Ákos Schmidt

Consistency Analysis of Modeling Languages

Nowadays, the Model Driven Architecture(MDA) has extensively spread in the design process of IT systems. According to MDA, in the first phase of the design a general platform independent model of the target system is created, from which concrete platform specific models will be generated automatically by model transformations. This platform specific representation forms the basis of sophisticated code generators that yield a product quality code of the target application. The objective of MDA is to save time and cost by revealing ~~the~~ faults in the system in an early phase of the design preferably in the platform independent model.

The UML - the dominating OO modeling language – does not guarantee the correctness of a user model. Despite its success in various application fields, UML has some major weaknesses: diagrams can be ambiguous, moreover, UML does not ensure the syntactic and semantic consistency between the different views of the model. The upcoming UML 2.0 will (probably) be able to check the syntactic consistency, but proving semantic consistency between diagrams is out of the scope. The aim of my project is to prove semantic consistency in any well-defined modeling languages of software engineering (like UML).

In the current semester, I have been dealing with the correctness (consistency analysis) of models taken from a single modeling languages. Given (i) a concrete model with any kind of modeling language as the specification, and (ii) a yes/no question as a requirement, we answer whether the model satisfies its requirements by model checking.

The concrete model, the metamodel and the semantical rules are the inputs of our system. They are represented with directed attributed graphs and graph-transformation rules. The main problem is to convert our system to a semantically equivalent transition system(TS) since leading model checker tools use as input a description of a transition system. Finally from the TS we have to generate the description in the actual language which can be processed by the model checker.

The main advantage of the program is the ability to check *any* model which can be transformed into directed graphs. So it will be able to check the consistency between the different UML views if they are represented in a common abstract graph structure.

So far, I have developed the target metamodels and currently, now I am working on the general construction of the transition system. In the future I am aiming to deal with the generation of the input language of a concrete model checker, and the semantic integration analysis of UML diagrams. Finally the capacity of the program will be evaulated by benchmark applications.