Automatic Generation of Compiled Model Transformations[1]

Ákos Horváth

Nowadays, Model Driven Architecture (MDA [1]) is an emerging paradigm in software development. Based on high-level modelling standards (e.g. UML), MDA separates business and application logic from underlying platform technology by using platform independent models (PIM) to capture the core functionality of the target system, and platform specific models (PSM) to specify the target system on the implementation platforms (Java, C#, C++). PSMs and platform-specific source codes are automatically generated from PIM and PSMs, respectively, by using model transformation (MT) methods, thus, the role of MT is unquestionable for the success of the overall process of MDA.

Models in MDA have a graph structure, so model transformations can be specified by graph transformation (GT), which is a declarative and rule based specification paradigm. The execution of a GT rule performs local manipulation on graph models by finding a matching of the pattern prescribed by its left-hand side graph in the model, and changing it according to the right-hand side graph. In order to be able to handle complex model transformations, control structures embedding GT rule applications are also needed.

As the complexity of model transformations is growing, a new demand has been arisen to separate the design from transformation execution by using high-level models at design time and by automatically deriving source code for the target platform from these high-level models. While high-level models ease the development, testing, debugging and validation of model transformations within a single transformation framework without relying on a highly optimized target transformation technology, compiled standalone versions of a model transformation in an underlying platform (e.g. Java, C#) are more efficient from runtime performance aspects.

Code generators that are used in the process of standalone version generation, are typically implemented in a standard programming language for specific model transformations, thus, it is difficult to reuse existing source code generators to different platforms with conceptual similarities (e.g. OO based programming languages) or to modify the output source code generation in order to integrate it to other MT tools.

The current paper presents a new approach using high order (generic and meta [2]) graph transformation rules for the source code generation from high-level model transformation specifications defined by a combination of graph transformations and abstract state machine (ASM) constructs (as used within the VIATRA2 framework [3], which is a general Eclipse based modelling framework tool having been developed at the TUB-DMIS). The essence of the approach is to store model transformation rules as ordinary models in the model space. This way the source code generator of the transformations can be handled within the modelling framework. As a result, the code generator can be reused by replacing only the output generation rules in order to port the transformations to new underlying platforms, and the correctness of the code generators can be validated by a huge range of graph algorithms (e.g mutant graphs).

References

[1] Model Driven Architecture — A Technical Perspective, *Object Management Group* (`http://www.omg.org`), September 2001.

[2] D. Varró and A. Pataricza. Generic and meta-transformations for model transformation engineering, in: *T. Baar, A. Strohmeier, A. Moreira and S. Mellor, editors*, *Proc. UML 2004: 7th International Conference on the Unified Modeling Language*, LNCS 3273 (2004), pp. 290–304.

[1]This work was partially supported by the SENSORIA European project (IST-3-016004).

[3] VIATRA2 Framework,
An Eclipse GMT Subproject (`http://www.eclipse.org/gmt/`).