Using Concrete Syntax in Graph-based Model Transformations

Grønmo, Roy

Doctoral thesis

View/Open
Groenmo-avhandling-publ.pdf (1.175Mb)

Year
2010

Permanent link
http://urn.nb.no/URN:NBN:no-24448

Abstract
The emergence of large and complex software systems increases the interest in model-driven engineering, as a way to lower the cost of development and maintenance of software. Models allow us to hide irrelevant details, provide different model viewpoints, and isolate and modularize models of cross-cutting concerns of a system. The emerging technologies for aspect-oriented modeling and weaving provide a systematic way to handle cross-cutting concerns at the modeling level. The success of model-driven engineering relies heavily on model transformations. This thesis describes how aspect-oriented modeling and many typical model transformations can be defined as aspects/rules that: (1) use the concrete syntax of the involved modeling languages, and (2) use graph transformation principles as its foundation. The thesis presents two main results. The first main result is an aspect language for UML 2 sequence diagrams. The language takes advantage of a formal model for sequence diagrams, which makes the matching and weaving process semantics-based. For this language we provide a confluence theory. The second main result is an approach to define many typical model transformations as graph transformations, where the transformation designer uses the concrete syntax of the involved modeling languages. Some typical model transformation examples are illustrated in this thesis, i.e. transformation from feature models to BPMN, UML activity diagram aspects, UML activity diagram refactoring, UML state machine refactoring, transformation from sequence diagrams to state machines, and transformations involving Petri nets. A collection operator has been introduced as a means to match and transform collections of similar subgraphs in graph transformations, using either concrete or abstract syntax. This allows for improved usability in transformations that would otherwise be complex or impractical to specify.

List of papers


Towards a Language for Graph-Based Model Transformation Design Patterns. Now, first I designed a language to express model transformation design patterns. I could have used an existing MTL as a notation, however this can lead developers to think the language itself is a transformation language and executable. DelTa metamodel. Just give concepts and give in details in concrete syntax. DelTa textual syntax. A textual development environment for DelTa in Eclipse using Xtext. Towards a Language for Graph-Based Model Transformation Design Patterns. Describe the textual syntax, sections etc. DelTa graphical syntax. Towards a Language for Graph-Based Model Transformation Design Patterns.