Semantic Anchoring with Model Transformations*

Kai Chen, Janos Sztipanovits, Sherif Abdelwalhed, and Ethan Jackson

Institute for Software Integrated Systems, Vanderbilt University,
P.O. Box 1829 Sta. B., Nashville, TN 37235, USA
{kai.chen,janos.sztipanovits,sherif.abdelwalhed,ethan.jackson}
@vanderbilt.edu

Abstract. Model-Integrated Computing (MIC) is an approach to Model-Driven Architecture (MDA), which has been developed primarily for embedded systems. MIC places strong emphasis on the use of domain-specific modeling languages (DSML-s) and model transformations. A metamodeling process facilitated by the Generic Modeling Environment (GME) tool suite enables the rapid and inexpensive development of DSML-s. However, the specification of semantics for DSML-s is still a hard problem. In order to simplify the DSML semantics, this paper discusses semantic anchoring, which is based on the transformational specification of semantics. Using a mathematical model, Abstract State Machine (ASM), as a common semantic framework, we have developed formal operational semantics for a set of basic models of computations, called semantic units. Semantic anchoring of DSML-s means the specification of model transformations between DSML-s (or aspects of complex DSML-s) and selected semantic units. The paper describes the semantic anchoring process using the meta-programmable MIC tool suite.

1 Introduction

The Model-Driven Architecture (MDA) advocates a model-based approach for software development. Model-Integrated Computing (MIC) [24,27] is a domain-specific approach to MDA, which has been developed primarily for embedded systems. The MIC approach eases the complicated task of embedded system design by equipping developers with domain-specific modeling languages [25] tailored to the particular constraints and assumptions of their various application domains. A well-made DSML captures the concepts, relationships, integrity constraints, and semantics of the application domain and allows users to program imperatively and declaratively through model construction.

While a metamodeling process enables the rapid and inexpensive development of DSML syntax, the semantics specification for DSML-s remains a challenge problem. Transformational specification of semantics [9], gives us a chance

* This research was supported by the NSF Grant CCR-0225610 "Foundations of Hybrid and Embedded Software System".

to simplify the DSML semantics design. This paper exploits the transformational semantics specification approach for creating a semantic anchoring infrastructure [22]. This infrastructure incorporates a set of metaprogrammable MIC tools, including: the Generic Model Environment (GME) [4] for metamodeling, the Graph Rewriting and Transformation (GReAT) [2] tool for model transformation, the Abstract State Machines (ASM) [12, 18], as a common semantic framework to define the semantic domain of DSML-s, and AsmL [1] – a high-level executable specification language based on the concepts of ASM for semantics specification.

The organization of this paper proceeds as follows: Section 2 describes the background for DSML specifications. Semantic anchoring is summarized in Section 3. In Section 4, we use a simple DSML that captures the finite state machine domain from Ptolemy II [5] as a case study to demonstrate the key steps in the semantic anchoring process. Conclusions and future work appear in Section 5.

2 Background: DSML Specification

A DSML can be formally defined as a 5-tuple $L = \langle A, C, S, M_S, M_C \rangle$ consisting of abstract syntax (A), concrete syntax (C), syntactic mapping (M_C), semantic domain (S) and semantic mapping (M_S) [28]. The syntax of a DSML consists of three parts: an abstract syntax, a concrete syntax, and a syntactic mapping. The abstract syntax A defines the language concepts, their relationships, and well-formedness rules available in the language. The concrete syntax C defines the specific notations used to express models, which may be graphical, textual, or mixed. The syntactic mapping, $M_C : C \rightarrow A$, assigns syntactic constructs to elements in the abstract syntax.

DSML syntax provides the modeling constructs that conceptually form an interface to the semantic domain. The semantics of a DSML provides the meaning behind each well-formed domain model composed from the syntactic modeling constructs of the language. For example, in MIC applications, the semantics of a domain model often prescribes the behavior that simulates an embedded system.

DSML semantics are defined in two parts: a semantic domain S and a semantic mapping $M_S : A \rightarrow S$ [21]. The semantic domain S is usually defined in some formal, mathematical framework, in terms of which the meaning of the models is explained. The semantic mapping relates syntactic concepts to those of the semantic domain. In DSML applications, semantics may be either structural or behavioral. The *structural semantics* describes the meaning of the models in terms of the structure of model instances: all of the possible sets of components and their relationships, which are consistent with the well-formedness rules, are defined by the abstract syntax. Accordingly, the semantic domain for structural semantics is defined by a *set-valued semantics*. The *behavioral semantics* may describe the evolution of the state of the modeled artifact along some time model. Hence, the behavioral semantics is formally captured by a mathematical framework representing the appropriate form of dynamics. In this paper, we focus on the behavioral semantics of a DSML.

3 Semantic Anchoring

Although DSML-s use many different notations, modeling concepts and model structuring principles for accommodating needs of domains and user communities, semantic domains for expressing basic behavior categories are more limited. A broad category of component behaviors can be represented by basic behavioral abstractions, such as Finite State Machine, Timed Automaton, Continuous Dynamics and Hybrid Automaton. This observation led us to the following strategy in defining behavioral semantics for DSML-s:

1. Define a set of minimal modeling languages $\{L_i\}$ for the basic behavioral abstractions and develop the precise specifications for all components of $L_i = \langle C_i, A_i, S_i, M_{Si}, M_{Ci} \rangle$. We use the term "semantic unit" to describe these basic modeling languages.
2. Define the behavioral semantics of an arbitrary $L = \langle C, A, S, M_S, M_C \rangle$ modeling language transformationally by specifying the $M_A : A \rightarrow A_i$ mapping. The $M_S : A \rightarrow S$ semantic mapping of L is defined by the $M_S = M_{Si} \circ M_A$ composition, which indicates that the semantics of L is anchored to the S_i semantic domain of the L_i modeling language.

The tool architecture supporting the semantic anchoring process above is shown in Figure 1. The GME tool suite [4] is used for defining the abstract syntax, A, for an $L = \langle C, A, S, M_S, M_C \rangle$ DSML using UML Class Diagrams [7] and OCL as metalanguage [8]. The $L_i = \langle C_i, A_i, S_i, M_{Si}, M_{Ci} \rangle$ semantic unit is defined as an AsmL specification [1] in terms of (a) an AsmL Abstract Data Model (which corresponds to the A_i, abstract syntax specification of the modeling language defining the semantic unit in the AsmL framework), (b) the S_i, semantic domain (which is implicitly defined by the ASM mathematical framework), and (c) the M_{Si}, semantic mapping (which is defined as a model interpreter written in AsmL).

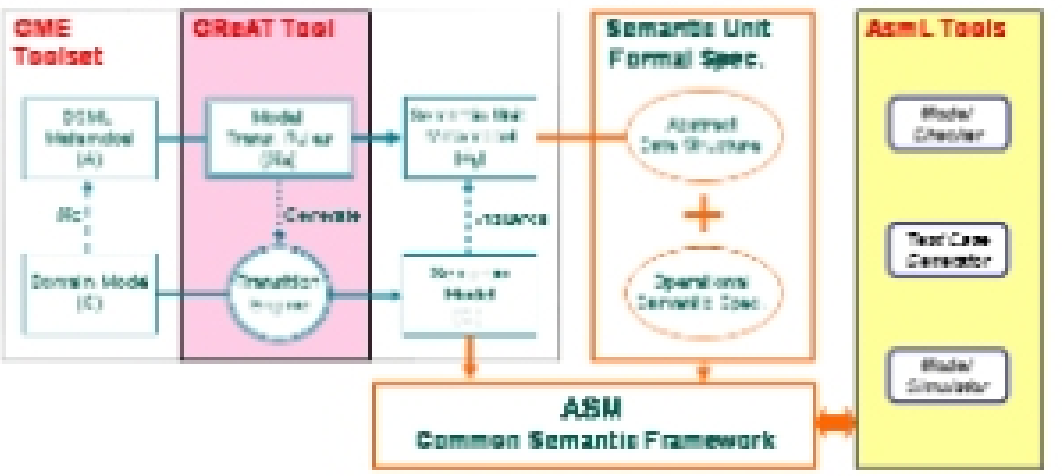

Fig. 1. Tool Architecture for DSML Design throughs Semantic: Anchoring

The $M_A : A \rightarrow A_i$ semantic anchoring of L to L_i is defined as a model transformation using the GReAT tool suite [2]. The abstract syntax A and A_i