating the test and adapting the order in which the messages are sent, all states in the state space of the test are covered. Paths in the state space are determined by a Colored Petri Net (CPN) [2] model of the Actors in a test. This CPN model permits deriving the state space of the test, in which the minimal set of paths to cover the whole state space are determined. As a proof of concept, eight cases have been implemented and tested. The main advantages include the guarantee of state space coverage as it is not dependent on heuristics, while it is not subject to false-positive test results. Furthermore, it is based on the Actor model and isolated tests, which limit state space explosion. However, this limits the scope of the tool to Actor-based software. Expanding its applicability to other models of concurrency is future work.

Keywords – concurrent software, testing, Actor model

References

