Unified Approach to Static & Runtime Verification

Extended Abstract

Olga Dedi, Andreas Werner, Robert Kaiser, Reinhold Kroeger [1]

Abstract: Today's systems become more and more complex, and even domain experts are sometimes in doubt regarding their correct behaviour in rare / non-standard situations. Especially embedded systems incorporate increasing functionality and have to interact with the environment through a wide spectrum of sensors and actors. Real-time properties requiring a guaranteed reaction of the system within a given limited time window are often associated as well, and safety is taken for granted by customers all-the-time when using them. Furthermore, these critical systems often have to face uncertainty, which may originate from unknown device configurations at design time or from unforeseen changes of the environment during operation. Uncertainty may also exist in control algorithms. For example, to guarantee a safe behaviour of trained AI algorithms in previously unseen situations is inherently difficult, if not impossible.

Under these conditions it is a complex and highly responsible task for developers to thoroughly test, or even to formally prove correct behaviour of the system's properties. Formal verification is often regarded as the ultimate way to achieve the highest levels of trust, therefore safety certification standards demand such proof for the highest Safety Integrity Levels. However, due to the ever-increasing complexity of current software and due to the non-deterministic nature of some mechanisms of the underlying hardware architecture, static verification, though sound in theory, is often impractical.

We do not believe that a full static verification and validation at design time is always possible. Instead, we have started to work on a methodology which distinguishes between verification activities carried out at design time and those at runtime. In a nutshell, at design time static verification takes place, i.e. specified system properties are formally proven to the highest possible degree at a reasonable effort. For properties which cannot be proven statically in this way, sufficiently strong monitors are generated which are executed at runtime to monitor correct system behaviour. Interestingly, this methodology can be applied to functional as well as non-functional (i.e. timing) properties. In the undesired case of detecting a property violation at runtime, the underlying system architecture is prepared to reconfigure the application to ensure acceptable behaviour. This adaptivity has to be supported by the application design.

In this paper, we report on an ongoing effort for tool-supported verification of functional and non-functional properties by combining static and runtime verification techniques.

Keywords: SPARK; WCET; static verification; runtime verification; OS microkernel; AQUAS

[1] RheinMain University of Applied Sciences; firstname.lastname@hs-rm.de