A Formal Modeling Framework for Dependable and Portable Embedded Operating Systems

Renata Martins Gomes and Marcel Baunach, TU Graz

Current approaches for portability of real-time operating systems (RTOSs) for embedded systems are largely based on manual coding, which is arduous and error prone. With increasing dependability requirements for cyber physical systems, specially within the Internet of Things (IoT), along with the expected great diversity of hardware platforms, software platforms will only remain competitive in the long run if they guarantee correct operation and easy deployment to every hardware platform. In this scenario, a new approach to the development and portability of RTOSs that guarantees correct implementations for all current and future devices and hardware architectures becomes indispensable.

We present a framework for automatic RTOS portability that integrates model-based design and formal methods into dependable embedded software development. First, we model the RTOS considering an abstract hardware. The model includes the interaction with the hardware, such as interrupt handling and context switches, but remains generic until the RTOS is completely modeled. Then, we refine the model into a hardware-concrete model for each target architecture. The hardware-concrete models only instantiate some hardware details and specify actions left under-specified in the model, and do not require much effort to be complete. From the hardware-concrete models, we can prove the RTOS's design, generating proofs that the model fulfills its functional and non-functional requirements.

The generator we are developing serves as a bridge between the formal language and programing languages, using the formal software models to generate target-specific code. The automatic code generation guarantees that the model is correctly translated to machine language, avoiding implementation mistakes common to manual coding. Changes on the software, for bug fixes or testing of new concepts, for example, do not require knowledge of the target architectures, since they are done on the model and are immediately reflected in all implementations upon code generation, assuring consistency across platforms.