Semantic Specifications for Domain-Specific Modeling Languages

Gabor Simko

Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN

Abstract. While there is a generic agreement that formal semantic specifications could resolve ambiguities in modeling languages, in practice, languages are often developed without such unambiguous specifications. In this paper, I propose a logic-based infrastructure for the specification of Domain-Specific Modeling Languages (DSML). The key advantage of the approach is the executability of the specifications: for model conformance checking, model checking and model finding.

1 Research Problem and Motivation

While there is a generic agreement that formal semantic specifications could resolve ambiguities in modeling languages, in practice, languages are often developed without such unambiguous specifications. The advantages of formal specifications are unquestionable in safety-critical applications: they serve as unambiguous documentations, facilitate formal reasoning (such as model checking or formal proofs), and help understanding design faults at an early stage.

In this paper, I propose an infrastructure based on a logic-based language called FORMULA [1] for the semantic specification of DSMLs. FORMULA is a fixed-point logic Constraint Logic Programming (CLP) language that uses algebraic data types for data represention.

As a motivational example, consider a modeling language for embedded systems, such as shown in Fig. 1. Such a language consists of many sub-languages: a data-flow language for representing controller software, a sub-language for describing hardware and networks, a sub-language for the deployment of software to hardware, and a language describing the timing of software execution.

Clearly, in such complex languages, there are many ambiguous parts (e.g., what is the time model for the data-flow language; or what is transmitted on the connections), for which we need to develop unambiguous definitions. In the following, I describe an approach towards this goal.

Section 2 contains the background and related work. In Section 3, my approach and its uniqueness are discussed. Finally, in Section 4, the results and contributions are described.

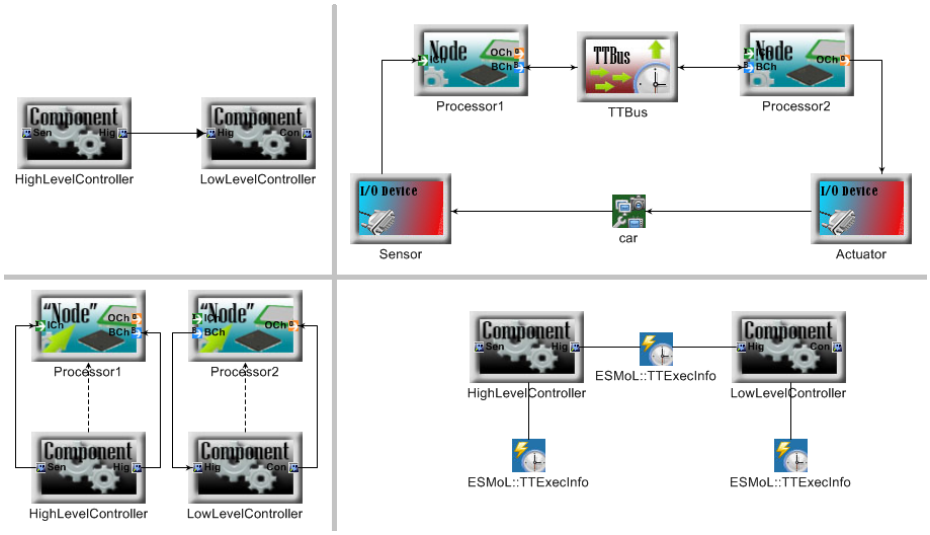

Fig. 1. An embedded controller model. Top-left: high-level data-flow between software components; top-right: hardware and network architecture; bottom-left: software to platform deployment; bottom-right: timing schedule.

2 Background and Related Work

The logic-based language FORMULA was first proposed by Jackson [12] as a formal language for specifying the structural semantics of DSMLs and later for specifying their operational semantics [13]. My research can be considered the continuation of these initiatives. In [21, 22], I used FORMULA for specifying the structural and denotational semantics of a physical modeling language and in [15], our research group specified the operational semantics of a state-chart language variant. FORMULA provides tools for executing these specifications, in particular they can be used for automated model finding, model conformance checking and linear temporal logic (LTL) based model checking.

A different line of research discussed by Rivera [16, 18] uses Maude, an equational logic and term rewriting-based language to specify the operational behavioral and structural semantics of DSMLs. Using Maude's rewriting engine, this representation can be used for LTL model checking, and by leveraging the Real-Time Maude framework it can be used for real-time simulations and analysis [17]. Furthermore, research by Romero [19], Egea [7], and Rusu [20] uses Maude-based formalizations for arguing about model sub-typing, type inference, model conformance and operational semantics of model transformations.

In [5] Chen et al. introduced a translational approach using the Abstract State Machines (ASM) and a semantic anchoring framework, and in [6] they show such a semantic anchoring framework can be used for compositional behavioral specifications. Gargantini [9] also introduces an ASM-based semantic framework

that includes translational approaches, semantic mapping, semantic hooking and semantic meta-hooking, and a weaving approach for semantic specifications.

Esfahasin [8] uses the Z notation to formally specify the behavioral semantics of an activity-oriented DSML modeled in GME. While Z is not executable, the formal specification provides an unambiguous guideline for automated code generation for their models.

A similar line of research is found for Ptolemy [10] [11], where the authors identified and investigated the composition of different models of computations: the models are chosen such that they represent a broad range of computational models.

BIP (Behavior, Interaction and Priority) [2] is a framework that supports the composition of heterogeneous computational systems. The key idea is the separation of component behaviors from component interactions. Such a separation of concerns facilitates the correct composition of components. In [3], the algebra of BIP is formulated, and in [4], the SOS style formalization of glue operators is described.

Structural and Behavioral Semantics

In general, models represent a structure and associated behaviors. Accordingly, specification of modeling languages requires support for specifying both structural and behavioral semantics [13].

Structural semantics describes the meaning of model instances in terms of their structure [5]. Structural semantics is described by a mapping from model instances into a two-valued domain, which distinguishes well-formed models from ill-formed models.

Behavioral semantics is represented as a mapping of the model into a semantic domain that is sufficiently rich for capturing essential aspects of the behavior [6] Although, there are many different representations for the behaviors of languages, in the following I focus on two of them: denotational semantics and operational semantics.

Denotational semantics describes the semantics of the language by mapping to a semantic domain (a domain with well-defined semantics), usually a mathematical domain. Therefore, we can specify the denotational semantics of a DSML by defining a semantic domain (possibly as a meta-model) and a denotational semantic mapping that transforms models of the DSML to models of the semantic domain. E.g., differential algebraic equations, difference equations, state-chart variants, parallel hybrid automata [22] are examples for semantic domains.

Operational semantics describes the step-wise execution of a computational language on an abstract machine. Formal specification of operational semantics involves defining the transformation that specifies how the system can evolve

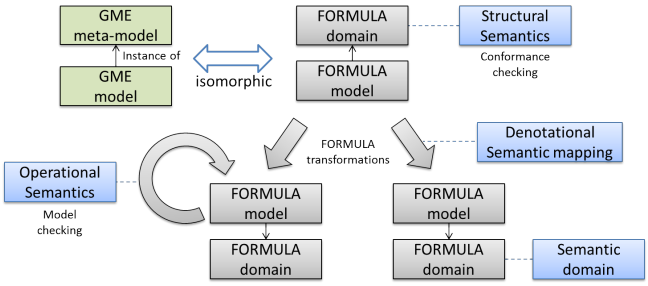

Fig. 2. Our approach for formal semantic specification

through its states. Our research group developed the operational semantics specification for our ESMoL state-chart dialect in [15].

3 Approach and Uniqueness

The essence of my approach is shown in Fig. 2. Here, on the left side, we have a GME (Generic Modeling Environment [14]) meta-model for the language in question, which provides us with a customized modeling environment (i.e., with a concrete syntax) and the abstract syntax for the language. By using the environment, we can draw models, such as shown earlier in Fig. 1. Notice that while the figure is based on GME meta-models and models, the idea is applicable to other modeling environments as well.

As a next step, we would like to assign semantics to these models by means of semantic mapping, but we cannot directly do so in GME. Therefore, we create isomorphic mappings between GME meta-models/models and FORMULA domains/models. Each concept of the meta-model is represented as an algebraic data type, and each element of the model is an instantiation of the corresponding data type.

By now, we have an equivalent FORMULA representation of the (meta-)model, and we can use deductive rules to assign semantics to the language.

Structural semantics is described as the deducibility of a special *conforms* statement. Because of the isomorphism between the GME model/meta-model and FORMULA model/domain, whenever a FORMULA model *conforms* to its domain, the GME model also conforms to its meta-model.

Behavioral semantics is represented by a semantic mapping: in the case of denotational semantics, we transform the FORMULA model to a model of a semantic domain (also represented using algebraic data types). For the operational semantics, we add behavioral concepts to the domain and model (such as the current state of the system), and write a FORMULA transformation to specify the evolution of the system. Based on the operational semantics, we can also perform model checking as described in [15].

There are three factors that contributes to the uniqueness of my approach: 1) using the same formal language for describing both the structure and the behavior establishes a tight connection between them (and which can be leveraged later on, when developing formal proofs); 2) being executable, the specifications can be used for model conformance checking, model checking, and for driving simulations; 3) support for model finding, i.e., automated search for models that satisfy desired properties.

4 Results and Contributions

The approach introduced in this paper was successfully applied to specify the semantics for a suite of Cyber-Physical Systems (CPS) modeling languages in DARPA's Adaptive Vehicle Make (AVM) program. So far, our research group has developed the specification for many languages. In particular, I have developed semantic specifications for a hybrid bond graph language [22] (a physical modeling language), a cyber-physical system integration language (CyPhyML), parts of the embedded systems modeling language (ESMoL) and the Simulink Stateflow language.

Altogether, our research group has developed more than 4000 lines of specifications. To our knowledge, this is the largest research project that aims complete, formal, and unambiguous semantic specifications. We expect invaluable feedback from the more than 1000 systems engineers and 200 designs teams who currently uses our languages in DARPA's FANG challenge (http://vehicleforge.org).

My contribution is the following: 1) the proposal of an infrastructure based on algebraic data types and a logic language for both structural and behavioral semantic specifications; 2) developing both structural and behavioral (denotational and operational) semantics in the proposed language; 3) demonstrating the applicability of the approach by specifying a suite of languages in an industry-sized project.

References

1. *FORMULA*. http://research.microsoft.com/en-us/projects/formula.
2. A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM*, pages 3 –12, Sept. 2006.
3. S. Bliudze and J. Sifakis. The algebra of connectors – structuring interaction in BIP. *IEEE Transactions on Computers*, 57(10):1315 –1330, Oct. 2008.
4. S. Bliudze and J. Sifakis. A notion of glue expressiveness for Component-Based systems. In *CONCUR*, page 508–522, 2008.
5. K. Chen, J. Sztipanovits, S. Abdelwalhed, and E. Jackson. Semantic anchoring with model transformations. In *Model Driven Architecture – Foundations and Applications*, volume 3748 of *LNCS*, pages 115–129. Springer, 2005.
6. K. Chen, J. Sztipanovits, and S. Neema. Compositional specification of behavioral semantics. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '07, page 906–911, San Jose, CA, USA, 2007. EDA Consortium.

7. M. Egea and V. Rusu. Formal executable semantics for conformance in the MDE framework. *Innovations in Systems and Software Engineering*, 6(1-2):73–81, Dec. 2010.
8. N. Esfahani, S. Malek, J. P. Sousa, H. Gomaa, and D. A. Menascé. A modeling language for activity-oriented composition of service-oriented software systems. In *Model Driven Engineering Languages and Systems*, volume 5795, pages 591–605. Springer, 2009.
9. A. Gargantini, E. Riccobene, and P. Scandurra. A semantic framework for metamodel-based languages. *Automated Software Engineering*, 16(3-4):415–454, Apr. 2009.
10. A. Goderis, C. Brooks, I. Altintas, E. Lee, and C. Goble. Composing different models of computation in kepler and ptolemy II. In *Computational Science – ICCS*, volume 4489, pages 182–190. Springer, 2007.
11. A. Goderis, C. Brooks, I. Altintas, E. Lee, and C. Goble. Heterogeneous composition of models of computation. *Future Generation Computer Systems*, 25(5):552–560, May 2009.
12. E. Jackson and J. Sztipanovits. Formalizing the structural semantics of domain-specific modeling languages. *Software and Systems Modeling*, 8(4):451–478, 2009.
13. E. Jackson, R. Thibodeaux, J. Porter, and J. Sztipanovits. Semantics of domain-specific modeling languages. *Model-Based Design for Embedded Systems*, 1:437, 2009.
14. A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi. The generic modeling environment. In *Workshop on Intelligent Signal Processing, Budapest, Hungary*, volume 17, 2001.
15. D. Lindecker, G. Simko, I. Madari, T. Levendovszky, and J. Sztipanovits. Multi-way semantic specification of domain-specific modeling languages. In *ECBS*, 2013.
16. J. E. Rivera, F. Duran, and A. Vallecillo. Formal specification and analysis of domain specific models using maude. *SIMULATION*, 85(11-12):778–792, Aug. 2009.
17. J. E. Rivera, F. Durán, and A. Vallecillo. On the behavioral semantics of real-time domain specific visual languages. In *Rewriting Logic and Its Applications*, volume 6381, pages 174–190. Springer, 2010.
18. J. E. Rivera and A. Vallecillo. Adding behavior to models. In *EDOC*, page 169. IEEE, Oct. 2007.
19. J. R. Romero, J. E. Rivera, F. Duran, and A. Vallecillo. Formal and tool support for model driven engineering with maude. *Journal of Object Technology*, 6(9):187–207, 2007.
20. V. Rusu. Embedding domain-specific modelling languages in maude specifications. *ACM SIGSOFT Software Engineering Notes*, 36(1):1–8, 2011.
21. G. Simko, T. Levendovszky, S. Neema, E. Jackson, T. Bapty, J. Porter, and J. Sztipanovits. Foundation for model integration: Semantic backplane. In *IDETC/CIE*, 2012.
22. G. Simko, D. Lindecker, T. Levendovszky, E. Jackson, S. Neema, and J. Sztipanovits. A framework for unambiguous and extensible specification of DSMLs for cyber-physical systems. In *ECBS*, 2013.