Formal Semantics for Conceptual Modeling Languages based on Model Theory

Victoria Döller
Supervisor: o. Univ.-Prof. Prof.h.c. Dr. Dimitris Karagiannis

University of Vienna
Faculty of Computer Science
Research Group Knowledge Engineering
Währingerstraße 29, 1090 Vienna, Austria
victoria.doeller@univie.ac.at

Abstract. The era of models as executable entities rather than descriptive pictures is on the rise. Nowadays modeling languages provide a variety of additional features like transformation, simulation, or code generation and the consumers of models are persons as well as machines. This development brings about the necessity to make models machine-processable, i.e. to formalize modeling languages. The goal of this research project is to find a suitable formalism for this purpose. This formalism will be grounded on mathematical theories, adopting their unambiguity and expressiveness and enabling the application of established mathematical methods. Therefore the plan for this research project is to examine several promising mathematical theories regarding their possibilities and limitations. A first result of this examination is presented in this paper: the formalization of the semantics of a metamodel based on model theory, a branch of mathematical logic. We outline the analogy between the concepts of metamodeling and model theory and describe how a metamodel can be formalized with this approach. A proof of concept is given by applying the formalism on a simple example, namely the entity-relationship modeling language.

Keywords: Metamodeling · Formalization · Formal Semantics · Model Theory

1 Introduction

In the past decade modeling languages have evolved from mere instruments for pictures supporting human understanding to highly specialized tools with value adding mechanisms like transformation, simulation or code generation. An increasing number of models are in fact not created for human consumption but for computational processing and execution. The most significant requirement

J. Ralyté and Y. Wautelet (Eds.): PoEM 2018 Doctoral Consortium Papers, pp. 61-73, 2018.

evolving from this fact is the inevitable need for appropriate and complete formalization of the underlying modeling languages, as machines cannot understand semi-formal or natural language specifications. Yet there is no commonly used approach or language to unambiguously define a modeling language much less a formalism which suffices for these needs. Thalheim [27] states in his reevaluation of notions of conceptual models that the establishment of formal foundations is one of the open research challenges in conceptual modeling. According to Bork and Fill [2] formalization enables an unambiguous intersubjective understanding of modeling methods and it enables models to act "... as machine-processable knowledge bases for answering queries, simulation behavior, performing reasoning, verification & validation, or generating executable code ..." (p. 3400). They analyze in their work six established modeling methods with respect to their degree of formalization. Their findings show that there exist several attempts to formalize modeling languages or at least parts thereof, but these attempts do not follow a common procedure or use a common formalism. This leads us to our first research question:

RQ1: *How can a suitable formalism for comprehensive applicability to describing metamodels, conceptual modeling languages and modeling methods be constructed?*

When talking about rigorous formalization the first things that come to mind are concepts from mathematics as these theories are formal, complete, and unambiguous by definition. Furthermore mathematical structures admit the exploitation of metamodels through the application of well elaborated algorithms or the discovery of hidden properties and connections by transforming them to related structures. A goal of this research project is therefore to examine appropriate theories that project concepts of metamodeling in a natural way to mathematical concepts.

RQ2: *Which mathematical concepts can serve as a formal foundation for modeling methods? Which opportunities do the different concepts provide? What are their limitations?*

Fill, Redmond and Karagiannis [8] develop a formal framework based on set theory and first order logic to describe the implementation of a modeling language on the ADOxx metamodeling platform. The difference to the approach at hand is that we do not aim at formalizing how a modeling language is implemented on a specific platform but at finding mathematical structures building the basis of a model with no limitations imposed by an implementation.

The mathematical concept that is most self-evident as a basis of a diagrammatic model is graph theory [1] which is also applied in a variety of existing research areas e.g. software engineering or language engineering [18]. The interpretation of models as graphs will serve as a starting point for our research. Furthermore, established techniques such as graph grammar and graph trans-

formation are promising for supporting mechanisms on models. Moreover, to support the conceptualizations of models we will also examine suitable structures to describe concepts and combine them with graph theory so that we can use the interdependency between them. A first selection of theories we want to investigate are conceptual graphs [24, 25], pattern theory [11], model theory [4] and formal concept analysis [9]. Category theory [13] may serve as the connecting link between them. In a first approach we build a bridge between graph theory and model theory, a concept from logic, and use the power of formal languages for modeling languages.

Another emerging demand in metamodeling which guides us in this research question is agility, a requirement modeling languages nowadays have to meet, see Karagiannis [15]. A suitable formalism for modeling methods has to be modifiable to the same extent as the metamodel is agile. Furthermore, modeling methods must be extensible and combinable.

At this quite early stage of the research project we address the specific question of finding ways to formalize semantics, similar to established approaches like formal semantics for programming languages. The aim is to enable model checking against the metamodel as well as checking the satisfiability of metamodel constraints.

RQ3: *How can the semantics of a modeling language be formalized in order to enable automated processing?*

Here we explicitly aim at a structurally founded formalism. In view of existing work on formalizing semantics, e.g. ConceptBase [14], we stress that our goal is not to develop a programming language. We want to describe a modeling language in a way that any program able to process the chosen mathematical structure can understand it. At this point in time we already started an attempt with the mathematical concepts of model theory, a branch of logic, which in contrast to several other approaches is not restricted to first-order logic a priori.

The rest of this paper is structured as follows: we start with an outline of the research plan. Then we provide a summary of the results achieved so far. We give an attempt to formalize the semantics of a modeling language with tools from mathematical logic. We do so by exhibiting an analogy between metamodels, the structure and concepts of graphs and formal and logical languages. We reason the parallelism of models and graphs and how the concepts of metamodeling can be mapped to the concepts of graph theory. Then we give a short introduction to model theory so that we can show the applicability of formal languages and models to metamodeling. As a final point we give a proof of concept by applying the approach to a simple example, namely the entity-relationship modeling language.

2 Methodology

Phase 1: We start with a literature review to identify and study other existing approaches in metamodeling focusing in particular on their formal foundations.

Phase 2: We study the concepts of metamodeling and abstract patterns and structures which can be described mathematically. We will proceed iteratively, concept by concept, component by component, and add a phase of review and consolidation after each iterative step. For the concepts of metamodels we follow the definitions of Kern, Hummerl and Kühne [17] and assemble them to components of modeling methods as described in the modeling framework of Karagiannis and Kühn [16]. The first iteration addresses RQ3 and is in progress. First results are presented in the next section.

Phase 3: We examine mathematical theories standing to reason and try to match the discovered patterns from phase 2. Due to the iterative approach this phase will to some extent be executed simultaneously to the second one. We try to elaborate a formalism for the concept or component under study and moreover try to consolidate or at least link the mathematical approaches used in former iterations. For proof of concept we will implement prototypes whenever this is meaningful.

Phase 4: We finally consolidate the results from the iteration steps in a holistic approach. The aim is to establish a final formalism comprising possibilities to formally define modeling languages or at least the most important components. This formalism will be a construct of structures, languages, methods and algorithms with high coherence and interoperability.

Phase 5: To validate our approach we will apply the constructed formalism on the diverse modeling languages available at the OMiLAB platform [20].

3 Preliminary Results

In this chapter we present the preliminary results of our ongoing work on RQ3. The goal is to find a mathematical formalism to describe a modeling language in a way that enables the definition of semantic constraints of the language. The approach is inspired by the research area of formal semantics in software engineering [22, 23].

3.1 Formal Semantics

Following the metamodeling framework of Karagiannis and Kühn [16] a modeling language consists of syntax, semantics and notation. Whilst the syntax is usually well defined by a metamodel semantics is often described in a vague manner. Even the term semantics is not used consistently in the literature. Some

authors stress that semantics is the assignment of meaning for the purpose of human understanding and communication [18], others consider semantics as the expression of meaning in a way well understood by the intended consumer indifferent if human or machine [16]. Some authors include context conditions and constraints under the heading of static semantics [10] others deny these aspects as part of semantics and allocate them in the syntax [12]. We stick to the first, less restrictive notion.

Nevertheless, all authors agree that semantics is a crucial but challenging thing to deal with and many approaches have been devised to formalize semantics for different areas. Especially for programming languages there are already well elaborated theories on how to assign meaning to expressions and programs. Historically the most important approaches are denotational semantics, operational semantics, translational semantics, axiomatic semantics and algebraic semantics [18, 22, 23]. Our approach using model theory for defining semantic constraints of modeling languages is a variant of algebraic semantics. We chose model theory because by definition its aim is to describe the semantics of a given structure as well as instances satisfying these semantical constraints. For the definition of metamodels we use the concepts identified by Kern, Hummerl and Kühne [17].

3.2 Models ↔ Graphs ↔ Formal Languages

In order to use the formal power of mathematical model theory in metamodeling we have to justify the analogy of the concepts on both sides, i.e. the validity of the red arrows in Figure 1. To do so we use as interim stage the concept of graphs as well as colors and labels on graphs. The procedure of creating a formal language for the structure of colored, labeled graphs is already established in model theory. The feasibility of representing models as graphs and the most important concepts of metamodeling, object types and relationship types, as concepts of colors and labels gives us the second analogy and closes the circle. The main benefit we gain by this mapping is the possibility to formally define the semantics of the modeling language in the unambiguous syntax of the specifically defined formal language $\mathcal{L}$ and logic, see Figure 2. The body of sentences representing the semantics of $\mathcal{L}$ form a so called $\mathcal{L}$-theory. A valid model according to the semantics is represented by an $\mathcal{L}$-structure satisfying all the sentences of the $\mathcal{L}$-theory. Tellingly in model theory an $\mathcal{L}$-structure with this property is called a model of the $\mathcal{L}$-theory.

Once we have derived a formal language from our metamodel, defined the semantic constraints, and expressed the models in the new language, the prerequisites for automated model checking are satisfied, see Figure 3.

3.3 Models as Graphs

When looking at diagrammatic models from a mathematical point of view it is natural to interpret them as graphs. Models consist of objects and relations between objects; graphs are defined as sets of vertices and edges i.e. ordered pairs of vertices. A mapping of subtler concepts, namely object types and relationship

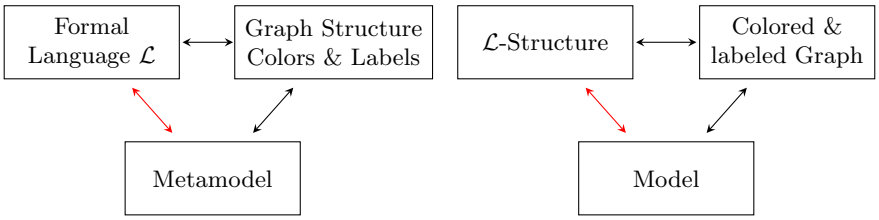

Fig. 1. The analogy of concepts from metamodeling, graph theory and model theory.

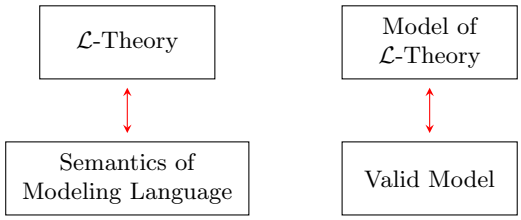

Fig. 2. The correspondence between the informal semantics of a modeling language and the formal semantic constraints in model theory.

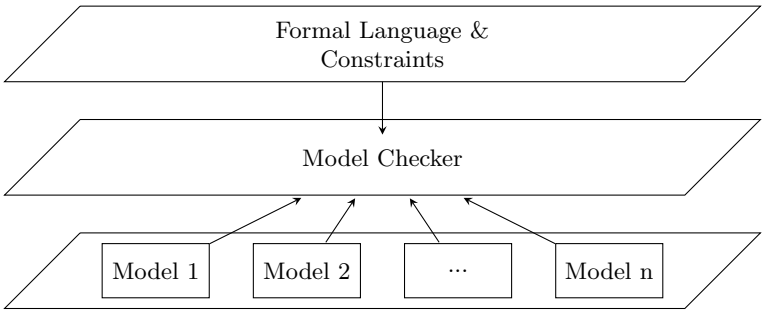

Fig. 3. The formal definition of the modeling language on the top layer and the formal description of the models on the bottom layer allow the automatic check of the latter one against the former one.

types, can be achieved through the concepts of colored vertices for the former ones or labeled edges for the latter ones.

Definition 1 (Graph). *A directed graph consist of a set of vertices V and a set of edges $E \subseteq V \times V$.*

To begin with this definition is sufficient. For more complex models with multiple edges between vertices we need the concept of multigraphs.

Definition 2 (Multigraph). *A directed multigraph consist of a set of vertices V, a set of edges E and two functions $s : E \to V$ and $t : E \to V$, where s assigns a source vertex to an edge and t assigns a target vertex.*

Definition 3 (Colored Graph). *A colored graph is a graph with an additional map $p : V \to P, P$ a set of colors, which assigns a color to each vertex.*

Definition 4 (Labeled Graph). *A labeled graph is a graph with an additional map $l : E \to L, L$ a set of labels, which assigns a label to each edge.*

Often authors use the terms vertex-labels and edge-labels instead of colors and labels, but for emphasizing the difference between object types and relationship types we prefer this terminology. The assigned color of a vertex symbolizes the object type of the corresponding object in the model, the assigned label of an edge the relationship type. Usually mathematicians consider graph colorings where no adjacent vertices have the same color. This is a restriction we do not follow.

3.4 Model Theory

Model theory is the study of mathematical structures from the viewpoint of mathematical logic. On the one hand its aim is to define a formal language and to describe a given structure with axioms. The axioms or sentences are written in the well known syntax of logical expressions and represent the semantics of the structure under study. On the other hand model theory aims at the study of structures fulfilling these axioms, the so called models. Model theory is a powerful tool for describing a multitude of mathematical structures such as rings, ordered groups or graphs and to define their semantics. In general, model theory is not confined to a specific type of logic. Nevertheless in the introduction and the example below we restrict ourselves to first-order logic.

Definition 5 (Language $\mathcal{L}$ and $\mathcal{L}$-Structure). *A language $\mathcal{L}$ is a collection of symbols, which are divided in three groups: function symbols denoted by f, relation symbols denoted by R and constant symbols denoted by c. Each function symbol and each relation symbol has associated a number in $\mathbb{N}$, which defines the arity.*

$$\mathcal{L} = \{f_i, R_j, c_k \mid i \in I, j \in J, k \in K\}$$

An $\mathcal{L}$-Structure $\mathcal{M}$ is a set M, called the universe of $\mathcal{M}$, and the interpretations of f_i, R_j and c_k in M:

- *An interpretation of a function symbol f is a function $f^{\mathcal{M}} : M^n \to M$ where n is the arity of f.*
- *An interpretation of a relation symbol R is a subset $R^{\mathcal{M}} \subseteq M^p$ where p is the arity of R.*
- *An interpretation of a constant symbol c is an element $c^{\mathcal{M}} \in M$.*

The structure $\mathcal{M}$ is denoted by

$$\mathcal{M} = \{M, f_i^{\mathcal{M}}, R_j^{\mathcal{M}}, c_k^{\mathcal{M}} \mid i \in I, j \in J, k \in K\}$$

Example 1 (Graphs). The language $\mathcal{L}$ for graphs contains only one symbol, a binary relation C representing the edge or connection between two vertices.

$$\mathcal{L} = \{C\}$$

Example 2 (Colored Graphs). The language $\mathcal{L}'$ for colored graphs contains additionally several unary relations P_i, the colors of the vertices.

$$\mathcal{L}' = \{C, P_i \mid i \in I\}$$

Remark 1 (Formulas, Sentences and Satisfaction). We forgo a precise definition of the from mathematical logic well-known terms formula and sentence. The interested reader can find them in [4]. Just recall that a *formula* is a construct of the symbols of the language together with the logical operators $=, \wedge, \vee, \neg, \to, \leftrightarrow, \forall, \exists, (,)$ as well as infinitely many variable symbols following the well known syntactical rules for logical expressions.
A *sentence* is a formula with no free variables, i.e. it is a formula with no variables or it is of the form $Q_1 x_1 ... Q_m x_m \phi(x_1, ..., x_n)$ with ϕ a quantifier-free formula, $Q_i \in \{\forall, \exists\}$ and $m = n$.
If an $\mathcal{L}$-Structure $\mathcal{M}$ satisfies a sentence ϕ we write

$$\mathcal{M} \models \phi.$$

Definition 6 ($\mathcal{L}$-Theory, Model of an $\mathcal{L}$-Theory). *An $\mathcal{L}$-Theory $\mathcal{T}$ is a set of sentences of the language $\mathcal{L}$. A Model of an $\mathcal{L}$-Theory $\mathcal{T}$ is an $\mathcal{L}$-Structure, which satisfies all sentences of $\mathcal{T}$.*

Example 3. Consider the language $\mathcal{L}$ for graphs from Example 1. Then the theory

$$\mathcal{T} = \{\forall x \forall y (C(x, y) \to C(y, x))\}$$

has as models all undirected graphs.

For a detailed introduction into model theory see [4]

4 Proof of Concept based on Entity-Relationship Models

A simplified version of the entity-relationship modeling language will serve as an example for the definition of the language inherent semantics with the presented approach. The simplified version under study comprises three concepts, namely entity, relation and attribute, only one undirected relationship type to connect any of the three types and does not admit multiple relationships between the same two objects. This example shows the simplicity and expressiveness of the presented approach.

4.1 The Language $\mathcal{ER}$

First we define the language for this special case of colored graphs or the meta-model of ER-models respectively:

$$\mathcal{ER} = \{C, E, R, A\}$$

where C is the binary relation of connections between the objects, E is the color or type of entities, R is the color or type of relations and A is the color or type of attributes. Now we can define the sentences for our Theory $\mathcal{T}$ for the language $\mathcal{ER}$:
The first three sentences ensure that every object has exactly one type, that the relation C is bidirectional (i.e. for every directed relation, also the inverse direction is element of the relation subset) and that no element is connected to itself.

$$\forall x((E(x) \vee R(x) \vee A(x)) \wedge (\neg(E(x) \wedge R(x)) \wedge \neg(E(x) \wedge A(x)) \wedge \neg(R(x) \wedge A(x)))) \tag{1}$$

$$\forall x \forall y(C(x, y) \rightarrow C(y, x)) \tag{2}$$

$$\forall x(\neg C(x, x)) \tag{3}$$

The following sentences are specific for ER-models:

$$\forall x \forall y((A(x) \wedge C(x, y)) \rightarrow \neg A(y)) \tag{4}$$

$$\forall x \forall y((E(x) \wedge C(x, y)) \rightarrow \neg E(y)) \tag{5}$$

$$\forall x \forall y((R(x) \wedge C(x, y)) \rightarrow \neg R(y)) \tag{6}$$

$$\forall x \exists y \exists z(R(x) \rightarrow (E(y) \wedge E(z) \wedge C(x, y) \wedge C(x, z) \wedge \neg(y = z))) \tag{7}$$

$$\forall x \forall y \forall z((A(x) \wedge C(x, y) \wedge C(x, z)) \rightarrow y = z) \tag{8}$$

Sentences (4) - (6) ensure that no element is connected to an element of the same type. Sentence (7) forces an element of type *relation* to have at least two connections to objects of type *entity*. Sentence (8) ensures that attributes are connected to only one other element.

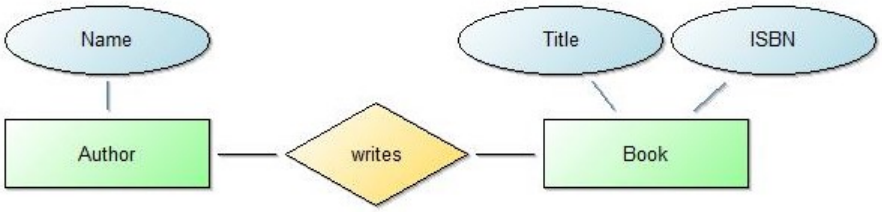

Fig. 4. A valid ER-model according to $\mathcal{T}$

4.2 The Models

Example 4. The $\mathcal{ER}$-Structure $\mathcal{M}_1$ for Figure 4 looks as follows (the elements of $C^{\mathcal{M}_1}$ should be read as unordered pairs) :

$$\mathcal{M}_1 = \{M_1, C^{\mathcal{M}_1}, E^{\mathcal{M}_1}, R^{\mathcal{M}_1}, A^{\mathcal{M}_1}\} \tag{9}$$

$$M_1 = \{Author, writes, Book, Name, Title, ISBN\} \tag{10}$$

$$C^{\mathcal{M}_1} = \{(Author, Name), (Author, writes), (writes, Book), (Book, Title), (Book, ISBN)\} \tag{11}$$

$$E^{\mathcal{M}_1} = \{Author, Book\} \tag{12}$$

$$R^{\mathcal{M}_1} = \{writes\} \tag{13}$$

$$A^{\mathcal{M}_1} = \{Name, Title, ISBN\} \tag{14}$$

The $\mathcal{ER}$-Structure $\mathcal{M}_1$ is a Model for the $\mathcal{ER}$-Theory $\mathcal{T}$, it fulfills all the sentences (1) - (8).

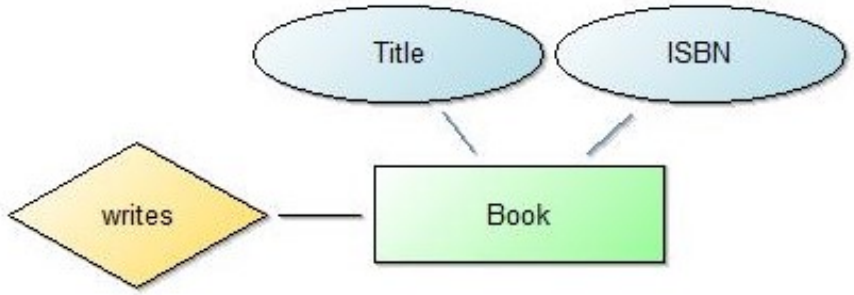

Fig. 5. A non valid ER-model according to $\mathcal{T}$

Example 5. The $\mathcal{ER}$-Structure $\mathcal{M}_2$ for Figure 5 is a substructure of $\mathcal{M}_1$:

$$\mathcal{M}_2 = \{M_2, C^{\mathcal{M}_2}, E^{\mathcal{M}_2}, R^{\mathcal{M}_2}, A^{\mathcal{M}_2}\} \tag{15}$$

$$M_2 = \{writes, Book, Title, ISBN\} \tag{16}$$

$$C^{\mathcal{M}_2} = \{(writes, Book), (Book, Title), (Book, ISBN)\} \tag{17}$$

$$E^{\mathcal{M}_2} = \{Book\} \tag{18}$$

$$R^{\mathcal{M}_2} = \{writes\} \tag{19}$$

$$A^{\mathcal{M}_2} = \{Title, ISBN\} \tag{20}$$

The $\mathcal{ER}$-Structure $\mathcal{M}_2$ is not a Model for the $\mathcal{ER}$-Theory $\mathcal{T}$, because sentence (7), denoted by ϕ, does not hold for $x = writes$.

$$\mathcal{M}_2 \not\models \phi$$

5 Conclusion and Future Work

In this paper we present the research goal of this PhD thesis - a formalism for modeling methods based on mathematical concepts - and present first results using model theory, a concept from logic. We show how a modeling language can be formalized in the notion of a formal language $\mathcal{L}$, how to enrich the language with semantic constraints written in first order logic and how models are then translated to so called $\mathcal{L}$-structures, which can be checked against the constraints.

At the moment we are also working on the application of particular concepts of model theory, i.e. sorted model theory and second order logic for the axioms, on modeling language definitions and apply them to more complex modeling methods like petri nets. With sorted model theory we can easily represent multigraphs. Furthermore, we are examining a procedure for merging modeling languages via the use of extensions and cartesian products of formal languages.

We will proceed with the literature review starting with literature on the application of graph theory in computer science, e.g. attributed graphs, type graphs, graph grammars and transformation [5–7, 18, 19, 21, 22, 26]. We expect to thereby find techniques to supplement our formalism.

Afterwards we will start to examine possible formal structures to enrich the semantics and achieve the necessary expressibility and conceptualization of modeling methods [9, 11, 24, 25] and then proceed with the iteration steps of investigating metamodeling concepts and components.

The main challenge in this research project is the balancing act of creating a feasible formalism. Thalheim, referencing the commandments of Bowen and Hinchey [3], asks: "Thou shalt formalise but not over-formalise." [27] (p. 25) We will strive to construct a formalism which is on the one hand powerful enough to define the syntax and semantics of a modeling language as well as to provide methods and algorithms like language merging, simulation and transformation and which is on the other hand intuitive and convenient enough so that the modeling method engineers can use it in a suitable way.

Acknowledgment

This doctoral project is supervised by o. Univ.-Prof. Prof.h.c. Dr. Dimitris Karagiannis, head of the research group Knowledge Engineering at the Faculty of Computer Science, University of Vienna.

References

1. Bondy, J.A., Murty, U.S.R.: Graph theory, volume 244 of Graduate Texts in Mathematics. Springer, New York (2008)
2. Bork, D., Fill, H.G.: Formal aspects of enterprise modeling methods: a comparison framework. In: System Sciences (HICSS), 2014 47th Hawaii International Conference on. pp. 3400–3409. IEEE (2014)
3. Bowen, J.P., Hinchey, M.G.: Ten commandments ten years on: lessons for asm, b, z and vsr-net. In: Rigorous Methods for Software Construction and Analysis, pp. 219–233. Springer (2009)
4. Chang, C.C., Keisler, H.J.: Model theory 3rd ed. North-Holland, Amsterdam (1990)
5. Depke, R., Heckel, R., Küster, J.M.: Formal agent-oriented modeling with uml and graph transformation. Science of Computer Programming **44**(2), 229–252 (2002)
6. Ehrig, H., Prange, U., Taentzer, G.: Fundamental theory for typed attributed graph transformation. In: International conference on graph transformation. pp. 161–177. Springer (2004)
7. Ehrig, H., Rozenberg, G., rg Kreowski, H.J.: Handbook Of Graph Grammars And Computing By Graph Transformations, Vol 2: Applications, Languages And Tools. World Scientific (1999)
8. Fill, H.G., Redmond, T., Karagiannis, D.: Fdmm: A formalism for describing adoxx meta models and models. In: Maciaszek, L., Cuzzocrea, A., Cordeiro, J. (eds.) ICEIS 2012 - 14th International Conference on Enterprise Information Systems. SciTePress (2012)
9. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer (1999)
10. Geisler, R., Klar, M., Pons, C.: Dimensions and dichotomy in metamodeling. Technische Universität Berlin, Fachbereich 13, Informatik (1998)
11. Grenander, U.: General pattern theory-A mathematical study of regular structures. Clarendon Press (1993)
12. Harel, D., Rumpe, B.: Meaningful modeling: what's the semantics of" semantics"? Computer **37**(10), 64–72 (2004)
13. Herrlich, H., Strecker, G.: Category theory. Allyn and Bacon (1973)
14. Jeusfeld, M.A.: Semcheck: Checking constraints for multi-perspective modeling languages. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 31–53. Springer (2016)
15. Karagiannis, D.: Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics. pp. 5–10. ACM (2015)
16. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Proceedings of the Third International Conference on E-Commerce and Web Technologies. p. 182. Springer (2002)

17. Kern, H., Hummel, A., Kühne, S.: Towards a comparative analysis of meta-metamodels. In: Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11. pp. 7–12. ACM (2011)
18. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley Professional (2008)
19. Limbourg, Q., Vanderdonckt, J.: Multipath transformational development of user interfaces with graph transformations. In: Human-Centered Software Engineering, pp. 107–138. Springer (2009)
20. Omilab - open models initiative laboratory, http://www.omilab.org, last accessed 07 October 2018
21. Osis, J., Donins, U.: Topological UML Modeling: An Improved Approach for Domain Modeling and Software Development. Elsevier (2017)
22. Schmidt, D.A.: Denotational Semantics, a Methodology for Language Development. Brown (1988)
23. Slonneger, K., Kurtz, B.L.: Formal Syntax and Semantics of Programming Languages (1995)
24. Sowa, J.F.: Conceptual structures: information processing in mind and machine (1983)
25. Sowa, J.F.: Chapter 5 conceptual graphs. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, Foundations of Artificial Intelligence, vol. 3, pp. 213 – 237. Elsevier (2008)
26. Stanciulescu, A., Vanderdonckt, J., Mens, T.: Colored graph transformation rules for model-driven engineering of multi-target systems. In: Proceedings of the third international workshop on Graph and model transformations. pp. 37–44. ACM (2008)
27. Thalheim, B.: Conceptual model notions–a matter of controversy: Conceptual modelling and its lacunas. Enterprise Modelling and Information Systems Architectures **13**, 9–27 (2018)