Model Checking of Visual Modeling Languages

Ákos Schmidt

In the past few years the Model Driven Architecture (MDA) has become a leading directive in the field of software engineering. According to the main concept of MDA, at the first phase during the design of the software system, a platform independent abstract model (PIM) is produced in a visual modeling language (mostly UML). The concrete platform specific models (defined by different visual modeling languages of UML dialects, for example for .NET, CORBA, or J2EE) can be derived from this abstract platform independent model by automatic model transformations. Finally, automatic code generators produce the majority of the final source code of the implementation.

Nowadays the wide use of visual modeling languages (such as UML) in software engineering has caused the quick spread of metamodeling and graph transformation techniques, as being an expressive and visual, but mathematically precise specification technique. Despite the mathematical accuracy it cannot guarantee that the components of the system model (the concrete model instances of the modeling languages) are free of design or modeling faults, which (without detection and correction) might deteriorate the safety or reliability of the system. The later a fault is detected during the design period, the more and more its correction will cost.

Typically, a wide range of model checkers (like, for instance, SMV, SAL, Murphi, or SPIN) are used in software engineering applications to detect such faults in the modeling phase automatically (where system properties are checked without human interaction). As their input specification language is a low-level and textual description instead of visual modeling languages widely used by engineers, several transformation has been developed to derive model checker input specifications from behavioral UML models automatically.

I present a method (with tool support of CheckVML [1]) for model checking arbitrary visual models defined by metamodeling and graph transformation techniques. First, a model checker independent mathematical representation (a transition system) is derived from our initial model, which is a common mathematical formalism that serves as the input specification of various model checker tools. For the second step of the transformation the tool generates a Promela description (into a file) from the transition system which can serves as the input for the SPIN model checker.

The model checking process for models of visual modeling languages consists of two steps: first, the model checker (SPIN in our case) input specification is generated by our tool (CheckVML). Finally, SPIN can verify different system properties (like safety, liveness, or deadlock freeedom), which can be expressed as LTL (Linear Temporal Logic) formulas.

I demonstrate the feasibility of the approach and transformation tool CheckVML on a well-known verification benchmark; namely transforming the model of dining philosophers into a SPIN specification, and verifying safety properties on the generated Promela code. The result of the runtime assessments [2] shows that the verification of a simple property by SPIN takes much longer than the transformation of the model from the visual description using CheckVML.

References

[1] Á. Schmidt and D. Varró. CheckVML: A tool for model checking visual modeling languages. In P. Stevens, J. Whittle, and G. Booch, editors, Proc. UML 2003: 6th International Conference on the Unified Modeling Languages, volume 2863 of LNCS, pages 92-95, San Francisco, CA, USA, October 20-24 2003. Springer

[2] Sz. Gyapay, Á. Schmidt, and D. Varró. Joint Optimization and Reachability Analysis in Graph Transformation Systems with Time. In Proc. GT-VMT 2004 International Workshop on Graph Transformation and Visual Modeling Techniques. In press. Barcelona, Spain, March 27-28 2004.