Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Morphism axioms

Florian Rabe [1]

Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany

ARTICLE INFO

Article history:
Received 14 November 2016
Received in revised form 27 June 2017
Accepted 4 July 2017
Available online xxxx
Communicated by D. Sannella

Keywords:
Logic
Specification
Institution
Theory morphism
Model morphism
Functorial

ABSTRACT

We introduce a new concept in the area of formal logic: axioms for model morphisms.
We work in the setting of specification languages that define the semantics of a theory as a category of models. While it is routine to use axioms to specify the class of models of a theory, there has so far been no analogue to systematically specify the morphisms between these models. This leads to subtle problems where it is difficult to give a theory that specifies the intended model category, or where seemingly isomorphic theories actually have non-isomorphic model categories. Our morphism axioms remedy this by providing new syntax for axiomatizing and reasoning about the properties of model morphisms.
Additionally, our system resolves a subtle incompatibility between theory morphisms and model morphisms: the semantics that maps theories to model categories is functorial. While this result is standard in principle, previous formulations had to restrict the allowed theory morphisms or the allowed model morphisms. Our system allows establishing the result in full generality.

1. Introduction

Motivation. One of the most important techniques in formal logic, especially in specification, is the use of axioms to restrict the class of admissible models of a theory. Informally, a theory consists of symbol declarations and axioms, and a model is an interpretation of the symbols that satisfies the axioms. Then the (model-theoretical) semantics of a theory Θ is given by the class $\mathbf{Mod}(\Theta)$ of models. The present paper explores two deep technical problems in this context.

Firstly, a frequent interest is to specify $\mathbf{Mod}(\Theta)$ not only as a class of models but as a category of models and model morphisms. Indeed, many interesting properties of models can be studied via the properties of the category, including initial models, product models, submodels, and quotient models. For example, we want the theory `Group` to give rise to the category $\mathbf{Mod}(\texttt{Group})$ of groups and group homomorphisms, and $\mathbf{Mod}(\texttt{Top})$ should be the category of topological spaces and continuous functions. For almost every logic, there is a canonical way to define **Mod** in such a way that $\mathbf{Mod}(\Theta)$ is indeed a useful category.

However, typically the author of Θ has only indirect and limited control over the choice of morphisms in $\mathbf{Mod}(\Theta)$. Moreover, subtle variations of Θ—even if they do not change the class of models—may yield very different model morphisms and thus different model categories. This is because theories usually provide no syntax for fine-tuning specifically the morphisms in $\mathbf{Mod}(\Theta)$: While theory authors can add axioms to Θ to change the class of models, they have no direct influence on the morphisms.

E-mail address: f.rabe@jacobs-university.de.

[1] This work was supported by DFG grant RA-18723-1 OAF.

http://dx.doi.org/10.1016/j.tcs.2017.07.006
0304-3975/

For example, there are many choices for the theory `Top` whose models are exactly the topological spaces. But different choices can yield very different model morphisms, which may or may not be the continuous functions.

Secondly, formal logic can use theory morphisms $\vartheta : \Theta \to \Theta'$ to translate between theories. This method—developed most deeply in the field of algebraic specification mostly through the concept of institutions [7]—yields a category **Th** of theories and theory morphisms. Informally, a theory morphism is a map of Θ-symbols to Θ'-expressions that preserves all Θ-axioms. The main properties of theory morphisms are that

- ϑ extends homomorphically to a mapping of Θ-formulas to Θ'-formulas, which is guaranteed to map Θ-theorems to Θ'-theorems.
- ϑ induces a model reduction functor $\mathbf{Mod}(\vartheta) : \mathbf{Mod}(\Theta') \to \mathbf{Mod}(\Theta)$.

This dual role of translating both syntax and semantics[2] has made theory morphisms an extremely valuable tool for structuring and relating large theories [16,6].

However, not every theory morphism is well-behaved with respect to model morphisms. While $\mathbf{Mod}(\vartheta)$ always reduces Θ'-models to Θ-models, not every Θ'-model morphism can be reduced to a Θ-model morphism. Thus, $\mathbf{Mod}(\vartheta)$ is not always a functor, and consequently **Mod** is not always a functor from **Th** to $\mathcal{CAT}^{op}$.

Combining both of the above problems, we can find isomorphic theories $\Theta \leftrightarrow \Theta'$, whose model categories are not isomorphic. Thus, when using theories to formally specify model categories, small changes in the syntax that appear inconsequential because they are justified by a theory isomorphism may significantly change the semantics. This is not a contrived problem—in fact, we will see below that it happens all the time, even for elementary examples like the theory of monoids.

Related work. The two problems described above have not received much attention in the literature so far. We can distinguish two fields that have avoided the practical consequences in two different ways.

On the one hand, algebraic specification languages of the OBJ tradition such as OBJ [8] or CASL [2] avoid the problem by restricting the allowed theory morphisms: they require that theory morphisms map symbols to *symbols* rather than to arbitrary *expressions*. In that special case, $\mathbf{Mod}(\vartheta)$ always yields a functor. Here by symbol-to-symbol maps, we mean that a theory morphism $\vartheta : \Theta \to \Theta'$ must map, e.g., every binary Θ-function symbol f to a binary Θ'-function *symbol*, whereas symbol-to-expression maps allow mapping f to a binary *function*, e.g., a λ-expression of the right type. An intermediate option was recently explored in [5]: Here theory morphisms map function symbols to expressions and predicate symbols to atomic expressions, and $\mathbf{Mod}(\vartheta)$ is proved to be a functor.

Interestingly, the restriction to symbol-to-symbol maps does not seem to have been motivated by the problems we described above. Algebraic specification languages tend to be based on first-order logic, where it is anyway more convenient to work only with symbol-to-symbol maps. Therefore, individual researchers in the field[3] may falsely believe that algebraic specification languages can be easily extended to allow symbol-to-expression maps.

Different choices of model morphisms in institutions are usually considered only by switching to a different institution. For example, [4] discusses the method of diagrams for a few institutions that differ only in the choice of model morphisms.

On the other hand, in type theory, theory morphisms (if they are used at all) routinely allow symbol-to-expression maps. This is because type theories tend to be based on λ-calculi, where symbol-to-expression maps are much more elegant. This is the case, for example, for the proof assistants Isabelle [11] and Coq [3] and the logical framework Twelf [14].

These languages do not encounter the problems described above because they do not consider model morphisms in the first place. There are two reasons for this. Firstly, the respective communities tend to be less interested in model theory to begin with. Secondly, for λ-calculi, model morphisms do not work very well at all, and logical relations going back to [13] usually have to be used instead.

Contribution. We introduce a general formalism for specifying the properties of not only the models but also of the model morphisms in $\mathbf{Mod}(\Theta)$. The key innovation is to allow theories to contain what we call *maxioms*[4] in analogy to axioms: Just like axioms specify the intended models, the maxioms specify the intended model morphisms.

Our approach provides an elegant solution of both of our motivating problems: It allows users to fine-tune the morphisms when specifying model categories, and it guarantees that every theory morphism ϑ yields a functor $\mathbf{Mod}(\vartheta)$.

Overview. We introduce a general framework for developing first-order style logics in Sect. 2. This allows us to later instantiate our results for different logics. It also has the added benefit to help us understand the limitations of our results by asking to which logics they do not apply.

Then Sect. 2 develops the syntax, proof theory, and model theory of our logics excluding model morphisms. Building on this, Sect. 3 discusses the technical problems regarding model morphisms in more detail. This discussion leads to our

[2] Note that syntax (formulas and proofs) and semantics (models) are translated in opposite directions. We follow the convention of algebraic specification that the $\to$ in $\Theta \to \Theta'$ indicates the direction of *syntax* translation. Readers that are used to working with model translations (e.g., forgetful functors or ML-style functors) may prefer flipping these arrows.

[3] Including until recently the author of this paper.

[4] We will systematically form new words by prepending the letter *m* in order to emphasize the symmetry between existing and new concepts.

Please cite this article in press as: F. Rabe, Morphism axioms, Theoret. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.tcs.2017.07.006