Towards an Alignment of Declarative Modeling and Model-to-Model Transformation Languages

Andreas Petter, Alexander Behring,

{a_petter,behring}@tk.informatik.tu-darmstadt.de

Declarative modeling and model driven software engineering seem to be two fields of research with completely different focus. However, the term “modeling” is used by both communities and both communities claim to use “declarative” techniques. Model-to-model transformations transform software engineering models. Similar to declarative modeling techniques they may be described using declarative model-to-model transformation languages. When it comes to finding similarities between both communities, these languages therefore classify as interesting candidates.

Our model-to-model transformation language, called “Solverational” can be used to define constraint problems in model-to-model transformations. Thereby, Solverational shows that model-to-model transformations are an interesting case study for constraint solving problems. QVT Relations, a declarative model-to-model transformation language, is using equations to set attribute values. Our languages enhances this concept by allowing to replace the equalities by inequalities. To narrow down the domain of the attributes, several inequalities may be used for a single attribute. Of course, this induces a constraint satisfaction problem.

The transformation engine is based on an Eclipse plugin (JAVA IDE) mapping Solverational to a constraint logic programming language (and system), ECLiPSe (not the IDE). Though this concept sounds simple, the implementation needs to be able to span constraints over associations and transformation rules. This is done by introducing “semi-delayed-goals”, which are delayed until all model elements have been generated.

As an example we transform an abstract user interface model into a concrete graphical user interface model using Solverational. A simple transformation rule maps Containers from the abstract model to Panels in the concrete model. The size and position of the Components residing in the Panel is determined during the transformation, but adheres to constraints we provide using Solverational. The constraints intuitively define that the Components may not be bigger as the Panel. The result is a complex CSP which can be solved by the constraint solver used in the transformation engine.

Therefore, we contribute to the alignment of declarative modelling and model-to-model transformation languages by examining the differences of models used for model-to-model transformation and declarative models, by presenting model-to-model transformation as a use case for declarative modeling, and by mapping Solverational to a CSP as an example for a larger set of graph grammars.