Correctness-proven code generation for MDA

András Balogh

Model Driven Architecture (MDA) is an emerging paradigm in software development, providing a framework for implementation platform independent modelling (PIM) of the target system, and automatic code generation by mapping it to a platform specific model (PSM). The basic idea is to separate the functional aspects of the software from the implementation specific ones. The PIM contains only the previous ones, and is created by the developers. Based on this model and the specification of the target platforms, the PSM is generated automatically using some model transformation methods. From this model, the majority of the source code can be automatically generated.

The correctness of the source code can be checked neither manually, nor with the widely used model-checking systems in large-scale systems. However, the PIM is a relatively small, easier-to-check one, and can be validated against the system requirements. If the source model can be treated as a correct one, the correctness of the final system depends on the correctness of the model transformation and code generation algorithms used in the MDA process. This way the correctness of the final implementation can be proven at the basic technology level.

Abstract State Machines (ASMs) provide a simple way to formally specify and hierarchically refine the behaviour of various dynamic systems. ASMs are widely used in telecommunication, programming language and hardware system design, both in academic and industrial environments.

A method for ASM based correctness proving of transformations is introduced in this paper. Model transformation algorithms (both model-to-model and model-to-code transformations) can be treated as mappings between different levels of abstraction of modelling. Therefore, if the input patterns and mapping results of the transformation are described as ASMs, and a refinement path can be found between them, the correctness of the specific transformation is proved. The basic idea of the proof is that in a rule based transformation, as used in our tool, the large problem of proving the correctness of the transformation can be split up into a set of smaller sub-problems proving the correctness of the individual rules.