Chapter 15
Formal Semantics for Metamodel-Based Domain Specific Languages

Paolo Arcaini
Università degli Studi di Milano, Italy

Elvinia Riccobene
Università degli Studi di Milano, Italy

Angelo Gargantini
Università di Bergamo, Italy

Patrizia Scandurra
Università di Bergamo, Italy

ABSTRACT

Domain Specific Languages (DSLs) are often defined in terms of metamodels capturing the abstract syntax of the language. For a complete definition of a DSL, both syntactic and semantic aspects of the language have to be specified. Metamodeling environments support syntactic definition issues, but they do not provide any help in defining the semantics of metamodels, which is usually given in natural language. In this chapter, the authors present an approach to formally define the semantics of metamodel-based languages. It is based on a translational technique that hooks to the language metamodel its precise and executable semantics expressed in terms of the Abstract State Machine formal method. The chapter also shows how different techniques can be used for formal analysis of models (i.e., instance of the language metamodel). The authors exemplify the use of their approach on a language for Petri nets.

INTRODUCTION

In the context of language and software development, *modeling* is beginning to take a more prominent role. Model-based approaches consider models as first-class entities that need to be maintained, analyzed, simulated and otherwise exercised, and mapped into programs and/or other models by automatic model transformations.

Domain Specific Languages (DSLs) themselves can be seen as artifacts of the model-based approach for language engineering. Indeed, in a model-based language definition, the abstract syntax of a language is defined in terms of an object-oriented model, called *metamodel*, that characterizes syntax elements and their relationships, so separating the abstract syntax and semantics of the language constructs from their

DOI: 10.4018/978-1-4666-6042-7.ch015

different concrete notations. Although a complete definition of a DSL requires both syntactic and semantic aspects of the language to be precisely specified, metamodeling environments (Eclipse/Ecore, GME/MetaGME, AMMA/KM3, XMF-Mosaic/Xcore, etc.) cope with most syntactic and transformation definition issues, but no standard and rigorous way exists for defining language semantics that is usually given in natural language. A rigorous approach to specify the semantics of metamodels is currently an open and crucial issue in the model-driven context.

In general, metamodel semantics can be given with different degrees of formality by a mapping to a sufficiently well-known domain or target platform (like the JVM). However, incomplete and informal specifications of a language make precise understanding of its syntax and semantics difficult. Moreover, the lack of formally specified language semantics can cause a semantic mismatch between design models and tools supporting the analysis of models of the language (Chen, Sztipanovits, & Neema, 2005). We believe these shortcomings can be avoided by integrating metamodeling techniques with formal methods providing the requested rigor and preciseness. Applying a formal method to a DSL defined in a metamodeling framework should have two main goals: (a) allowing the definition of the semantics of models conforming to the DSL and (b) providing several techniques and methods for the formal analysis (e.g., validation, property proving, model checking, etc.) of such models. Indeed, a semantics is essential to communicate the meaning of models or programs to stakeholders in the development process, and a formal definition of the semantics of a DSL is a key prerequisite for the verification of the correctness of models specified using such a DSL.

In (Gargantini, Riccobene, & Scandurra, 2009; Gargantini, Riccobene, & Scandurra, 2010), the feasibility and the advantages of integrating metamodeling techniques and formal methods in the context of the ASM (Abstract State Machine) formalism(Börger & Stärk, 2003) is analyzed. (Gargantini, Riccobene, & Scandurra, 2009) proposes different techniques to endow metamodel-based languages with precise and executable semantics. These techniques imply a different level of automation, user freedom, possible reuse, and user effort in defining semantics, but they all share a common unifying formal ASM framework.

In this chapter, we present the application of one of these techniques to express the semantics (possibly executable) of metamodel-based DSLs. We present the semantic *hooking* approach that allows designers to hook to the language metamodel an ASM, which contains all data structures modeling elements of the metamodel with their relationships, and all transition rules representing behavioral aspects of the language. Model-to-text transformations are used to map metamodel elements into corresponding ASM constructs. We exemplify the application of our approach on a language for Petri nets. This work extends (Gargantini, Riccobene, & Scandurra, 2009) by showing how different techniques, like simulation, scenario-based validation, model review, and model property checking, can be performed for formal analysis of DSL models.

The chapter is organized as follows:

- *Background* provides an overview on the approaches existing in the literature for defining semantics of DSLs. It briefly introduces the Abstract State Machine formal method and the set of tools for model analysis the method supports. Furthermore, a very concise description of the two main formal analysis activities, *model validation* and *model verification,* is given.
- *DSL Case Study: Petri Nets* presents the abstract syntax of a language for Petri nets used, throughout the chapter, as case study to exemplify the approach of semantic definition and model analysis.