Designing Correct Runtime-Monitors for Erlang

Aldrin Seychell and Adrian Francalanza

Faculty of ICT, University of Malta, Msida Malta,
asey0001@um.edu.mt, adrian.francalanza@um.edu.mt

In runtime verification, a monitor continuously checks the execution of a program that is running concurrently with it. Normally, the runtime monitor checks that the system does not violate a correctness property. Any runtime monitor is expected to satisfy the following:

If a system does not obey a property ϕ, then the monitor for ϕ MUST flag a failure.

Showing that this statement holds is not trivial since we have to consider all possible execution paths of the monitor. In our work, we show how one can prove this indirectly by proving a number of correctness criteria on our synthesised monitors.

For instance, a desirable property for monitors is that the execution of a monitor does not result in divergence since this would result in infinite computation by the monitor. Having divergence in the monitor results in an unacceptable computational overhead that would hinder the performance of the monitored system. If the computational overhead incurred on the system is unacceptably high, the monitor cannot even be considered as a valid monitor since it will never be used.

Another desirable property that monitors should observe is that there exists at least one execution path that when the program does not satisfy a property, the monitor will give a *fail* notification. This is generally enough if the monitors are executed in a sequential manner.

The monitors might have to execute using concurrent processes for example to check multiple branches of a property at the same time. Thus, in a concurrent environment it is not enough that there exists a single path that gives the expected result. Having a proof of determinism would extend the latter property to be valid for all execution paths. Determinism also means that if a specific execution trace is monitored multiple times, the monitor will always give the same result.

In this talk, a design of monitors in Erlang for Erlang programs is discussed. A proof that this design of monitors satisfies the theorem stated above is also given. The design uses concurrent processes in order to monitor multiple sub-properties combined under conjunction. These synthesised monitors are shown to be non-divergent, are correct and are determinate. In order to show that the monitors preserves determinism, the monitors are proved to be confluent since confluence implies determinism.