Morphism Axioms

Florian Rabe[1 a]

[a] *Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany, f.rabe@jacobs-university.de*

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.

Keywords: logic, specification, institution, theory morphism, model morphism, functorial

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

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

initial models, product models, submodels, and quotient models. For example, we want the theory `Group` to give rise to the category **Mod**(`Group`) of groups and group homomorphisms, and **Mod**(`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 **Mod**(Θ) is indeed a useful category.

However, typically the author of Θ has only indirect and limited control over the choice of morphisms in **Mod**(Θ). 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 **Mod**(Θ): While theory authors can add axioms to Θ to change the class of models, they have no direct influence on the morphisms.

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 [GB92]—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 [SW83, FGT92].

However, not every theory morphism is well-behaved with respect to model morphisms. While **Mod**(ϑ) always reduces Θ'-models to Θ-models, not every Θ'-model morphism can be reduced to a Θ-model morphism. Thus, **Mod**(ϑ) 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.

[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.

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 [GWM+93] or CASL [CoF04] 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 [Dia16]: 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, [Dia08] 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 [Pau94] and Coq [Coq15] and the logical framework Twelf [RS09].

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 [Rey74] 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.

[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.

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 solution in Sect. 4, which presents our main results. Specifically, we instantiate our framework with plain, typed, and polymorphic first-order logic. Sect. 5 discusses further generalizations for subtypes and partial functions.

2. A Framework for First-Order Logics

We give a systematic definition of the syntax, proof theory, and model theory of first-order logic. Our languages fixes the general shape of formulas and models: the formulas are the ones of typed first-order logic, and models interpret types as sets and terms as elements. But it abstracts from other language features such as type and proof system.

Following the author's treatment of logical frameworks in [Rab14], it is possible to give a more general definition that also abstracts from the shape of formulas and models. That would allow, e.g., to treat modal logic as an incarnation of our language. We do not do that here and instead stay close to the concepts and notations of typed first-order logic, for which model morphisms are particularly important. However, several of our notations are inspired by such general treatments, in particular we use the theory morphisms from [Rab14] and the variable binding of LF[5] [HHP93].

2.1. Syntax

2.1.1. Theories and Their Expressions

The syntax of our logic distinguishes four ontological concepts: **types** A, **terms** T, **formulas** F, and **proofs** P, which are collectively called **expressions**. **Theories** Θ can contain four different kinds of declarations, one for each concept: type symbols a, function symbols t, predicate symbols φ, and axioms a, which are collectively called **symbols**. Similarly, **contexts** Γ declare **variables** of each concept. The following table gives an overview:

[5] Readers familiar with λ-calculi should read the notations $\{x : A\}B$ and $[x : A]T$, which we introduce below, as $\Pi x : A.B$ and $\lambda x : A.T$, respectively.

Symbol Declaration		Expressions	Judgment
type	$a : \{\Gamma\}\mathtt{tp}$	types A	$\Gamma \vdash_\Theta A : \mathtt{tp}$
function	$t : \{\Gamma\}A$	terms T of type A	$\Gamma \vdash_\Theta T : A$
predicate	$\varphi : \{\Gamma\}\mathtt{form}$	formulas F	$\Gamma \vdash_\Theta F : \mathtt{form}$
axiom	$a : \{\Gamma\}F$	proofs P of formula F	$\Gamma \vdash_\Theta P : F$
generic case to unify all of the above			
symbol	$c : \{\Gamma\}C$	expressions E	$\Gamma \vdash_\Theta E : C$

Before giving the precise definition, we introduce our running example to explain our notations:

Example 2.1. The theory `SemiGroup` consists of

- a type declaration: $u : \{\}\mathtt{tp}$
- a binary function symbol taking inputs x and y of type u and returning a term of type u

$$\circ : \{x : u, y : u\}u$$

- an axiom

$$\mathtt{assoc} : \{\}\ \forall x : u.\forall y : u.\forall z : u.x \circ (y \circ z) \doteq_u (x \circ y) \circ z$$

where we write $\circ$ as infix for readability.

The context Γ declares the arguments of each symbol: $\circ$ takes two arguments whereas u and `assoc` take none. Types with arguments allow for type operators, and axioms with arguments allow for axiom schemata.

In the sequel, we may omit $\{\Gamma\}$ if Γ is empty, i.e., if a symbol takes no arguments.

Expressions are well-formed relative to a theory Θ and a context Γ. It is convenient to write all well-formedness judgments as typing judgments. Therefore, we use the special symbols `tp` for the universe of all types and `form` for the universe of all formulas. Terms are typed by types, and proofs are typed by the formulas that they prove. We will write c and E for arbitrary symbols and expressions (i.e., type, term, formula, or proof), and C stands for any type A, any formula F, or either of the universes `tp` and `form`. Then all well-formedness judgments are of the form $\Gamma \vdash_\Theta E : C$.

The formal definition of the above is as follows:

Definition 2.2 (Theories)**.** A **theory** is a list of declarations $D_1, \ldots, D_m$ where each D_i is of the form $c : \{\Gamma\}C$ such that

- c is a unique name for the declared symbol, i.e., the names declared by D_i and D_j are different if $i \neq j$,
- Γ is a context over the theory $D_1, \ldots, D_{i-1}$,
- C is one of the four cases described below.

Γ declares the arguments of c, and C describes what c returns.

A **context** Γ over a theory Θ is a list of declarations $x_1 : C_1, \ldots, x_n : C_n$ such that Θ, Γ is a theory. The x_i are called **variables**.

Based on the shape of C, we distinguish four kinds of **declarations** $c : \{\Gamma\}C$ in a theory or context:

- $C = \mathtt{tp}$: c is a **type symbol** and returns a new type.
- $C = A$ for some $\Gamma \vdash_\Theta A : \mathtt{tp}$: c is a **function symbol** and returns a term of type A.
- $C = \mathtt{form}$: c is a **predicate symbol** and returns a formula.
- $C = F$ for some $\Gamma \vdash_\Theta F : \mathtt{form}$: c is an **axiom**[6] and returns a proof of the formula F.

Symbol declarations in theories may take parameters (as declared by Γ), whereas variables may not. This gives our logic its first-order flavor: For example, we can declare function symbols in theories, but we cannot quantify over them because we cannot declare them as variables.

Expressions are formed inductively from the declared symbols and variables:

Definition 2.3 (Expressions)**.** Types, terms, formulas, and proofs are formed according to the following grammar

A	$::=$	$x \mid a(E_1, \dots, E_n)$	atomic types
T	$::=$	$x \mid t(E_1, \dots, E_n)$	atomic terms
F	$::=$	$x \mid \varphi(E_1, \dots, E_n)$	atomic formulas
	$\mid$	$F \wedge F' \mid F \vee F' \mid F \Rightarrow F' \mid \neg F \mid F \Leftrightarrow F'$	propositional logic
	$\mid$	$T \doteq_A T$	typed equality
	$\mid$	$\forall x : A.F(x) \mid \exists x : A.F(x)$	typed quantification
P	$::=$	$x \mid p(E_1, \dots, E_n)$	atomic proofs
	$\mid$	as in Fig. 1	natural deduction proofs

In examples, we will omit the type arguments of equality and quantifiers if they can be inferred.

All four kinds share the well-formedness rules for atomic expressions:

$$\frac{c : \{x_1 : C_1, \dots, x_n : C_n\}C \text{ in } \Theta \qquad \Gamma \vdash_\Theta E_i : C_i(\vec{E}) \text{ for } i = 1, \dots, n}{\Gamma \vdash_\Theta c(E_1, \dots, E_n) : C(\vec{E})}$$

$$\frac{x : C \text{ in } \Gamma}{\Gamma \vdash_\Theta x : C}$$

Here each x_i may occur in $C_{i+1}, \dots, C_n, C$, and we write $C(\vec{E})$ for the result of substituting each x_i with E_i.

We omit the straightforward well-formedness rules for non-atomic formulas.

The well-formedness rules for non-atomic proofs are given in Fig. 1. There we use the usual notation $\Gamma \vdash_\Theta F$ in proof rules, i.e., we omit the proof expression P in the judgment $\Gamma \vdash_\Theta P : F$. It is straightforward to form proof expressions from the names of the proof rules.

[6] Because we allow parameters, we should technically speak of an *axiom schema*, but we will use the word *axiom* for brevity.

Definition 2.4 (Substitution)**.** We write $E[x_1/E_1, \ldots, x_n/E_n]$ for the result of (capture-avoiding) **substitution** of E_i for x_i in E.

If the free variables in E are clear from the context, we also write this as $E(E_1, \ldots, E_n)$.

	Introduction	Elimination
$\wedge$	$\frac{\Gamma\vdash_\Theta F \quad \Gamma\vdash_\Theta G}{\Gamma\vdash_\Theta F\wedge G}$ `conjI`	$\frac{\Gamma\vdash_\Theta F\wedge G}{\Gamma\vdash_\Theta F}$ `conjEl` $\frac{\Gamma\vdash_\Theta F\wedge G}{\Gamma\vdash_\Theta G}$ `conjEr`
$\vee$	$\frac{\Gamma\vdash_\Theta F}{\Gamma\vdash_\Theta F\vee G}$ `disjIl` $\frac{\Gamma\vdash_\Theta G}{\Gamma\vdash_\Theta F\vee G}$ `disjIr`	$\frac{\Gamma\vdash_\Theta F\vee G \quad \Gamma,p{:}F\vdash_\Theta H \quad \Gamma,p{:}G\vdash_\Theta H}{\Gamma\vdash_\Theta H}$ `disjE`
$\rightarrow$	$\frac{\Gamma,p{:}F\vdash_\Theta G}{\Gamma\vdash_\Theta F\rightarrow G}$ `implI`	$\frac{\Gamma\vdash_\Theta F\rightarrow G \quad \Gamma\vdash_\Theta F}{\Gamma\vdash_\Theta G}$ `implE`
$\Leftrightarrow$	$\frac{\Gamma,p{:}F\vdash_\Theta G \quad \Gamma,p{:}G\vdash_\Theta F}{\Gamma\vdash_\Theta F\Leftrightarrow G}$ `equivI`	$\frac{\Gamma\vdash_\Theta F\Leftrightarrow G \quad \Gamma\vdash_\Theta F}{\Gamma\vdash_\Theta G}$ `equivEl` $\frac{\Gamma\vdash_\Theta F\Leftrightarrow G \quad \Gamma\vdash_\Theta G}{\Gamma\vdash_\Theta F}$ `equivEr`
$\neg$	$\frac{\Gamma,p{:}F,f{:}\mathtt{form}\vdash_\Theta f}{\Gamma\vdash_\Theta \neg F}$ `negI`	$\frac{\Gamma\vdash_\Theta \neg F \quad \Gamma\vdash_\Theta F \quad \Gamma\vdash_\Theta H{:}\mathtt{form}}{\Gamma\vdash_\Theta H}$ `negE`
`true`	$\frac{}{\Gamma\vdash_\Theta \mathtt{true}}$ `trueI`	
`false`		$\frac{\Gamma\vdash_\Theta \mathtt{false} \quad \Gamma\vdash_\Theta H{:}\mathtt{form}}{\Gamma\vdash_\Theta H}$ `falseE`
$\forall$	$\frac{\Gamma,x{:}A\vdash_\Theta F \quad x\notin\Gamma}{\Gamma\vdash_\Theta \forall x{:}A.\,F}$ `forallI`	$\frac{\Gamma\vdash_\Theta \forall x{:}A.\,F \quad \Gamma\vdash_\Theta T{:}A}{\Gamma\vdash_\Theta F[x/T]}$ `forallE`
$\exists$	$\frac{\Gamma\vdash_\Theta F[x/T] \quad \Gamma\vdash_\Theta T{:}A}{\Gamma\vdash_\Theta \exists x{:}A.\,F}$ `existsI`	$\frac{\Gamma\vdash_\Theta \exists x{:}A.\,F \quad \Gamma,x{:}A,p{:}F\vdash_\Theta H \quad x\notin\Gamma,H}{\Gamma\vdash_\Theta H}$ `existsI`
$\doteq$	$\frac{\Gamma\vdash_\Theta T{:}A}{\Gamma\vdash_\Theta T\doteq_A T}$ `equalI`	$\frac{\Gamma\vdash_\Theta E(S){:}C(S) \quad \Gamma\vdash_\Theta S\doteq_A T}{\Gamma\vdash_\Theta E(T){:}C(T)}$ `equalE`

Figure 1: Natural Deduction Rules

Examples and Remarks. We can obtain several logics as special cases:

Example 2.5 (Variant Logics). Untyped intuitionistic first-order logic (FOL) arises as the special case where

- there is a single fixed type symbol u : `tp` and other type declarations are not allowed,
- contexts may declare only term variables.

Thus, all FOL-contexts are of the form $x_1 : u, \ldots, x_n : u$. `SemiGroup` from Ex. 2.1 is an example of a FOL-theory.

Typed first-order logic (TFOL) arises as the special case where

- type declarations must be of the form $a : \mathtt{tp}$ (i.e., types may not take parameters),
- contexts may declare only term variables.

Thus, all TFOL-contexts are of the form $x_1 : a_1, \ldots, x_n : a_n$.

Typed first-order logic with polymorphism (PFOL), which has recently received more systematic attention [BP13], arises as the special case where

- type declarations must be of the form $a : \{x_1 : \mathtt{tp}, \ldots, x_n : \mathtt{tp}\}\mathtt{tp}$ (i.e., n-ary type operators),
- contexts may only declare type and term variables.

Thus, up to reordering, all PFOL-contexts are of the form $x_1 : \mathtt{tp}, \ldots, x_m : \mathtt{tp}, y_1 : A_1, \ldots, y_n : A_n$.

All of the above logics can be classical or intuitionistic. The classical variant arises by adding the axiom $\mathtt{classical} : \{x : \mathtt{form}\}x \vee \neg x$.

Example 2.6 (Continuing Ex. 2.1). There are two equivalent ways to extend the theory SemiGroup to the theory of monoids. Let $\mathtt{isunit}(x)$ abbreviate $\forall y : u.y \circ x \doteq_u y \wedge x \circ y \doteq_u y$. The theory MonoidCon uses a constant for the unit

$$e : u,\ \mathtt{neut} : \mathtt{isunit}(e)$$

The theory MonoidAx uses an axiom for the existence of a unit

$$\mathtt{unit} : \exists x : u.\, \mathtt{isunit}(x)$$

It is well-known that these two theories induce isomorphic model *classes*. However, as we will see in Ex. 2.12, they do not induce isomorphic model *categories* in standard first-order logic.

Remark 2.7 (Empty Types). Our logic allows types to be empty. This is important for applications in mathematics, where many important theories naturally have empty models, e.g., the theory of sets or the theory of orders. This breaks with the convention of standard first-order logic that all types are non-empty, i.e., that $\exists x : A.\mathtt{true}$ is a theorem.

For readers not familiar with this effect, this may be surprising because our natural deduction rules appear to be exactly the ones of standard first-order logic. But our proof rules subtly deviate from standard first-order logic by using the assumption $\Gamma \vdash_\Theta T : A$ in the rules forallE and existsI. If we dropped these assumptions, we could instantiate the rules with variables not in Γ. That is equivalent to assuming that all types are non-empty.

If we want to recover the standard convention, we can add an axiom $\mathtt{nonempty} : \{y : \mathtt{tp}\}\exists x : y.\, \mathtt{true}$.

Remark 2.8 (Separating Formulas from Terms and Types). Our logic uses a layered approach where types and terms are separated from formulas and proofs. Many logics use a simpler ontology where formulas are special cases of terms or types.

For example, higher-order logic assumes $\vdash \texttt{form} : \texttt{tp}$, i.e., `form` is a special type and all formulas are special cases of terms. Similarly, type theories following the propositions-as-types paradigm assume $\texttt{form} = \texttt{tp}$, i.e., formulas and proofs are the same, respectively, as types and terms.

All statements of Sect. 2 easily specialize to those simpler ontologies. However, these simpler ontologies do not interact well with model morphisms. The reasons are subtle, and we will come back to this in Sect. 3. Essentially, when working with model morphisms, we have to treat formulas differently than terms and types. Therefore, we strictly segregate them from the beginning.

2.1.2. Theory Morphisms and Their Action on Expressions

Our logic allows for a very general definition of theory morphisms $\Theta \to \Theta'$. These are compositional translations, which map

- Θ-contexts to Θ'-contexts,
- Θ-expressions to Θ'-expressions.

Definition 2.9 (Theory Morphisms)**.** Consider two theories Θ and Θ'.

A **theory morphism** $\vartheta : \Theta \to \Theta'$ contains
for every declaration $c : \{\Gamma\}C$ in Θ
exactly one assignment $c \mapsto [\vartheta(\Gamma)]E$
such that $\vartheta(\Gamma) \vdash_{\Theta'} E : \vartheta(C)$.

The **homomorphic extension** $\vartheta(-)$ is defined as follows: To obtain $\vartheta(X)$
replace every occurrence of an atomic expression $c(E_1, \ldots, E_n)$ in X
where $c \mapsto [\vartheta(\Gamma)]E$ in ϑ
with $E(E_1, \ldots, E_n)$.

The key property of theory morphisms is that they preserve all judgments:

Theorem 2.10 (Judgment Preservation)**.** *Given a theory morphism $\vartheta : \Theta \to \Theta'$, then*

$$\Gamma \vdash_\Theta E : C \qquad \text{implies} \qquad \vartheta(\Gamma) \vdash_{\Theta'} \vartheta(E) : \vartheta(C)$$

Note that Thm. 2.10 contains as a special case the preservation of theorems: If F is provable in Θ, then $\vartheta(F)$ is provable in Θ'. Judgment preservation holds for a wide variety of formal systems—a proof for a very general case that subsumes the logic we use here can be found in [Rab14].

The theories and theory morphisms form a category:

Definition 2.11 (Category of Theories)**.** We obtain the category of theories and theory morphisms as follows:

- The identity morphism id_Θ is given by $c \mapsto [\vartheta(\Gamma)]c(x_1, \ldots, x_n)$ for every $c : \{x_1 : C_1, \ldots, x_n : C_n\}C$ of Θ.
- For two theory morphisms $\vartheta : \Theta \to \Theta'$ and $\vartheta' : \Theta' \to \Theta''$, the composition $\vartheta' \circ \vartheta$ is given by $c \mapsto [\vartheta'(\vartheta(\Gamma))]\vartheta'(\vartheta(c(x_1, \ldots, x_n)))$ for every $c : \{x_1 : C_1, \ldots, x_n : C_n\}C$ of Θ.

Example 2.12 (Continuing Ex. 2.6). We can give a theory morphism `axCon` : `MonoidAx` $\to$ `MonoidCon`. It maps all symbols to themselves except for the axiom `unit`. It maps `unit` to a `MonoidCon`-proof by `unit` $\mapsto$ `existsI(e, neut)`.

`axCon` is essentially an isomorphism in the category of theories and morphisms. The only caveat is that first-order logic is not expressive enough to write down the inverse morphism. This limitation is not critical for this example, and `axCon` becomes an isomorphism if we extend the expressivity of first-order logic. For example we can add a description operator ι that obeys the rule

$$\frac{\Gamma, x : A \vdash_\Theta F : \texttt{form} \qquad \Gamma \vdash_\Theta \exists^1 x : A.F}{\Gamma \vdash_\Theta \iota x : A.F : A}$$

(where $\exists^1$ is the quantifier of unique existence) as well as the axiom that F holds for $\iota x : A.F$.

Then we can map the constant `e` of `MonoidCon` to the uniquely determined unit element:

$$\texttt{e} \mapsto \iota x : u.\, \texttt{isunit}(x)$$

The occurrence of the description operator here is indeed well-typed because we can give a proof of $\exists^1 x : u.\, \texttt{isunit}\, x$ in `MonoidAx`. This yields an isomorphism `MonoidCon` $\to$ `MonoidAx` that is the inverse of `axCon`.

Remark 2.13 (Equality of Theory Morphisms). We have some flexibility when defining the equality of theory morphisms: while our definition distinguishes any two morphisms that differ syntactically, it is often desirable to quotient the theory morphisms by provable equality.

Two theory morphisms ϑ and ϑ' are *provably equal* if they map all Θ-declarations to provably equal expressions. And two expressions are *provably equal* under the following conditions:

- types: if they are syntactically equal,
- terms t and t' of the same type A: if $t \doteq_A t'$ is provable,
- formulas F and F': if $F \Leftrightarrow F'$ is provable,
- proofs of the same formula: always (proof-irrelevance).

Provably equal expressions are interpreted equally by any sound model theory. Therefore, the distinction between syntactic and provable equality can usually be ignored. In this paper, the distinction only affects one aspect: the use of the phrase *theory isomorphism*. In practice, the composition of a theory morphism and its intended inverse is usually only provably equal but not syntactically equal to the identity morphism. For example, in Ex. 2.12, `axCon` will only be an isomorphism up to provable equality.

We introduce one more running example:

Example 2.14. Consider the theories

$$\texttt{Special} = u : \texttt{tp},\ \texttt{sp} : \{x : u\}\texttt{form}$$

of a unary predicate on a type u and

$$\texttt{Order} = u : \texttt{tp},\ \texttt{leq} : \{x : u, y : u\}\texttt{form}, \ldots$$

of orders on a type u (where we omit the axioms).

We define a theory morphism $\texttt{least} : \texttt{Special} \to \texttt{Order}$ by

$$\texttt{least} = u \mapsto u,\ \texttt{sp} \mapsto [x : u]\forall y : u.\,\texttt{leq}(x, y)$$

It interprets the predicate of being special as the property of being the least element.

We will see in Ex. 3.8 that the standard definition of first-order logic does not yield a model reduction functor for this theory morphism.

2.2. Semantics

Our primary interest is in *classical* logics. Therefore, we give a bivalent semantics, i.e., the formula `classical` from Ex. 2.5 holds in all models. Nonetheless, it is interesting to generalize our results to logics like intuitionistic and many-valued logics. While we do not do this generalization here, our formulations try to anticipate it, and Rem. 3.21 gives some pointers what changes would have to be made.

2.2.1. Models and Their Interpretation Functions

Before concisely stating our general definition of models, it is instructive to recall the general structure of how models are defined for the simple case of typed first-order logic (TFOL):

Remark 2.15 (Standard Definition of Models). For an TFOL-theory Θ, a model M

1. (a) provides a set a^M for each Θ-type symbol $a : \texttt{tp}$,
   (b) induces (by induction on types A) a set A^M for every type A (which is trivial for TFOL because the type symbols are the only types),
2. (a) provides an interpretation for each Θ-function symbol $t : \{x_1 : A_1, \ldots, x_n : A_n\}a$ as a mapping

$$t^M : (A_1^M \times \ldots \times A_n^M) \to A^M$$

(b) induces (by induction on terms T) the interpretation $T^{M,\alpha} \in A^M$ under assignment α for every term T of type A (where an assignment $(\alpha_1, \ldots, \alpha_n)$ maps variables $x_i : A_i$ to values $\alpha_i \in A_i^M$),

3. (a) provides an interpretation for each Θ-predicate symbol $\varphi : \{x_1 : A_1, \ldots, x_n : A_n\}\mathtt{form}$ as a mapping

$$\varphi^M : (A_1^M \times \ldots \times A_n^M) \to \{0, 1\}$$

(b) induces (by induction on formulas F) the interpretation $F^{M,\alpha} \in \{0, 1\}$ under assignment α for every formula F,

4. (a) must satisfy each Θ-axiom symbol $a : \{\Gamma\}F$, i.e., $F^{M,\alpha} = 1$ for all assignments α

(b) is shown to satisfy (by induction on proofs P) every Θ-theorem F with proof P.

Our definition of model follows quite straightforwardly from applying these principles. The only subtlety is that we use a particular treatment of assignments that allows for a very concise definition:

Remark 2.16 (Treatment of Assignments). We interpret every context as the set of assignments for its variables. Thus, $(x_1 : A_1, \ldots, x_n : A_n)^M$ is the set of assignments $\alpha = (\alpha_1, \ldots, \alpha_n)$ that assign α_i to x_i.

We will also write $\alpha.e$ for the tuple $(\alpha_1, \ldots, \alpha_n, e)$.

Then we are ready to state our definitions of models and interpretation function:

Definition 2.17 (Models)**.** Given a theory Θ, the class $\mathbf{Mod}(\Theta)$ of **models** contains tuples $(\ldots, c^M, \ldots)$ providing for every Θ-symbol $c : \{\Gamma\}C$ a denotation c^M that maps $\alpha \in \Gamma^M$ to $c^M(\alpha) \in C^{M,\alpha}$.

Every such model induces an **interpretation function** that maps
- every Θ-context Γ to the set Γ^M of assignments for Γ,
- for every assignment $\alpha \in \Gamma^M$, every expression $\Gamma \vdash_\Theta E : C$ to its denotation $E^{M,\alpha} \in C^{M,\alpha}$

and is defined as follows:
- for contexts $\Gamma = x_1 : C_1, \ldots, x_n : C_n$, assignments α to M are n-tuples and
$$(\alpha_1, \ldots, \alpha_n) \in \Gamma^M \quad \text{iff} \quad \alpha_i \in C_i^{M,(\alpha_1,\ldots,\alpha_{i-1})}$$
- for the universes
$$\mathtt{tp}^{M,\alpha} = \mathcal{SET}$$
$$\mathtt{form}^{M,\alpha} = \{0, 1\}$$
- for variables x_i declared in the context
$$x_i^{M,\alpha} = \alpha_i$$

- for atomic expressions $E = c(E_1, \ldots, E_n)$

$$E^{M,\alpha} = c^M(E_1^{M,\alpha}, \ldots, E_n^{M,\alpha})$$

- for non-atomic expressions see Def. 2.20.

Definition 2.18 (Satisfaction)**.** As usual, we write $M \models F$ if $F^{M,\alpha} = 1$ for all assignments α to the context of F and say that M **satisfies** F.

Remark 2.19 (Interpretation of Proofs). Proofs of F are interpreted as elements of the interpretation of F. This corresponds to the *formulas as types* interpretation where every formula serves as the type of its proofs.

A formula is interpreted as either 0 or 1, and we assume the usual definitions of $0 := \varnothing$ and $1 := \{0\}$, i.e., the interpretation of F is either empty or a singleton. Consequently, the interpretation of a proof $\vdash_\Theta P : F$ is either impossible or uniquely determined. Thus, well-definedness of the interpretation function subsumes the soundness of the proof calculus: There may only ever be proofs of F if F is satisfied by every model M.

This corresponds to the *proof irrelevance* interpretation: It matters only that a proof can be interpreted, but its interpretation is irrelevant (because it is uniquely determined). In particular, even though a model M must provide an interpretation p^M of every Θ-axiom p, only the existence of p^M is relevant and two models can never differ in the choice of p^M. Therefore, when giving concrete models, we can drop the interpretations p^M from the notation.

Def. 2.17 does not define the interpretation of non-atomic expressions. These can be supplied separately depending on the choice of non-atomic expressions. For all expressions from Def. 2.3, the interpretation is straightforward:

Definition 2.20 (Interpretation Function)**.** We define the interpretation of non-atomic expressions in the usual way

$$\forall x : A.F(x)^{M,\alpha} = \begin{cases} 1 & \text{if } F^{M,\alpha.e} = 1 \text{ for all } e \in A^{M,\alpha} \\ 0 & \text{otherwise} \end{cases}$$

$$S \doteq_A T^{M,\alpha} = \begin{cases} 1 & \text{if } S^{M,\alpha} = T^{M,\alpha} \\ 0 & \text{otherwise} \end{cases}$$

and accordingly for the other non-atomic formulas. The interpretation of all non-atomic proofs is uniquely determined: All proofs are interpreted as the element of $\{0\}$.

2.2.2. The Action of Theory Morphisms on Models

The following definition provides the semantic analogue to Def. 2.9. Just like theory morphisms map expressions in one direction, they map models in the opposite direction:

Definition 2.21 (Model Reduction)**.** Consider a theory morphism $\vartheta : \Theta \to \Theta'$.

We define $\mathbf{Mod}(\vartheta) : \mathbf{Mod}(\Theta') \to \mathbf{Mod}(\Theta)$ as follows. We consider $M' \in \mathbf{Mod}(\Theta')$ and define $M = \mathbf{Mod}(\vartheta)(M') \in \mathbf{Mod}(\Theta)$ by:
For every declaration $c : \{\Gamma\}C$ in Θ,
if $c \mapsto [\vartheta(\Gamma)]\vartheta(E)$ is the corresponding assignment in ϑ,
then $c^M(\alpha) = E^{M',\alpha}$ for every $\alpha \in \Gamma^M$.

The well-definedness of Def. 2.21 is proved together with the main theorem about model reduction, which forms the semantic analogue to Thm. 2.10:

Theorem 2.22 (Denotation Condition)**.** *Given a theory morphism $\vartheta : \Theta \to \Theta'$ and an expression $\Gamma \vdash_\Theta E : C$, we have for every model $M' \in \mathbf{Mod}(\Theta')$ and every assignment α from $\vartheta(\Gamma)$ to M':*

$$\vartheta(\Gamma)^{M'} = \Gamma^{\mathbf{Mod}(\vartheta)(M')}$$

$$\vartheta(E)^{M',\alpha} = E^{\mathbf{Mod}(\vartheta)(M'),\alpha}.$$

Proof. The straightforward proof proceeds by induction on expressions. Proofs of this form are well-known from the study of institutions [GB92] (whose satisfaction condition is a special case of our denotation condition). □

Thus, the reduced model $\mathbf{Mod}(\vartheta)(M')$ is essentially the same mathematical structure as M'. It only differs by being framed as a model of Θ according to ϑ.

Example 2.23 (Continuing Ex. 2.14). A model $N' \in \mathbf{Mod}(\mathtt{Order})$ is given by $u^{N'} = \mathbb{N}$, $\mathtt{leq}^{N'} = \leq$ (and $p^{N'} = 0$ for every axiom p). Model reduction yields $N = \mathbf{Mod}(\mathtt{least})(N')$ with $u^N = \mathbb{N}$ and $\mathtt{sp}^N : n \mapsto \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases}$

Another model $Z' \in \mathbf{Mod}(\mathtt{Order})$ is given by $u^{Z'} = \mathbb{Z}$, $\mathtt{leq}^{Z'} = \leq$. Model reduction yields $Z = \mathbf{Mod}(\mathtt{least})(Z')$ with $u^Z = \mathbb{Z}$ and $\mathtt{sp}^Z : n \mapsto 0$.

Thus, **Mod** is a functor from the category of theories and theory morphisms to the category of classes and mappings. Our work is motivated by the observation that functoriality fails if we naively add model morphisms to $\mathbf{Mod}(\Theta)$.

3. Motivating Considerations

3.1. Limitations of Standard Model Morphisms

Before stating our definition of model morphism, let us recall the standard definition for the special case of TFOL in analogy to Rem. 2.15:

Remark 3.1 (Standard Definition of Model Morphisms). For a TFOL-theory Θ, a model morphism $m : M \to M'$ between Θ-models

1. (a) provides a function $a^m : a^M \to a^{M'}$ for each Θ-type symbol a,
   (b) induces (by induction on types A) a function $A^m : A^M \to A^{M'}$ for every type A (which is trivial for TFOL because the type symbols are the only types),
2. (a) must commute with the interpretation of each Θ-function symbol $t : \{x_1 : A_1, \ldots, x_n : A_n\}a$, i.e.,

$$a^m\big(t(x_1, \ldots, x_n)^{M,\alpha}\big) = t(x_1, \ldots, x_n)^{M',m(\alpha)}$$

   (b) is shown (by induction on terms T) to commute with the interpretation of every Θ-term T,
3. (a) must preserve the interpretation of each Θ-predicate symbol φ : $\{x_1 : A_1, \ldots, x_n : A_n\}\texttt{form}$, i.e.,

$$a^m\big(\varphi(x_1, \ldots, x_n)^{M,\alpha}\big) \le \varphi(x_1, \ldots, x_n)^{M',m(\alpha)}$$

   (b) cannot be shown by induction on formulas F to preserve the interpretation of all Θ-formulas,
4. (a) does nothing with Θ-axioms,
   (b) has no relation with Θ-theorems.

Note that (3b) already fails if negation is present in the logic (see also Rem. 3.5).

To understand the above notations, we have to explain how model morphisms treat assignments in analogy to Rem. 3.1:

Remark 3.2 (Treatment of Assignments). A model morphism m maps every M-assignments α for Γ component-wise to an M'-assignment for Γ. Formally, we define a function $\Gamma^m : \Gamma^M \to \Gamma^{M'}$ as follows: Let $\Gamma = \ldots, x_i : A_i, \ldots$. Then $\Gamma^m(\ldots, \alpha_i, \ldots) = (\ldots, A_i^m(\alpha_i), \ldots)$,

We abbreviate $\Gamma^m(\alpha)$ as $m(\alpha)$ if Γ is clear.

Rem. 3.1 indicates a broken symmetry between models and morphisms: The cases (3b) and (4) do not extend to model morphisms. To understand this better, we discuss the relation between model morphisms and formulas.

The ideal but unrealistic invariants of model morphisms are concisely captured by the following table:

Expression	Interpretation by	
	model M	morphism $m : M \to M'$
Type A	set $A^{M,\alpha}$	function $A^{m,\alpha} : A^{M,\alpha} \to A^{M',\alpha}$
Term T of A	elements $T^{M,\alpha} \in A^{M,\alpha}$	commutativity $A^{m,\alpha}(T^{M,\alpha}) = T^{M',m(\alpha)}$
Formula F	truth value $F^{M,\alpha} \in \{0, 1\}$	preservation $F^{M,\alpha} \le F^{M',m(\alpha)}$
Proof P of F	uniquely determined	commutativity trivial

The upper half of the table works well. Models interpret types as sets and terms as elements of these sets. Model morphisms interpret types as functions between these sets that commute with the interpretation of terms.

Example 3.3 (First-Order Logic). Recall FOL from Ex. 2.5, which has a single type u. Thus, a model morphism m must provide a single function $u^m : u^M \to u^{M'}$. For a function symbol $t : \{x_1 : u, \ldots, x_n : u\}u$, the commutativity conditions becomes the well-known $u^m(t^M(e_1, \ldots, e_n)) = t^{M'}(u^m(e_1), \ldots, u^m(e_n))$.

For example, for $n = 2$ and $t = \circ$, this yields the condition for semigroup homomorphisms: $u^m(e_1 \circ^M e_2) = u^m(e_1) \circ^{M'} u^m(e_2)$.

We might now try to generalize these results to the lower half of the table. Ideally, a model morphism would provide a function $F^{M,\alpha} \to F^{M',m(\alpha)}$ for every formula F. Using $0 = \varnothing$ and $1 = \{0\}$, we see that such a function exists uniquely or not at all. It exists iff the morphism preserves F, i.e., if $F^{M,\alpha} = 1$ implies $F^{M',m(\alpha)} = 1$. We can write this concisely as $F^{M,\alpha} \leq F^{M',m(\alpha)}$. In that case, the commutativity condition for proofs is trivial because the interpretation of proofs is determined uniquely anyway. However, if F is not preserved, morphisms cannot map proofs at all, and the commutativity condition cannot even be stated anymore.

Example 3.4 (First-Order Logic). For a predicate symbol $\varphi : \{x_1 : u, \ldots, x_n : u\}\mathtt{form}$, the preservation of atomic formulas becomes the well-known "if $\varphi^M(e_1, \ldots, e_n)$ then $\varphi^{M'}(u^m(e_1), \ldots, u^m(e_n))$".

For example, for $n = 2$ and $\varphi = \leq$, this yields the condition for monotonic maps between orders: if $e_1 \leq^M e_2$, then $u^m(e_1) \leq^{M'} u^m(e_2)$.

Thus, it is natural to require that all formulas be preserved by model morphisms. However—while elegant and consistent—this is impractical: It is a very strong condition that is not satisfied by many interesting maps between models. For example, the preservation of the formula $\neg x \doteq_u y$ by a model morphism m is already equivalent to the injectivity of u^m. Similarly, universally quantified formulas can usually only be preserved by surjective morphisms. But model morphisms are used extensively to describe submodels and quotient models—which are non-trivial exactly for non-surjective and non-injective morphisms, respectively.

Therefore, the preservation condition on formulas must be relaxed to be practical for mathematical applications. This leads to the standard definition described in Rem. 3.1: require preservation only for the *atomic* formulas. This choice is made, for example, in the CASL standard for first-order logic [CoF04].

Remark 3.5 (Proving Preservation Inductively). Starting with the knowledge that all atomic formulas are preserved, one might try to prove the preservation of all formulas F by induction on F. This is not possible. The induction step succeeds for equality, truth, falsity, disjunction, conjunction, and existential quantification. But it fails for negation, implication, and universal quantification.

Remark 3.6 (Segregating Formulas (following up on Rem. 2.8)). We can now explain why our logic does not treat formulas as a special case of terms: Formulas generally do not commute with model morphisms, i.e., we do not expect $F^{M,\alpha} = F^{M',m(\alpha)}$ to hold in general.

We might require this stronger property in order to allow treating terms and formulas uniformly, but that would be impractically strong. At best, we would have to restrict it to atomic formulas—the resulting model morphisms are called *closed* in [Dia08]. But many interesting model morphisms are not closed. For example, order homomorphisms are the ones that preserve (but do not necessarily commute with) atomic formulas.

The standard choice of requiring the preservation of atomic formulas is often but not necessarily the best choice. Ultimately, it is the reason for the two motivating problems described in Sect. 1: It fixes one set of morphisms when other choices might be interesting as well, and it prevents the functoriality of $\mathbf{Mod}(\vartheta)$ in the general case.

Example 3.7 (Monoid Morphisms (Continuing Ex. 2.12). Applying the standard definition of model morphisms to `MonoidCon` yields the usual monoid morphisms. But because the standard definition ignores axioms entirely, the theory `MonoidAx` induces the same model morphisms as the theory `SemiGroup`.

Thus, $\mathbf{Mod}$(`MonoidAx`) also allows semigroup morphisms between monoids that are not monoid morphisms. An example is the inclusion from $(\mathbb{N} \setminus \{0\}, \max)$ to $(\mathbb{N}, \max)$ (where max returns the greater one of two numbers): Both are monoids, and the uniquely determined units are 1 and 0, respectively. The inclusion commutes with the binary operation but not with the unit. Thus, it is a $\mathbf{Mod}$(`MonoidAx`)-morphism but not a $\mathbf{Mod}$(`MonoidCon`)-morphism.

Therefore, the categories $\mathbf{Mod}$(`MonoidAx`) and $\mathbf{Mod}$(`MonoidCon`) are not isomorphic, and $\mathbf{Mod}$(`axCon`) is not an isomorphic functor between them.

If we want to use `MonoidAx` instead of `MonoidCon`, we have no way to fine-tune the `MonoidAx`-model morphisms to exclude such morphisms. Or rather, the only way to fine-tune the morphisms is to add a constant for the unit, in which case we end up with the theory `MonoidCon`.

Example 3.8 (Continuing Ex. 2.23). We can give a model morphism $i' : N' \to Z'$ by the inclusion mapping $u^{i'} : n \mapsto n$. It preserves the ordering and thus conforms to the standard definition: If $\mathtt{leq}(x,y)^{N',(m,n)} = 1$ then $\mathtt{leq}(x,y)^{Z',i(m,n)} = 1$.

The standard definition of $\mathbf{Mod}(\vartheta)$ reduces i' to i with $u^i = u^{i'}$. i does not preserve the predicate `sp` and thus is not a standard model morphism: $\mathtt{sp}(x)^{N,(0)} = 1$ but $\mathtt{sp}(x)^{Z,i(0)} = 0$.

Thus, $\mathbf{Mod}$(`least`) is not a functor if we use the standard definition of $\mathbf{Mod}(-)$.

3.2. Interesting Properties of Model Morphisms

To understand better what kind of properties we might require of model morphisms, we discuss some properties they may or may not have.

Consider two models M and M' and a model morphism $m : M \to M'$. For the purposes of this section, it does not matter how model morphisms are defined in detail. All we need is that m induces the translation functions $\Gamma^m : \Gamma^M \to \Gamma^{M'}$ (abbreviated $m(-)$) and $A^{m,\alpha} : A^{M,\alpha} \to A^{M',\alpha}$ from above. These translation functions allow translating values and assignments in M to corresponding entities in M'.

Definition 3.9 (Preserving Formulas)**.** We say that m **preserves** $\Gamma \vdash_\Theta F : \texttt{form}$ if $F^{M,\alpha} \leq F^{M',m(\alpha)}$ for all $\alpha \in \Gamma^M$.

Definition 3.10 (Reflecting Formulas)**.** We say that m **reflects** $\Gamma \vdash_\Theta F : \texttt{form}$ if $F^{M,\alpha} \geq F^{M',m(\alpha)}$ for all $\alpha \in \Gamma^M$.

Definition 3.11 (Commuting with Formulas)**.** We say that m **commutes** with a formula $\Gamma \vdash_\Theta F : \texttt{form}$ if $F^{M,\alpha} = F^{M',m(\alpha)}$ for all $\alpha \in \Gamma^M$.

Definition 3.12 (Commuting with Terms)**.** We say that m **commutes** with a term $\Gamma \vdash_\Theta T : A$ if $A^{m,\alpha}(T^{M,\alpha}) = T^{M',m(\alpha)}$ for all $\alpha \in \Gamma^M$.

We have the following relations betweens these notions:

Theorem 3.13 (Terms)**.** *Consider a term $\Gamma \vdash_\Theta T : A$. A model morphism commutes with T iff it preserves $y \doteq_A T$ for a fresh variable $y : A$.*

Proof. Consider a model morphism $m : M \to M'$. m preserves $y \doteq_A T$ iff $(y \doteq_A T)^{M,\alpha.e} = 1$ implies $(y \doteq_A T)^{M',m(\alpha.e)} = 1$ for all assignments $\alpha.e$ for $\Gamma, y : A$. Because $m(-)$ is defined component-wise, we have $m(\alpha.e) = m(\alpha).(A^{m,\alpha}(e))$. Then the above is equivalent to $e = T^{M,\alpha}$ implies $A^{m,\alpha}(e) = T^{M',m(\alpha)}$ for all $\alpha.e$. That is equivalent to m commuting with T. □

Theorem 3.14 (Formulas)**.** *Consider a formula F. A model morphism*

- *reflects F iff it preserves $\neg F$,*
- *preserves F iff it reflects $\neg F$,*
- *commutes with F iff it commutes with $\neg F$,*
- *commutes with F iff it preserves and reflects F.*

Proof. All proofs are straightforward.

As an example, we prove the right-to-left direction of the first statement. Assume $m : M \to M'$ preserves $\neg F$ (1). We want to prove that m reflects F. So we assume $M' \models F$ (2) and have to prove $M \models F$ $(*)$. Because our models are bivalent, (2) implies $(\neg F)^{M'} = 0$. Then (1) yields $(\neg F)^M = 0$ and thus $(*)$. □

Theorem 3.15 (Proofs)**.** *Consider a proof P of F. Every model morphism commutes with P in the sense that $P^{M,\alpha} = P^{M',m(\alpha)}$ for all α.*

Proof. This holds trivially because all models interpret proofs in the same way. □

Remark 3.16 (Types). Consider a type A. Interesting model morphisms usually do not commute with A, i.e., we usually do not have $A^{M,\alpha} = A^{M',m(\alpha)}$. This is because the whole point of a model morphism is to relate two different models, i.e., two models for which A^M and $A^{M'}$ are different.

In some situations, it may be useful to speak about whether a morphism commutes with a type, but we will not consider this further here.

Considering Thm. 3.14, Thm. 3.15, and Rem. 3.16, we see that the commutation properties are most interesting for terms and can be ignored for types, formulas, and proofs. Moreover, considering Thm. 3.13 and Thm. 3.14, we see that — for classical logic — many interesting properties can be expressed in terms of preservation alone.

Example 3.17 (Equality and Injectivity). All model morphisms m preserve the formula $x \doteq_a y$ in context $x : a, y : a$. This expresses the fact that two equal values cannot be teased apart by a model morphism.

Conversely, m preserves $\neg x \doteq_a y$ iff a^m is injective.

Example 3.18 (Preserving All Formulas and Surjectivity). Model isomorphisms preserve and reflect all formulas.

More generally, elementary equivalences are defined as those model morphisms that preserve all formulas. For example the inclusion model morphism $i : (\mathbb{Q}, <) \hookrightarrow (\mathbb{R}, <)$ preserves all formulas over the theory of dense linear orders without end points.

This example also shows that we cannot specify the surjectivity of model morphisms in terms of preservation: i preserves all formulas but is not surjective. This is reminiscent of the result that we cannot specify the class of all term-generated models using any set of first-order formulas.

Example 3.19 (Continuing Ex. 3.7). We can now state the problem of Ex. 2.12 concisely. We want to specify the category containing only those `MonoidAx`-model morphisms that preserve the formula $\texttt{isunit}(x)$ in context $x : u$.

Example 3.20 (Continuing Ex. 3.8). We can now state the problem of Ex. 3.8 concisely.

The model morphism i' preserves the formula $x : u, y : u \vdash_{\texttt{Order}} \texttt{leq}(x, y) : \texttt{form}$ but not the formula $x : u \vdash_{\texttt{Order}} \forall y : u.\, \texttt{leq}(x, y) : \texttt{form}$.

Thus, the theory morphism `least` maps `sp` to a non-preserved formula. Therefore, i does not preserve $x : u \vdash_{\texttt{Special}} \texttt{sp}(x) : \texttt{form}$.

Ex. 3.19 and 3.20 indicate our solution. Just like theories may declare axioms that specify the desired properties of models, theories should be allowed to make declarations that specify the desired properties of model morphisms. We will develop this solution in Sect. 4.

Remark 3.21 (Intuitionistic Logic). Thm. 3.13 and 3.14 holds because our models are always bivalent and thus classical. In order to generalize the above results to intuitionistic logics, we would have to make the following changes.

Every model M must provide a Heyting algebra H, and we define $\texttt{form}^{M,\alpha} = H$ instead of using the boolean algebra $\{0, 1\}$. Accordingly, every model morphism $m : M \to M'$ must provide a monotone map $h : H \to H'$.

Then we change the definition of preserving formulas to use $h(F^{M,\alpha}) \leq F^{M',m(\alpha)}$, and we change the definitions of reflecting and commuting with formulas accordingly. If we put $\texttt{form}^{m,\alpha} = h$, all definition are elegantly analogous to commuting with terms: in all cases, we have an expression $\Gamma \vdash_{\Theta} E : C$ and relate $C^{m,\alpha}(E^{M,\alpha})$ to $E^{M',m(\alpha)}$.

With these changes, some theorems do not hold anymore: the right-to-left implication in Thm. 3.13 and the right-to-left implications of the items in Thm. 3.14 that involve negation do not hold. (The left-to-right directions still hold.)

Generalizations to other many-valued logics can be made using arbitrary ordered sets of truth values instead of Heyting algebras. In that case, of course, the proof calculus may have to be changed as well.

4. A Logic With Model Morphisms

In this section, we extend the basic logic defined in Sect. 2.1. Our main goal is to define model morphisms in a way that introduces flexibility while guaranteeing that every theory morphism induces a model reduction functor.

Our central idea is to add a new kind of declaration in theories: *maxiom* declarations postulate a property for all model morphisms in the same way in which axiom declarations postulate a property for all models. Just like formulas are properties of models asserted by axioms, we introduce *mormulas* as properties of model morphisms that are asserted by maxioms. Just like theorems are formulas that are implied by the axioms, we introduce *meorems* as the mormulas implied by the maxioms. Finally, just like proofs are the expressions that establish theorems, we introduce *moofs* as the expressions that establish meorems.

The following tables summarize the new concepts for model morphisms and their analogy to the existing concepts for models:

Declared symbols	Expressions	Judgment
-	mormulas V	$\Gamma \vdash_{\Theta} V : \texttt{morm}$
maxioms $q : V$	moofs Q of mormula V	$\Gamma \vdash_{\Theta} Q : V$

	Models	Model morphisms
property	formula F	mormula V
postulated property	axiom $p : F$	maxiom $q : V$
established property	theorem F	meorem V
establishing evidence	proof $\vdash_\Theta P : F$	moof $\vdash_\Theta Q : V$

The empty cell in the first table raises the question whether there should also be declarations of mormula constructors. These would be analogous to predicate symbols. Our logic could easily accommodate such declarations. But we omit them here because we have so far not found a use for such declarations.

Rem. 3.1 pointed out a lack of symmetry between the definitions of models and model morphisms. Our logic will yield the following symmetry between models and model morphisms:

- Regarding types
  - A model interprets every type as a set.
  - A model morphism interprets every type as a map.
- Regarding terms
  - A model interprets every term as an element of the set interpreting its type.
  - A model morphism may or may not commute with a term.
- Regarding formulas and mormulas
  - A model interprets every formula as a truth value.
  - A model morphism interprets every mormula as a truth value.
- Regarding proofs and moofs
  - A model must satisfy all axioms; proofs show that it also satisfies all theorems.
  - A model morphism must satisfy all maxioms; moofs show that it also satisfies all meorems.

4.1. Syntax: Mormulas and Maxioms

To allow maxiom declarations in theories, we amend Def. 2.2 as follows:

Definition 4.1 (Mormulas, Moofs, Maxioms)**.** In addition to `tp` and `form`, we have the universe `morm` of mormulas. Expressions $\Gamma \vdash_\Theta V : \texttt{morm}$ are called **mormulas**.

Expressions $\Gamma \vdash_\Theta Q : V$ are called **moofs** of Q.

In addition to the declarations of Def. 2.2, theories may contain declarations of the form

- **maxioms** $q : \{\Gamma\}V$ for $\Gamma \vdash_\Theta V : \texttt{morm}$

The key design choice to make now is what mormulas to allow. Our choice here is inspired by Sect. 3.2, and we discuss other options in Rem. 4.8:

Definition 4.2 (Mormulas)**.** Mormulas are formed according to the following grammar

$$\begin{array}{rcll} V & ::= & \reflectbox{C}_\Gamma T & \text{commutation with } T \\ & | & \reflectbox{P}_\Gamma F & \text{preservation of } F \\ & | & \reflectbox{R}_\Gamma F & \text{reflection of } F \end{array}$$

Their typing rules are

$$\frac{\Delta, \Gamma \vdash_\Theta T : A}{\Delta \vdash_\Theta \reflectbox{C}_\Gamma T : \texttt{morm}} \qquad \frac{\Delta, \Gamma \vdash_\Theta F : \texttt{form}}{\Delta \vdash_\Theta \#_\Gamma F : \texttt{morm}} \text{ for } \# \in \{\reflectbox{P}, \reflectbox{R}\}$$

where each Γ may declare only term variables.

All mormulas have a similar syntax: They bind term variables in a formula. Thus, they behave similarly to $\forall$ and $\exists$, which inspired their notations as mirrored upper case letters.

However, contrary to formulas, mormulas are not nested. For example, we can only write $\reflectbox{C}_{x:a,y:b}T$ but not $\reflectbox{C}_{x:a}\reflectbox{C}_{y:b}T$. This is important because we will define the interpretation of $\reflectbox{C}_{x:a,y:b}T$ in one step rather than in two compositional steps.

We can immediately recover the standard definition of model morphisms in the sense of Rem. 3.1:

Definition 4.3 (Standard Maxioms)**.** For a theory Θ, the standard maxioms are:

- for each function symbol $t : \{\Gamma\}A$ in Θ:

$$t^{\text{std}} : \reflectbox{C}_\Gamma t(x_1, \ldots, x_n)$$

- for each predicate symbol $\varphi : \{\Gamma\}\texttt{form}$ in Θ:

$$\varphi^{\text{std}} : \reflectbox{P}_\Gamma \varphi(x_1, \ldots, x_n)$$

where in each case $x_1, \ldots, x_n$ are the variables declared in Γ.

We do not require that the standard maxioms are present in every theory Θ. But it is good to have a name for them because they are present very often.

Example 4.4 (Continuing Ex. 3.19). We can now solve the problem of Ex. 3.7 concisely. The theory `MonoidCon` can use the standard maxioms. These are:

$$\circ^{\text{std}} : \reflectbox{C}_{x:u,y:u} x \circ y, \quad e^{\text{std}} : \reflectbox{C} e$$

But in the theory `MonoidAx`, the standard maxiom $\circ^{\text{std}}$ is not enough. We need the additional maxiom

$$\texttt{unit\_commute} : \reflectbox{P}_{x:u}\, \texttt{isunit}(x)$$

With these maxioms, both MonoidCon and MonoidAx will yield isomorphic model categories.

Remark 4.5 (Expressivity of the Standard Maxioms). One might argue that the standard maxioms are already sufficient because all maxioms can be seen as a special case of the standard maxiom for a fresh symbol. For example, we can replace the maxiom $q : ꟼ_\Gamma F$ where $\Gamma = \ldots, x_i : A_i, \ldots$ with the following declarations:

- a new predicate symbol $\varphi : \{\Gamma\}\texttt{form}$
- an axiom $p : \forall x_1 : A_1 . \ldots \forall x_n : A_n . \varphi(x_1, \ldots, x_n) \Leftrightarrow F$

Now φ^{std} has the same effect as the maxiom q.

Our solution goes beyond this lightweight approach in several ways:

- It allows using fewer than the standard maxioms, e.g., when
  - a standard maxiom is not satisfied by the intended model morphisms as in Ex. 5.1,
  - a standard maxiom is redundant because it is a meorem, i.e., implied by the other maxioms, as in Ex. 4.6,
  - a theory uses auxiliary symbols that should be ignored by model morphisms, e.g., the base of a vector space or the size of a finite group.[7]
- It allows us to reason about meoremhood in theory morphisms as we will see in Sect. 4.2. This is the key step to show that $\mathbf{Mod}(\vartheta)$ is always functorial.

Example 4.6 (Continuing Ex. 4.4). To extend the theory MonoidCon to the theory Group, we have to add the symbols

$$\texttt{i} : \{x : u\}u, \quad \texttt{inverse} : \forall x : u.\, x \circ \texttt{i}(x) \doteq e \wedge \texttt{i}(x) \circ x \doteq e$$

We do not have to add a maxiom—it is well known that the category of groups is a full subcategory of the theory of monoids, i.e., every monoid morphism between groups is already a group morphism. In our terminology, that means that the mormula $\supset_{x:u} \texttt{i}(x)$ is a meorem of the theory Group. Indeed, using the moof calculus from Sect. 4.2, we give a moof Q such that $\vdash_{\texttt{Group}} Q : \supset_{x:u} \texttt{i}(x)$ in Ex. 4.11.

In fact, groups are even a full subcategory of semigroups. Therefore, Group does not need the maxiom e^{std} either—that is also a meorem. However, monoids are not a full subcategory of semigroups. Therefore, the theory MonoidCon does need the maxiom e^{std}.

[7] These examples are taken from the category library of SageMath [S+13], which specifies model categories common in mathematics.

Example 4.7 (Elementary Equivalences). We can specify elementary equivalences by using the maxiom $\texttt{ee} : \{x : \texttt{form}\}\text{ꟼ}x$, which asserts the preservation of all formulas.

Remark 4.8 (Other Choices of Mormulas). In light of Sect. 3.2, we see that Ↄ and Я are definable in terms of ꟼ in classical logic. Thus, one might argue that they are redundant. We prefer to include them for several reasons.

Firstly, it follows the spirit of first-order logic to include important but redundant connectives. Secondly, the definition of Ↄ in terms of ꟼ would be awkward in practice, and using the dual connectives ꟼ and Я together makes the moof calculus somewhat more elegant. Thirdly, in other variants such as intuitionistic logic, the equivalences are lost.

Finally, our choice is not meant to be final. Instead, it serves as a starting point that exemplifies the approach and indicates the extensibility of the language. There are several candidate connectives that could be added to Ↄ, ꟼ, and Я, e.g.,

- $\text{Ↄ}_\Gamma F$ to express commuting with a formula (i.e., to preserve and reflect a formula),
- $\texttt{surj}\,A$ to express the surjectivity of A^m,
- analogues of ꟼ and Я that take two different formulas to be evaluated in M and M', e.g., $m \models \text{ꟼ}_\Gamma F, G$ might mean $F^{M,\alpha} \leq G^{M',m(\alpha)}$,
- analogues of ꟼ and Я whose semantics quantifies existentially over M-assignments, e.g., $m \models \text{ꟼ}'_\Gamma F$ might mean that if $F^{M',\beta} = 1$, then $F^{M,\alpha} = 1$ for some[8] $m(\alpha) = \beta$ (which would allow defining $\texttt{surj}\,A := \text{ꟼ}'_{x:A}\texttt{true}$).

Moreover, it is straightforward to add propositional connectives for mormulas. For example, if we add, e.g., conjunction $V \wedge V'$, then $(\text{ꟼ}_\Gamma F) \wedge (\text{ꟼ}_\Gamma F')$ is not definable in terms of ꟼ: it is stronger than $\text{ꟼ}_\Gamma(F \wedge F')$.

4.2. Proof Theory: Moofs and Syntactic Meorems

4.2.1. Theories

The deduction rules for **moofs** are given in Fig. 2. They can be verbalized as follows:

- `commvar`: Variables commute.
- `commsubs`: Substituting commuting terms into commuting terms yields commuting terms.
- `commpres`: A term commutes if its identity is preserved.
- `subs`: Substituting commuting terms into a preserved (reflected) formula, yields a preserved (reflected) formula.
- `equiv`: If F is preserved (reflected), so is every equivalent formula.
- `true` and `false`: `true` and `false` are preserved (reflected) by all morphisms.

[8]Our ꟼ and Я arise by using "for all" here instead of "for some".

- `conj` and `disj`: Conjunctions and disjunctions are preserved (reflected) if their arguments are.
- `neg`: $\neg F$ is preserved if F is reflected, and vice versa for reflection.
- `impl`: $F \Rightarrow G$ is preserved if F is reflected and G preserved, and vice versa for reflection.
- `equal`: An equality between commuting terms is preserved.
- `exists`: An existential quantifications is preserved if its argument is.
- `forall`: A universal quantifications is reflected if its argument is.

Definition 4.9 (Meorems)**.** A mormula V is called a **syntactic**[9] **meorem** if there is a moof Q such that $\vdash_\Theta Q : V$, i.e., if $\vdash_\Theta V$.

Example 4.10 (Theorems and Contradictions are Meorems). Theorems and contradictions F have the same truth values in all models. Therefore, they are trivially preserved by all model morphisms and thus $\text{ꟼ}F$ and $\text{Я}F$ are meorems.

Our calculus derives them using the rules for `equiv`, `true`, and `false`. This is possible because `true` $\Leftrightarrow F$ is a theorem if F is, and `false` $\Leftrightarrow F$ is a theorem if F is a contradiction.

Example 4.11 (Continuing Ex. 4.6). We show that $\text{Ↄ}e$ is a meorem of the theory `Group` using $\circ^{\mathrm{std}}$ as the only maxiom:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\texttt{Group}} \text{Ↄ}_{y:u} y \circ y}\ \circ^{\mathrm{std}} \qquad \dfrac{}{\vdash_{\texttt{Group}} \text{Ↄ}_{y:u} y}\ \texttt{commvar}}{\vdash_{\texttt{Group}} \text{ꟼ}_{y:u} y \circ y \doteq_u y}\ \texttt{equal} \qquad P}{\vdash_{\texttt{Group}} \text{ꟼ}_{y:u} y \doteq_u e}\ \texttt{equiv}}{\vdash_{\texttt{Group}} \text{Ↄ}e}\ \texttt{commpres}$$

where $y : u \vdash_{\texttt{Group}} P : y \circ y \doteq_u y \Leftrightarrow y \doteq_u e$ is a straightforward proof.

The moof of $\vdash_{\texttt{Group}} \text{Ↄ}_{x:u}\, \texttt{i}(x)$ proceeds in essentially the same way. The key step is to apply `equiv` with the equivalence $y \doteq_u \texttt{i}(x) \Leftrightarrow x \circ y \doteq_u e$ and then use $\circ^{\mathrm{std}}$ and $\text{Ↄ}e$.

4.2.2. Theory Morphisms

Def. 2.9 for theory morphisms can be applied without change to theories that contain maxioms. In particular, a theory morphism $\vartheta : \Theta \to \Theta'$ must map every maxiom to a moof of the corresponding mormula, and induces a homomorphic extension that maps all Θ-mormulas to moofs to Θ'-counterparts. Concretely,

- ϑ must provide for every maxiom $q : \{\Gamma\}V$ in Θ, an assignment $q \mapsto [\vartheta(\Gamma)]Q$ such that $\vartheta(\Gamma) \vdash_{\Theta'} Q : \vartheta(V)$,

[9] We use the qualifier *syntactic* to distinguish from *semantic* meorems, which are defined model theoretically in Sect. 4.3. We discuss the soundness/completeness relation between the two notions in Sect. 4.4.

Rules for atomic moofs $q(E_1, \ldots, E_n)$ for a maxiom q: as in Def. 2.3

Rules for commutation:

$$\frac{x : A \text{ in } \Gamma}{\triangle \vdash_\Theta \reflectbox{C}_\Gamma x}\,\texttt{commvar}$$

$$\frac{\triangle \vdash_\Theta \reflectbox{C}_{x_1:A_1,\ldots,x_n:A_n} T \qquad \triangle \vdash_\Theta \reflectbox{C}_\Gamma T_1 \qquad \ldots \qquad \triangle \vdash_\Theta \reflectbox{C}_\Gamma T_n}{\triangle \vdash_\Theta \reflectbox{C}_\Gamma T[\ldots, x_i/T_i, \ldots]}\,\texttt{commsubs}$$

$$\frac{\Gamma \vdash_\Theta T : A \qquad \triangle \vdash_\Theta \reflectbox{P}_{\Gamma,y:A}\, y \doteq_A T}{\triangle \vdash_\Theta \reflectbox{C}_\Gamma T}\,\texttt{commpres}$$

Common rules for preservation/reflection for $\# \in \{\reflectbox{P}, \reflectbox{R}\}$:

$$\frac{\triangle \vdash_\Theta \#_{x_1:A_1,\ldots,x_n:A_n} F \qquad \triangle \vdash_\Theta \reflectbox{C}_\Gamma T_1 \qquad \ldots \qquad \triangle \vdash_\Theta \reflectbox{C}_\Gamma T_n}{\triangle \vdash_\Theta \#_\Gamma F[\ldots, x_i/T_i, \ldots]}\,\texttt{subs}$$

$$\frac{\triangle \vdash_\Theta \#_\Gamma F \qquad \Gamma \vdash_\Theta F \Leftrightarrow G}{\triangle \vdash_\Theta \#_\Gamma G}\,\texttt{equiv} \qquad \frac{}{\triangle \vdash_\Theta \#_\Gamma \texttt{true}}\,\texttt{true} \qquad \frac{}{\triangle \vdash_\Theta \#_\Gamma \texttt{false}}\,\texttt{false}$$

$$\frac{\triangle \vdash_\Theta \#_\Gamma F \qquad \triangle \vdash_\Theta \#_\Gamma G}{\triangle \vdash_\Theta \#_\Gamma F \wedge G}\,\texttt{conj} \qquad \frac{\triangle \vdash_\Theta \#_\Gamma F \qquad \triangle \vdash_\Theta \#_\Gamma G}{\triangle \vdash_\Theta \#_\Gamma F \vee G}\,\texttt{disj}$$

Rules specific to preservation:

$$\frac{\triangle \vdash_\Theta \reflectbox{R}_\Gamma F}{\triangle \vdash_\Theta \reflectbox{P}_\Gamma \neg F}\,\texttt{neg} \qquad \frac{\triangle \vdash_\Theta \reflectbox{R}_\Gamma F \qquad \triangle \vdash_\Theta \reflectbox{P}_\Gamma G}{\triangle \vdash_\Theta \reflectbox{P}_\Gamma F \Rightarrow G}\,\texttt{impl}$$

$$\frac{\triangle \vdash_\Theta \reflectbox{P}_{\Gamma,x:A} F}{\triangle \vdash_\Theta \reflectbox{P}_\Gamma \exists x : A.F}\,\texttt{exists} \qquad \frac{\triangle \vdash_\Theta \reflectbox{C}_\Gamma T \qquad \triangle \vdash_\Theta \reflectbox{C}_\Gamma T'}{\triangle \vdash_\Theta \reflectbox{P}_\Gamma T \doteq_A T'}\,\texttt{equal}$$

Rules specific to reflection:

$$\frac{\triangle \vdash_\Theta \reflectbox{P}_\Gamma F}{\triangle \vdash_\Theta \reflectbox{R}_\Gamma \neg F}\,\texttt{neg} \qquad \frac{\triangle \vdash_\Theta \reflectbox{P}_\Gamma F \qquad \triangle \vdash_\Theta \reflectbox{R}_\Gamma G}{\triangle \vdash_\Theta \reflectbox{R}_\Gamma F \Rightarrow G}\,\texttt{impl}$$

$$\frac{\triangle \vdash_\Theta \reflectbox{R}_{\Gamma,x:A} F}{\triangle \vdash_\Theta \reflectbox{R}_\Gamma \forall x : A.F}\,\texttt{forall}$$

Figure 2: Moof Calculus

- the homomorphic extension of ϑ satisfies
  - $\vartheta(\mathtt{morm}) = \mathtt{morm}$
  - $\vartheta(\#_\Gamma F) = \#_{\vartheta(\Gamma)}\vartheta(F)$ for $\# \in \{\text{Ↄ}, \text{ꟼ}, \text{Я}\}$,
  - $\vartheta(Q)$ is the moof arising from Q by replacing all references to maxioms q with the moof assigned to q by ϑ, and by mapping all other expressions occurring in Q recursively.

Then Thm. 2.10 holds for maxioms without change:

Theorem 4.12 (Meorem Preservation)**.** *Let $\vartheta : \Theta \to \Theta'$ be a theory morphism. If V is a syntactic Θ-meorem, then $\vartheta(V)$ is a syntactic Θ'-meorem.*

Proof. Like Thm. 2.10 this is a special case of a more general result: Our definitions of theories and theory morphisms are systematically formulated in such a way that they are special cases of the corresponding concepts of the MMT framework [Rab14]. MMT proves once and for all the preservation of all judgments along theory morphisms, in particular, if $\vdash_\Theta Q : V$, then $\vdash_{\Theta'} \vartheta(Q) : \vartheta(V)$. That yields the theorem. □

Example 4.13 (Continuing Ex. 4.4)*.* We can now extend the theory morphism $\mathtt{axCon} : \mathtt{MonoidAx} \to \mathtt{MonoidCon}$ from Ex. 2.12.

Both theories contain the standard maxiom $\circ^{\mathrm{std}}$, and we can simply use the assignment

$$\circ^{\mathrm{std}} \mapsto \circ^{\mathrm{std}}$$

For the additional maxiom of `MonoidAx` we use

$$\mathtt{unit\_commute} \mapsto Q$$

where Q is the moof given by the following derivation:

$$\frac{\dfrac{\dfrac{}{\vdash_{\mathtt{MonoidCon}} \text{Ↄ}_{x:u} x}\,\mathtt{commvar} \qquad \dfrac{\dfrac{}{\vdash_{\mathtt{MonoidCon}} \text{Ↄ} e}\, e^{\mathrm{std}}}{\vdash_{\mathtt{MonoidCon}} \text{Ↄ}_{x:u} e}\,\mathtt{commsubs}}{\vdash_{\mathtt{MonoidCon}} \text{ꟼ}_{x:u} x \doteq_u e}\,\mathtt{equal} \qquad P}{\vdash_{\mathtt{MonoidCon}} \text{ꟼ}_{x:u}\, \mathtt{isunit}(x)}\,\mathtt{equiv}$$

for some proof $x : u \vdash_{\mathtt{MonoidCon}} P : x \doteq_u e \Leftrightarrow \mathtt{isunit}(x)$.

With the same caveat about a description operator as in Ex. 2.12, `axCon` is now essentially an isomorphism.

Example 4.14 (Continuing Ex. 3.20)*.* We can now solve the problem of Ex. 3.8 concisely.

We can choose not to add any maxioms to the theory `Special`. Then we obtain the category of sets with special elements whose morphisms do not have to preserve specialty. `least` is a theory morphism.

Alternatively, we can add the standard maxiom $\texttt{sp}^{\mathrm{std}}$, thus obtaining a different theory $\texttt{Special}'$. This theory yields the category whose morphisms must preserve specialty. Consequently, we cannot extend `least` to a theory morphism out of $\texttt{Special}'$ because there is no fitting moof in the theory `Order`.

4.3. Model Theory: Model Categories and Semantic Meorems

So far all definitions were generic, covering all variants of first-order logic allowed by our framework of Sect. 2. To state our main results, we now restrict attention to PFOL as defined in Ex. 2.5.

Our definitions specialize to all fragments of PFOL, in particular FOL and TFOL. We can generalize our results to more complex logics, but the definitions become increasingly complex. In fact, already the case for PFOL is so complex that it is instructive to consider the special case of TFOL first.

4.3.1. Model Categories for TFOL

We avoid the dilemmas described in Sect. 3.1 by using a weak definition of model morphism that only requires the maps a^m and imposes no commutativity or preservation conditions. Any properties of model morphisms are then imposed later by declaring maxioms. This is analogous to the standard definition of models, which uses a weak definition of model first and then allows imposing conditions on models by declaring axioms.

Definition 4.15 (Model Morphisms)**.** Consider a fixed TFOL-theory Θ and two models $M, M' \in \mathbf{Mod}(\Theta)$.

A **model morphism** $m : M \to M'$ is a tuple $(\ldots, a^m, \ldots)$ containing for every Θ-type symbol $a : \texttt{tp}$ a function

$$a^m : a^M \to a^{M'}$$

such that for every Θ-maxiom $q : \{\Gamma\}V$, we have that $m \models V(E_1, \ldots, E_n)$ for every mormula $\vdash_\Theta V(E_1, \ldots, E_n) : \texttt{morm}$ that arises by substituting for the free variables of V.

$m \models V$ will be defined in Def. 4.16 and Def. 4.17.

Every model morphism $m : M \to M'$ induces an interpretation function Γ^m. For contexts Γ, we have that $\Gamma^m : \Gamma^M \to \Gamma^{M'}$ translates M-assignments to M'-assignments. We will abbreviate $\Gamma^m(\alpha)$ it by $m(\alpha)$ if Γ is clear.

Similarly to formulas, a mormula $\vdash_\Theta V : \texttt{morm}$ is interpreted as its truth value $V^m \in \{0, 1\}$. Accordingly, the interpretation of a moof is uniquely determined.

Definition 4.16 (Interpretation Function)**.** The **interpretation function** of a model morphism $m : M \to M'$ is defined as follows:

- for contexts $\Gamma = \ldots, x_i : a_i, \ldots$:

$$\Gamma^m : \Gamma^M \ni (\ldots, \alpha_i, \ldots) \mapsto (\ldots, a_i^m(\alpha_i), \ldots) \in \Gamma^{M'}$$

- for the universe

$$\texttt{morm}^m = \{0, 1\}$$

- for non-atomic mormulas V:
  - if V is closed:

$$(\text{Ↄ}_\Gamma T)^m = \begin{cases} 1 & \text{if } A^{m,\alpha}(T^{M,\alpha}) = T^{M',m(\alpha)} \text{ for all } \alpha \in \Gamma^M \\ 0 & \text{otherwise} \end{cases}$$

    where A is the type of T

$$(\text{ꟼ}_\Gamma F)^m = \begin{cases} 1 & \text{if } F^{M,\alpha} \leq F^{M',m(\alpha)} \text{ for all } \alpha \in \Gamma^M \\ 0 & \text{otherwise} \end{cases}$$

$$(\text{Я}_\Gamma F)^m = \begin{cases} 1 & \text{if } F^{M,\alpha} \geq F^{M',m(\alpha)} \text{ for all } \alpha \in \Gamma^M \\ 0 & \text{otherwise} \end{cases}$$

  - if V has free variables $x_1, \ldots, x_n$: V'^m for all well-formed closed formulas V' of the form $V(E_1, \ldots, E_n)$
- for non-atomic moofs: Q^m is uniquely determined

Note that mormulas with free variables are interpreted by quantifying over all ground instances. This is in contrast to formulas with free variable, whose interpretation is relative to an assignment.

Definition 4.17 (Satisfaction)**.** We write $m \models V$ if $V^m = 1$.

It is straightforward to package models and model morphisms into model categories:

Definition 4.18 (Model Categories)**.** For every theory Θ, we define $\mathbf{Mod}(\Theta)$ to be the category of
- models as in Def. 2.17,
- model morphisms as in Def. 4.15,
- identity and composition of model morphisms in the obvious way.

We can now state the semantic analogue to Def. 4.9. Just like a (semantic) theorem is a formula that is satisfied by all models, a semantic meorem is a mormula that is satisfied by all model morphisms:

Definition 4.19 (Meorems)**.** A Θ-mormula V is called a **semantic meorem** if $m \models V$ for all Θ-model morphisms m.

4.3.2. Model Categories for PFOL

To generalize Def. 4.15 to polymorphic theories, we need an auxiliary definitions first:

Definition 4.20. Let $\mathcal{FUN}$ be the class of triples $(A \in \mathcal{SET}, B \in \mathcal{SET}, f : A \to B)$. For a tuple $(k_1, \ldots, k_n) \in \mathcal{FUN}^n$ and $i = 1, 2$ we write $k^{(i)} \in \mathcal{SET}^n$ for the tuple $(k_{1i}, \ldots, k_{ni})$.

The intuition behind $\mathcal{FUN}$ is that for every type A, the triple $(A^M, A^{M'}, A^m) \in \mathcal{FUN}$ packages the interpretations of A by the models and by the model morphism $m : M \to M'$. Because PFOL-types may contain type variables, we have to carry an assignment α when interpreting them. Therefore, we will actually work with triples $(A^{M,\alpha}, A^{M',m(\alpha)}, A^{m,\alpha}) \in \mathcal{FUN}$.

Definition 4.21 (Model Morphisms)**.** A PFOL-**model morphism** $m : M \to M'$ is defined as in Def. 4.15 except that the tuple $(\ldots, a^m, \ldots)$ contains for every Θ-type symbol $a : \{x_1 : \mathtt{tp}, \ldots, x_n : \mathtt{tp}\}\mathtt{tp}$ a dependent function

$$a^m : (k \in \mathcal{FUN}^n) \to a^M(k^{(1)}) \to a^{M'}(k^{(2)}).$$

Note that this definition contains Def. 4.15 as the special case where $n = 0$. Similarly, the interpretation of mormulas from Def. 4.16 remains essentially unchanged. We only have to extend the translation of assignments because PFOL-contexts may contain type variables:

Definition 4.22 (Assignment Translation)**.** A PFOL-model morphism $m : M \to M'$ translates assignments for a context $\Gamma = \ldots, x_i : C_i, \ldots$ as follows:

$$\Gamma^m : \Gamma^M \ni (\ldots, \alpha_i, \ldots) \mapsto (\ldots, C_i^{m,(\alpha_1,\ldots,\alpha_{i-1})}(\alpha_i), \ldots) \in \Gamma^{M'}$$

There are two cases for the C_i:

- If $C = \mathtt{tp}$, we need a function from $\mathtt{tp}^{M,\alpha} = \mathcal{SET}$ to $\mathtt{tp}^{M',m(\alpha)} = \mathcal{SET}$ and define $\mathtt{tp}^{m,\alpha} = id_{\mathcal{SET}}$. Thus m maps type assignments to themselves.
- If C is a type A, we define $A^{m,\alpha} : A^{M,\alpha} \to A^{M',m(\alpha)}$ by

  $$x^{m,\alpha} = id_{\alpha(x)}$$

  and

  $$a(\ldots, A_i, \ldots)^{m,\alpha} : a^M(\ldots, A_i^{M,\alpha}, \ldots) \to a^{M'}(\ldots, A_i^{M',m(\alpha)}, \ldots)$$

  $$a(\ldots, A_i, \ldots)^{m,\alpha} = a^m(\ldots, (A_i^{M,\alpha}, A_i^{M',m(\alpha)}, A_i^{m,\alpha}), \ldots)$$

With the assignment translation in place, all remaining definitions remain unchanged. In general, the assignment translation is the key part of the definition that has to be generalized when considering more general logics.

Example 4.23 (Lists). For a theory of lists, we use the declarations

$$\mathtt{list} : \{X : \mathtt{tp}\}\mathtt{tp}$$

$$\mathtt{nil} : \{X : \mathtt{tp}\}\,\mathtt{list}(X), \quad \mathtt{cons} : \{X : \mathtt{tp}, x : X, l : \mathtt{list}(X)\}\,\mathtt{list}(X)$$

along with the usual axioms for inductive types as well as the standard maxioms

$$\mathtt{nil}^{\mathrm{std}} : \{X : \mathtt{tp}\}\beth\,\mathtt{nil}(X), \quad \mathtt{cons}^{\mathrm{std}} : \{X : \mathtt{tp}\}\beth_{x:X,l:\mathtt{list}(X)}\,\mathtt{cons}(X, x, l)$$

Let M and M' be two models of this theory using two different interpretations of lists:

$$\mathtt{list}^M(S) = \{(n, l) | n \in \mathbb{N}, l : \{1, \ldots, n\} \to S\}$$

$$\mathtt{list}^{M'}(S) = \bigcup_{n \in \mathbb{N}} S^n$$

along with the obvious values for $\mathtt{nil}^M$, $\mathtt{nil}^{M'}$, $\mathtt{cons}^M$, and $\mathtt{cons}^{M'}$.

To give a model morphism $m : M \to M'$, we define $\mathtt{list}^m$ by

$$\mathtt{list}^m : ((S, S', f) \in \mathcal{FUN}) \to \mathtt{list}^M(S) \to \mathtt{list}^{M'}(S')$$

$$\mathtt{list}^m((S, S', f)) : (n, l) \mapsto (f(l(1)), \ldots, f(l(n)))$$

In fact, because the axioms make `list` an inductive type, the maxioms (which guarantee that morphisms commute with all constructors) already uniquely determine $\mathtt{list}^m$.

4.3.3. Model Reduction Functors

We can now establish our main result: For every PFOL-theory morphism ϑ, we obtain a model reduction functor $\mathbf{Mod}(\vartheta)$, and $\mathbf{Mod}$ is itself functorial.

Definition 4.24 (Model Reduction)**.** For every PFOL-theory morphism $\vartheta : \Theta \to \Theta'$, we define $\mathbf{Mod}(\vartheta) : \mathbf{Mod}(\Theta') \to \mathbf{Mod}(\Theta)$ to be the functor that maps

- models as in Def. 2.21,
- model morphisms $m' : M' \to N'$ to model morphisms $m : \mathbf{Mod}(\vartheta)(M') \to \mathbf{Mod}(\vartheta)(N')$ such that $a^m = \vartheta(a)^{m'}$ for all type symbols a declared in Θ.

The central technical step is to establish the following extension of Thm. 2.22:

Theorem 4.25 (Denotation Condition)**.** *Given a theory morphism $\vartheta : \Theta \to \Theta'$ and a mormula or moof $\vdash_\Theta E : C$, we have for every morphism $m' : M' \to N'$ in $\mathbf{Mod}(\Theta')$:*

$$\vartheta(E)^{m'} = E^{\mathbf{Mod}(\vartheta)(m')}$$

Proof. The statement is trivial if E is a moof because their interpretation is uniquely determined.

Because mormulas are interpreted as truth values, the case where E is a mormula is an equivalence: the left side of the equation must be 1 iff the right side is. We prove the left-to-right direction for the case $E = \text{ꟼ}_\Gamma F$.

We abbreviate $\Gamma' = \vartheta(\Gamma)$ and $F' = \vartheta(F)$ as well as $M = \mathbf{Mod}(\vartheta)(M')$, $N = \mathbf{Mod}(\vartheta)(N')$, and $m = \mathbf{Mod}(\vartheta)(m')$. Assume the left side equals 1 (1). To prove the right side equals 1, assume an assignment $\alpha \in \Gamma^M$ such that $F^{M,\alpha} = 1$ (2). We have to prove $F^{N,\Gamma^m(\alpha)} = 1$ $(*)$. Note that $\Gamma'^{M'} = \Gamma^M$ and $\Gamma'^{m'}(\alpha) = \Gamma^m(\alpha)$. Observe that we can treat

- Θ, Γ and $\Theta, \vartheta(\Gamma)$ as theories,
- ϑ, id_Γ as a theory morphism between them,
- M', α and $N', \Gamma'^{m'}(\alpha)$ as models of the latter theory,
- M, α and $N, \Gamma^m(\alpha)$ as the reducts of these models along ϑ, id_Γ.

Using this, we can apply Thm. 2.22 to the theory morphism ϑ, id_Γ and (2). This yields $F'^{M',\alpha} = 1$ (3). Applying (1) to (3) yields $F'^{N',\Gamma'^{m'}(\alpha)} = 1$ (4). Applying the first statement to the theory morphism ϑ, id_Γ and (4) yields $(*)$.

The proof of the right-to-left direction proceeds accordingly.

The cases for $E = \text{Я}_\Gamma F$ and $E = \text{Ɔ}_\Gamma T$ proceed accordingly. □

The following corollary is often useful:

Theorem 4.26 (Satisfaction Condition)**.** *Consider a theory morphism $\vartheta : \Theta' \to \Theta$.*

For every Θ-formula F and every Θ'-model M'

$$M' \models \vartheta(F) \quad \text{iff} \quad \mathbf{Mod}(\vartheta)(M') \models F$$

For every Θ-mormula V and every Θ'-model morphism $m' : M' \to N'$

$$m' \models \vartheta(V) \quad \text{iff} \quad \mathbf{Mod}(\vartheta)(m') \models V$$

Proof. Both statements are special cases of Thm. 2.22 and 4.25, respectively. □

Building on this, we can establish the functoriality condition, whose failure motivated the research presented in this article:

Theorem 4.27 (Functorial Model Reduction)**.** $\mathbf{Mod}(\vartheta)$ *is a functor for every theory morphism ϑ.*

Proof. Consider a theory morphism $\vartheta : \Theta \to \Theta'$.

We have to show that

- $\mathbf{Mod}(\vartheta)(M')$ is a Θ-model for every Θ'-model M',
- $\mathbf{Mod}(\vartheta)(m') : \mathbf{Mod}(\vartheta)(M'_1) \to \mathbf{Mod}(\vartheta)(M'_2)$ is a Θ-model morphism for every Θ'-model morphism $m' : M'_1 \to M'_2$.

- the functor law for identity $\mathbf{Mod}(\vartheta)(id_{M'}) = id_{\mathbf{Mod}(\vartheta)(M')}$,
- the functor law for composition $\mathbf{Mod}(\vartheta)(m_2' \circ m_1') = \mathbf{Mod}(\vartheta)(m_2') \circ \mathbf{Mod}(\vartheta)(m_1')$.

Note that only the second of these properties is new — the others do not depend on maxioms and moofs at all and are well-known for most logics similar to ours.

From now on and whenever appropriate, we abbreviate $M = \mathbf{Mod}(\vartheta)(M')$, $m = \mathbf{Mod}(\vartheta)(m')$ and so on.

Regarding the second property, it is easy to see that m is a Θ-model morphisms except for possibly not satisfying all Θ-maxioms. To show that m satisfies every Θ-maxiom $q : \{\Gamma\}V$, we proceed as follows:

1. Assume terms $E_1, \ldots, E_n$ such that $W = V(E_1, \ldots, E_n)$ is closed and well-formed. We have to show that $m \models W$.
2. Let $E_i' = \vartheta(E_i)$ and $W' = \vartheta(W)$, i.e., $W' = V'(E_1', \ldots, E_n')$. We have $\vartheta(\Gamma) \vdash_{\Theta'} \vartheta(q) : V'$ by the definition of theory morphisms. Then the substitution rules of the moof calculus yield a moof of W', i.e., W' is a syntactic meorem of Θ'.
3. The soundness of the moof calculus (see Thm. 4.30) yields $m' \models W'$.
4. Now Thm. 4.26 yields the needed $m \models W$.

The critical step in the above argument is the appeal to $\vartheta(q)$ — this is the step that fails without the maxioms and moofs introduced in this work.

The first property is proved accordingly using the soundness of the proof calculus and the corresponding part of Thm. 4.26.

Finally, the functor laws are inherited from the corresponding properties of sets because $\mathbf{Mod}(\vartheta)(m)$ is defined in terms of functions between sets. □

Theorem 4.28 (Functorial Model Reduction)**.** $\mathbf{Mod}$ *is a functor from the category of PFOL-theories and theory morphisms to* $\mathcal{CAT}^{op}$*.*

Proof. Thm. 4.27 shows that $\mathbf{Mod}$ indeed maps theories and morphisms to categories and functors. We only have to verify the functor laws:

- $\mathbf{Mod}(id_\Theta) = id_{\mathbf{Mod}(\Theta)}$,
- $\mathbf{Mod}(\vartheta' \circ \vartheta) = \mathbf{Mod}(\vartheta') \circ \mathbf{Mod}(\vartheta)$.

Both proofs are straightforward. □

Finally, we can package the model theoretical properties of our logic into a single statement:

Theorem 4.29. *PFOL forms an institution in the sense of [GB92].*

Proof. The institutions consists of

- the category $\mathbf{Th}$ of PFOL-theories and theory morphisms,
- the functor $\mathbf{Sen} : \mathbf{Th} \to \mathcal{SET}$ which maps
  - every theory Θ to the set of all F such that $\vdash_\Theta F : \texttt{form}$
  - every theory morphism ϑ to the function $F \mapsto \vartheta(F)$,

- the functor $\mathbf{Mod} : \mathbf{Th} \to \mathcal{CAT}^{op}$,
- the Θ-indexed family of relations defined by $M \models F$ for $M \in \mathbf{Mod}(\Theta)$ and $F \in \mathbf{Sen}(\Theta)$.

Thm. 4.28 yields the well-definedness of **Mod**, and Thm. 4.26 yields the satisfaction condition. □

The corresponding results hold for (classical or intuitionistic) fragments of PFOL, including FOL and TFOL.

4.4. Soundness and Completeness

It remains to relate syntactic and semantic meorems. Ideally, the notions should coincide, i.e., the moof calculus should be sound and complete. However, we have only been able to prove soundness at this point:

Theorem 4.30 (Soundness)**.** *The moof calculus is sound, i.e., if V is a syntactic meorem, then V is a semantic meorem.*

Proof. The proof proceeds by induction on derivations. The induction hypothesis is as follows: if $\Delta \vdash_\Theta E : C$, then

- for types, terms, formulas, and proofs E: $E^{M,\alpha} \in C^{M,\alpha}$ for every $M \in \mathbf{Mod}(\Theta)$ and $\alpha \in \Delta^M$
- for mormulas and moofs E: $E^m \in C^m$ for every $m : M \to N \in \mathbf{Mod}(\Theta)$

Recall that our notation $\triangle \vdash_\Theta \text{ꟼ}_\Gamma V$ is shorthand for $\Delta \vdash_\Theta Q : V$ for some Q. In this case, the induction hypothesis becomes: $m \models V$ for all morphisms m.

All cases are straightforward. As examples, we prove the soundness of two rules.

We prove the soundness of the rule `subs` for the case $\# = \text{ꟼ}$:

$$\frac{\triangle \vdash_\Theta \text{ꟼ}_{x_1:A_1,\ldots,x_n:A_n} F \qquad \triangle \vdash_\Theta \text{Ↄ}_\Gamma T_1 \qquad \ldots \qquad \triangle \vdash_\Theta \text{Ↄ}_\Gamma T_n}{\triangle \vdash_\Theta \text{ꟼ}_\Gamma F^*}\,\texttt{subs}$$

for $F^* = F[\ldots, x_i/T_i, \ldots]$.

If Δ is not empty, the corresponding induction hypotheses quantify over an arbitrary ground substitution for Δ. It is sufficient to prove soundness for an arbitrary such substitution, and therefore it is sufficient to prove soundness for the case where Δ is empty. Similarly, the corresponding induction hypotheses quantify over all model morphisms $m : M \to M'$. It is sufficient to prove soundness for the case of an arbitrary such m. Then applying the induction hypothesis to the assumptions yields

$$m \models \text{ꟼ}_{x_1:A_1,\ldots,x_n:A_n} F \quad (1) \qquad m \models \text{Ↄ}_\Gamma T_i \text{ for } i = 1, \ldots, n \quad (2)$$

and we have to prove

$$m \models \text{ꟼ}_\Gamma F^*$$

To prove that, we assume an arbitrary α such that ${F^*}^{M,\alpha} = 1$ (3) and prove ${F^*}^{M',m(\alpha)} = 1$ $(*)$.

Let τ and τ' be the assignments for $x_1 : A_1, \ldots, x_n : A_n$ given by $\tau_i = T_i^{M,\alpha}$ and $\tau'_i = T_i^{M',m(\alpha)}$. Using (2), we obtain $A_i^m(\tau_i) = \tau'_i$ and therefore $m(\tau) = \tau'$. We have $F^{*M,\alpha} = F^{M,\tau}$ and $F^{*M',m(\alpha)} = F^{M',\tau'}$. (These follow from Thm. 2.22 using the theory $\Theta, x_1 : A_1, \ldots, x_n : A_n$ and the theory morphism $id_\Theta, x_1 \mapsto T_1, \ldots, x_n \mapsto T_n$ into Θ, Γ.) With those equalities, applying (1) to (3) yields $(*)$.

We prove the soundness of the rule `forall`:

$$\frac{\triangle \vdash_\Theta Я_{\Gamma,x:A} F}{\triangle \vdash_\Theta Я_\Gamma \forall x : A.F}\texttt{forall}$$

Again it is sufficient to consider an arbitrary model morphism $m : M \to M'$. The induction hypothesis for the assumption yields $m \models Я_{\Gamma,x:A} F$ (1), and we have to prove $m \models Я_\Gamma \forall x : A.F$. To prove that, we assume an arbitrary α such that $(\forall x : A.F)^{M',m(\alpha)} = 1$ (2) and prove $(\forall x : A.F)^{M,\alpha} = 1$. To prove that, we assume an arbitrary $e \in A^{M,\alpha}$ and prove $F^{M,\alpha.e} = 1$ $(*)$. (2) yields $F^{M',m(\alpha).A^m(e)} = F^{M',m(\alpha.e)} = 1$ (3). Applying (1) to (3) for the assignment $\alpha.e$ yields $(*)$.

□

We conjecture that the moof calculus for meorems is also complete. More precisely, we conjecture it is complete for any variant of our logic for which the proof calculus for theorems is complete. However, we do not have a proof at this point.

Our conjecture is motivated by an informal analysis of possible moof rules that one might add to the calculus. All rules we could conceive of were easily seen to be either derivable or unsound.

A promising proof idea uses that, in light of Ex. 4.10, any counter-example to completeness (i.e., a semantic meorem that is not a syntactic meorem) would have to be a Θ-formula F that is neither a theorem nor a contradiction. Consequently, both F and $\neg F$ would be consistent with Θ, and the completeness of the proof calculus would yield models M of $\Theta \cup \{F\}$ and M' of $\Theta \cup \{\neg F\}$. Completeness of the moof calculus would follow if we could exhibit an appropriate model morphism from M to M'.

5. Generalizations

5.1. Power Types

In this section, we extend PFOL with a built-in unary type operator for power types. That is necessary because power types cannot be axiomatized in first-order logic even if we use PFOL-type operators. We apply the extended logic to axiomatize the category of topological spaces and continuous functions.

We extend PFOL with a type constructor $\texttt{Set}(A)$. Our guiding intuition is that $\Gamma \vdash_\Theta T : \texttt{Set}(A)$ holds if T is a subtype of A.

Syntax. We add the following productions to the grammar of Def. 2.3:

A	$::=$	$\texttt{Set}(A)$	power type of A
T	$::=$	$\{x : A \,\vert\, F\}$	subtype of A given by F
F	$::=$	$T \in T'$	membership test for a subtype
P	$::=$	see below	
Q	$::=$	see below	

with the following typing rules

$$\frac{\Gamma \vdash_\Theta A : \texttt{tp}}{\Gamma \vdash_\Theta \texttt{Set}(A) : \texttt{tp}}$$

$$\frac{\Gamma, x : A \vdash_\Theta F : \texttt{form}}{\Gamma \vdash_\Theta \{x : A \,|\, F\} : \texttt{Set}(A)} \qquad \frac{\Gamma \vdash_\Theta U : \texttt{Set}(A) \quad \Gamma \vdash_\Theta T : A}{\Gamma \vdash_\Theta T \in U : \texttt{form}}$$

We also add the following proof rules

$$\frac{\Gamma, x : A \vdash_\Theta F : \texttt{form} \quad \Gamma \vdash_\Theta T : A}{\Gamma \vdash_\Theta \big(T \in \{x : A \,|\, F(x)\}\big) \Leftrightarrow F(T)}\texttt{reduce}$$

$$\frac{}{\Gamma, X : \texttt{tp}, y : \texttt{Set}(X) \vdash_\Theta y \doteq_{\texttt{Set}(X)} \{x : X \,|\, x \in y\}}\texttt{expand}$$

Everything so far is straightforward: If we used higher-order logic[10], we could define $\texttt{Set}(A) := A \to \texttt{bool}$. Then $\{x : A \,|\, F\}$, $T \in U$, $\texttt{reduce}$, and $\texttt{expand}$, respectively, would become special cases of λ-abstraction, application, β-reduction, and η-expansion.

The key novelty now is that we can add the maxiom

$$\frac{}{X : \texttt{tp} \vdash_\Theta \,\mathsf{q}_{x:X, y:\texttt{Set}(X)} x \in y}\texttt{preserve}$$

which guarantees that membership is preserved by model morphisms. Notably, we do not require that membership is reflected. Thus, model morphisms do not commute with all terms. As we will see below, that requirement would be impractically strong.

Abbreviations. We can immediately define the usual lattice operations on the type $\texttt{Set}(A)$:

$$\bot_A := \{x : A \,|\, \texttt{false}\}$$

$$\top_A := \{x : A \,|\, \texttt{true}\}$$

$$S \cap T := \{x : A \,|\, x \in S \wedge x \in T\}$$

[10]Recall that using higher-order logic in its entirety would make it difficult to define model morphisms.

$$S \cup T := \{x : A \,|\, x \in S \vee x \in T\}$$
$$\complement S := \{x : A \,|\, \neg x \in S\}$$
$$S \subseteq T := \forall x : A.\, x \in S \Rightarrow x \in T$$

We can also define the big operators that map from $\mathtt{Set}(\mathtt{Set}(A))$ to $\mathtt{Set}(A)$:

$$\bigcap K := \{x : A \,|\, \forall k : \mathtt{Set}(A).k \in K \Rightarrow x \in k\}$$
$$\bigcup K := \{x : A \,|\, \exists k : \mathtt{Set}(A).k \in K \wedge x \in k\}$$

Semantics. We add the following cases to the interpretation of expressions in models in Def. 2.20:

$$\mathtt{Set}(A)^{M,\alpha} = \mathcal{P}(A^{M,\alpha})$$
$$\{x : A \,|\, F\}^{M,\alpha} = \{e \in A^{M,\alpha} \,|\, F^{M,\alpha.e} = 1\}$$
$$(T \in U)^{M,\alpha} = \begin{cases} 1 & \text{if } T^{M,\alpha} \in U^{M,\alpha} \\ 0 & \text{otherwise} \end{cases}$$

This fixes the expected interpretation of power types as power sets.

Moreover, we add one case to the translation function of model morphisms in Def. 4.22:

$$\mathtt{Set}(A)^{m,\alpha} : \mathcal{P}(A^{M,\alpha}) \to \mathcal{P}(A^{M',m(\alpha)})$$
$$\mathtt{Set}(A)^{m,\alpha} : S \mapsto \{A^{m,\alpha}(s) : s \in S\}$$

This extends maps between sets to maps between power sets point-wise, i.e., m maps a set S of points $s \in A^{M,\alpha}$ to the set of points $A^{m,\alpha}(s) \in A^{M',m(\alpha)}$.

Extending the proof of soundness in Thm. 4.30 with cases for the new proof and moof rules is straightforward.

Non-Commutation of Terms. Model morphisms do not have to commute with terms $\{x : A \,|\, F\}$. Using the moof rule we added, it is easy to show that for all terms $\Gamma \vdash_\Theta U : \mathtt{Set}(A)$, we have $U^{M,\alpha} \subseteq U^{M',m(\alpha)}$. But equality may or may not hold.

For example, it is easy to show that model morphisms commute with $\bot_A$, $\cup$, $\cap$, and $\bigcup$ but not with $\top_A$, $\complement$, and $\bigcap$. For example, if a^m is not surjective, the point-wise definition yields $(\top_a)^M \subsetneq (\top_a)^{M'}$.

Applications to Topology. There are multiple conceptually different but isomorphic theories whose models are the topological spaces. But if the standard maxioms are used, these isomorphic theories yield very different model morphisms and thus do not yield isomorphic model categories.

The following table gives an overview:

Axiomatization is based on $\mathtt{point} : \mathtt{tp}$ and	Standard maxioms yield model morphisms that
open sets $\mathtt{open} : \{s : \mathtt{Set(point)}\}\mathtt{form}$	preserve openness
neighborhoods of a point $\mathtt{neighborhood} : \{p : \mathtt{point}, n : \mathtt{Set(point)}\}\mathtt{form}$	preserve openness
closed sets $\mathtt{closed} : \{s : \mathtt{Set(point)}\}\mathtt{form}$	preserve closedness
closure operator $\mathtt{closure} : \{s : \mathtt{Set(point)}\}\mathtt{Set(point)}$	preserve closedness
closeness relation between points and sets $\mathtt{close} : \{p : \mathtt{point}, s : \mathtt{Set(point)}\}\mathtt{form}$	are continuous

Moreover, similar to Ex. 2.6, subtle variations can further influence the resulting model morphisms. For example, the standard maxioms yield different morphisms if the open sets are given by a set $\mathtt{open}' : \mathtt{Set(Set(point))}$ than if they are given by a unary predicate $\mathtt{open} : \{s : \mathtt{Set(point)}\}\mathtt{form}$.

The following example looks at two of the above in more detail:

Example 5.1 (Topological Spaces). The theory `Open` of topological spaces based on open sets contains the declarations

$$\mathtt{point} : \mathtt{tp}, \ \ \mathtt{open} : \{x : \mathtt{Set(point)}\}\mathtt{form}$$

The theory `Closed` of topological spaces based on closed sets contains the declarations

$$\mathtt{point} : \mathtt{tp}, \ \ \mathtt{closed} : \{x : \mathtt{Set(point)}\}\mathtt{form}$$

In both cases, we omit the well-known axioms.

It is straightforward to give theory isomorphisms between these theories. Both directions maps $\mathtt{point} \mapsto \mathtt{point}$. Additionally, the isomorphism $\mathtt{oc} : \mathtt{Open} \to \mathtt{Closed}$ maps $\mathtt{open} \mapsto [x : \mathtt{point}]\, \mathtt{closed}(\mathbb{C}x)$, and the isomorphism $\mathtt{co} : \mathtt{Closed} \to \mathtt{Open}$ maps $\mathtt{closed} \mapsto [x : \mathtt{point}]\, \mathtt{open}(\mathbb{C}x)$. We show that these morphisms are indeed isomorphisms (up to provable equality) by checking that they compose to the identity, which follows easily using $\mathbb{C}(\mathbb{C}x) \doteq_{\mathtt{Set(point)}} x$.

Mod(`Open`) and **Mod**(`Closed`) have bijective model classes, and **Mod**(`oc`) and **Mod**(`co`) are bijections between them.

But if we use the standard maxioms, the two categories have different model morphisms. **Mod**(`Open`)-morphisms preserve openness (i.e., map open sets to open sets), whereas **Mod**(`Closed`)-morphisms preserve closedness (i.e., map closed sets to closed sets). Neither of these two conditions implies the other, and neither is sufficient or necessary for being a continuous function.

If, instead, we do not use the standard maxioms, it becomes possible to give maxioms that yield the continuous functions: To `Closed`, we add[11]

$$\mathtt{continuous} : ꟼ_{p:\mathtt{point}, s:\mathtt{Set(point)}}\, p \in \mathtt{closure}(s)$$

where $\texttt{closure}(s) := \bigcap\{S : \texttt{Set}(\texttt{point}) \mid s \subseteq S \wedge \texttt{closed}(S)\}$ is the smallest closed superset of s.

The theory `Open` can be extended accordingly. It is now straightforward to extend the isomorphisms `oc` and `co`, which then imply that the resulting model categories are isomorphic.

5.2. Partial Functions

The theory of fields poses an awkward problem in first-order-based algebraic specification: It is an elementary theory in mathematics but the partiality of the division function cannot naturally be handled with plain first-order logic.

There are several solutions. For example we can use PFOL and introduce option types akin to Ex. 4.23. Among others, this would use the declarations

$$\texttt{option} : \{x : \texttt{tp}\}\texttt{tp}, \quad u : \texttt{tp}, \quad / : \{x : u, y : u\}\, \texttt{option}(u)$$

Another common solution is to extend FOL with a definedness predicate and partial function symbols.

Here we explore an alternative that uses guarded arguments. This example has the added benefit that it explores some advanced details of our treatment of model morphisms. We show how it can be used first and discuss the subtleties afterwards:

Example 5.2 (Fields). The theory of rings contains among others the declarations

$$u : \texttt{tp},\ 0 : u\ , 1 : u,\ + : \{x : u, y : u\}u,\ - : \{x : u, y : u\}u,\ * : \{x : u, y : u\}u$$

To obtain the theory `Field` of fields, we add a ternary function symbol for division, whose third argument is a guard that prevents division by zero:

$$\texttt{div} : \{x : u, y : u, p : \neg y \doteq_u 0\}u$$

`Field` only needs the standard maxioms $+^{\mathrm{std}}$ and $*^{\mathrm{std}}$. The other standard maxioms are meorems that we can establish in essentially the same way as in Ex. 4.11.

However, to make all this work, we have to extend our syntax to allow quantifying over proof variables. Firstly, we need it to state the axiom

$$\texttt{mult\_inv} : \forall x : u.\, \forall p : \neg x \doteq_u 0.\ \texttt{div}(x, x, p) \doteq_u 1$$

We cannot use an implication $\forall x : u.\, x \doteq_u 0 \Rightarrow \texttt{div}(x, x, ?) \doteq_u 1$ anymore because we need a name p for the assumption so that we can use it as the

[11] Maybe surprisingly, the seemingly obvious maxiom $Я_{x:\texttt{Set}(\texttt{point})}\, \texttt{open}(x)$ is not strong enough to yield the continuous functions: It states "S is open if $f(S)$ is" whereas continuity of f is the subtly different "$f^{-1}(S)$ is open if S is".

guard. Secondly, we need it in the mormula of the standard axiom for division (which is a meorem):

$$\texttt{div\_comm} : \text{Ↄ}_{x:u,y:u,p:\neg y \doteq_u 0}\, \texttt{div}(x, y, p)$$

Quantifying over proof variables is a technically major but intuitively straightforward generalization of TFOL.

With that in place, we can establish the meorem that non-zeroness is preserved:

$$\texttt{nz} : \text{ꟼ}_{x:u} \neg x \doteq_u 0$$

The key part of the moof is

$$\dfrac{\dfrac{\dfrac{(\text{ omitted })}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u,y:u} x * y \doteq 1}}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u} \exists y : u.\, x * y \doteq_u 1}\texttt{exists} \qquad P}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u} \neg x \doteq_u 0}\texttt{equiv}$$

where $x : u \vdash_{\texttt{Field}} P : \neg x \doteq_u 0 \Leftrightarrow \exists y : u.\, x * y \doteq_u 1$.

From this, we can prove an important meorem about field morphisms: that they are injective (i.e., preserve inequality). The moof is

$$\dfrac{\dfrac{\dfrac{}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u} \neg x \doteq_u 0}\texttt{nz} \qquad \dfrac{(\text{ omitted })}{\vdash_{\texttt{Field}} \text{Ↄ}_{x:u,y:u} x - y}}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u,y:u} \neg x - y \doteq_u 0}\texttt{commsubs} \qquad P}{\vdash_{\texttt{Field}} \text{ꟼ}_{x:u,y:u} \neg x \doteq_u y}\texttt{equiv}$$

where $x : u, y : u \vdash_{\texttt{Field}} P : \neg(x - y \doteq_u 0) \Leftrightarrow \neg(x \doteq_u y)$.

Working out the extension of TFOL used in Ex. 5.2 leads to a subtle problem. The assignment translation function Γ^m is defined for contexts Γ that declare only term variables in Def. 4.16 and generalized to type variables in Def. 4.22. But it cannot easily be generalized to proof variables.

Consider a context $p : F$ for a closed formula F and a model morphism $m : M \to M'$. We want to define $(p : F)^m$. Note that $(p : F)^M$ is non-empty iff $F^M = 1$. Therefore, it is possible that no function $(p : F)^M \to (p : F)^{M'}$ exists. This happens if $F^M = 1$ but $F^{M'} = 0$, which is equivalent to $(\text{ꟼ}F)^m = 0$.

Therefore, it is non-obvious how to define the needed extension of TFOL. A promising idea is to allow proof variable declarations $p : F$ only in contexts in which $\text{ꟼ}F$ is a meorem. Indeed, the proof variables used in Ex. 5.2 have this property. We leave the details to future work.

6. Conclusion

We have introduced new syntactic concepts for specifying and reasoning about model morphisms. Our new concepts of mormulas, maxioms, moofs, and meorems are for model morphisms what the existing concepts of formulas, axioms, proofs, and theorems are for models.

This allows designing specification languages in which users have more fine-grained control over the intended semantics of a theory: they can now specify not only the models but also the model morphisms in the categories. The most important property of these languages is that the semantics of every theory morphism is a model reduction functor.

We developed a general framework for such specification languages and instantiated it with untyped, typed, and polymorphic first-order logic. Moreover, we discussed further generalizations to power types and partial functions.

Existing implementations of specification languages such as in the Hets system [MML07] can be easily augmented with our new concepts. Besides allowing for more precise specifications of categories, this enables tool support for reasoning about model morphisms.

An open question for future work is whether our approach covers all interesting practical cases. Just like not every interesting property of models is expressible using formulas, we cannot expect every interesting property of model morphisms to be expressible using mormulas. As indicated in Rem. 4.8, there are alternative choices of mormula operators that may become desirable as more and more practical applications are investigated. Our current design is simple enough to implement relatively easily and strong enough to cover many practically important examples. That provides a good infrastructure for conducting major case studies, which we expect will drive the future fine-tuning of the mormula language.

[BP13] J. Blanchette and A. Paskevich. TFF1: The TPTP Typed First-Order Form with Rank-1 Polymorphism. In M. Bonacina, editor, *Automated Deduction*, pages 414–420. Springer, 2013.

[CoF04] CoFI (The Common Framework Initiative). *CASL Reference Manual*, volume 2960 of *LNCS*. Springer, 2004.

[Coq15] Coq Development Team. The Coq Proof Assistant: Reference Manual. Technical report, INRIA, 2015.

[Dia08] R. Diaconescu. *Institution-independent Model Theory*. Birkhäuser, 2008.

[Dia16] R. Diaconescu. Functorial semantics of first-order views. *Theoretical Computer Science*, 2016. in press, see `http://dx.doi.org/10.1016/j.tcs.2016.09.009`.

[FGT92] W. Farmer, J. Guttman, and F. Thayer. Little Theories. In D. Kapur, editor, *Conference on Automated Deduction*, pages 467–581, 1992.

[GB92] J. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GWM+93] J. Goguen, Timothy Winkler, J. Meseguer, K. Futatsugi, and J. Jouannaud. Introducing OBJ. In J. Goguen, D. Coleman, and R. Gallimore, editors, *Applications of Algebraic Specification using OBJ*. Cambridge, 1993.

[HHP93] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

[MML07] T. Mossakowski, C. Maeder, and K. Lüttich. The Heterogeneous Tool Set. In O. Grumberg and M. Huth, editor, *Tools and Algorithms for the Construction and Analysis of Systems 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522, 2007.

[Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994.

[Rab14] F. Rabe. How to Identify, Translate, and Combine Logics? *Journal of Logic and Computation*, 2014. doi:10.1093/logcom/exu079.

[Rey74] J. Reynolds. On the relation between direct and continuation semantics. In *Second Colloq. Automata, Languages and Programming*, LNCS, pages 141–156. Springer, 1974.

[RS09] F. Rabe and C. Schürmann. A Practical Module System for LF. In J. Cheney and A. Felty, editors, *Proceedings of the Workshop on Logical Frameworks: Meta-Theory and Practice (LFMTP)*, pages 40–48. ACM Press, 2009.

[S+13] W. Stein et al. *Sage Mathematics Software*. The Sage Development Team, 2013. http://www.sagemath.org.

[SW83] D. Sannella and M. Wirsing. A Kernel Language for Algebraic Specification and Implementation. In M. Karpinski, editor, *Fundamentals of Computation Theory*, pages 413–427. Springer, 1983.