A Logical Framework Combining Model and Proof Theory

F L O R I A N   R A B E

Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany, f.rabe@jacobs-university.de

Mathematical logic and computer science drive the design of a growing number of logics and related formalisms such as set theories and type theories. In response to this population explosion, logical frameworks have been developed as formal meta-languages in which to represent, structure, relate, and reason about logics.

Research on logical frameworks has diverged into separate communities with often conflicting backgrounds and philosophies. In particular, among the most important logical frameworks are the framework of institutions from the area of model theory based on category theory, and the Edinburgh Logical Framework LF from the area of proof theory based on dependent type theory. Even though their ultimate motivations overlap – for example in applications to software verification – these take fundamentally different perspectives on logic.

We design a logical framework that integrates the frameworks of institutions and LF in a way that combines their complementary advantages while retaining the elegance of either one. In particular, our framework takes a balanced approach between model theory and proof theory and permits the representation of logics in a way that comprises all major ingredients of a logic: syntax, models, satisfaction, judgments, and proofs. This provides a theoretical basis for the systematic study of logics in a comprehensive logical framework. Our framework has been applied to obtain a large library of structured and machine-verified encodings of logics and logic translations.

1. Introduction

Since the foundational crisis of mathematics in the early 20th century, logic has been an important research topic in mathematics and later in computer science. A central issue has always been what a logic actually is (let alone a logic translation). Over the course of the last hundred years researchers have provided very different answers to this question; for example, even in 2005 the World Congress on Universal Logic held a contest about what a logic is Béziau (2005). Research areas that were initially connected have diverged and evolved into separate fields, and while this specialization has led to very successful results, it has also created divisions in the research on logic that are sometimes detrimental.

In response to these divisions, logical frameworks have been introduced. They unify

different logical formalisms by representing them in a fixed meta-language. Logical frameworks have been used successfully for logic-independent investigations both on the theoretical (e.g., in the textbook Diaconescu (2008)) and on the practical level (e.g., in the proof assistant Paulson (1994)).

However, today we observe that one division remains and that there are two groups of logical frameworks: *model theoretical* and *proof theoretical* ones. While some of these frameworks integrate the other side, such as the general proof theory developed in Meseguer (1989), almost all of them lean towards either model or proof theory. Often these sides are divided not only by research questions but also by "conflicting cultures and attitudes", and "attempts to bridge these two cultures are rare and rather timid" (quoting an anonymous reviewer of a related paper). This paper makes one such attempt, trying to provide a balanced framework that integrates and subsumes both views on logic in a way that preserves and exploits their respective advantages.

Model theoretical frameworks are based on set theoretical (such as Zermelo-Fraenkel set theoretical Fraenkel (1922); Zermelo (1908)) or category theoretical (Mac Lane (1998)) foundations of mathematics and characterize logics model theoretically in a way that goes back to Tarski's view of logical consequence (Robinson (1950); Tarski (1933); Tarski and Vaught (1956)). The central concept is that of a **model**, which interprets the non-logical symbols, and the **satisfaction** relation between formulas and models. Often category theory and initial models (Goguen et al. (1978)) are employed to study the model classes. The most important such framework is that of institutions (Goguen and Burstall (1992)).

Proof theoretical frameworks are based on type theoretical foundations of mathematics (such as the principia Whitehead and Russell (1913) or simple type theory Church (1940)) and characterize logics proof theoretically (e.g., Gentzen (1934); Gödel (1930)) in a way most prominently expressed in Hilbert's program (Hilbert (1926)). The central concept is that of a **proof**, which derives valid **judgments** from axioms via inference rules. Often the Curry-Howard correspondence (Curry and Feys (1958); Howard (1980)) is employed to study proofs as expressions of a formal language. The most important such frameworks are Automath (de Bruijn (1970)), Isabelle (Paulson (1994)), and the Edinburgh Logical Framework (LF Harper et al. (1993)).

The ontological intersection of model and proof theory is the syntax of logical formulas and the **consequence** relation between formulas, which includes the study of **theorems**. Thus, both provide ontological frameworks for formulas and consequence, and this intersection will be our central interest in this work. But both frameworks also go beyond this intersection. Model theory studies the properties of models in general such as for example categoricity and initiality. Similarly, proof theory studies the properties of proofs in general such as for example normalization and program extraction. Our work may provide a starting point to investigate whether these advanced properties have interesting analogues in the respective other field.

Our unified framework picks one of the most successful frameworks from each side and combines them: institutions and LF.

Institutions provide an abstract definition of the syntax and the model theoretical

semantics of a logic. Research on institutions focuses on signatures, sentences, models, and satisfaction but deemphasizes proofs and derivability. Among the most characteristic features of the framework of institutions are: (i) It abstracts from the syntax of formulas and only assumes an abstract set of sentences; similarly, it uses an abstract class of models and an abstract satisfaction relation. (ii) Using a Galois connection between sentences and models, it permits viewing the relation between syntax and (model theoretical) semantics as a pair of adjoint functors (Lawvere (1969)). (iii) Using category theory (Mac Lane (1998)), it provides an abstract notion of translations, both of translations within a logic – called signature or theory morphisms – and of translations between logics (Goguen and Rosu (2002); Mossakowski et al. (2007)) – called institution (co)morphisms.

LF is a dependent type theory related to Martin-Löf's type theory (Martin-Löf (1974)) featuring kinded type families and dependent function types. Research on logic encodings in LF focuses on syntax, proofs, and provability and dually deemphasizes models and satisfaction. Among the most characteristic features of this framework are: (i) It represents the logical and non-logical symbols as constants of the type theory using higher-order abstract syntax to represent binders. (ii) Using the judgments-as-types paradigm (Martin-Löf (1996)), it permits viewing syntax and (proof theoretical) semantics via judgments and inference systems and thus without appealing to a foundation of mathematics. (iii) Using a constructive point of view, translation functions are given as provably terminating programs (e.g., in Naumov et al. (2001)).

These two frameworks have been strongly influenced by the respective approaches, and this has created complementary strengths and weaknesses. For example, the bias towards an abstraction from the syntax exhibited by institutions is very useful for a logic-independent meta-logical analysis (Diaconescu (2008)). But LF's bias towards concrete syntax excels when defining individual logics and extracting programs from individual translations (Pfenning (2001)). Therefore, we focus on combining the respective strengths and motivations: While our work is technically formulated within institution theory, it inherits a strictly formalist justification from LF.

In the **literature** on individual logics, the combination of proof and model theoretical perspective is commonplace, and logics are usually introduced giving their syntax, model theory, and proof theory. The major difference is often the order of the latter two. Sometimes the model theory is given primacy as *the* semantics, and then a proof theory is given and proved sound and complete (e.g., in Barwise (1977); Smullyan (1995)). And sometimes it is the other way round, e.g., in Andrews (1986); Lambek and Scott (1986). Often this difference reflects a philosophical inclination of the author.

A similar difference in primacy is found when looking at families of logics. For example, the model theoretical view takes precedence in description logic (see, e.g., Baader et al. (2003); Brachman and Schmolze (1985)) where the model theory is fixed and proof theory is studied chiefly to provide reasoning tools. This view is arguably paramount as it dominates accounts of (classical) first-order logic and most uses of logic in mathematics. On the other hand, the proof theoretical view takes precedence in, for example, higher-order logic (proof theory Church (1940), model theory Henkin (1950)); and some logics such as intuitionistic logics (Brouwer (1907)) are explicitly defined by their proof theory.

At the level of logical frameworks, proof theoretical principles have been integrated into model theoretical frameworks in several ways. Parchments are used in Goguen and Burstall (1986); Mossakowski et al. (1997) to express the syntax of a logic as a formal language. And for example, in Fiadeiro and Sernadas (1988); Meseguer (1989); Mossakowski et al. (2005), proof theory is formalized in the abstract style of category theory. These approaches are very elegant but often fail to exploit a major advantage of proof theory – the constructive reasoning over concrete syntax. Vice versa, model theoretical principles have been integrated into proof theoretical frameworks, albeit to a lesser extent. For example, models and model classes can be represented using axiomatic type classes in Isabelle Haftmann and Wenzel (2006); Paulson (1994) or using structures in Mizar (Trybulec and Blair (1985)).

But a logical framework that systematically subsumes frameworks from both perspectives has so far been lacking, and our contribution consists in providing such a framework. Our work is separated into three main parts.

Firstly – in Sect. 3 – we extend institutions with an abstract notion of proof theory arriving at what we call *logics*. Our logics are defined in the spirit of institutions, in particular retaining the abstraction from concrete syntax. Thus, they are very similar to the general logics of Meseguer (1989) and to the proof theoretic institutions of Diaconescu (2006); Mossakowski et al. (2005). We also discuss the use of meta-logics, which can be seen as a response to Tarlecki (1996), where the use of a meta-institution is explored in general and the use of LF suggested in particular.

Secondly – in Sect. 4 – we give a logic $\mathbb{M}$ based on LF that we will use as our meta-logic. The central idea is that a signature of $\mathbb{M}$ captures syntax, proof theory, and model theory of an object logic. Similarly, $\mathbb{M}$-signature morphisms capture translations between object logics, including the translations of syntax, proof theory, and model theory. $\mathbb{M}$ is defined in the spirit of LF, in particular retaining the focus on concrete syntax. Consequently, judgments are represented as types and proofs as terms. Moreover – inspired by the ideas of Lawvere (Lawvere (1963)) – models are represented as signature morphisms from the logical theory into a signature representing the foundation of mathematics.

In this paper, we will use the term "foundation of mathematics" to refer to a fixed language in which mathematical objects are expressed. While mathematicians usually do not explicate a foundation (often implicitly assuming a variant of first-order set theory as the foundation), the choice of foundation is more difficult in computational logic, where for example higher-order logic (Church (1940)) and the calculus of constructions (Coquand and Huet (1988)) are often used instead. By representing such a foundation as an LF signature itself, our framework can formulate model theory without committing to a specific foundation.

Thirdly – in Sect. 5 – we demonstrate how to use $\mathbb{M}$ as a meta-logic. We also show how existing logics and translations can be encoded in $\mathbb{M}$ and how we can reason about the adequacy of such encodings. This section reconciles the type/proof and the set/model theoretical perspective: institution-based logics are expressed using the syntax of an LF-based meta-logic. We will exemplify all our definitions by giving a simple logic translation from modal logic to first-order logic as a running example. Moreover, we have

implemented our framework and evaluated it in large scale case studies (Codescu et al. (2011); Horozal and Rabe (2011); Iancu and Rabe (2011)).

We discuss and evaluate our framework in Sect. 6 and conclude in Sect. 7. We give introductions to the frameworks of institutions and LF in Sect. 2.

2. Logical Frameworks

2.1. *The Model Theoretical Framework of Institutions*

Institutions were introduced in Goguen and Burstall (1992) as as means to manage the population explosion among logics in use. The key idea is to abstract from the satisfaction relation between the sentences and the models. We will only give an intuitive introduction to the notion of an institution here. A rigorous and more comprehensive treatment is inherent in our definitions in Sect. 3.

The components of an institution are centered around a class **Sig** of **signatures**. To each signature $\Sigma \in$ **Sig** are associated a set **Sen**(Σ) of **sentences**, a class **Mod**(Σ) of **models**, and the **satisfaction** relation $\models_\Sigma \subseteq$ **Mod**$(\Sigma) \times$ **Sen**(Σ).

Example 1 (Modal Logic). For a simple institution $\mathbb{ML}$ of modal logic, the signatures in **Sig**$^{\mathbb{ML}}$ are the finite sets of propositional variables (chosen from some fixed set of symbols). The set **Sen**$^{\mathbb{ML}}(\Sigma)$ is the smallest set containing p for all $p \in \Sigma$ and closed under sentence formation $F \supset G$ and $\Box F$.

A model $M \in$ **Mod**$^{\mathbb{ML}}(\Sigma)$ is a Kripke model, i.e., a tuple $(W, \prec, V)$ for a set W of worlds, an accessibility relation $\prec \subseteq W \times W$, and a valuation $V : \Sigma \times W \to \{0, 1\}$ for the propositional variables. V is extended to a mapping **Sen**$(\Sigma) \times W \to \{0, 1\}$ in the usual way, and the satisfaction relation $M \models_\Sigma^{\mathbb{ML}} F$ holds iff $V(F, w) = 1$ for all worlds $w \in W$.

This abstract definition yields a rich setting in which institutions can be studied (see, e.g., Diaconescu (2008)). Most importantly, we can define an abstract notion of **theory** as a pair (Σ, Θ) for a set $\Theta \subseteq$ **Sen**(Σ) of axioms. (See Term. 1 for different uses of the term "theory".) A sentence $F \in$ **Sen**(Σ) is a (Σ, Θ)-**theorem** iff for all models $M \in$ **Mod**(Σ) we have that $M \models_\Sigma A$ for all $A \in \Theta$ implies $M \models_\Sigma F$. This is the model theoretical consequence relation and written as $\Theta \models_\Sigma F$.

The above concepts have a second dimension when all classes are extended to categories (Mac Lane (1998)). **Sig** is in fact a category, and **Sen** a functor **Sig** $\to \mathcal{SET}$. Similarly, **Mod** is a functor **Sig** $\to \mathcal{CAT}^{op}$ (see also Not. 1). Signature morphisms represent translations between signatures, and the actions of **Sen** and **Mod** on signature morphism extend these translations to sentences and models. Sentence and model translations go in opposite directions, which corresponds to an adjunction between syntax and semantics. Finally $\models_\Sigma$ is subject to the satisfaction condition, which guarantees that the satisfaction of a sentence in a model is invariant under signature morphisms.

Example 2 (Continued). In the case of modal logic, signature morphisms $\sigma : \Sigma \to \Sigma'$ are substitutions of Σ'-sentences for the propositional variables in Σ, and **Sen**$^{\mathbb{ML}}(\sigma) :$ **Sen**$^{\mathbb{ML}}(\Sigma) \to$ **Sen**$^{\mathbb{ML}}(\Sigma')$ is the induced homomorphic extension. The functor **Mod**$^{\mathbb{ML}}(\sigma)$

reduces a model $(W, \prec, V') \in \mathbf{Mod}^{\mathbb{ML}}(\Sigma')$ to the model $(W, \prec, V) \in \mathbf{Mod}^{\mathbb{ML}}(\Sigma)$ by putting $V(p, w) = V'(\sigma(p), w)$.

A **theory morphism** $\sigma : (\Sigma, \Theta) \to (\Sigma', \Theta')$ is a signature morphism $\sigma : \Sigma \to \Sigma'$ such that for all axioms $A \in \Theta$, we have that $\mathbf{Sen}(\sigma)(A)$ is a (Σ', Θ')-theorem. This implies that $\mathbf{Sen}(\sigma)$ maps (Σ, Θ)-theorems to (Σ', Θ')-theorems, i.e., theory morphisms preserve truth. Adjointly, one can show that σ is a theory morphism iff $\mathbf{Mod}(\sigma)$ maps (Σ', Θ')-models to (Σ, Θ)-models.

Institutions can be described elegantly as functors out of a signature category into a fixed category C of *rooms*. This yields a notion of institution translations via the well-known notion of the lax slice category of functors into C. Such translations are called institution **comorphisms** (Goguen and Rosu (2002); Meseguer (1989); Tarlecki (1996)). This is the basis of a number of logic translations that have been expressed as translations between institutions (see, e.g., Mossakowski et al. (2007) for examples).

A comorphism from $\mathbb{I}$ to $\mathbb{I}'$ translates signatures and sentences from $\mathbb{I}$ to $\mathbb{I}'$ and models in the opposite direction. The sentence translation preserves the (model theoretical) consequence relation $\Theta \models_\Sigma F$. If the model translations are surjective – called the model expansion property – the consequence relation is also reflected. This is the basis of the important borrowing application (Cerioli and Meseguer (1997)), where the consequence relation of one institution is reduced to that of another, e.g., one for which an implementation is available.

In fact, very few logic translations can be expressed as comorphisms in this sense – instead, it is often necessary to translate $\mathbb{I}$-signatures to $\mathbb{I}'$-theories. However, such translations can be represented as institution comorphisms from $\mathbb{I}$ to $Th(\mathbb{I}')$. Here $Th(\mathbb{I}')$ is a new institution whose signatures are the theories of $\mathbb{I}'$, a straightforward construction that is possible for all institutions.

2.2. *The Proof Theoretical Framework LF*

LF (Harper et al. (1993)) is a dependent type theory related to Martin-Löf type theory (Martin-Löf (1974)). It is the corner of the λ-cube (Barendregt (1992)) that extends simple type theory with dependent function types. We will work with the Twelf implementation of LF (Pfenning and Schürmann (1999)).

The main use of LF and Twelf is as a **logical framework** in which deductive systems are represented. Typically, **kinded type families** are declared to represent the syntactic classes of the system. For example, to represent higher-order logic, we use a signature HOL with declarations

tp	:	$\texttt{type}$
tm	:	$tp \to \texttt{type}$
$bool$	:	tp
ded	:	$tm\ bool \to \texttt{type}$

Here $\texttt{type}$ is the LF-kind of types, and tp is an LF-type whose LF-terms represent the STT-types. $tp \to \texttt{type}$ is the kind of type families that are indexed by terms of

LF-type *tp*; then *tm A* is the LF-type whose terms represent the STT-terms of type *A*. For example, *bool* represents the HOL-type of propositions such that HOL-propositions are represented as LF-terms of type *tm bool*. LF employs the Curry-Howard correspondence to represent proofs-as-term (Curry and Feys (1958); Howard (1980)) and extends it to the **judgments-as-types** methodology (Martin-Löf (1996)): *ded* declares the truth judgment on HOL-propositions, and the type *ded F* represents the judgment that *F* is derivable. HOL-derivations of *F* are represented as LF-terms of type *ded F*, and *F* is provable in HOL iff there is an LF term of type *ded F*.

Additionally, **typed constants** are declared to construct expressions of these syntactic classes, i.e., types (e.g., *bool* above), terms, propositions, and derivations of HOL. For example, consider:

$$
\begin{array}{lll}
\Rightarrow & : & tm\ bool \to tm\ bool \to tm\ bool \\
\Rightarrow_E & : & \{F : bool\}\ \ \{G : bool\}\ \ ded\ (F \Rightarrow G)\ \ \to\ \ ded\ F\ \ \to\ \ ded\ G \\
\forall & : & \{A : tp\}\ \ (tm\ A \to bool)\ \ \to\ \ bool
\end{array}
$$

Here $\Rightarrow$ encodes implication as a binary proposition constructor. $\Rightarrow_E$ encodes the implication elimination rule (modus ponens): It takes first two propositions *F* and *G* as arguments, and then two derivations of $F \Rightarrow G$ and *F*, respectively, and returns a derivation of *G*. Note that the types of the later arguments depend on the values *F* and *G* of the first two arguments. The encoding of the universal quantifier $\forall$ uses **higher-order abstract syntax** to declare binders as constants: LF terms of type $S \to T$ are in bijection to LF terms of type *T* with a free variable of type *S*. Thus, $\forall$ encodes a proposition constructor that takes a HOL type *A* and a proposition with a free variable of type *tm A*.

We will always use Twelf notation for the LF primitives of binding and application: The type $\Pi_{x:A} B(x)$ of dependent functions taking $x : A$ to an element of $B(x)$ is written $\{x : A\}\, B\ x$, and the function term $\lambda_{x:A} t(x)$ taking $x : A$ to $t(x)$ is written $[x : A]\, t\ x$. As usual, we write $A \to B$ instead of $\{x : A\}\, B$ if x does not occur in B.

This yields the following grammar for the three levels of LF expressions:

$$
\begin{array}{llll}
\text{Kinds:} & K & ::= & \texttt{type}\,|\,\{x : A\}\, K \\
\text{Type families:} & A, B & ::= & a\,|\,A\ t\,|\,[x : A]\, B\,|\,\{x : A\}\, B \\
\text{Terms:} & s, t & ::= & c\,|\,x\,|\,[x : A]\, t\,|\,s\ t
\end{array}
$$

$\{x : A\}\, K$ and $\{x : A\}\, B$ are used to abstract over variables occurring in kinds and types, respectively. Similarly, there are two λ-abstractions and two applications. The three productions for kind level abstraction are given in gray.

The **judgments** are $\Gamma \vdash_\Sigma K$ `kind` for well-formed kinds, $\Gamma \vdash_\Sigma A : K$ for well-kinded type families, and $\Gamma \vdash_\Sigma s : A$ for well-typed terms. They are defined relative to a signature Σ and a context Γ, which declare the symbols and variables, respectively, that can occur in expressions. Moreover, all levels come with an **equality** judgment, and both abstractions respect α, β, and η-conversion. All judgments are decidable Harper et al. (1993).

The grammar for signatures and contexts and their translations is as follows:

Signatures	Σ	::=	$\cdot \mid \Sigma,\ c : A \mid \Sigma,\ a : K,$
Morphisms	σ	::=	$\cdot \mid \sigma,\ c := t \mid \Sigma,\ a := A$
Contexts	Γ	::=	$\cdot \mid \Gamma,\ x : A$
Substitutions	σ	::=	$\cdot \mid \gamma,\ x := t$

An LF **signature** Σ holds the globally available declared names; it is a list of kinded type family declarations $a : K$ and typed constant declarations $c : A$. Similarly, a Σ-**context** holds the locally available declared names; it is a list typed variable declarations $x : A$.

Both signatures and contexts come with a notion of homomorphic translations. Given two signatures Σ and Σ', a **signature morphism** $\sigma : \Sigma \to \Sigma'$ is a typing- and kinding-preserving map of Σ-symbols to closed Σ'-expressions. Thus, σ maps every constant $c : A$ of Σ to a term $\sigma(c) : \overline{\sigma}(A)$ and every type family symbol $a : K$ to a type family $\sigma(a) : \overline{\sigma}(K)$. Here, $\overline{\sigma}$ is the homomorphic extension of σ to all closed Σ-expressions, and we will write σ instead of $\overline{\sigma}$ from now on.

Signature morphisms preserve typing, kinding, and equality of expressions, i.e., if $\cdot \vdash_\Sigma E : F$, then $\cdot \vdash_{\Sigma'} \sigma(E) : \sigma(F)$, and similarly for equality. In particular, because σ must map all axioms or inference rules declared in Σ to proofs or derived inference rules, respectively, over Σ', signature morphisms preserve the provability of judgments. Two signature morphisms are equal if they map the same constants to $\alpha\beta\eta$-equal expressions. Up to this equality, signatures and signature morphisms form a category, which we denote by $\mathbb{LF}$.

$\mathbb{LF}$ has inclusion morphisms $\Sigma \hookrightarrow \Sigma, \Sigma'$ and pushouts along inclusions (Harper et al. (1994)). Moreover, a coherent system of pushouts along inclusions can be chosen canonically: Given $\sigma : \Sigma \to \Sigma'$ and an inclusion $\Sigma \hookrightarrow \Sigma,\ c : A$, the canonical **pushout** is given by

$$\sigma,\ c := c\ :\ \Sigma,\ c : A\ \to\ \Sigma',\ c : \sigma(A)$$

(except for possibly renaming c if it is not fresh for Σ'). The canonical choices for pushouts along other inclusions are obtained accordingly.

Similarly, given two Σ-contexts Γ and Γ', a **substitution** $\gamma : \Gamma \to \Gamma'$ is a typing-preserving map of Γ-variables to Γ'-expressions (which leaves the symbols of Σ fixed). Thus, σ maps every variable $x : A$ of Γ to a term $\gamma(x) : \overline{\gamma}(A)$. Again $\overline{\gamma}$ is the homomorphic extension of γ, and we write γ instead of $\overline{\gamma}$. Similar to signature morphisms, substitutions preserve typing, kinding, and equality of expressions in contexts, e.g., if $\Gamma \vdash_\Sigma E : F$, then $\Gamma' \vdash_\Sigma \gamma(E) : \gamma(F)$. Two substitutions are equal if they map the same variables to $\alpha\beta\eta$-equal terms. Up to this equality, for a fixed signature Σ, the contexts over Σ and the substitutions between them form a category.

Finally a signature morphism $\sigma : \Sigma \to \Sigma'$ can be applied component-wise to contexts and substitutions: $\sigma(\cdot) = \cdot$ and $\sigma(\Gamma,\ x : A) = \sigma(\Gamma),\ x : \sigma(A)$ for contexts, and $\sigma(\cdot) = \cdot$, and $\sigma(\gamma,\ x := t) = \sigma(\gamma),\ x := \sigma(t)$ for substitutions. We have the invariant that if $\gamma : \Gamma \to \Gamma'$ over Σ and $\sigma : \Sigma \to \Sigma'$, then $\sigma(\gamma) : \sigma(\Gamma) \to \sigma(\Gamma')$. Categorically, $\sigma(-)$ is a functor from the category of Σ-contexts to the category of Σ'-contexts.

3. A Comprehensive Logical Framework

Our primary objective is to encode logics and logic translations in an LF-based meta-logic. Therefore, we must do three things (i) define what logics and logic translations are, which we do in Sect. 3.1 and 3.2, respectively, (ii) define what it means to encode them in a meta-logic, which we do in Sect. 3.3, and (iii) give the LF-based meta-logic, which we do in Sect. 4. Note that (ii) and (iii) could also be given the other way round. We give (ii) first for an arbitrary meta-logic so that it is possible to substitute other meta-logics later.

3.1. *Logics*

Model Theories and Institutions In order to give a unified approach to model and proof theory, we will refactor the definition of institutions by splitting an institution into syntax and model theory:

Definition 1 (Logic Syntax). A **logic syntax** is a pair $(\mathbf{Sig}, \mathbf{Sen})$ such that $\mathbf{Sig}$ is a category and $\mathbf{Sen} : \mathbf{Sig} \to \mathcal{SET}$ is a functor.

Definition 2 (Model Theory). For a logic syntax $(\mathbf{Sig}, \mathbf{Sen})$, a **model theory** is a tuple $(\mathbf{Mod}, \models)$ such that $\mathbf{Mod} : \mathbf{Sig} \to \mathcal{CAT}^{op}$ is a functor and $\models_\Sigma \subseteq \mathbf{Sen}(\Sigma) \times |\mathbf{Mod}(\Sigma)|$ is family of relations such that the satisfaction condition holds: For all $\sigma : \Sigma \to \Sigma'$, $F \in \mathbf{Sen}(\Sigma)$, and $M' \in |\mathbf{Mod}(\Sigma')|$, we have $\mathbf{Mod}(\sigma)(M') \models_\Sigma F$ iff $M' \models_{\Sigma'} \mathbf{Sen}(\sigma)(F)$.

In particular, institutions are exactly the pairs of a logic syntax and a model theory for it.

Notation 1. In the literature, the model theory functor is usually given as $\mathbf{Mod} : \mathbf{Sig}^{op} \to \mathcal{CAT}$. We choose the dual orientation here in order to emphasize the symmetry between model and proof theory later on.

Example 3 (Modal Logic). $(\mathbf{Sig}^{\mathbb{ML}}, \mathbf{Sen}^{\mathbb{ML}})$ and $(\mathbf{Mod}^{\mathbb{ML}}, \models^{\mathbb{ML}})$ from Ex. 1 are examples of a logic syntax and a model theory for it. To complete the example, we have to show the satisfaction condition: Assume $\sigma : \Sigma \to \Sigma'$, $F \in \mathbf{Sen}^{\mathbb{ML}}(\Sigma)$, and $(W, \prec, V) \in \mathbf{Mod}^{\mathbb{ML}}(\Sigma')$; then the satisfaction condition follows if we show that $V(F, w) = V'(\mathbf{Sen}^{\mathbb{ML}}(\sigma)(F), w)$, which is easy to see.

Note that we proved that σ preserves truth in all worlds, whereas the satisfaction condition only requires the weaker condition that $\mathbf{Sen}^{\mathbb{ML}}(\sigma)(F)$ holds in all worlds of $(W, \prec, V)$ iff F holds in all worlds of $(W, \prec, V')$. This extension of the satisfaction condition to all components of syntax and model theory is typical but different for each institution. A uniform formulation was given in Aiguier and Diaconescu (2007) using stratified institutions.

Proof Theories and Logics Like in the case of model theories, for a given logic syntax, we will define proof theories as pairs $(\mathbf{Pf}, \vdash)$. First, we define proof categories, the values of the $\mathbf{Pf}$ functor:

Definition 3 (Proof Categories). A **proof category** is a category P with finite products (including the empty product). $\mathcal{PFCAT}$ is the category of proof categories together with the functors preserving finite products.

Notation 2. If a category has the product $(F_1, \ldots, F_n)$, we write it as $(F_i)_1^n$ or simply F_1^n. Similarly, for $p_i : E \to F_i$, we write p_1^n for the universal morphism $(p_1, \ldots, p_n) : E \to F_1^n$.

We think of the objects of P as **judgments** about the syntax that can be assumed and derived. The intuition of a morphism p_1^n from E_1^m to F_1^n is that it provides **evidence** p_i for each judgment F_i under the assumptions $E_1, \ldots, E_m$.

Example 4 (Continued). We obtain a proof theory functor $\mathbf{Pf}^{\mathbb{ML}} : \mathbf{Sig}^{\mathbb{ML}} \to \mathcal{PFCAT}$ as follows. The proof category $\mathbf{Pf}^{\mathbb{ML}}(\Sigma)$ is the category of finite families F_1^n for $F_i \in \mathbf{Sen}^{\mathbb{ML}}(\Sigma)$. The morphisms in $\mathbf{Pf}^{\mathbb{ML}}(\Sigma)$ from E_1^m to F_1^n are the families p_1^n such that every p_i is a proof of F_i from assumptions $E_1, \ldots, E_m$. We can choose any calculus to make the notion of proofs precise, e.g., a Hilbert calculus with rules for modus ponens and necessitation.

$\mathbf{Pf}^{\mathbb{ML}}(\Sigma)$ has finite products by taking products of families. The action of $\mathbf{Pf}^{\mathbb{ML}}$ on signature morphisms $\sigma : \Sigma \to \Sigma'$ is induced in the obvious way.

Remark 1 (Proof Categories). By assuming proof categories to have products in Def. 7, we are implicitly assuming the rules of weakening (given by morphisms from (A, B) to A), contraction (given by morphisms from A to (A, A)), and exchange (given by morphisms from (A, B) to (B, A)). A substructural framework, e.g., one akin to linear logic (Girard (1987)), could be obtained by weakening the condition on the existence of products.

We only require finite products because deductive systems typically use words over countable alphabets, in which only finite products of judgments can be expressed. This amounts to a restriction to compact deductive systems, which corresponds to the constructive flavor of proof theory.

It is natural to strengthen the definition by further assuming that all objects of a proof category P can be written as products of irreducible objects, in which case proof categories become many-sorted Lawvere categories (Lawvere (1963)). Then, up to isomorphism, the judgments can be seen as the finite multi-sets of irreducible objects, and canonical products are obtained by taking unions of multi-sets. This is the typical situation, but we do not assume it here because we will not make use of this additional assumption.

An even stronger definition could require that the irreducible objects of $\mathbf{Pf}(\Sigma)$ are just the sentences in $\mathbf{Sen}(\Sigma)$. This is the case in Ex. 4 where the irreducible objects are the singleton multi-sets and thus essentially the sentences. But it is desirable to avoid this assumption in order to include deductive systems where auxiliary syntax is introduced, such as hypothetical judgments (as used in natural deduction, see Ex. 5) or signed formulas (as used in tableaux calculi).

Example 5. The proof theory of the first-order natural deduction calculus arises as follows. For a $\mathbb{FOL}$-signature Σ, an atomic Σ-judgment is of the form $\Gamma; \Delta \vdash F$ where Γ is a context declaring some free variables that may occur in Δ and F; Δ is a list of Σ-formulas; and F is a Σ-formula. The objects of $\mathbf{Pf}^{\,\mathbb{FOL}}(\Sigma)$ are the multi-sets of atomic Σ-judgments, and products are given by unions of multi-sets. A morphism in $\mathbf{Pf}^{\,\mathbb{FOL}}(\Sigma)$ from J_1^m to a singleton multi-set (K) is a natural deduction proof tree whose root is labelled with K and whose leaves are labelled with one element of $J_1, \ldots, J_m$. Morphisms with other codomains are obtained from the universal property of the product.

Giving a proof category $\mathbf{Pf}\,(\Sigma)$ for a signature Σ is not sufficient to define a proof theoretical semantics of Σ — we also have to relate the Σ sentences and the Σ-judgments. For that purpose, we use a map $\vdash_\Sigma\colon \mathbf{Sen}(\Sigma) \to \mathbf{Pf}\,(\Sigma)$ assigning to every sentence its truth judgment. In particular, $\mathbf{Pf}\,(\Sigma)$-morphisms from the empty family () to $\vdash_\Sigma F$ represent the proofs of F. Corresponding to the satisfaction condition, the truth judgment should be preserved under signature morphisms. This yields the following definition:

Definition 4 (Proof Theory). For a logic syntax $(\mathbf{Sig}, \mathbf{Sen})$, a **proof theory** is a tuple $(\mathbf{Pf}, \vdash)$ such that $\mathbf{Pf} : \mathbf{Sig} \to \mathcal{PFCAT}$ is a functor and $\vdash_\Sigma$ is a mapping from $\mathbf{Sen}(\Sigma)$ to $\mathbf{Pf}\,(\Sigma)$ such that for all $\sigma : \Sigma \to \Sigma'$ and $F \in \mathbf{Sen}(\Sigma)$, we have $\mathbf{Pf}\,(\sigma)(\vdash_\Sigma F) = \vdash_{\Sigma'} \mathbf{Sen}(\sigma)(F)$.

Finally, we can put together syntax, model theory, and proof theory to form a logic:

Definition 5 (Logics). A **logic** is a tuple $\mathbb{I} = (\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models, \mathbf{Pf}, \vdash)$ such that $(\mathbf{Sig}, \mathbf{Sen})$ is a logic syntax and $(\mathbf{Mod}, \models)$ and $(\mathbf{Pf}, \vdash)$ are a model theory and a proof theory, respectively, for it.

Example 6 (Continued). We obtain a proof theory and thus a logic (in our formal sense) for modal logic by using $\mathbf{Pf}^{\,\mathbb{ML}}$ as in Ex. 4 and putting $\vdash_\Sigma^{\mathbb{ML}} F = (F)$ where (F) is the family containing a single element F. Indeed, we have $\mathbf{Pf}^{\,\mathbb{ML}}(\sigma)(\vdash_\Sigma^{\mathbb{ML}} F) = (\mathbf{Sen}^{\mathbb{ML}}(\sigma)(F)) = \vdash_{\Sigma'}^{\mathbb{ML}} \mathbf{Sen}^{\mathbb{ML}}(\sigma)(F)$.

Thus, a logic consists of a category of signatures $\mathbf{Sig}$, a sentence functor $\mathbf{Sen}$, a model theory functor $\mathbf{Mod}$, a proof theory functor $\mathbf{Pf}$, and model and proof theoretical definitions of truth $\models$ and $\vdash$, respectively. This is visualized on the right for a signature morphism $\sigma : \Sigma \to \Sigma'$. Technically, this is not a diagram in the sense of category theory. But if we

$$
\begin{array}{ccc}
\mathbf{Mod}(\Sigma) \xleftarrow{\ \mathbf{Mod}(\sigma)\ } \mathbf{Mod}(\Sigma') & \mathcal{CAT} \\[4pt]
\models_\Sigma \Big\downarrow \qquad\qquad\qquad \Big\downarrow \models_{\Sigma'} & \\[4pt]
\mathbf{Sen}(\Sigma) \xrightarrow{\ \mathbf{Sen}(\sigma)\ } \mathbf{Sen}(\Sigma') & \mathcal{SET} \\[4pt]
\vdash_\Sigma \Big\downarrow \qquad\qquad\qquad \Big\downarrow \vdash_{\Sigma'} & \\[4pt]
\mathbf{Pf}\,(\Sigma) \xrightarrow{\ \mathbf{Pf}\,(\sigma)\ } \mathbf{Pf}\,(\Sigma') & \mathcal{PFCAT}
\end{array}
$$

treat the sets $\mathbf{Sen}(\Sigma)$ and $\mathbf{Sen}(\Sigma')$ as discrete categories, then the lower half is a commuting diagram of categories. Moreover, if we forget morphisms and treat functors as special cases of relations between classes, then the upper half is a commutative diagram in the category of classes and relations. This is made more precise in the following remark.

Remark 2. The definition of logics can be phrased more categorically, which also makes the formal symmetry between model and proof theory more apparent. Let $\mathcal{CLASS}$ and $\mathcal{REL}$ be the categories of classes with mappings and relations, respectively, and let $|-|:\mathcal{CAT}^{op}\rightarrow\mathcal{CLASS}$ and $|-|^r:\mathcal{CAT}^{op}\rightarrow\mathcal{REL}$ be the functors forgetting morphisms. Let us also identify $\mathcal{SET}$ with its two inclusions into $\mathcal{REL}$ and $\mathcal{PFCAT}$. Then $\models$ and $\vdash$ are natural transformations $\models: \mathbf{Sen}\rightarrow|-|^r\circ\mathbf{Mod}$ and $\vdash:\mathbf{Sen}\rightarrow|-|\circ\mathbf{Pf}$.

Then a logic $(\mathbf{Sig},\mathbf{Sen},\mathbf{Mod},\models,\mathbf{Pf},\vdash)$ yields the following diagram in the category $\mathcal{CAT}$, where double arrows indicate natural transformations between the functors making up the surrounding rectangles:

$$
\begin{array}{ccc}
 & \mathcal{CAT}^{op} \xrightarrow{\;|-|^r\;} \mathcal{REL} & \\
\mathbf{Mod}\nearrow & \Big\Uparrow \models \nearrow & \\
\mathbf{Sig}\xrightarrow{\mathbf{Sen}} \mathcal{SET} & & \\
\mathbf{Pf}\searrow & \Big\Downarrow \vdash \searrow & \\
 & \mathcal{PFCAT}\xrightarrow[|-|]{} \mathcal{CLASS} &
\end{array}
$$

Remark 3. Even more categorically than in Rem. 2, assume $U:\mathcal{SET}\rightarrow\mathcal{REL}\times\mathcal{CLASS}$ maps a set S to (S,S), and $V:\mathcal{CAT}^{op}\times\mathcal{PFCAT}\rightarrow\mathcal{REL}\times\mathcal{CLASS}$ maps a pair of a model category and a proof category to itself but forgetting morphisms. Then logics are the functors from some category of signatures $\mathbf{Sig}$ to the comma category $(U\downarrow V)$.

Theories and Consequence While the notion of theories only depends on a logic syntax, both the model theoretical semantics and the proof theoretical semantics induce one consequence relation each: As usual, we write $\models$ and $\vdash$ for the model and proof theoretical consequence, respectively. Consequently, we obtain two notions of theory morphisms.

Definition 6 (Theories). Given a logic syntax $(\mathbf{Sig},\mathbf{Sen})$, a **theory** is a pair (Σ,Θ) for $\Sigma\in\mathbf{Sig}$ and $\Theta\subseteq\mathbf{Sen}(\Sigma)$. The elements of Θ are called the **axioms** of (Σ,Θ).

Definition 7 (Consequence). Assume a logic $\mathbb{I}=(\mathbf{Sig},\mathbf{Sen},\mathbf{Mod},\models,\mathbf{Pf},\vdash)$. For a theory (Σ,Θ) and a Σ-sentence F, we define **consequence** as follows:

— Θ **entails** F proof theoretically, written $\Theta\vdash^{\mathbb{I}}_{\Sigma}F$, iff there exist a finite subset $\{F_1,\ldots,F_n\}\subseteq\Theta$ and a $\mathbf{Pf}(\Sigma)$-morphism from $(\vdash^{\mathbb{I}}_{\Sigma}F_i)^n_1$ to $\vdash^{\mathbb{I}}_{\Sigma}F$,

— Θ **entails** F model theoretically, written $\Theta\models^{\mathbb{I}}_{\Sigma}F$, iff every model $M\in\mathbf{Mod}(\Sigma)$ satisfying all sentences in Θ also satisfies F.

We will drop the superscript $\mathbb{I}$ if it is clear from the context.

A signature morphism $\sigma:\Sigma\rightarrow\Sigma'$ is called a model theoretical (or proof theoretical) **theory morphism** from (Σ,Θ) to (Σ',Θ') if for all $F\in\Theta$, we have $\Theta'\models_{\Sigma'}\mathbf{Sen}(\sigma)(F)$ (or $\Theta'\vdash_{\Sigma'}\mathbf{Sen}(\sigma)(F)$ in case of proof theoretical theory morphisms).

Terminology 1. In the literature, the term *theory* is sometimes used only for a set of sentences that is closed under the consequence relation. In that case theories in our sense are called *presentations*.

Theory morphisms preserve the respective consequence relation:

Lemma 1 (Truth Preservation). Assume a (i) model theoretical or (ii) proof theoretical theory morphism $\sigma : (\Sigma, \Theta) \to (\Sigma', \Theta')$ and a sentence $F \in \mathbf{Sen}(\Sigma)$. Then in case

(i): $\Theta \models_\Sigma F$ implies $\Theta' \models_{\Sigma'} \mathbf{Sen}(\sigma)(F)$,
(ii): $\Theta \vdash_\Sigma F$ implies $\Theta' \vdash_{\Sigma'} \mathbf{Sen}(\sigma)(F)$.

The well-known concepts of soundness and completeness relate the two consequence relations:

Definition 8 (Soundness and Completeness). A logic is **sound** iff $\varnothing \vdash_\Sigma F$ implies $\varnothing \models_\Sigma F$ for all signatures Σ and all sentences F. It is **strongly sound** iff $\Theta \vdash_\Sigma F$ implies $\Theta \models_\Sigma F$ for all theories (Σ, Θ) and all sentences F. Similarly, a logic is **strongly complete** or **complete** if the respective converse implication holds.

In particular, if a logic is strongly sound and strongly complete, then proof and model theoretical consequence and theory morphisms coincide.

3.2. *Logic Translations*

Logic Translations We will now define translations between two logics. Since our framework is based on institutions, we will build upon the existing notion of an institution *comorphism*.

Definition 9 (Logic Comorphism). Assume two logics $\mathbb{I} = (\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models, \mathbf{Pf}, \vdash)$ and $\mathbb{I}' = (\mathbf{Sig}', \mathbf{Sen}', \mathbf{Mod}', \models', \mathbf{Pf}', \vdash')$. A **logic comorphism** from $\mathbb{I}$ to $\mathbb{I}'$ is a tuple $(\Phi, \alpha, \beta, \gamma)$ consisting of a functor $\Phi : \mathbf{Sig} \to \mathbf{Sig}'$ and natural transformations $\alpha : \mathbf{Sen} \to \mathbf{Sen}' \circ \Phi$, $\beta : \mathbf{Mod} \to \mathbf{Mod}' \circ \Phi$, and $\gamma : \mathbf{Pf} \to \mathbf{Pf}' \circ \Phi$ such that

1. for all $\Sigma \in \mathbf{Sig}$, $F \in \mathbf{Sen}(\Sigma)$, $M' \in \mathbf{Mod}'(\Phi(\Sigma))$:

$$\beta_\Sigma(M') \models_\Sigma F \quad \text{iff} \quad M' \models'_{\Phi(\Sigma)} \alpha_\Sigma(F),$$

2. for all $\Sigma \in \mathbf{Sig}$, $F \in \mathbf{Sen}(\Sigma)$:

$$\gamma_\Sigma(\vdash_\Sigma F) \;=\; \vdash'_{\Phi(\Sigma)} \alpha_\Sigma(F).$$

With the obvious choices for identity and composition, we obtain the category $\mathcal{LOG}$ of logics and comorphisms.

Recall that β_Σ is a morphism in the category $\mathcal{CAT}^{op}$ and thus the same as a functor $\beta_\Sigma : \mathbf{Mod}'(\Phi(\Sigma)) \to \mathbf{Mod}(\Sigma)$; thus, $\beta_\Sigma(M')$ is a well-formed functor application. Note that the syntax and the proof theory are translated from $\mathbb{I}$ to $\mathbb{I}'$, whereas the model theory is translated in the opposite direction. The two conditions on comorphisms are

truth preservation conditions: The model and proof theory translations must preserve model and proof theoretical truth, respectively.

Just like logics can be decomposed into syntax, model theory, and proof theory, a logic comorphism consists of a

— a syntax translation $(\Phi, \alpha) : (\mathbf{Sig}, \mathbf{Sen}) \to (\mathbf{Sig}', \mathbf{Sen}')$,

— for a given syntax translation (Φ, α), a model translation $\beta : (\mathbf{Mod}, \models) \to (\mathbf{Mod}', \models')$ which must satisfy condition 1. ,

— for a given syntax translation (Φ, α), a proof translation $\gamma : (\mathbf{Pf}, \vdash) \to (\mathbf{Pf}', \vdash')$ which must satisfy condition 2. .

Remark 4. Continuing Rem. 3, we observe that if logics are slices $\mathbb{I}_i : \mathbf{Sig}_i \to (U \downarrow V)$, then a logic comorphisms from $\mathbb{I}_1$ to $\mathbb{I}_2$ is a functor $\Phi : \mathbf{Sig}_1 \to \mathbf{Sig}_2$ together with a natural transformation from $\mathbb{I}_1$ to $\mathbb{I}_2 \circ \Phi$. Thus, $\mathcal{LOG}$ can be seen as the lax slice category of objects over $(U \downarrow V)$.

Comorphism Modifications The category of logics and logic comorphisms can be turned into a 2-category. Again we are inspired by the existing notion for institutions, for which the 2-cells are called institution *comorphism modifications* (Diaconescu (2002)):

Definition 10 (Modifications). Assume two logic comorphisms $\mu^i = (\Phi^i, \alpha^i, \beta^i, \gamma^i) : \mathbb{I} \to \mathbb{I}'$ for $i = 1, 2$. A **comorphism modification** from μ^1 to μ^2 is a natural transformation $m : \Phi_1 \to \Phi_2$ such that the following diagrams commute:

where the diagram for the model theory is drawn in $\mathcal{CAT}$.

Thus, a comorphism modification m is a family $(m_\Sigma)_{\Sigma \in \mathbf{Sig}}$ of $\mathbb{I}'$-signature morphisms $m_\Sigma : \Phi^1(\Sigma) \to \Phi^2(\Sigma)$. It can be understood as modifying μ_1 in order to make it equal to μ_2: For example, the sentence translations $\alpha_\Sigma^1 : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\Phi^1(\Sigma))$ and $\alpha_\Sigma^2 :$

$\mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\Phi^2(\Sigma))$ may be quite different. Yet, by composing α_Σ^1 with $\mathbf{Sen}'(m_\Sigma)$, the difference can be bridged. We obtain a comorphism modification if syntax, model, and proof translation can be bridged uniformly, i.e., by using the respective translations induced by m_Σ.

3.3. *Meta-Logics*

Logic comorphisms $\mathbb{I} \to \mathbb{I}'$ permit representing a logic $\mathbb{I}$ in a logic $\mathbb{I}'$. An important special case arises when this representation is **adequate** in the intuitive sense that the semantics of $\mathbb{I}$ is preserved when representing it in $\mathbb{I}'$. Then our understanding of $\mathbb{I}'$ can be applied to study properties of $\mathbb{I}$. If we use a fixed logic $\mathbb{M} = \mathbb{I}'$, in which multiple other logics are represented adequately, we speak of **logic encodings** in a **meta-logic**. Occasionally, the term "universal logic" is used instead of "meta-logic", e.g., in Tarlecki (1996), but we avoid it here to prevent confusion with the general field of "universal logic", which investigates common structures of all logics.

Alternatively, instead of giving a logic comorphism from a given logic $\mathbb{I}$ to $\mathbb{M}$, we can start with a partial logic – e.g., an institution or even only a category $\mathbf{Sig}$ – and translate only that into $\mathbb{M}$. The missing components of a logic can then be inherited – often called **borrowed** – from $\mathbb{M}$. We speak of **logic definitions**.
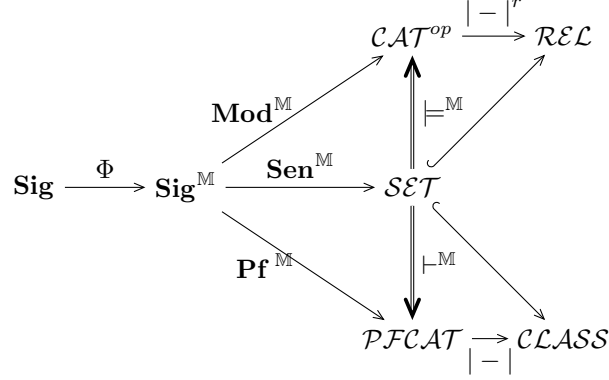
Similarly, a meta-logic can be applied to define or encode logic comorphisms so that we have four cases in the end: defining/encoding a logic (comorphism). In the following we will derive the general properties of these four concepts for an arbitrary meta-logic $\mathbb{M}$. Then Sect. 4 will give the concrete definition of our meta-logic $\mathbb{M}$ as well as examples.

Defining Logics Given a signature category $\mathbf{Sig}$, to **define a logic** in a meta-logic $\mathbb{M}$ means to relate $\mathbf{Sig}$ to $\mathbb{M}$ in such a way that the syntax, model theory, and proof theory of $\mathbb{M}$ can be borrowed to extend $\mathbf{Sig}$ to a logic.

Notation 3. If $\models$ is a family $(\models_\Sigma)_{\Sigma \in \mathbf{Sig}}$ and $\Phi : \mathbf{Sig}' \to \mathbf{Sig}$ is a functor, then we write $\models \circ \Phi$ for the family $(\models_{\Phi(\Sigma)})_{\Sigma \in \mathbf{Sig}'}$. $\vdash \circ \Phi$ is defined accordingly. Considering Rem. 2, this is simply the composition of a functor and a natural transformation.

Definition 11. Let $\mathbb{M}$ be a logic. A **logic definition** in $\mathbb{M}$ consists of a category $\mathbf{Sig}$ and a functor $\Phi : \mathbf{Sig} \to \mathbf{Sig}^\mathbb{M}$. Such a logic definition induces a logic $\mathbb{M} \circ \Phi = (\mathbf{Sig}, \mathbf{Sen}^\mathbb{M} \circ \Phi, \mathbf{Mod}^\mathbb{M} \circ \Phi, \models^\mathbb{M} \circ \Phi, \mathbf{Pf}^\mathbb{M} \circ \Phi, \vdash^\mathbb{M} \circ \Phi)$.

Remark 5. With the notation of Rem. 2, this yields the diagram

$$\begin{array}{ccc}
& \mathcal{CAT}^{op} \xrightarrow{\;|-|^r\;} \mathcal{REL} & \\
\mathbf{Mod}^{\mathbb{M}} \nearrow & \big\Vert \models^{\mathbb{M}} & \nearrow \\
\mathbf{Sig} \xrightarrow{\;\Phi\;} \mathbf{Sig}^{\mathbb{M}} \xrightarrow{\;\mathbf{Sen}^{\mathbb{M}}\;} \mathcal{SET} & & \\
\mathbf{Pf}^{\mathbb{M}} \searrow & \big\Vert \vdash^{\mathbb{M}} & \searrow \\
& \mathcal{PFCAT} \underset{|-|}{\to} \mathcal{CLASS} &
\end{array}$$

Moreover, Φ induces a canonical comorphism $\mathbb{M} \circ \Phi \to \mathbb{M}$.

Theorem 1 (Logic Definitions). $\mathbb{M} \circ \Phi$ is indeed a logic. Moreover, it is (strongly) sound or (strongly) complete if $\mathbb{M}$ is.

Proof. That $\mathbb{M} \circ \Phi$ is a logic follows immediately from the diagram in Rem. 5. In particular, $\models^{\mathbb{M}} \circ \Phi$ and $\vdash^{\mathbb{M}} \circ \Phi$ are natural transformation because they arise by composing a natural transformation with a functor.

It is inherent in Def. 11 that $\Theta \vdash_{\Sigma}^{\mathbb{M} \circ \Phi} F$ iff $\Theta \vdash_{\Phi(\Sigma)}^{\mathbb{M}} F$, and similarly for the entailment relation. That yields the (strong) soundness/completeness result. $\square$

Remark 6. In Rem. 3, we stated that institutions are functors into $(U \downarrow V)$, and thus in particular $\mathbb{M} : \mathbf{Sig}^{\mathbb{M}} \to (U \downarrow V)$. Then $\mathbb{M} \circ \Phi$ is indeed the composition of two functors (which justifies our notation).

Encoding Logics While a logic definition starts with a signature category and borrows all other notions from $\mathbb{M}$, a **logic encoding** starts with a whole logic $\mathbb{I} = (\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models$ $, \mathbf{Pf}, \vdash)$ and a functor $\Phi : \mathbf{Sig} \to \mathbf{Sig}^{\mathbb{M}}$:

Definition 12 (Logic Encoding). An **encoding** of a logic $\mathbb{I}$ in $\mathbb{M}$ is a logic comorphism $(\Phi, \alpha, \beta, \gamma) : \mathbb{I} \to \mathbb{M}$. (Equivalently, we can say that $(id, \alpha, \beta, \gamma)$ is a logic comorphism $\mathbb{I} \to \mathbb{M} \circ \Phi$.)

When encoding logics in a meta-logic, a central question is **adequacy**. Intuitively, adequacy means that the encoding is correct with respect to the encoded logic. More formally, a logic encoding yields two logics $\mathbb{I}$ and $\mathbb{M} \circ \Phi$, and adequacy is a statement about the relation between them.

Ideally, α, β, and γ are isomorphisms, in which case $\mathbb{I}$ and $\mathbb{M} \circ \Phi$ are isomorphic in the category $\mathcal{LOG}$. However, often it is sufficient to show that the consequence relations in $\mathbb{I}$ and $\mathbb{M} \circ \Phi$ coincide. This motivates the following definition and theorem:

Definition 13 (Adequate Encodings). In the situation of Def. 12, the encoding is called **adequate** if

— each β_{Σ} is surjective on objects (model theoretical adequacy),

— each γ_Σ is "surjective" on morphisms in the following sense: for any A, B, if there is a morphism from $\gamma_\Sigma(A)$ to $\gamma_\Sigma(B)$, then there is also one from A to B (proof theoretical adequacy).

Model theoretical adequacy is just the *model expansion property* (e.g., Cerioli and Meseguer (1997)); intuitively, it means that a property that holds for all models of $\mathbf{Mod}^{\mathbb{M}}(\Phi(\Sigma))$ can be used to establish the corresponding property for all models of Σ. Proof theoretical adequacy means that proofs found in $\mathbb{M} \circ \Phi$ give rise to corresponding proofs in $\mathbb{I}$. More precisely:

Theorem 2 (Adequacy). Assume an adequate logic encoding $(\Phi, \alpha, \beta, \gamma)$ of $\mathbb{I}$ in $\mathbb{M}$. Then for all $\mathbb{I}$-theories (Σ, Θ) and all Σ-sentences F:

$$\Theta \models^{\mathbb{I}}_\Sigma F \qquad \text{iff} \qquad \alpha_\Sigma(\Theta) \models^{\mathbb{M}}_{\Phi(\Sigma)} \alpha_\Sigma(F)$$

$$\Theta \vdash^{\mathbb{I}}_\Sigma F \qquad \text{iff} \qquad \alpha_\Sigma(\Theta) \vdash^{\mathbb{M}}_{\Phi(\Sigma)} \alpha_\Sigma(F)$$

In particular, $\mathbb{I}$ is strongly sound or complete if $\mathbb{M}$ is.

Proof. It is easy to show that the left-to-right implications hold for any logic comorphism.

The right-to-left implications are independent and only require the respective adequacy assumption. The model theoretical result is essentially the known borrowing result for institutions (see Cerioli and Meseguer (1997)). For the proof theoretical result, recall that γ_Σ is a functor $\mathbf{Pf}^{\mathbb{I}}(\Sigma) \to \mathbf{Pf}^{\mathbb{M}}(\Phi(\Sigma))$; since provability is defined by the existence of morphisms in the proof category, the proof theoretical adequacy yields the result.

Then the soundness and completeness results follow by using Thm. 1. □

Mixing Defining and Encoding Logic definitions, where we start with $\mathbf{Sig}$ and inherit all other components from $\mathbb{M}$, and logic encodings, where we start with a logic and encode all components in $\mathbb{M}$, are opposite extremes. We can also start, for example, with an institution $\mathbb{I} = (\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models)$ and give an institution comorphism $(\Phi, \alpha, \beta) : \mathbb{I} \to \mathbb{M}$: This encodes syntax and model theory of $\mathbb{I}$ in $\mathbb{M}$ and inherits the proof theory. Dually, we can start with syntax and proof theory $\mathbb{I} = (\mathbf{Sig}, \mathbf{Sen}, \mathbf{Pf}, \vdash)$ and give a comorphism $(\Phi, \alpha, \gamma) : \mathbb{I} \to \mathbb{M}$: This encodes syntax and proof theory and inherits the model theory.

The former of these two is often used to apply an implementation of the proof theoretical consequence relation of $\mathbb{M}$ to reason about the model theoretical consequence relation of $\mathbb{I}$, a technique known as *borrowing* (Cerioli and Meseguer (1997)). The borrowing theorem can be recovered as a special case of the above, and we briefly state it in our notation:

Theorem 3. Let $\mathbb{I}$ be an institution and $\mathbb{M}$ a logic, and let $\mu = (\Phi, \alpha, \beta)$ be an institution comorphism from $\mathbb{I}$ to $\mathbb{M}$ (qua institution). Then we extend $\mathbb{I}$ to a logic $\mathbb{I}'$ by putting $\mathbf{Pf}' := \mathbf{Pf}^{\mathbb{M}} \circ \Phi$ and $\vdash' := (\vdash^{\mathbb{M}} \circ \Phi) \circ \alpha$. Moreover, we have that $(\Phi, \alpha, \beta, id) : \mathbb{I}' \to \mathbb{M}$ is a logic comorphism that is an adequate encoding if all β_Σ are surjective on objects.

Proof. It follows from basic properties of category theory that $\mathbb{I}'$ is indeed a logic. Model

theoretical adequacy holds due to the surjectivity assumption, and proof theoretical follows because the proof translation is the identity. $\square$

Defining Logic Comorphisms Now we turn to the treatment of logic comorphisms. To define a logic comorphism we give a family of signature morphisms in $\mathbb{M}$:

Definition 14 (Defining Comorphisms). Assume two logic definitions $\Phi^i : \mathbf{Sig}^i \to \mathbf{Sig}^{\mathbb{M}}$ for $i = 1, 2$. A **comorphism definition** consists of a functor $\Phi : \mathbf{Sig}^1 \to \mathbf{Sig}^2$ and a natural transformation $m : \Phi^1 \to \Phi^2 \circ \Phi$. Such a comorphism definition induces a logic comorphism $(\Phi, \mathbb{M} \circ m) = (\Phi, \mathbf{Sen}^{\mathbb{M}} \circ m, \mathbf{Mod}^{\mathbb{M}} \circ m, \mathbf{Pf}^{\mathbb{M}} \circ m)$ from $\mathbb{M} \circ \Phi^1$ to $\mathbb{M} \circ \Phi^2$.

Remark 7. With the notation of Rem. 5, this yields the diagram
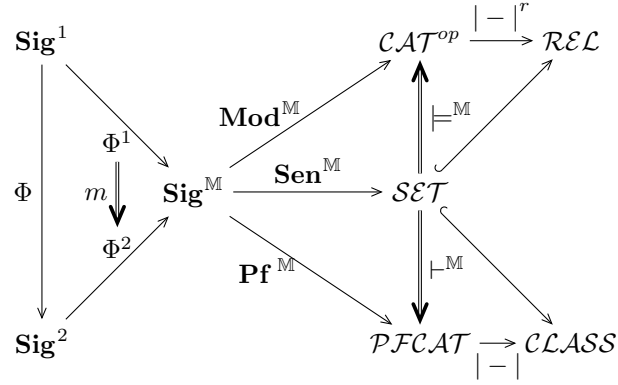
where double arrows indicate natural transformations as before. Thus, m_Σ is a family of $\mathbf{Sig}^{\mathbb{M}}$-morphisms $m_\Sigma : \Phi^1(\Sigma) \to \Phi^2(\Phi(\Sigma))$ for $\Sigma \in \mathbf{Sig}^1$. Sentence, proof, and model translation are given by $\mathbf{Sen}^{\mathbb{M}}(m_\Sigma)$, $\mathbf{Mod}^{\mathbb{M}}(m_\Sigma)$, and $\mathbf{Pf}^{\mathbb{M}}(m_\Sigma)$.

Theorem 4 (Defining Comorphisms). $(\Phi, \mathbb{M} \circ m)$ is indeed a logic comorphism.

Proof. From the diagram in Rem. 7, it is clear that the types of the components of $(\Phi, \mathbb{M} \circ m)$ are correct because they arise by composing functors and natural transformations.

It remains to show Condition 1. and 2. in Def. 9. For the latter, we substitute the respective definitions for $\vdash$, $\vdash'$, α, and γ and obtain

$$(\mathbf{Pf}^{\mathbb{M}} \circ m)_\Sigma \big((\vdash^{\mathbb{M}} \circ \Phi^1)_\Sigma \, F \big) \; = \; (\vdash^{\mathbb{M}} \circ \Phi^2)_{\Phi(\Sigma)} \, (\mathbf{Sen}^{\mathbb{M}} \circ m)_\Sigma(F).$$

This simplifies to

$$\mathbf{Pf}^{\mathbb{M}}(m_\Sigma) \, (\vdash^{\mathbb{M}}_{\Phi^1(\Sigma)} \; F) \; = \; \vdash^{\mathbb{M}}_{\Phi^2(\Phi(\Sigma))} \; \mathbf{Sen}^{\mathbb{M}}(m_\Sigma)(F)$$

which is exactly the condition from Def. 4 for the morphism m_Σ.

In the same way, Condition 1. simplifies to the satisfaction condition from Def. 2. $\square$

Remark 8. Recalling Rem. 3 and continuing Rem. 6, we observe that $\mathbb{M} \circ m$ is a natural transformation $\mathbb{M} \circ \Phi^1 \to \mathbb{M} \circ \Phi^2 \circ \Phi$ between functors $\mathbf{Sig} \to (U \downarrow V)$. Thus, $(\Phi, \mathbb{M} \circ m) :$

$\mathbb{M} \circ \Phi^1 \to \mathbb{M} \circ \Phi^2$ is indeed a morphism in the lax slice category of objects over $(U \downarrow V)$ (which justifies our notation). This yields a simpler, more abstract proof of Thm. 4.

Encoding Logic Comorphisms Finally, we study the encoding of existing logic comorphisms.

Definition 15 (Encoding Comorphisms). Assume a logic comorphism $\mu : \mathbb{I}^1 \to \mathbb{I}^2$ and two logic encodings $\mu^i : \mathbb{I}^i \to \mathbb{M}$. An **encoding** of μ relative to μ_1 and μ_2 is a logic comorphism modification $\mu^1 \to \mu^2 \circ \mu$.

To understand this definition better, let us assume a comorphism $\mu = (\Phi, \alpha, \beta, \gamma) : \mathbb{I}^1 \to \mathbb{I}^2$ and two encodings $\mu^i = (\Phi^i, \alpha^i, \beta^i, \gamma^i) : \mathbb{I}^i \to \mathbb{M}$. Then an encoding m of μ is a natural transformation $\Phi^1 \to \Phi^2 \circ \Phi$ such that the following diagrams commute for all $\Sigma \in \mathbf{Sig}^1$:

$$
\begin{array}{ccc}
\mathbf{Sen}^1(\Sigma) & \xrightarrow{\alpha_\Sigma} & \mathbf{Sen}^2(\Phi(\Sigma)) \\
\alpha^1_\Sigma \downarrow & & \downarrow \alpha^2_{\Phi(\Sigma)} \\
\mathbf{Sen}^{\mathbb{M}}(\Phi^1(\Sigma)) & \xrightarrow{\mathbf{Sen}^{\mathbb{M}}(m_\Sigma)} & \mathbf{Sen}^{\mathbb{M}}(\Phi^2(\Phi(\Sigma)))
\end{array}
$$

$$
\begin{array}{ccc}
\mathbf{Mod}^1(\Sigma) & \xleftarrow{\beta_\Sigma} & \mathbf{Mod}^2(\Phi(\Sigma)) \\
\beta^1_\Sigma \uparrow & & \uparrow \beta^2_{\Phi(\Sigma)} \\
\mathbf{Mod}^{\mathbb{M}}(\Phi^1(\Sigma)) & \xleftarrow{\mathbf{Mod}^{\mathbb{M}}(m_\Sigma)} & \mathbf{Mod}^{\mathbb{M}}(\Phi^2(\Phi(\Sigma)))
\end{array}
$$

$$
\begin{array}{ccc}
\mathbf{Pf}^1(\Sigma) & \xrightarrow{\gamma_\Sigma} & \mathbf{Pf}^2(\Phi(\Sigma)) \\
\gamma^1_\Sigma \downarrow & & \downarrow \gamma^2_{\Phi(\Sigma)} \\
\mathbf{Pf}^{\mathbb{M}}(\Phi^1(\Sigma)) & \xrightarrow{\mathbf{Pf}^{\mathbb{M}}(m_\Sigma)} & \mathbf{Pf}^{\mathbb{M}}(\Phi^2(\Phi(\Sigma)))
\end{array}
$$

Intuitively, $m_\Sigma : \Phi^1(\Sigma) \to \Phi^2(\Phi(\Sigma))$ is a signature morphism in the meta-logic such that $\mathbf{Sen}^{\mathbb{M}}(m_\Sigma)$ has the same effect as α_Σ. This is particularly intuitive in the typical case where α^1_Σ and $\alpha^2_{\Phi(\Sigma)}$ are bijections. Similarly, $\mathbf{Mod}^{\mathbb{M}}(m_\Sigma)$ and $\mathbf{Pf}^{\mathbb{M}}(m_\Sigma)$ must have the same effect as β_Σ and γ_Σ.

4. A Meta-Logic

We will now define a specific logic $\mathbb{M} = (\mathbf{Sig}^{\mathbb{M}}, \mathbf{Sen}^{\mathbb{M}}, \mathbf{Mod}^{\mathbb{M}}, \models^{\mathbb{M}}, \mathbf{Pf}^{\mathbb{M}}, \vdash^{\mathbb{M}})$ that we use as our meta-logic. $\mathbb{M}$ is based on the dependent type theory of LF.

We could define $\mathbb{M}$ such that the signatures of $\mathbb{M}$ are the LF signatures. There are several choices for the sentences of such a logic, the most elegant one uses the types

as the sentences. In that case the proof categories are the categories of contexts, and models are usually based on locally cartesian closed categories (Cartmell (1986); Seely (1984)). For example, we gave a (sound and complete) logic in this style but without the terminology used here in Awodey and Rabe (2011).

However, this is not our intention here. We want to use $\mathbb{M}$ as a logical framework, in which *all* components of an object logic are represented in terms of the syntax of the meta-logic. In order to capture all aspects of a logic, the $\mathbb{M}$-signatures must be more complex.

Below we first define the syntax, proof theory, and model theory of $\mathbb{M}$ in Sect. 4.1, 4.2, and 4.3, respectively, before defining $\mathbb{M}$ and discussing its basic properties in Sect. 4.4. We use propositional modal logic as a running example. $\mathbb{M}$ is foundation-independent: Object logic models are represented using the syntax of the meta-logic in a way that does not commit to a particular foundation of mathematics. We discuss the use of foundations in more detail in Sect. 4.5.

4.1. Syntax

The encoding of logical syntax in LF has been well-studied, and we motivate our definitions of $\mathbb{M}$ signatures by analyzing existing logic encodings in LF.

Firstly, logic encodings in LF usually employ a distinguished type `form` for the formulas and a distinguished judgment `ded : form → type` for the truth judgment. But the names of these symbols may differ, and in some cases `form` is not even a symbol. Therefore, we fix a signature *Base* as follows

```
form   :   type
ded    :   form → type
```

Now by considering morphisms out of *Base*, i.e., pairs of a signature Σ and a morphism $base : Base \to \Sigma$, we can explicate these distinguished types. $base(\texttt{form})$ is a type over Σ representing the syntactic class of formulas. And $base(\texttt{ded})$ is a type family over Σ representing the truth of formulas. By investigating morphisms out of *Base*, we are restricting attention to those LF signatures that define formulas and truth − the central properties that separate logic encodings from other LF signatures.

Secondly, we want to represent models of a signature Σ as signature morphisms out of Σ. However, models often have components that are not mentioned in the syntax, e.g., the set of worlds in a Kripke model. Therefore, we will use two LF-signatures Σ^{syn} and Σ^{mod} to represent syntax and model theory separately.

Σ^{syn} declares the syntax of a logical language, i.e., LF symbols that represent syntactic classes (e.g., terms or formulas), sorts, functions, predicates, etc. Σ^{mod} declares the model theory, i.e., LF symbols that axiomatize the properties of models, typically in terms of a foundational semantic language like set theory, type theory, or category theory. A morphism $\mu : \Sigma^{syn} \to \Sigma^{mod}$ interprets the syntax in terms of the model theory and represents the inductive interpretation function that translates syntax into the semantic realm. Finally individual models are represented as morphisms out of Σ^{mod}.

Finally, because of the symmetry between model and proof theory, it is elegant to split

syntax and proof theory as well by introducing an LF-signature Σ^{pf} and a morphism $\pi : \Sigma^{syn} \to \Sigma^{pf}$. Σ^{pf} declares the proof theory, i.e., LF symbols that represent judgments about the syntax and inference rules for them. It typically deviates from Σ^{syn} by declaring auxiliary syntax such as signed formulas in a tableaux calculus. In many cases, π is an inclusion.

Then we arrive at the following diagram:

$$Base \xrightarrow{\ base\ } \Sigma^{syn} \xrightarrow[\mu]{\pi} \begin{array}{c} \Sigma^{pf} \\ \\ \Sigma^{mod} \end{array}$$

The symbol $\mathtt{ded}$ plays a crucial duplicate role. *Proof theoretically*, the Σ^{pf}-type

$$\pi(base(\mathtt{ded})\ F)$$

is the type of proofs of F. A proof from assumptions Γ is a Σ^{pf}-term in context Γ. This yields the well-known Curry-Howard representation of proofs as terms (Martin-Löf (1996); Pfenning (2001)). *Model theoretically*, the Σ^{mod}-type $\mu(base(\mathtt{form}))$ is the type of truth values, and $\mu(base(\mathtt{ded}))$ is a predicate on it giving the designated truth values. Below we will define models as morphisms $m : \Sigma^{mod} \to M$, and (M, m) satisfies F iff the type

$$m(\mu(base(\mathtt{ded})\ F))$$

is inhabited over M. This yields the well-known representation of models as morphisms out of an initial object in a suitable category (Goguen et al. (1978); Lawvere (1963)). Thus, $\mathtt{ded}$ is the mediator between proof and model theory, and Σ^{pf} and Σ^{mod} encode the two different ways to give meaning to Σ^{syn}.

We summarize the above in the following definition:

Definition 16 (Signatures). The **signatures** of $\mathbb{M}$ are tuples

$$\Sigma = (\Sigma^{syn}, \Sigma^{pf}, \Sigma^{mod}, base, \pi, \mu)$$

forming a diagram of LF signatures as given above. $base(\mathtt{ded})$ must be a constant.

The restriction of $base(\mathtt{ded})$ to constants is a bit inelegant. But it is not harmful in practice and permits to exclude some degenerate cases that would make later definitions more complicated.

As expected, sentences are the LF terms of type $\mathtt{form}$ over Σ^{syn}:

Definition 17 (Sentences). Given an $\mathbb{M}$ signature Σ as in Def. 16, we define the **sentences** by

$$\mathbf{Sen}^{\mathbb{M}}(\Sigma) = \{F \mid \cdot \vdash_{\Sigma^{syn}} F : base(\mathtt{form})\}$$

Example 7 (Modal Logic). For our simple modal logic $\mathbb{ML}$ from Ex. 1, we use a signature ML^{syn} as follows:

```
form   :   type
⊃      :   form → form → form
□      :   form → form
ded    :   form → type
```

Here $\supset$ and $\square$ are formula constructors for implication and necessity. The morphism $base : Base \to ML^{syn}$ is simply an inclusion. If we want to declare propositional variables, we can add them using declarations $p :$ form. Let PQ^{syn} be the extension of ML^{syn} with $p :$ form and $q :$ form. Then an example sentence is $T = \square \; p \supset p \in \mathbf{Sen}^{\mathbb{M}}(PQ)$, an instance of the axiom scheme T. As here for $\supset$, we will use intuitive infix and bracket elimination rules when giving example expressions.

Definition 18 (Signature Morphism). An $\mathbb{M}$-**signature morphism** $\sigma : \Sigma \to \Sigma'$ is a tuple $(\sigma^{syn}, \sigma^{pf}, \sigma^{mod})$ such that

— the diagram below commutes,
— for every $F \in \mathbf{Sen}^{\mathbb{M}}(\Sigma)$ there is an F' such that $\sigma^{syn}(base(\mathtt{ded}) \; F) = base'(\mathtt{ded}) \; F'$.

Identity and composition for $\mathbf{Sig}^{\mathbb{M}}$-morphisms are defined component-wise.

$$
\begin{array}{ccc}
 & \Sigma^{pf} \xrightarrow{\;\;\sigma^{pf}\;\;} \Sigma'^{pf} & \\
\overset{\pi}{\nearrow} & & \overset{\pi'}{\nearrow} \\
\Sigma^{syn} \xrightarrow{\;\;\sigma^{syn}\;\;} \Sigma'^{syn} & & \\
\underset{\mu}{\searrow} & & \underset{\mu'}{\searrow} \\
 & \Sigma^{mod} \xrightarrow{\;\;\sigma^{mod}\;\;} \Sigma'^{mod} &
\end{array}
$$

σ is **simple** if additionally $\sigma^{syn} \circ base = base'$.

Remark 9. The diagram in Def. 18 omits $Base$ and thus does not imply $\sigma^{syn} \circ base = base'$, which only holds for simple morphisms. It is natural to restrict attention to simple morphisms. In particular, the second condition in Def. 18 is redundant for simple morphisms as we always have $F' = \sigma^{syn}(F)$.

However, our slightly weaker definition is crucial for the representation of many logic translations as we will see in Ex. 18 and discuss in Rem. 18.

Def. 18 is just strong enough to guarantee the existence of a **sentence translation**:

Definition 19 (Sentence Translation). In the situation of Def. 18, we put

$$\mathbf{Sen}^{\mathbb{M}}(\sigma)(F) = \text{ the } F' \text{ such that } \sigma^{syn}(base(\mathtt{ded}) \; F) = base'(\mathtt{ded}) \; F'$$

This is well-defined because F' exists by Def. 18; and if it exists, it is necessarily unique.

Remark 10. In particular, sentence translation along simple morphisms is just the application of the morphism σ^{syn}: $\sigma^{syn}(base(o)) = base'(o)$ and $\mathbf{Sen}^{\mathbb{M}}(\sigma)(F) = \sigma^{syn}(F)$.

Example 8 (Continued). A simple $\mathbf{Sig}^{\mathbb{M}}$-morphism $w : PQ \to PQ$ can contain the component

$$w^{syn} \;\; = \;\; id, \; q := \square q, \; p := q$$

where id is the identity morphism for PQ^{syn}. Then we have $w^{syn}(\texttt{form}) = \texttt{form}$ and $w^{syn}(\texttt{ded}) = \texttt{ded}$, and $w^{syn}(\texttt{ded}\ T) = \texttt{ded}\ \Box\ q \supset q$. Thus, we obtain, $\textbf{Sen}^{\mathbb{M}}(w)(T) = \Box q \supset q$.

4.2. *Proof Theory*

The idea behind the proof theory of $\mathbb{M}$ is the Curry-Howard correspondence (Curry and Feys (1958); Howard (1980)): judgments $(E_1, \ldots, E_n)$ correspond to contexts $x_1 : E_1, \ldots, x_n : E_n$ where the variables act as names for hypotheses. Moreover, a morphism from $\Gamma = x_1 : E_1, \ldots, x_m : E_m$ to $\Gamma' = x_1 : F_1, \ldots, x_n : F_n$ provides one proof over Γ for every judgment assumed in Γ', i.e., it is a tuple p_1^n of terms in context $\Gamma \vdash_{\Sigma^{pf}} p_i : F_i$. Such a tuple is simply a substitution from Γ' to Γ, and the proof categories are the well-understood categories of contexts and substitutions. The finite products are given by concatenations of contexts (possibly using α-renaming to avoid duplicate variables).

Definition 20 (Proof Categories). Given an $\mathbb{M}$ signature Σ as in Def. 16, we define the proof category $\textbf{Pf}^{\mathbb{M}}(\Sigma)$ as the dual of the category of contexts and substitutions over Σ^{pf}:

— the objects are the contexts over Σ^{pf},
— the morphisms from Γ to Γ' are the substitutions from Γ' to Γ.

Example 9 (Continued). For our modal logic, we can define ML^{pf} by extending ML^{syn} with a calculus for provability. Thus, π_{ML} is an inclusion. There are a number of ways to do that (see Avron et al. (1998)); for a simple Hilbert-style calculus, ML^{pf} consists of axioms for propositional logic and the declarations

$$
\begin{aligned}
mp\ &:\ \{x : \texttt{form}\}\, \{y : \texttt{form}\}\, \texttt{ded}\ x \supset y \to \texttt{ded}\ x \to \texttt{ded}\ y \\
nec\ &:\ \{x : \texttt{form}\}\, \texttt{ded}\ x \to \texttt{ded}\ \Box\ x \\
K\ &:\ \{x : \texttt{form}\}\, \{y : \texttt{form}\}\, \texttt{ded}\ \Box(x \supset y) \supset (\Box x \supset \Box y)
\end{aligned}
$$

Note how the Π-binder $\{x : \texttt{form}\}$ is used to declare schema variables.

Similarly, we define PQ^{pf} by extending ML^{pf} with $p : \texttt{form}$ and $q : \texttt{form}$. Then the proof category $\textbf{Pf}^{\mathbb{M}}(PQ)$ contains objects like

$$\Gamma_1\ =\ x : \texttt{ded}\ \Box(p \supset q),\ y : \texttt{ded}\ \Box p$$

and

$$\Gamma_2\ =\ z : \texttt{ded}\ \Box q.$$

A $\textbf{Pf}^{\mathbb{M}}(PQ)$-morphism from Γ_1 to Γ_2 is the substitution

$$\gamma_1\ =\ z := mp\ \underline{\Box p}\ \underline{\Box q}\ \left(mp\ \underline{\Box(p \supset q)}\ \underline{\Box p \supset \Box q}\ (K\ \underline{p}\ \underline{q})\ x\right)\ y$$

It gives a proof of the goal labelled z in terms of the two assumptions labelled x and y. To enhance readability, we have underlined the arguments that instantiate the schema variables.

Just like the proof categories are the categories of contexts and substitutions over the LF signature Σ^{pf}, the proof translation functor is given by the application of the LF morphism σ^{pf} to contexts and substitutions:

Definition 21 (Proof Translation). Given an $\mathbb{M}$-signature morphism $\sigma : \Sigma \to \Sigma'$ as in Def. 18, the proof translation functor $\mathbf{Pf}^{\,\mathbb{M}}(\sigma) : \mathbf{Pf}^{\,\mathbb{M}}(\Sigma) \to \mathbf{Pf}^{\,\mathbb{M}}(\Sigma')$ is defined by

$$\mathbf{Pf}^{\,\mathbb{M}}(\sigma)(\Gamma) = \sigma^{pf}(\Gamma) \qquad \text{for } \Gamma \in |\mathbf{Pf}^{\,\mathbb{M}}(\Sigma)|$$
$$\mathbf{Pf}^{\,\mathbb{M}}(\sigma)(\gamma) = \sigma^{pf}(\gamma) \qquad \text{for } \gamma \in \mathbf{Pf}^{\,\mathbb{M}}(\Sigma)(\Gamma, \Gamma')$$

Example 10 (Continued). Reusing the morphism w and the object Γ_1 from our running example, we obtain

$$\mathbf{Pf}^{\,\mathbb{M}}(w)(\Gamma_1) \;\; = \;\; x : \mathtt{ded}\,\Box(q \supset \Box q),\; y : \mathtt{ded}\,\Box q$$

and

$$\mathbf{Pf}^{\,\mathbb{M}}(w)(\gamma_1) \;\; = \;\; z := mp\,\underline{\Box q}\,\underline{\Box\Box q}\,\big(mp\,\underline{\Box(q \supset \Box q)}\,\underline{\Box q \supset \Box\Box q}\,(K\,\underline{q}\,\underline{\Box q})\,x\big)\,y$$

which is an $\mathbf{Pf}^{\,\mathbb{M}}(PQ)$-morphism from $\mathbf{Pf}^{\,\mathbb{M}}(w)(\Gamma_1)$ to $\mathbf{Pf}^{\,\mathbb{M}}(w)(\Gamma_2)$.

As expected, the truth judgment is given by the image of $\mathtt{ded}$ in Σ^{pf}:

Definition 22 (Truth Judgment). Given an $\mathbb{M}$ signature Σ as in Def. 16, the truth judgment $\vdash_\Sigma^{\mathbb{M}} F$ is defined as the context $x : \pi(base(\mathtt{ded})\,F)$ for an arbitrary fixed variable x.

Example 11 (Continued). We have

$$\vdash_{PQ}^{\mathbb{M}} \Box(p \supset q) \;\times\; \vdash_{PQ}^{\mathbb{M}} \Box p \;\; \cong \;\; \Gamma_1$$

$$\vdash_{PQ}^{\mathbb{M}} \Box q \;\; \cong \;\; \Gamma_2$$

The projection out of the product Γ_1 are just inclusion substitutions. And the isomorphism in the second case is a variable renaming. Thus, the morphism γ_1 proves $\{\Box(p \supset q), \Box p\} \vdash_{PQ}^{\mathbb{M}} \Box q$. Similarly, the morphism $\mathbf{Pf}^{\,\mathbb{M}}(w)(\gamma_1)$ proves $\{\Box(q \supset \Box q), \Box q\} \vdash_{PQ}^{\mathbb{M}} \Box\Box q$.

4.3. Model Theory

The definition of Σ^{mod} is the most difficult part because Σ^{mod} must declare symbols for all components that are present in a model. Since a model may involve any mathematical object, Σ^{mod} must be expressive enough to define arbitrary mathematical objects. Therefore, Σ^{mod} must include a representation of the whole foundation of mathematics. The foundation is flexible; for example, we have represented Zermelo-Fraenkel set theory, Mizar's set theory, and higher-order logic HOL as foundations in the applications of our framework (Codescu et al. (2011); Horozal and Rabe (2011); Iancu and Rabe (2011)). For simplicity, we will use higher-order logic as the foundation here. We start with our running example:

Example 12 (Continued). To describe the model theory of modal logics, we reuse the signature HOL that was introduced as a running example in Sect. 2.2. We assume it also provides a declaration

$$\forall \quad : \quad (tm\ A \to tm\ bool) \to tm\ bool$$

for universal quantification over elements of A.

Then ML^{mod} can be given by extending HOL with

$$
\begin{array}{lll}
worlds & : & tp \\
acc & : & tm\ worlds \to tm\ worlds \to tm\ bool
\end{array}
$$

where *worlds* and *acc* represent the set of words and the accessibility relation of a Kripke model. Now the morphism $\mu_{ML} : ML^{syn} \to ML^{mod}$ can be defined as

$$
\begin{array}{lll}
\texttt{form} & := & tm\ worlds \to tm\ bool \\
\supset & := & [f : tm\ worlds \to tm\ bool]\,[g : tm\ worlds \to tm\ bool] \\
& & \quad [w : tm\ worlds]\,(f\ w \Rightarrow g\ w) \\
\Box & := & [f : tm\ worlds \to tm\ bool] \\
& & \quad [w : tm\ worlds]\,\forall\,[w' : tm\ worlds]\,(acc\ w\ w' \Rightarrow f\ w') \\
\texttt{ded} & := & [f : tm\ worlds \to tm\ bool]\,ded\ (\forall\,[w : tm\ worlds]\,f\ w)
\end{array}
$$

μ_{ML} formalizes that formulas are interpreted as functions from the set of worlds to the set of booleans. Consequently, $\mu_{ML}(\supset)$ takes two and returns one, and $\mu_{ML}(\Box)$ takes one and returns one such functions. These formalize the interpretation of formulas in Kripke models in the usual way. Finally, $\mu_{ML}(\texttt{ded})$ formalizes the satisfaction relation: A formula holds in a model if it is true in all worlds.

To avoid confusion, keep in mind that $\texttt{ded}$ formalizes truth in the object-logic ML, whereas *ded* formalizes truth in the mathematical foundation HOL, which is used to express the model theory. As common in model theoretical definitions of truth, the truth of formulas is defined in terms of the truth of the mathematical foundation.

The signature PQ^{mod} arises by extending ML^{mod} with declarations $p : \mu(\texttt{form})$, i.e., $p : tm\ worlds \to tm\ bool$, and similarly for q. The morphism $\mu_{PQ} : PQ^{syn} \to PQ^{mod}$ is given by μ, $p := p$, $q := q$.

Now a model of PQ must provide specific values for *worlds*, *acc*, p, and q in terms of HOL. Thus, we can represent models as morphisms from PQ^{mod} to HOL.

Then we arrive at the following definition of model categories:

Definition 23 (Model Categories). Given an $\mathbb{M}$ signature Σ as in Def. 16, we define the model category $\mathbf{Mod}^{\mathbb{M}}(\Sigma)$ as the slice category of objects under Σ^{mod}:

— objects are the pairs (M, m) for an LF-signature morphism $m : \Sigma^{mod} \to M$,
— morphisms from (M, m) to (M', m') are LF-signature morphisms $\varphi : M \to M'$ such that $\varphi \circ m = m'$,
— the identity morphism of (M, m) is id_M,
— the composition of $\varphi : (M, m) \to (M', m')$ and $\varphi' : (M', m') \to (M'', m'')$ is $\varphi' \circ \varphi$.

Here we use an arbitrary LF signature as the codomain of the models to avoid a commitment to a particular foundation. We get back to that in Rem. 12.

The model translation functor is given by composition with σ^{mod}:

Definition 24 (Model Translation). Given an $\mathbb{M}$-signature morphism $\sigma : \Sigma \to \Sigma'$ as in Def. 18, we define $\mathbf{Mod}^{\mathbb{M}}(\sigma) : \mathbf{Mod}^{\mathbb{M}}(\Sigma') \to \mathbf{Mod}^{\mathbb{M}}(\Sigma)$ by

$$\mathbf{Mod}^{\mathbb{M}}(\sigma)(M, m) = (M, m \circ \sigma^{mod})$$

$$\mathbf{Mod}^{\mathbb{M}}(\sigma)(\varphi) = \varphi$$

The following commutative diagram illustrates the functor $\mathbf{Mod}^{\mathbb{M}}$:

$$
\begin{array}{ccc}
\Sigma^{mod} & \xrightarrow{\sigma^{mod}} & \Sigma'^{mod} \\
& m \swarrow \quad \searrow m' & \\
M & \xrightarrow{\quad \varphi \quad} & M'
\end{array}
$$

Example 13 (Continued). We have an example model $(HOL, m_1) \in \mathbf{Mod}^{\mathbb{M}}(PQ)$ by defining the LF-signature morphism m_1 to map

— all symbols of HOL to themselves,
— *worlds* to the set $\mathbb{N}$,
— *acc* to the $\leq$ relation on $\mathbb{N}$,
— p and q to the predicates *odd* and *nonzero* on $\mathbb{N}$ that are true for odd and non-zero numbers, respectively.

The model theoretical truth, i.e., the satisfaction relation, is given by the inhabitation of the type $base(\mathtt{ded})\ F$ in a model:

Definition 25 (Satisfaction). For a Σ-model (M, m) and a Σ-sentence F, we define

$$(M, m) \models^{\mathbb{M}}_{\Sigma} F \qquad \text{iff} \qquad \text{there exists } t \text{ such that } \cdot \vdash_M t : m(\mu(base(\mathtt{ded})\ F)).$$

Example 14 (Continued). We can assume that HOL has expressions $true : tm\ bool$ and $false : tm\ bool$ such that $ded\ true$ is inhabited but $ded\ false$ is empty. Thus, we obtain for the satisfaction in the model (HOL, m_1) from above

$$(HOL, m_1) \models^{\mathbb{M}}_{PQ} F \qquad \text{iff} \qquad \text{exists } t \text{ such that } \cdot \vdash_{HOL} t : ded\ (\forall[w : tm\ \mathbb{N}]\ m_1(F)\ w).$$

That holds iff $m_1(F)$ is the constant function returning $true$ (up to provable equality in HOL). For example, if we assume that HOL contains proof rules to derive common HOL-theorems, we obtain $(HOL, m_1) \models^{\mathbb{M}}_{PQ} p \supset \Box q$ because we can simplify $m_1(p \supset \Box q)\ w$ to the formula

$$odd\ w \Rightarrow \forall[w' : tm\ \mathbb{N}]\ ((w \leq w') \Rightarrow nonzero\ w')$$

which is true for all w.

Remark 11 (Lax Model Categories). Def. 23 can be generalized by using lax slice categories under Σ^{mod}. Then model morphisms from (M, m) to (M', m') are pairs (φ, r) for an LF-signature morphism $\varphi : M \to M'$ and a 2-cell $r : \varphi \circ m \Rightarrow m'$. Model reduction can be generalized accordingly. This is problematic, however, because there is no natural

way to turn the category of LF signatures into a 2-category. The same remark applies when other type theories are used instead of LF.

Recently we developed a theory of logical relations for LF as relations between LF signature morphisms Sojakova (2010). Binary logical relations do not form a 2-category either but behave like 2-cells in many ways. We expect that our present definition can be improved along those lines but omit the details here.

Remark 12 (Restricted Model Categories). This definition of models as morphisms (M, m) is very general permitting, e.g., the trivial model $(\Sigma^{mod}, id_{\Sigma^{mod}})$. It is often desirable to restrict attention to certain models or certain model morphisms.

Most importantly, we are usually interested in those models where the codomain is a fixed LF-signature $\mathcal{F}$ representing the foundation of mathematics such as $\mathcal{F} = HOL$ above. Then the LF signature morphism $m \circ \mu$ maps the syntax into the foundation $\mathcal{F}$, i.e., every syntactical expression to its interpretation in the language of mathematics. We will get back to this in Sect. 4.5.

More generally, we can supplement our definition of signatures with a formal language in which structural properties of models and morphisms can be formulated. For example, Σ^{mod} might contain a declaration that requires m to be an elementary embedding. The key unsolved difficulty here is to design a formal language that is reasonably expressive without making the framework overly complex.

4.4. *Collecting the Pieces*

We conclude the definition of $\mathbb{M}$ with the following result.

Theorem 5. $\mathbb{M} = (\mathbf{Sig}^{\mathbb{M}}, \mathbf{Sen}^{\mathbb{M}}, \mathbf{Mod}^{\mathbb{M}}, \models^{\mathbb{M}}, \mathbf{Pf}^{\mathbb{M}}, \vdash^{\mathbb{M}})$ is a logic.

Proof. We have to prove a number of conditions:

— **Sig** is a category. The only non-obvious aspect here is that the existence of F' in the definition of signature morphisms is preserved under composition. It is easy to see.
— **Sen**$^{\mathbb{M}}$ is a functor. This is straightforward.
— **Mod**$^{\mathbb{M}}$ is a functor. This is a standard result of category theory for slice categories (see, e.g., Mac Lane (1998)).
— It is a standard about the category of contexts in type theories that $\mathbf{Pf}^{\mathbb{M}}$ is a functor $\mathbf{Sig}^{\mathbb{M}} \to \mathcal{CAT}$ (see, e.g., Pitts (2000)). Thus, we only need to show that $\mathbf{Pf}^{\mathbb{M}}(\Sigma)$ has finite products and that $\mathbf{Pf}^{\mathbb{M}}(\sigma)$ preserves these products. The products are given by $\Gamma_1 \times \ldots \times \Gamma_n = \Gamma'_1, \ldots, \Gamma'_n$ where Γ'_i arises from Γ_i by α-renaming to eliminate duplicate variable names. The empty product (terminal object) is the empty context.
— $\models^{\mathbb{M}}$ meets the satisfaction condition. Assume $\sigma : \Sigma \to \Sigma'$, $F \in \mathbf{Sen}^{\mathbb{M}}(\Sigma)$, and $(M, m) \in \mathbf{Mod}^{\mathbb{M}}(\Sigma')$. Then

$\mathbf{Mod}^{\mathbb{M}}(\sigma)(M, m) \models^{\mathbb{M}}_{\Sigma} F$ iff
there is an M-term of type $(m \circ \sigma^{mod})\big(\mu(base(\mathtt{ded})\, F)\big)$ iff
there is an M-term of type $m\big(\mu'\big(\sigma^{syn}(base(\mathtt{ded})\, F)\big)\big)$ iff
there is an M-term of type $m\big(\mu'\big(base'(\mathtt{ded})\, \mathbf{Sen}^{\mathbb{M}}(\sigma)(F)\big)\big)$ iff
$(M, m) \models^{\mathbb{M}}_{\Sigma'} \mathbf{Sen}^{\mathbb{M}}(\sigma)(F)$.

$\vdash^{\mathbb{M}}$ is preserved. Assume $\sigma : \Sigma \to \Sigma'$ and $F \in \mathbf{Sen}^{\mathbb{M}}(\Sigma)$. Then

$$\mathbf{Pf}^{\mathbb{M}}(\sigma)(\vdash_{\Sigma}^{\mathbb{M}} F) \quad = \quad x : \sigma^{pf}(\pi(base(\mathtt{ded})\ F)) \quad =$$

$$x : \pi'(\sigma^{syn}(base(\mathtt{ded})\ F)) \quad = \quad x : \pi'(base'(\mathtt{ded})\ \mathbf{Sen}^{\mathbb{M}}(\sigma)(F)) \quad =$$

$$\vdash_{\Sigma'}^{\mathbb{M}}\ \mathbf{Sen}^{\mathbb{M}}(\sigma)(F)$$

$\square$

Soundness and Completeness $\mathbb{M}$ is intentionally neither sound nor complete because it is designed to be a meta-logic in which other possibly unsound or incomplete logics are represented. However, we have the following soundness criterion:

Definition 26. An $\mathbb{M}$-signature $(\Sigma^{syn}, \Sigma^{pf}, \Sigma^{mod}, base, \pi, \mu)$ is called **sound** if there is an LF signature morphism $\psi : \Sigma^{pf} \to \Sigma^{mod}$ such that the following diagram commutes:

This definition is justified by:

Theorem 6. The logic arising from $\mathbb{M}$ by considering only the sound $\mathbb{M}$-signatures is strongly sound.

Proof. Let Σ be a sound signature (i) as in Def. 26, Θ a set of Σ-sentences, and F a Σ-sentence. Assume $\Theta \vdash_{\Sigma}^{\mathbb{M}} F$ (ii); then we need to show $\Theta \models_{\Sigma}^{\mathbb{M}} F$. So assume an arbitrary model $(M, m) \in \mathbf{Mod}^{\mathbb{M}}(\Sigma)$ such that $(M, m) \models_{\Sigma}^{\mathbb{M}} A$ for every $A \in \Theta$ (iii); then we need to show $(M, m) \models_{\Sigma}^{\mathbb{M}} F$.

By (ii), there must be a substitution from $\vdash_{\Sigma}^{\mathbb{M}} F$ to $\vdash_{\Sigma}^{\mathbb{M}} F_1 \times \ldots \times \vdash_{\Sigma}^{\mathbb{M}} F_n$ for some $\{F_1, \ldots, F_n\} \subseteq \Theta$. Using the LF type theory, it is easy to show that this is equivalent to having a closed Σ^{pf}-term t of type

$$\pi\big(base(\mathtt{ded})\ F_1 \to \ldots \to base(\mathtt{ded})\ F_n \to base(\mathtt{ded})\ F\big).$$

By (i), we obtain a closed M-term $m(\psi(t))$ of type

$$m\big(\mu\big(base(\mathtt{ded})\ F_1 \to \ldots \to base(\mathtt{ded})\ F_n \to base(\mathtt{ded})\ F\big)\big).$$

Now by (iii), there are M-terms t_i of type $m(\mu(base(\mathtt{ded})\ F_i))$. Thus, there is also an M-term $m(\psi(t))\ t_1\ \ldots\ t_n$ of type $m(\mu(base(\mathtt{ded})\ F))$ and therefore $(M, m) \models_{\Sigma}^{\mathbb{M}} F$. $\square$

We will refine this into a criterion to establish the soundness of a logic in Thm. 10 below.

Remark 13 (Completeness). It does not make sense to use the analogue of Def. 26 for completeness: Σ^{mod} is typically much stronger than Σ^{pf} – often as strong as mathematics

as a whole as in Sect. 4.5 – so that there can be no signature morphism $\Sigma^{mod} \to \Sigma^{syn}$. Moreover, even if we had such a morphism, it would not yield completeness because the proof of Thm. 6 crucially uses the composition $m \circ \psi$.

This is not surprising. A soundness result is naturally proved by showing inductively that the interpretation function maps every derivable syntactic statement to a true semantic statement; this is exactly what a signature morphism does. A completeness result on the other hand is usually proved by exhibiting a canonical model, which has a very different flavor.

However, it is still very promising to undertake completeness proofs within our framework: The canonical model is usually formed from the syntax, which is already formally represented in Σ^{syn}. A framework like LF is ideal to build canonical models as abstract data types. However, to show completeness, we still have to reflect these LF-level syntactical models into Σ^{mod}, e.g., by interpreting an LF type as the set of LF terms of that type. This technique is related to the definition of abstract data types in Isabelle/HOL (Nipkow et al. (2002)) and the quotations of Chiron (Farmer (2010)). We have to leave a further investigation to future work.

Theories Due to the Curry-Howard representation of proofs, there is no significant conceptual difference between signatures and theories from an LF perspective. In fact, $\mathbb{M}$-signatures subsume the finite theories already:

Notation 4. If $\Theta = \{F_1, \ldots, F_n\}$, the $\mathbb{M}$-theory (Σ, Θ) behaves in the same way as the $\mathbb{M}$-signature arising from Σ by appending $a_1 : base(\mathtt{ded})\ F_1$, ..., $a_n : base(\mathtt{ded})\ F_n$ to Σ^{syn}. We write $(\Sigma, \Theta)^{syn}$ for this signature. Accordingly, we obtain (by pushout along $\Sigma^{syn} \to (\Sigma, \Theta)^{syn}$) the signatures $(\Sigma, \Theta)^{pf}$ and $(\Sigma, \Theta)^{mod}$.

Correspondingly, we can express $\mathbb{M}$-theory morphisms as LF-signature morphisms:

Theorem 7. Assume an $\mathbb{M}$-signature morphism $\sigma : \Sigma \to \Sigma'$ and finite theories (Σ, Θ) and (Σ', Θ').

1. The following properties of σ are equivalent:

    — σ is a proof theoretical theory morphism $(\Sigma, \Theta) \to (\Sigma', \Theta')$.

    — There is an LF-signature morphism $(\sigma, \vartheta)^{pf}$ such that the left diagram below commutes.

2. The following properties of σ are equivalent:

    — σ is a model theoretical theory morphism $(\Sigma, \Theta) \to (\Sigma', \Theta')$.

    — There is an LF-signature morphism $(\sigma, \vartheta)^{mod}$ such that the right diagram below commutes.

$$\begin{array}{ccc}
\Sigma^{pf} \xrightarrow{\quad \sigma^{pf} \quad} \Sigma^{pf} & \qquad & \Sigma^{mod} \xrightarrow{\quad \sigma^{mod} \quad} \Sigma^{mod} \\
\Big\uparrow \qquad \qquad \Big\downarrow & & \Big\uparrow \qquad \qquad \Big\uparrow \\
(\Sigma, \Theta)^{pf} \xrightarrow{(\sigma, \vartheta)^{pf}} (\Sigma', \Theta')^{pf} & & (\Sigma, \Theta)^{mod} \xrightarrow{(\sigma, \vartheta)^{mod}} (\Sigma', \Theta')^{mod}
\end{array}$$

Proof.

1. σ being a proof theoretical theory morphism is equivalent to there being proof terms $p_i : \sigma^{syn}(\pi(base'(\mathtt{ded})\ F_i))$ over the LF-signature $(\Sigma', \Theta')^{pf}$ for every $F_i \in \Theta$, i.e., for every $a_i : base(\mathtt{ded})\ F_i$ in $(\Sigma, \Theta)^{syn}$. Then mapping every a_i to p_i yields the LF-signature morphism $(\sigma, \vartheta)^{pf} : (\Sigma, \Theta)^{pf} \to (\Sigma', \Theta')^{pf}$. The inverse direction is proved accordingly.

2. First, we observe, using the definition of satisfaction, that a Σ-model (M, m) satisfies all axioms in Θ iff $m : \Sigma^{mod} \to M$ can be factored through $(\Sigma, \Theta)^{mod}$ (in other words: m can be extended to a signature morphism $(\Sigma, \Theta)^{mod}$).

   Thus, given $(\sigma, \vartheta)^{mod}$, models of Σ' yield models of Σ simply by composition, which proves σ is a model theoretical theory morphism. Conversely, one obtains $(\sigma, \vartheta)^{mod}$ by factoring the Σ-model $((\Sigma', \Theta')^{mod}, \sigma^{mod})$ through $(\Sigma, \Theta)^{mod}$.

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

The criterion for proof theoretical theory morphisms is quite intuitive and useful: Via the Curry-Howard representation, axioms are mapped to proof terms. The model theoretical criterion is much less useful because there are many models in $\mathbb{M}$, which makes it a very strong condition. It guarantees that σ is a model theoretical theory morphism for any foundation. However, often the property of being a model theoretical theory morphism depends on the chosen foundation. In Sect. 4.5, we will look at a weaker, foundation-specific condition.

4.5. *Foundations*

We call $\mathbb{M}$ **foundation-independent** because it is not committed to a fixed foundation, in which the model theory of object logics is defined. Thus, we are able to express logics using different mathematical foundations and even translations between them. However, this is also a drawback because for any specific logic, we are usually only interested in certain models (M, m), namely in those where M is the semantic realm of all mathematical objects. Therefore, we restrict $\mathbb{M}$ accordingly using a fixed LF-signature $\mathcal{F}$ that represents the foundation of mathematics.

The intuition is as follows. $\mathcal{F}$ is a meta-language, which is used by Σ^{mod} to specify models. Thus, Σ^{mod} includes $\mathcal{F}$ and adds symbols to it that represent the free parameters of a model, e.g., the universe and the function and predicate symbols for first-order models. Models must interpret those added symbols as expressions of the foundation in a way that preserves typing (and thus via Curry-Howard, in a way that satisfies the axioms). Therefore, we can represent models as signature morphisms $m : \Sigma^{mod} \to \mathcal{F}$.

However, representing a foundation in LF can be cumbersome due to the size of the theory involved. Our representation of ZFC that we use in Horozal and Rabe (2011) requires several thousand declarations just to build up the basic notions of ZFC set theory to a degree that supports simple examples. Moreover, the automation of foundations is a very difficult problem, and type checking or automated reasoning for untyped set theories is much harder than for most logics.

We remedy these drawbacks with a slightly more general approach. We use two LF-signatures $\mathcal{F}_0$ and $\mathcal{F}$ with a morphism $P : \mathcal{F}_0 \to \mathcal{F}$. The idea is that $\mathcal{F}_0$ is a fragment or an approximation of the foundation $\mathcal{F}$ that is refined into $\mathcal{F}$ via the morphism P. It turns out that manageably simple choices of $\mathcal{F}_0$ are expressive enough to represent the model theory of most logics.

This has the additional benefit that the same $\mathcal{F}_0$ can be used together with different values for P and $\mathcal{F}$. This corresponds to the mathematical practice of not detailing the foundation unless necessary. For example, standard constructions and results like the real numbers are usually assumed without giving or relying on a specific definition. For example, in Horozal and Rabe (2011), we use a simply-typed higher-order logic for $\mathcal{F}_0$, ZFC set theory as $\mathcal{F}$, and P interprets types as sets and terms as elements of these sets. $\mathcal{F}_0$ can be written down in LF within a few minutes, and strong implementations are available (Gordon (1988); Harrison (1996); Nipkow et al. (2002)).

If we combine this approach with a module system for LF – as we do for example in Codescu et al. (2011) – this gives rise to the paradigm of **little foundations**. It corresponds to the little theories used implicitly in Bourbaki (1974) and explicitly in Farmer et al. (1992). Using little foundations, we specify models in $\mathcal{F}_0$ making as few foundational assumptions as possible and refine it to a stronger foundation when needed. In fact, Σ^{syn} can be seen as the extreme case of making no assumptions about the foundation, and $\mu : \Sigma^{syn} \to \Sigma^{mod}$ as a first refinement step. At the opposite end of the refinement chain, we might find the extension $ZF \hookrightarrow ZFC$, which adds the axiom of choice.

We formalize these intuitions as follows:

Definition 27 (Foundations). Given an LF-signature $\mathcal{F}_0$, $\mathbf{Sig}^{\mathbb{M}}_{\mathcal{F}_0}$ is the subcategory of $\mathbf{Sig}^{\mathbb{M}}$ that is restricted to

— those signatures Σ for which there is an inclusion from $\mathcal{F}_0$ into Σ^{mod},
— those signature morphisms σ for which σ^{mod} is the identity on $\mathcal{F}_0$.

Thus, $\mathbf{Sig}^{\mathbb{M}}_{\mathcal{F}_0}$ is the full subcategory of inclusions of the slice category of objects under $\mathcal{F}_0$. Note that we recover $\mathbf{Sig}^{\mathbb{M}}$ when $\mathcal{F}_0$ is the empty signature.

Definition 28 (Founded Models). For an LF-signature morphism $P : \mathcal{F}_0 \to \mathcal{F}$, we define $\mathbf{Mod}^{\mathbb{M}}_P$ as the functor $\mathbf{Sig}^{\mathbb{M}}_{\mathcal{F}_0} \to \mathcal{CAT}^{op}$ that maps

— signatures Σ to the subcategory of $\mathbf{Mod}^{\mathbb{M}}(\Sigma)$ restricted to the models $(\mathcal{F}, m)$ for which m agrees with P on $\mathcal{F}_0$,
— signature morphisms in the same way as $\mathbf{Mod}^{\mathbb{M}}$.

These definitions lead to the following commutative diagram for a Σ'-model $(\mathcal{F}, m)$ translated along σ:

$$\Sigma^{mod} \xrightarrow{\sigma^{mod}} \Sigma'^{mod}$$

And these specializations indeed yield a logic:

Theorem 8. For arbitrary $P : \mathcal{F}_0 \to \mathcal{F}$, we obtain a logic

$$\mathbb{M}_P = (\mathbf{Sig}^{\mathbb{M}}_{\mathcal{F}_0}, \mathbf{Sen}^{\mathbb{M}}, \mathbf{Mod}^{\mathbb{M}}_P, \models^{\mathbb{M}}, \mathbf{Pf}^{\mathbb{M}}, \vdash^{\mathbb{M}})$$

(where $\mathbf{Sen}^{\mathbb{M}}$, $\models^{\mathbb{M}}$, $\mathbf{Pf}^{\mathbb{M}}$, and $\vdash^{\mathbb{M}}$ are the appropriate restrictions according to $\mathbf{Sig}^{\mathbb{M}}_{\mathcal{F}_0}$ and $\mathbf{Mod}^{\mathbb{M}}_P$).

Proof. This follows immediately from Thm. 5. The only non-trivial aspect is to show that $\mathbf{Mod}^{\mathbb{M}}_P(\sigma)$ is well-defined, and that follows from the commutativity of the above diagram. $\square$

Example 15 (Continued). In Ex. 12, we have already used $\mathbb{M}_P$-models: We used $\mathcal{F}_0 = \mathcal{F} = HOL$, and P was the identity.

Now we can revisit Thm. 7, which stated that model theoretical theory morphisms $\sigma : (\Sigma, \Theta) \to (\Sigma', \Theta')$ in $\mathbb{M}$ are equivalent to LF-signature morphisms $(\sigma, \vartheta)^{mod} : (\Sigma, \Theta)^{mod} \to (\Sigma', \Theta')^{mod}$. If we restrict attention to $\mathbb{M}_P$, the situation is a bit more complicated:

Theorem 9. Assume an $\mathbb{M}_P$-signature morphism $\sigma : \Sigma \to \Sigma'$ and finite theories (Σ, Θ) and (Σ', Θ'). The following properties of σ are equivalent:

— σ is a model theoretical theory morphism $(\Sigma, \Theta) \to (\Sigma', \Theta')$ in the logic $\mathbb{M}_P$.
— For every LF-signature morphism $m' : (\Sigma', \Theta')^{mod} \to \mathcal{F}$, there is an LF-signature morphism $m : (\Sigma, \Theta)^{mod} \to \mathcal{F}$ such that the following diagram commutes:

$$\Sigma^{mod} \xrightarrow{\sigma^{mod}} \Sigma'^{mod}$$

Proof. This follows immediately because σ is a model theoretical theory morphism iff $\mathbf{Mod}^{\mathbb{M}}_P(\sigma)$ maps Θ'-models to Θ-models. $\square$

Using Thm. 9, extending σ^{mod} to an LF-signature morphism $(\sigma, \vartheta)^{mod} : (\Sigma, \Theta)^{mod} \to (\Sigma', \Theta')^{mod}$ is sufficient to make σ a model theoretical $\mathbb{M}_P$-theory morphism. But it is not a necessary condition. Counter-examples arise when $\mathcal{F}_0$ is too weak and cannot prove all theorems of $\mathcal{F}$. In Horozal and Rabe (2011), we show that is a necessary condition in the special case $\mathcal{F}_0 = \mathcal{F} = ZFC$. The general case is open.

5. Defining and Encoding Logics

We can combine the results of Sect. 3 and 4 to define logics and logic translations in our framework. Logics are defined or represented using functors $\Phi : \mathbf{Sig} \to \mathbf{Sig}^{\mathbb{M}}$. But the formal details can be cumbersome in practice: $\mathbb{M}$-signatures are 6-tuples and morphisms 3-tuples with some commutativity constraints. Therefore, we introduce the notion of uniform logic representations, which are the typical situation in practice and which can be given conveniently.

Uniform logics are generated by a fixed $\mathbf{Sig}^{\mathbb{M}}$ signature L. The intuition is that L^{syn} represents the logical symbols, and extensions $L^{syn} \hookrightarrow \Sigma^{syn}$ add one declaration for each non-logical symbols. This is very similar to the use of "uniform" in Harper et al. (1994), from which we borrow the name. Then proof and model theory are given in general by L^{pf} and L^{mod}, and these induce extensions Σ^{pf} and Σ^{mod} for the proof and model theory of a specific choice of non-logical symbols. Similarly, translations interpret logical symbols in terms of logical symbols and non-logical ones in terms of non-logical ones.

5.1. Non-Logical Symbols

Recall from Sect. 2.2 that we write $\mathbb{LF}$ for the category of LF signatures and LF signature morphisms. Because $\mathbf{Sig}^{\mathbb{M}}$ morphisms are triples of $\mathbb{LF}$ morphisms and composition is defined component-wise, $\mathbf{Sig}^{\mathbb{M}}$ inherits inclusions and pushouts from $\mathbb{LF}$. In particular:

Lemma 2. $\mathbf{Sig}^{\mathbb{M}}$ has inclusion morphisms in the following sense: The subcategory of $\mathbf{Sig}^{\mathbb{M}}$ which contains

— all objects and
— only those morphisms $(\sigma^{syn}, \sigma^{pf}, \sigma^{mod}) : \Sigma \to \Sigma'$ for which σ^{syn}, σ^{pf}, and σ^{mod} are inclusions in $\mathbb{LF}$

is a partial order. We write $\Sigma \hookrightarrow \Sigma'$ for inclusion morphisms from Σ to Σ'

Proof. Morphisms in $\mathbf{Sig}^{\mathbb{M}}$ are triples, and composition is defined component-wise. Therefore, the properties of inclusions follow immediately from those of inclusions in $\mathbb{LF}$.
□

Remark 14. More generally, the inclusions of $\mathbb{LF}$ induce a weak inclusion system in the sense of Căzănescu and Roşu (1997). This property is inherited by $\mathbf{Sig}^{\mathbb{M}}$ as well.

We will use inclusions $L \hookrightarrow \Sigma$ to represent a logic L together with non-logical symbols given by Σ. When giving the non-logical symbols, it is not necessary to give Σ; instead, we can just give an $\mathbb{LF}$ inclusion $L^{syn} \hookrightarrow \Sigma^{syn}$ and obtain the remaining components of Σ by pushout constructions. We make that precise in the following definitions:

Definition 29. An inclusion $L^{syn} \hookrightarrow \Sigma^{syn}$ is called *compatible* with L if Σ^{syn} does not add declarations for names that are also declared in L^{pf} or L^{mod}.

Definition 30. We define the category $\mathbb{M} + \mathbb{LF}$ as follows

— objects are the pairs (L, Σ^{syn}) of an $\mathbb{M}$ signature L and an $\mathbb{LF}$ inclusion $L^{syn} \hookrightarrow \Sigma^{syn}$ that is compatible with L,
— morphisms from (L, Σ^{syn}) to (L', Σ'^{syn}) are the commuting rectangles

$$
\begin{array}{ccc}
L'^{syn} & \hookrightarrow & \Sigma'^{syn} \\
\uparrow l^{syn} & & \uparrow \sigma^{syn} \\
L^{syn} & \hookrightarrow & \Sigma^{syn}
\end{array}
$$

The restriction to compatible inclusions in Def. 30 is a technicality needed for Def. 31: If Σ^{syn} added a name that is also declared in L^{pf} or L^{mod}, the respective canonical pushout in $\mathbb{LF}$ would not exist. This restriction is harmless in practice because namespaces can be used to ensure the uniqueness of names.

For compatible inclusions, the canonical pushouts exist and every $\mathbb{M} + \mathbb{LF}$ object (L, Σ^{syn}) induces an $\mathbb{M}$ signature and every $\mathbb{M} + \mathbb{LF}$ morphism (l, σ^{syn}) induces an $\mathbb{M}$ signature morphism:

Definition 31. We define a functor $\overline{\cdot} : \mathbb{M} + \mathbb{LF} \to \mathbf{Sig}^{\mathbb{M}}$ as follows
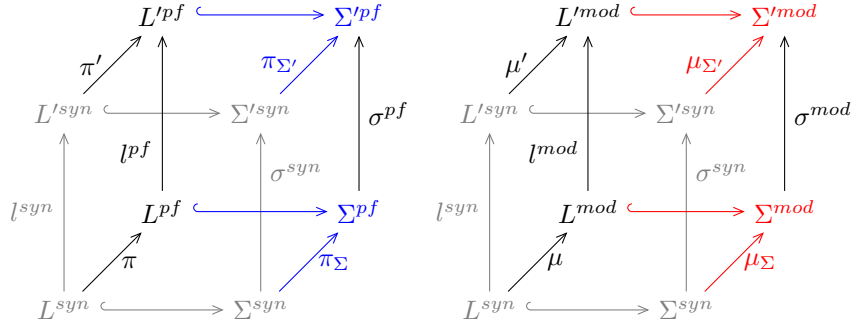
— For objects (L, Σ^{syn}), we define the $\mathbb{M}$ signature

$$\overline{(L, \Sigma^{syn})} = (\Sigma^{syn}, \Sigma^{pf}, \Sigma^{mod}, base_\Sigma, \pi_\Sigma, \mu_\Sigma)$$

such that the following diagram commutes and Σ^{pf} and Σ^{mod} are canonical pushouts in $\mathbb{LF}$

— For morphisms $(l, \sigma^{syn}) : (L, \Sigma^{syn}) \to (L', \Sigma'^{syn})$, we define the $\mathbb{M}$ signature morphism $\overline{(l, \sigma^{syn})} = (\sigma^{syn}, \sigma^{pf}, \sigma^{mod}) : \overline{(L, \Sigma^{syn})} \to \overline{(L', \Sigma'^{syn})}$ such that σ^{pf} and σ^{mod} are the unique morphisms that make the following $\mathbb{LF}$ diagrams commute

5.2. *Uniform Logics*

We obtain a uniform logic $\mathbb{M}_P^L$ for every $\mathbb{M}$ signature L by pairing L with all possible choices of non-logical symbols. In $\mathbb{M}_P^L$, we also fix a foundation for the model theory:

Definition 32 (Uniform Logics). Assume an LF signature morphism $P : \mathcal{F}_0 \to \mathcal{F}$ and a $\mathbf{Sig}_{\mathcal{F}_0}^{\mathbb{M}}$-signature L. We write $L + \mathbb{LF}$ for the subcategory of $\mathbb{M} + \mathbb{LF}$ containing the objects (L, Σ^{syn}) and the morphisms (id_L, σ^{syn}). Then we obtain a logic

$$\mathbb{M}_P^L = \mathbb{M}_P \circ \overline{\;\cdot\;}|_{L+\mathbb{LF}}.$$

Here $\overline{\;\cdot\;}|_{L+\mathbb{LF}} : L + \mathbb{LF} \to \mathbf{Sig}^{\mathbb{M}}$ is the restriction of $\overline{\;\cdot\;}$ to $L + \mathbb{LF}$.

Uniform logics are defined completely within LF – signatures, sentences, proofs, and models are given as syntactical entities of LF. We already know how to represent existing logics in LF, namely via Thm. 12. Uniform logic encodings arise when we use $\mathbb{M}_P^L$ as the meta-logic.

Example 16 (Modal Logic). If we take all examples from Sect. 4 together, we obtain a uniform encoding of $\mathbb{ML}$ using the $\mathbb{M}$-signature

$$ML = (ML^{syn}, ML^{pf}, ML^{mod}, incl, incl, \mu_{ML})$$

(where *incl* denotes inclusion morphisms). We define a functor $\Phi^{\mathbb{ML}} : \mathbf{Sig}^{\mathbb{ML}} \to ML + \mathbb{LF}$ as follows. For a modal logic signature $\Sigma = \{p_1, \ldots, p_n\} \in |\mathbf{Sig}^{\mathbb{ML}}|$ and a signature morphism $\sigma : \Sigma \to \Sigma'$, we define $\Phi^{\mathbb{ML}}(\Sigma) = (ML, \Sigma^{syn})$ and $\Phi^{\mathbb{ML}}(\sigma) = (id, \sigma^{syn})$ with

$$\Sigma^{syn} = ML^{syn},\ p_1 : \texttt{form},\ \ldots,\ p_n : \texttt{form}$$
$$\sigma^{syn} = id_{ML^{syn}},\ p_1 := \sigma(p_1),\ \ldots,\ p_n := \sigma(p_n)$$

This yields $\overline{\Phi^{\mathbb{ML}}(\Sigma)} = (\Sigma^{syn}, \Sigma^{pf}, \Sigma^{mod}, incl, incl, \mu_\Sigma)$ and $\overline{\Phi^{\mathbb{ML}}(\sigma)} = (\sigma^{syn}, \sigma^{pf}, \sigma^{mod})$ with

$$\Sigma^{pf} = ML^{pf},\ p_1 : \texttt{form},\ \ldots,\ p_n : \texttt{form}$$
$$\Sigma^{mod} = ML^{mod},\ p_1 : tm\ worlds \to tm\ bool,\ \ldots,\ p_n : tm\ worlds \to tm\ bool$$
$$\mu_\Sigma = \mu_{ML},\ p_1 := p_1,\ \ldots,\ p_n := p_n$$
$$\sigma^{pf} = id_{ML^{pf}},\ p_1 := \sigma(p_1),\ \ldots,\ p_n := \sigma(p_n)$$
$$\sigma^{mod} = id_{ML^{mod}},\ p_1 := \sigma(p_1),\ \ldots,\ p_n := \sigma(p_n)$$

In particular, we have

$$\overline{\Phi^{\mathbb{ML}}(PQ)} = (PQ^{syn}, PQ^{pf}, PQ^{mod}, incl, incl, \mu_{PQ}) \quad \text{and}$$
$$\overline{\Phi^{\mathbb{ML}}(w)} = (w^{syn}, w^{pf}, w^{mod})$$

as defined throughout the running example.

Moreover it is straightforward to show that $\Phi^{\mathbb{ML}}$ can be extended to a logic comorphism $(\Phi^{\mathbb{ML}}, \alpha, \beta, \gamma)$ from $\mathbb{ML}$ to $\mathbb{M}_P^{ML}$ where P is as in Ex. 15. α_Σ is an obvious bijection. β_Σ maps every LF-signature morphism $m : \Sigma^{mod} \to HOL$ to a Kripke-model of Σ. Finally, γ_Σ is a straightforward encoding of judgments as types and proofs as terms.

It is very easy to show the proof theoretical adequacy of γ. The discussion of the model theoretical adequacy of β is complicated as it depends on which sets are eligible as Kripke frames in ML-models (see Rem. 16). If HOL can express all sets that are eligible as Kripke frames, we obtain model theoretical adequacy.

Remark 15 (Declaration Patterns). Def. 32 uses all extensions of L^{syn} as the signatures of $\mathbb{M}_P^L$. However, often we want to work with a subcategory instead. For example, for modal logic, we would like to use only those extensions with declarations of the form $p : \texttt{form}$. This requires an extension of LF with what we call *declaration patterns* (similar to the *block* declarations already present in Twelf, Pfenning and Schürmann (1999)). Such an extension is currently designed in joint work with Fulya Horozal.

Remark 16 (Model Theoretical Adequacy). Ex. 16 raises the question when model theoretical adequacy holds. Let us call a mathematical object **definable** if it can be denoted by an expression of the foundational language. For example, since there can only be countably many definable objects, a set theory with uncountably many objects has undefinable objects. But even if we can prove the existence of undefinable objects, we can never actually give one. On the other hand, it is easy to see that a model can be represented as an LF signature morphism iff all its components are definable.

Therefore, the surjectivity of β_Σ is and thus model theoretical adequacy depend on the philosophical point of view we take.

The type theoretical school often rejects a commitment to set theory – recall that LF and thus the essence of $\mathbb{M}$ can be developed using only formal languages and inference systems without appealing to a foundation of mathematics like set theory. Type theory does not object to model theory per se; but it would restrict attention to definable models. In fact, our models-as-signature-morphisms paradigm is an appealing way to define *model* from a type theoretical point of view. This is particularly elegant if we observe that the approach unifies the study of models of theories and that of implementations of specifications. In that case, of course all models are expressible as signature morphisms, and adequacy holds.

The set theoretical school does not only define individual models but also the class of all models. Moreover, they see the sets of LF signatures and signature morphisms as defined within set theory. Then adequacy means the existence of a certain between a class of models and a set of morphisms. But then a simple cardinality argument shows

that adequacy usually does not hold. In order to maintain Thm. 2, we have to study weaker notions of adequacy. We leave this to further work.

Besides their simplicity, uniform representations have the advantage, that the properties of soundness can be proved formally within the logical framework:

Theorem 10. A uniform logic $\mathbb{M}_P^L$ is strongly sound if L is a sound $\mathbb{M}$ signature.

Proof. Using Thm. 6, we only have to show that for every Σ^{syn} extending L^{syn} there is an $\mathbb{LF}$-morphism from Σ^{pf} to Σ^{mod} with a certain commutativity property. That follows immediately from the universal property of the pushout Σ^{pf} using the soundness of L. $\square$

We sketch another example – a fragment of our encoding of first-order logic from Horozal and Rabe (2011) – in order to give an example of a logic translation below.

Example 17 (First-Order Logic). To define first-order logic $\mathbb{FOL}$, we give a uniform functor $\Phi^{\mathbb{FOL}}$ over the $\mathbb{M}$ signature FOL. The central declarations of FOL^{syn} are:

i	:	$\texttt{type}$	terms
o	:	$\texttt{type}$	formulas
$\supset$	:	$o \to o \to o$	implication
$\forall$	:	$(i \to o) \to o$	universal
ded	:	$o \to \texttt{type}$	truth

FOL^{pf} adds natural deduction proof rules in a straightforward way. For simplicity, we assume that FOL^{mod} extends the same signature $\mathcal{F}_0 = HOL$ as ML^{mod}. This extension includes a declaration $univ : tp$ for the universe. $base_{FOL}$ and π_{FOL} are inclusions, and μ_{FOL} maps i to $tm\ univ$, o to $tm\ bool$, and all formulas to their semantics in the usual way.

Assume a $\mathbb{FOL}$ signature $\Sigma = (\Sigma_f, \Sigma_p, arit)$ where Σ_f and Σ_p are the sets of function and predicate symbols with arity function $arit : \Sigma_f \cup \Sigma_p \to \mathbb{N}$. Then $\Phi^{\mathbb{FOL}}(\Sigma)$ extends FOL^{syn} with declarations $f : i \to \ldots \to i \to i$ and $p : i \to \ldots \to i \to o$, respectively, for each function or predicate symbol.

5.3. *Uniform Logic Translations*

Logic translations between uniform logics are already possible using the general methods of Thm. 4. We speak of uniform translations in the special case where a fixed $\mathbb{M}$-signature morphism is used to translate the logical symbols:

Theorem 11 (Uniform Logic Translations). For $i = 1, 2$, assume functors $\Phi^i : \mathbf{Sig}^i \to L_i + \mathbb{LF}$. Assume

— a functor $\Phi : \mathbf{Sig}^1 \to \mathbf{Sig}^2$,
— a $\mathbf{Sig}_{\mathcal{F}_0}^{\mathbb{M}}$-morphism $l = (l^{syn}, l^{pf}, l^{mod}) : L_1 \to L_2$,
— a natural transformation $m : \Phi^1 \to \Phi^2 \circ \Phi$ (seen as functors $\mathbf{Sig}^1 \to \mathbb{M} + \mathbb{LF}$) such that each m_Σ is of the form (l, α_Σ).

Then $(\varphi, \mathbb{M}_P \circ \overline{\ \cdot\ } \circ m)$ is a logic comorphism from $\mathbb{M}_P^{L_1} \circ \Phi^1$ to $\mathbb{M}_P^{L_2} \circ \Phi^2$.

Proof. Observe that $\overline{\;\cdot\;}\circ m$ is a natural transformation $\overline{\;\cdot\;}\circ\Phi^1\to\overline{\;\cdot\;}\circ\Phi^2\circ\Phi$ between functors $\mathbf{Sig}^1\to\mathbf{Sig}^{\mathbb{M}}$, and also that $\mathbb{M}_P^{L_i}\circ\Phi^i=\mathbb{M}_P\circ\overline{\;\cdot\;}\circ\Phi^i$. Then the result follows immediately from Thm. 4. $\qquad\square$

Finally we can give a uniform logic translation from modal logic to first-order logic:

Example 18 (Translating ML to FOL). Based on Ex. 16 and 17, we represent the well-known translation from modal logic to first-order logic that makes worlds explicit and relativizes quantifiers.

Firstly, the functor $\Phi:\mathbf{Sig}^{\mathbb{ML}}\to\mathbf{Sig}^{\mathbb{FOL}}$ maps every modal logic signature Σ to the $\mathbb{FOL}$ signature that contains a binary predicate symbol acc (for the accessibility relation) and one unary predicate symbol p for every propositional variable $p\in\Sigma$.

Secondly, we cannot give a $\mathbb{M}$ signature morphism $ML\to FOL$ because the translation requires the fixed predicate symbol acc, which is present in every $\Phi(\Sigma)$ but not in FOL^{syn}. Therefore, we put $L_1=ML$ and $L_2=FOL'=\overline{(FOL,\Sigma_0)}$ where Σ_0 is the LF signature FOL^{syn}, $acc:i\to i\to o$. Then $\Phi^{\mathbb{FOL}}\circ\Phi$ is indeed a functor $\mathbf{Sig}^{\mathbb{ML}}\to FOL'+\mathbb{LF}$.

Thirdly, we give a $\mathbf{Sig}^{\mathbb{M}}$ morphism $l:ML\to FOL'$ as follows. l^{syn} is given by

$$
\begin{array}{lcl}
\texttt{form} & := & i\to o\\
\supset & := & [f:i\to o]\,[g:i\to o]\,[x:i]\,(f\,x)\supset(g\,x)\\
\square & := & [f:i\to o]\,[x:i]\,\forall[y:i]\,((acc\,x\,y)\supset(f\,y))\\
\texttt{ded} & := & [f:i\to o]\,ded\,\forall[x:i]\,f\,x
\end{array}
$$

ML-sentences are mapped to FOL-formulas with a free variable (i.e., terms of type $i\to o$). The intuition is that the free variable represents the world in which the ML-sentence is evaluated. $\supset$ and $\square$ are translated in the expected way. Finally, the ML-truth judgment $\texttt{ded}\,F$ is mapped to the FOL-judgment $ded\,\forall[x]\,l^{syn}(F)\,x$; note how $\forall$ is used to bind the above-mentioned free variable and to obtain a sentence.

l^{pf} extends l^{syn} with assignments that map every proof rule of modal logic to a proof in first-order logic. These are technical but straightforward. l^{mod} is the identity for all symbols of HOL and maps

$$
\begin{array}{lcl}
worlds & := & univ\\
acc & := & acc
\end{array}
$$

It is easy to establish the commutativity requirements on (l^{syn},l^{pf},l^{mod}).

Fourthly, we define the natural transformation m. m_Σ must be a $\mathbb{M}+\mathbb{LF}$ morphism (l,α_Σ) from $\Phi^{\mathbb{FOL}}(\Phi(\Sigma))$ to $\Phi^{\mathbb{ML}}(\Sigma)$. For $\Sigma=\{p_1,\ldots,p_n\}$, the domain of α_Σ is

$$ML^{syn},p_1:\texttt{form},\ \ldots,\ p_n:\texttt{form},$$

and the codomain is

$$FOL^{syn},\ acc:i\to i\to o,\ p_1:i\to o,\ \ldots,\ p_n:i\to o.$$

Thus, we put $\alpha_\Sigma=l^{syn}$, $p_1:=p_1,\ \ldots,\ p_n:=p_n$.

Remark 17. In the typical case where Φ^1 and Φ^2 are injective, we can modify Def. 11 to use a functor $\Phi':L_1+\mathbb{LF}\to L_2+\mathbb{LF}$ such that $\Phi^2\circ\Phi=\Phi'\circ\Phi^1$. Φ' represents the same signature translation as Φ but lives within the meta-logic $\mathbb{M}$.

Then in the case of Ex. 18, Φ' and the natural transformation m are given simply by pushout along l. This yields a significantly simpler notion of uniform logic translation induced by l alone. However, not all signature translations can be obtained in this way. A counter-example is the translation from sorted to unsorted FOL translates single declarations in Σ to multiple declarations in $\Phi(\Sigma)$.

In general, Φ' and m can be formalized using the declaration patterns from Rem. 15, but we leave the details to future work.

Remark 18 (Non-simple Morphisms). Note that l in Ex. 18 is not simple, which finally justifies the particular choice in Def. 18 explained in Rem. 9. The induced sentence translation $\mathbf{Sen}^{\mathbb{M}}(l)$ can be factored into two steps: First a compositional translation maps F of type `form` to $l^{syn}(F)$ of type $i \to o$; it is followed by a non-compositional step that is only applied once on toplevel, which maps $l^{syn}(F)$ to $\forall[x]\, l^{syn}(F)\, x$ of type o.

This is typical for non-trivial logic translations: If the semantics of $\mathbb{I}_1$ is encoded using the syntax of $\mathbb{I}_2$, then $\mathbb{I}_1$-sentences are translated to some $\mathbb{I}_2$-expressions, which must be lifted to $\mathbb{I}_2$-sentences using a final toplevel step. Another example is the translation of many-valued propositional logic to first-order logic where formulas are translated compositionally to terms and a final toplevel step applies a unary predicate for designated truth values.

More generally, we can distinguish two kinds of sentence translation functions: the **within-logic** translations $\mathbf{Sen}(\sigma) : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma)$ translate along a signature morphism between two signatures of the same logic, and the **across-logic** translations $\alpha_\Sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\Phi(\Sigma))$ translate along a logic translation between two signatures of different logics. The corresponding observation applies to model and proof translations.

Within-logic translations are typically straightforward homomorphic mappings that can be represented as simple $\mathbf{Sig}^{\mathbb{M}}$-morphisms. But across-logic translations may involve complex encoding steps that may not be easily expressible using homomorphic mappings. When using a Grothendieck institution as in Diaconescu (2002), within-logic and across-logic translations are unified, and the latter are those that employ a non-trivial institution comorphism.

But if we use a meta-logic like $\mathbb{M}$, both within- and across-logic translations must be induced by $\mathbf{Sig}^{\mathbb{M}}$-morphisms. Therefore, we need our more general definition of $\mathbf{Sig}^{\mathbb{M}}$-morphisms that includes non-simple ones. This problem is a consequence of using a meta-logic in which a limited formal language is used to express translations. The problem does not arise when using institutions (as we do in Sect. 3) because both within-logic and across-logic translations are represented as mappings and functors, the most general notion of a translation operation.

6. Discussion

Our basic motivation was to provide a logical framework in which both model theoretical and proof theoretical logics as well as their combinations can be formulated and studied. But a logical framework – or any framework for that matter – can only serve as an auxiliary device: it helps get work done but does not do the work by itself. In fact, a

good logical framework should be general and weak in the sense of being foundationally uncommitted (see also de Bruijn (1991)).

Therefore, to evaluate an individual framework, we must ask how it supports the solution of which concrete problems. More concretely, we must ask *(i)* what logics can be represented in it, *(ii)* what applications become possible for represented logics, and *(iii)* whether the same could be achieved with other, more elegant frameworks. We will discuss these questions in Sect. 6.1, 6.2, and 6.3, respectively.

6.1. *Coverage and Limitations*

Logics Our definitions are chosen carefully to subsume both the framework of institutions and the one of LF so that existing logic representations can be reused or − in case they only cover either proof or model theory − complemented. The available institution-based representations of model theoretical logics (e.g., Borzyszkowski (2000); CoFI (The Common Framework Initiative) (2004); Goguen and Burstall (1992); Mossakowski et al. (2007)) become logic representations in our sense once their syntax and proof theory are made explicit in LF. Similarly, the available LF-based proof theoretical logic representations (e.g., Avron et al. (1992, 1998); Harper et al. (1993, 1994); Pfenning (2000, 2001)) become uniform logic definitions in our sense after encoding their model theory. In many cases, existing institution and LF-based representations of the same logic can be paired to complement each other.

Our definition of logic covers a wide range of logics including propositional, first-order, or higher-order logics; logics based on untyped, simply typed, dependently typed, or polymorphic type theories; modal, temporal, and description logics; and logics with classical or intuitionistic semantics. Multiple truth values such as in three-valued logics are supported if some truth values are *designated* to obtain a two-valued satisfaction relation in the models (as we did in Goguen et al. (2007)), or if the syntax includes constants for the truth values.

The syntax of a logic can typically be expressed in our meta-logic $\mathbb{M}$. A limitation concerns logics where the notion of well-formed formulas is undecidable (such as in PVS, Owre et al. (1992)). This requires to merge Σ^{syn} and Σ^{pf} to axiomatize well-formedness and take a subtype of $base(\texttt{form})$ as the type of sentences. This is possible with the more general framework we used in Rabe (2008), but we have omitted this here for simplicity.

Regarding proof theories, $\mathbb{M}$ inherits from LF the support of Hilbert, natural deduction, sequent, or tableaux calculi. Resolution-based proof theories cannot be represented as elegantly. Also, our notions of proof categories is biased against substructural logics, see Rem. 1. This is a bias shared with both institutions and LF, the latter has even been extended to a linear variant for that reason (Cervesato and Pfenning (2002)).

The representation of model theories in our meta-logic depends on two things. Firstly, the foundation must be expressible in LF; this is typically the case. Secondly, it must be adequate to represent models as morphisms; we discussed this in Rem. 16.

Our logical framework permits expressing signatures, sentences, models, and proofs of logics as LF objects. In addition, we identified two extensions of LF that we need to reach a fully comprehensive framework. Firstly, logical relations should be developed and used

to represent model morphisms (see Rem. 11). Secondly, a language of declaration patterns is needed to express the category of signatures as a whole as opposed to individual signatures (see Rem. 15).

Logic Translations For logic translations, the situation is more complicated. Firstly, logic translations are less well classified than logics, which complicates a systematic study of which classes of logic translations are covered. Secondly, the representation of logic translations is significantly harder than that of logics. In general, features present in covered syntax translations are expressing the model theory of one logic using the syntax of another, or coding sentences of one logic as terms of another logic.

A sentence translation is covered if it is compositional up to a final toplevel step as in Ex. 18.

A proof theory translation is covered if it translates inference rules to derivable rules. This excludes the non-compositional elimination of admissible rules such as cut elimination (where sentences and models are translated to themselves and proofs to cut-free proofs). Representing such translations requires a stronger notion of LF signature morphism that supports computation, e.g., along the lines of Twelf's logic programs, Delphin (Poswolsky and Schürmann (2008)), or Beluga (Pientka and Dunfield (2010)). But the typing invariants of such signature morphisms would be a lot more complicated.

Model theory translations are typically covered if the model theories themselves are covered.

In Rem. 18, we have already discussed the need for non-simple $\mathbf{Sig}^{\mathbb{M}}$-morphisms. We expect this to be a general phenomenon: While within-logic translations will always be represented by simple morphisms, the representation of more complex across-logic translations will require more complex notions of $\mathbf{Sig}^{\mathbb{M}}$-morphisms. Consequently, we expect future work to further generalize the definition of $\mathbf{Sig}^{\mathbb{M}}$-morphisms.

For example, some translations can only be encoded if the commutativity condition of Def. 18 is relaxed, especially for the lower rectangle dealing with the model translation. For example, the semantics of description logic interprets concepts as subsets of the universe. But the translation of description logic into first-order logic translates concepts to unary predicates, which the semantics of first-order logic interprets as functions from the universe to the booleans. Thus, the model translation commutes only up to the isomorphism between subsets and their characteristic functions. Recently a weaker condition than commutativity was shown to be sufficient to obtain logic comorphisms in Sojakova (2010), namely commutativity up to certain logical relations (see also Rem. 11).

Finally there are some limitations of logic comorphisms, which are inherited from institution comorphisms. Firstly, logic comorphisms are limited to total translations. This excludes partial translations such as a translation from higher-order to first-order logic that is undefined for expressions containing λ. Such translations are important in practice to borrow (semi-)automated theorem provers (e.g., the use of first-order provers in Isabelle) but difficult to describe in a logical framework. To overcome that, we recently designed an extension of $\mathbb{LF}$ that supports partial signature morphisms (Dumbrava and Rabe (2010)).

Secondly, logic comorphisms separate the signature and the sentence translation. But there are applications of borrowing where a different signature translation is used for different conjectures over the same signature. For example, the Leo-II prover (Benzmüller et al. (2008)) uses a translation from higher-order logic to sorted first-order logic by creating a new first-order sort for every function type that is mentioned in the higher-order conjecture. Such translations remain future work.

Case Studies In Rabe and Kohlhase (2011), we designed a generic module system based on theories and theory morphisms. In Rabe and Schürmann (2009), we extended the Twelf implementation of LF with the corresponding instance of this module system. We have used this implementation to conduct several large case studies where logics and logic translations are defined and machine-checked efficiently.

A large number of case studies has been conducted in the LATIN project Codescu et al. (2011). These include various incarnations of first-order logics, higher-order logics, modal and description logics as well as a number of set and type theoretical foundations. These are written as modular LF signatures and morphisms, and the overall size of the atlas exceeds 1000 modules.

The most extensive case studies were presented in Horozal and Rabe (2011) and Iancu and Rabe (2011). In Horozal and Rabe (2011), we give a comprehensive representation of first-order logic. It includes a formalization of ZFC set theory to formalize the model theory of FOL and a formalized soundness proof of FOL in the sense of Thm. 6.

The biggest investment when representing logics in $\mathbb{M}$ is the formalization of the foundation of mathematics. Therefore, we formalized three important foundations in Iancu and Rabe (2011): Zermelo Fraenkel set theory (Fraenkel (1922); Zermelo (1908)), the logical framework Isabelle and the higher-order logic Isabelle/HOL (Nipkow et al. (2002); Paulson (1994)), and the Mizar system for Tarski-Grothendieck set theory (Trybulec and Blair (1985)). These include translations between the foundations, which permits even the translation of models across foundations.

6.2. *Applications*

Our framework was designed in response to a number of specific problems we encountered in practice. In all cases, the lack of a comprehensive framework like ours had previously precluded general solutions.

An Atlas of Logics An important application of logical frameworks is to structure and relate the multitude of logics in use. This goal was a central motivation of institutions and has been pursued in LF as well (Pfenning et al. (2003)). And both in model and proof theoretical frameworks, there are substantial collections of presentations of model and proof theoretical logics, respectively. But a collection comprising both model and proof theory has so far been lacking.

Our framework provides the theoretical base of the LATIN project (Codescu et al. (2011), see also Sect. 6.1), which systematically builds such a collection of logic presentations. The LATIN atlas contains formalizations of logics and logic translations presented

as a diagram of $\mathbb{M}$ signatures. It is made available as a web-based portal of documented logic definitions.

Logic-Aware Machines A central motivation of logical frameworks has been the goal to generically mechanize arbitrary logics: The implementation of one logical framework induces implementations of individual logics defined in it. This requires an abstract definition of logic and a machine-understandable language in which individual logics can be presented. Our framework provides both by presenting individual uniform logics as $\mathbb{M}$-signatures, i.e., LF spans. Therefore, we can now use implementations of LF – such as Twelf (Pfenning and Schürmann (1999)) – to mechanically process logic presentations.

We have given an example for this work flow in Codescu et al. (2012). The Hets system (Mossakowski et al. (2007)) implements the framework of institutions and acts as a mediator between parsers, static analyzers, and theorem provers for various object logics. But previously Hets could only work with a fixed number of institutions and comorphisms implemented individually in the underlying programming language. Our framework connects Hets to declarative logic presentations in LF, and Hets is now able to work with arbitrary uniform logics defined in $\mathbb{M}$.

Rapidly Prototyping Logics The definition of a new logic is typically the result of a long quest, and the evaluation of candidate logics often requires large case studies, which in turn usually require sophisticated machine support. Logical frameworks can support this process by permitting the swift implementation of candidate logics.

Using our framework, users can present candidate logics concisely and quickly, and Twelf immediately induces an implementation. In particular, Twelf provides type reconstruction and module system out of the box. Both features require a major investment to realize for individual logics but are indispensable in practice. Moreover, the module system can be used to build a library of logic components that serve as building blocks for large logics.

We employ this methodology in the LATIN atlas: Each logic feature (such as a connective or a type formation operator) is presented separately along with its proof and model theoretical semantics, and logics are composed using colimits. Therefore, the presentation of each logic only requires formalizing those features not already present in LATIN. Moreover, LATIN includes a library of modular formalizations of a variety of type and set theories so that logics can be built easily on top of and interpreted in a wide range of languages.

Verified Heterogeneous Reasoning At the moment mechanically verified formalizations of mathematics are restricted to individual proof theoretical logics, whose implementations find and verify theorems, e.g., Isabelle/HOL Nipkow et al. (2002) or Mizar Trybulec and Blair (1985). In this context, heterogeneous reasoning, i.e., the reuse of theorems along logic translations, is problematic because the translation engine would lie outside the verifying system. Applications are usually limited to translating and dynamically reverifying a suite of theorems (e.g., Krauss and Schropp (2010), Keller and Werner (2010)).

On the other hand, model theoretical adequacy (in the sense of Def. 13) has been used very successfully in the institution community to borrow proof systems along a statically verified logic translation (Cerioli and Meseguer (1997)). But while there are systems (like Hets Mossakowski et al. (2007)) that permit presenting and applying such translations, there is no mechanized support for verifying them.

Our framework provides such support. For example, Sojakova (2010) uses it to prove the model theoretical adequacy of the translation from modal logic to first-order logic. Well-formedness and adequacy of the translation reduce to type checking in LF, which is verified mechanically by Twelf.

Logical Knowledge Management In Rabe and Kohlhase (2011), we have developed the MMT interface language for logic-related applications. The latter include both deduction systems such as theorem provers, proof assistants, or type checkers as well as management systems such as databases, browsers, IDEs, or search engines. MMT combines a modular representation format with a scalable knowledge management infrastructure (Kohlhase et al. (2010)) to provide an exchange format for logics, signatures, and theories as well as their morphisms, for foundations, and for expressions, proofs, and models.

For example, these objects can be presented using Twelf, exported as MMT, and imported by any other application. In particular, authors can use Twelf's human-oriented concrete syntax including the reconstruction of omitted terms and types, and the importing application can use fully reconstructed and thus easily machine-processable MMT format. This work flow is used in Codescu et al. (2012) to present logics in Twelf and use them in Hets.

6.3. *Related Work*

Frameworks based on Model Theory There are several closely related frameworks that define logics and logic comorphisms (possibly with different names) as certain tuples of categorical objects.

Our definitions of logics and logic comorphisms follow and subsume those of *institutions* and institution comorphisms (Goguen and Burstall (1992)). We obtain a forgetful functor from logics to institutions by dropping the proof theory from a logic and the proof translation from a logic comorphism. Similarly, if we drop the condition on proof translations in Def. 10, we recover institution comorphism modifications. These were introduced in Diaconescu (2002), albeit with a weaker condition; in Tarlecki (1996) a similar concept was called a representation map. Our Def. 10 extends the definition of institution comorphism modification given in Mossakowski (2005) in the expected way.

In Meseguer (1989), *general logics* are introduced as tuples $(\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models, \vdash)$ where $\vdash$ is an *entailment system* between the sentences. Entailment systems formalize the provability relation without formalizing proofs. Our use of $\vdash_\Sigma F$ as a truth judgment is different from entailment systems, but our provability relation $\Theta \vdash_\Sigma F$ is always an entailment system. In addition, Meseguer (1989) defines a proof calculus as a functor from theories to a fixed but arbitrary category **Str**. Our definition of proof categories can be

seen as the special case **Str** $= \mathcal{PFCAT}$; in this special case, proofs can be represented as morphisms.

In Mossakowski et al. (2005) and Diaconescu (2006) *proof theoretic institutions* are introduced as essentially tuples $(\mathbf{Sig}, \mathbf{Sen}, \mathbf{Mod}, \models, \mathbf{Pf})$. Here the relation between sentences and objects of the proof category is predetermined because the objects of the proof categories are always the sets of sentences. Our definition of proof categories is more general and uses the truth judgment $\vdash$ to relate sentences to objects in the proof category. In particular, the objects of our proof categories can be the multi-sets of sentences, which solves a problem discovered by us in an early revision of Diaconescu (2006): There the free generation of a proof system by a set of rules is restricted to logics with signature morphisms $\sigma : \Sigma \to \Sigma'$ for which $\mathbf{Sen}(\sigma)$ is injective. Otherwise, σ would not induce a canonical functor $\mathbf{Pf}(\sigma)$ between proof categories. Later Lawvere theories were used to overcome this problem in the revised version of Mossakowski et al. (2005): There, no size restriction on the products is used, and proof categories are not small.

Mossakowski et al. (2005) also generalizes to $\mathbb{C}/\mathbb{D}$-*institutions* where $\mathbb{C}$ and $\mathbb{D}$ are the codomains of **Pf** and **Mod**, respectively, and **Sen** is obtained by composing **Pf** with a fixed functor $\mathbb{C} \to \mathcal{SET}$. Our logics are almost $\mathcal{PFCAT}/\mathcal{CAT}$ institutions; the difference is again that we give **Sen** and **Pf** separately and relate them via $\vdash$.

In Fiadeiro and Sernadas (1988), Π-*institutions* are introduced as tuples $(\mathbf{Sig}, \mathbf{Sen}, Cn)$ where Cn is a closure operator on sets of formulas defining consequence. Cn is treated proof theoretically, and theory morphisms are studied proof theoretically.

The idea of using a meta-logic like $\mathbb{M}$ is well-known, and was studied for institutions in e.g., Tarlecki (1996). Our generalization to logic encodings is not surprising, and our notion of model theoretical adequacy is known as the model expansion property. The major novelty in our work is the use of LF as a concrete meta-logic.

In Martí-Oliet and Meseguer (1996), *rewriting logic* is proposed as another concrete meta-logic within an informal framework similar to general logics. Rewriting logic arises by combining sorted first-order logic with term rewriting, which makes it very different from LF. In rewriting logic, typically, the syntax of an object logic is represented as a sorted first-order theory, and the proof theory as a set of rewriting rules; the model theory is not represented syntactically but via the model theory of rewriting logic. The main advantage of rewriting logic over LF is the use of rewriting to simplify expressions; the main advantage of LF is the use of dependent type theory and of higher-order abstract syntax to represent proofs and variable binding as expressions.

The most advanced implementations of such frameworks are the Hets implementation of institutions (Mossakowski et al. (2007)) and the Maude implementation of a simplified variant of rewriting logic (Clavel et al. (1996)). Hets uses Haskell as an implementation language for the syntax of institutions and comorphisms. It maintains a graph of logics via which theories can be translated between logics and serves as middleware between implementations of individual logics. Maude uses theories of rewriting logic to represent the syntax and proof theory of object logics as theories of the meta-logic. It implements term rewriting to provide computation within the logical framework.

Hets implements an abstract framework (institutions) whereas Maude implements a

concrete meta-logic within a given framework (general logics). The latter corresponds to our approach: Twelf (Pfenning and Schürmann (1999)) implements $\mathbb{LF}$ and thus $\mathbb{M}$ within the framework of our logics; in particular, logics are represented declaratively. The former is complementary to our approach, for example, we have used $\mathbb{M}$ as a convenient way to add logics to Hets in Codescu et al. (2012). Unlike Hets and Maude, our approach also permits the mechanized representation of model theory.

Frameworks based on Proof Theory Our definition of $\mathbb{M}$ subsumes the logical framework LF. The use of LF signatures L^{syn} and L^{pf} to obtain proof theoretical logic encodings are well-understood (see, e.g., Pfenning (2001)). The reasoning about adequacy of these encodings corresponds to our comorphisms (Φ, α, γ). A difference is that in the proof theoretical community, such encodings are considered adequate if α and γ are isomorphisms; we require only a weaker condition on γ here.

Isabelle (Paulson (1994)) is an alternative logical framework, which uses higher-order logic with shallow polymorphism (Church (1940)) instead of dependent type theory. The main advantage over LF are recursive and inductive computation and (semi-)automated reasoning support. Other type theories such as Martin-Löf type theory in Agda (Martin-Löf (1974); Norell (2005)) or the calculus of constructions in Coq (Bertot and Castéran (2004); Coquand and Huet (1988)) are sometimes used as logical frameworks as well and have similar advantages. The main advantage of LF are the use of higher-order abstract syntax, which facilitates adequacy proofs, and the support for signature morphisms in Twelf (Pfenning and Schürmann (1999)).

Our definition of $\mathbb{M}$ and our results about it depend only on a few properties of LF. The main assumptions are that the signatures and for each signature the contexts form categories, and that there are pushouts along inclusions. This is the case for almost all type theories. Thus, our definitions can be generalized easily to most type theories subsuming LF such as Martin-Löf type theory or the calculus of constructions. Even though the signature *Base* uses dependent types, it is also easy to adapt the definitions to (simply-typed) higher-order logic. Then the signature *Base* contains `ded : form → prop` where `prop` is the type of propositions. This corresponds to logic encodings in Isabelle where `ded` is typically called `Trueprop`.

While logic translations play a major role in model theoretical frameworks, they have been studied less systematically for proof theoretical frameworks. The most comprehensive approach was the Logosphere project (Pfenning et al. (2003)), which employed LF as a framework for logic translations. These were implemented in two ways: firstly, using the operational semantics of Twelf based on logic programming, secondly using the specifically developed functional programming language Delphin (Poswolsky and Schürmann (2008)). Our use of LF signature morphisms differs because our translations are represented in a declarative language.

Other logic translations are typically represented ad-hoc, i.e., as implementations outside a logical framework, e.g., Keller and Werner (2010); Krauss and Schropp (2010); McLaughlin (2006); Obua and Skalberg (2006). Instead of appealing to general results about the framework, the correctness of such a translation can be guaranteed by veri-

fying the translated proofs. The disadvantage is that proofs must be produced, stored, translated, and verified for each translated theorem.

Representing Models Our representation of models as morphisms goes back to Lawvere's representation of models as functors (Lawvere (1963)) and the use of initial algebras as the semantics of theories (Goguen et al. (1978)). This approach has been applied most systematically in the area of categorical models of type theories (see, e.g., Pitts (2000)) and has been integrated into model theoretical frameworks (e.g., Fiadeiro and Sernadas (1988); Goguen and Burstall (1986); Goguen et al. (2007)). The representation of models as signature morphisms into some fixed signature has also been used to show the amalgamation property of given institutions (see, e.g., Diaconescu (2008)).

From a model theoretical perspective, the novelty of our approach is to use morphisms in a category – namely $\mathbb{LF}$ – whose objects and morphisms are given in terms of concrete syntax and do not depend on any foundation of mathematics. That makes it possible to represent models as expressions in a mechanized language. Moreover, by separating Σ^{syn} and Σ^{mod} and by using morphisms out of Σ^{mod} as models (rather than morphisms out of Σ^{syn}), we can represent logics whose models have a different structure than the syntax. For example, Kripke models of modal logic use a set of worlds and an accessibility relation, which are not present in the syntax.

In type theories, model theory has been represented occasionally in a similar way. For example, in Benton et al. (2009); Coquand and Dybjer (1997) the syntax of formal languages is represented as an inductive data type and models as inductive functions out of it. Isabelle (Paulson (1994)) provides locales and interpretations, which behave very similarly to LF signatures and signature morphisms.

From a type theoretical perspective, the novelty of our approach is the systematic design of a logical framework on top of this intuition. Moreover, we abstract from the foundation by permitting the use of an arbitrary $\mathbb{LF}$ signature. Thus, we avoid the commitment to a specific foundation such as higher-order logic or Martin-Löf type theory that is often inherent in type theoretical representations of models.

7. Conclusion

A Comprehensive Logical Framework We gave a logical framework that permits comprehensive logic representations: The syntax, model theory, and proof theory of logics as well as their translations are represented within a formal logical framework. In particular, our approach combines the model theoretical and the proof theoretical perspectives. Despite the ontological and philosophical differences of these two perspectives on logic, we are able to preserve the flavor and advantages of either one.

Specifically, we extended the model theoretical framework of institutions (Goguen and Burstall (1992)) with the notions of judgments and proof categories. Our definitions preserve the elegance and abstraction of institutions while permitting a natural integration of proof theoretical logics, which can be seen as the continuation of work undertaken in Meseguer (1989) and Mossakowski et al. (2005).

Correspondingly, we extended the proof theoretical framework LF (Harper et al. (1993)) with the notions of homomorphisms and model categories. We use a logic based on LF as a meta-logic in which object logics are represented, a reply to a suggestion made in Tarlecki (1996). Using this meta-logic, all logical notions are defined as syntactic objects in the LF type theory and can thus be verified mechanically. This includes a special LF signature representing the foundation of mathematics so that we can represent model theory even though our meta-logic is foundationally uncommitted.

Evaluation Using the criteria we gave at the beginning of Sect. 6, we can conclude as follows. *(i)* As described in Sect. 6.1, our framework preserves the large collection of existing logic encodings in institutions and LF. Moreover, since these encodings usually covered either the model theory or the proof theory, in many cases we can now give comprehensive encodings for the first time. Regarding logic translations, our work errs on the side of simplicity in the simplicity-expressivity trade-off, and extensions remain future work.

(ii) In Sect. 6.2, we described a number of applications that are enabled by our framework, focusing on those that are already underway. The key feature here is that our framework provides both an abstract definition of logics and a simple declarative language, in which such logics can be presented concisely and easily.

(iii) Finally, our framework has been designed systematically as the simplest possible evolution of the existing frameworks which combines the above two properties. A special strength is the symmetric treatment of model theory $(\mathbf{Mod}, \models)$ and proof theory $(\mathbf{Pf}, \vdash)$. Moreover, the models-as-morphisms paradigm yields an elegant formalist representation of platonist model theory.

Future Work Future work will focus on three lines of research. Firstly, the theoretical framework and the tool support that is in place now can be leveraged in the practical applications outlined in Sect. 6.2. In particular, we have started the work on a logic atlas in the LATIN project (Codescu et al. (2011)).

Secondly, some of the limitations discussed in Sect. 6.1 can be overcome by using different underlying categories than $\mathbb{LF}$. For example, we can use a type theory with non-compositional or partial functions. Our results can be extended easily to such type theories.

Thirdly, our framework can be used for the theoretical analysis of a logic's meta-theory. This includes the mechanization of soundness and completeness proofs: Soundness will use Thm. 10; completeness is currently open, but we consider the approach along the lines of Rem. 13 very promising. A related application is the modular development of logics as envisioned in Harper et al. (1994) and now implemented in the LATIN project.

References

Aiguier, M. and Diaconescu, R. (2007). Stratified institutions and elementary homomorphisms. *Information Processing Letters*, 103(1):5–13.

Andrews, P. (1986). *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press.

Avron, A., Honsell, F., Mason, I., and Pollack, R. (1992). Using typed lambda calculus to implement formal systems on a machine. *Journal of Automated Reasoning*, 9(3):309–354.

Avron, A., Honsell, F., Miculan, M., and Paravano, C. (1998). Encoding modal logics in logical frameworks. *Studia Logica*, 60(1):161–208.

Awodey, S. and Rabe, F. (2011). Kripke Semantics for Martin-Löf's Extensional Type Theory. *Logical Methods in Computer Science*, 7(3).

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Barendregt, H. (1992). Lambda calculi with types. In Abramsky, S., Gabbay, D., and Maibaum, T., editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press.

Barwise, J. (1977). An Introduction to First-Order Logic. In Barwise, J., editor, *Handbook of Mathematical Logic*, pages 5–46. North-Holland.

Benton, N., Kennedy, A., and Varming, C. (2009). Some Domain Theory and Denotational Semantics in Coq. In Berghofer, S., Nipkow, T., Urban, C., and Wenzel, M., editors, *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 115–130. Springer.

Benzmüller, C., Paulson, L., Theiss, F., and Fietzke, A. (2008). LEO-II - A Cooperative Automatic Theorem Prover for Classical Higher-Order Logic (System Description). In Armando, A., Baumgartner, P., and Dowek, G., editors, *Automated Reasoning*, pages 162–170. Springer.

Bertot, Y. and Castéran, P. (2004). *Coq'Art: The Calculus of Inductive Constructions*. Springer.

Béziau, J., editor (2005). *Logica Universalis*. Birkhäuser Verlag.

Borzyszkowski, T. (2000). Higher-Order Logic and Theorem Proving for Structured Specifications. In Choppy, C., Bert, D., and Mosses, P., editors, *Workshop on Algebraic Development Techniques*, LNCS, pages 401–418.

Bourbaki, N. (1974). *Algebra I*. Elements of Mathematics. Springer.

Brachman, R. and Schmolze, J. (1985). An Overview of the KL-ONE Knowledge Representation Scheme. *Cognitive Science*, 9(2).

Brouwer, L. (1907). *Over de grondslagen der wiskunde*. PhD thesis, Universiteit van Amsterdam. English title: On the Foundations of Mathematics.

Cartmell, J. (1986). Generalized algebraic theories and contextual category. *Annals of Pure and Applied Logic*, 32:209–243.

Căzănescu, V. and Roşu, G. (1997). Weak inclusion systems. *Mathematical Structures in Computer Science*, 7(2):195–206.

Cerioli, M. and Meseguer, J. (1997). May I Borrow Your Logic? (Transporting Logical Structures along Maps). *Theoretical Computer Science*, 173:311–347.

Cervesato, I. and Pfenning, F. (2002). A Linear Logical Framework. *Information and Computation*, 179(1):19–75.

Church, A. (1940). A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(1):56–68.

Clavel, M., Eker, S., Lincoln, P., and Meseguer, J. (1996). Principles of Maude. In Meseguer, J., editor, *Proceedings of the First International Workshop on Rewriting Logic*, volume 4, pages 65–89.

Codescu, M., Horozal, F., Kohlhase, M., Mossakowski, T., and Rabe, F. (2011). Project Abstract: Logic Atlas and Integrator (LATIN). In Davenport, J., Farmer, W., Rabe, F., and Urban, J., editors, *Intelligent Computer Mathematics*, volume 6824 of *Lecture Notes in Computer Science*, pages 287–289. Springer.

Codescu, M., Horozal, F., Kohlhase, M., Mossakowski, T., Rabe, F., and Sojakova, K. (2012). Towards Logical Frameworks in the Heterogeneous Tool Set Hets. In Mossakowski, T. and Kreowski, H., editors, *Recent Trends in Algebraic Development Techniques 2010*, volume 7137 of *Lecture Notes in Computer Science*, pages 139–159. Springer.

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

Coquand, T. and Dybjer, P. (1997). Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science*, 7(1):75–94.

Coquand, T. and Huet, G. (1988). The Calculus of Constructions. *Information and Computation*, 76(2/3):95–120.

Curry, H. and Feys, R. (1958). *Combinatory Logic*. North-Holland, Amsterdam.

de Bruijn, N. (1970). The Mathematical Language AUTOMATH. In Laudet, M., editor, *Proceedings of the Symposium on Automated Demonstration*, volume 25 of *Lecture Notes in Mathematics*, pages 29–61. Springer.

de Bruijn, N. (1991). A Plea for Weaker Frameworks. In Huet, G. and Plotkin, G., editors, *Logical Frameworks*, pages 40–67. Cambridge University Press.

Diaconescu, R. (2002). Grothendieck institutions. *Applied Categorical Structures*, 10(4):383–402.

Diaconescu, R. (2006). Proof systems for institutional logic. *Journal of Logic and Computation*, 16(3):339–357.

Diaconescu, R. (2008). *Institution-independent Model Theory*. Birkhäuser.

Dumbrava, S. and Rabe, F. (2010). Structuring Theories with Partial Morphisms. In *Workshop on Algebraic Development Techniques*.

Farmer, W. (2010). Chiron: A set theory with types, undefinedness, quotation, and evaluation. SQRL Report 38, McMaster University.

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

Fiadeiro, J. and Sernadas, A. (1988). Structuring theories on consequence. In *Recent Trends in Data Type Specification*, pages 44–72. Springer.

Fraenkel, A. (1922). Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237. English title: On the Foundation of Cantor-Zermelo Set Theory.

Gentzen, G. (1934). Untersuchungen über das logische Schließen. *Math. Z.*, 39. English title: Investigations into Logical Deduction.

Girard, J. (1987). Linear Logic. *Theoretical Computer Science*, 50:1–102.

Gödel, K. (1930). Die Vollständigkeit der Axiome des Logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360. English title: The Completeness of the Axioms of the Logical Calculus of Functions.

Goguen, J. and Burstall, R. (1986). A study in the foundations of programming methodology: specifications, institutions, charters and parchments. In Pitt, D., Abramsky, S., Poigné, A., and Rydeheard, D., editors, *Workshop on Category Theory and Computer Programming*, pages 313–333. Springer.

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

Goguen, J., Mossakowski, T., de Paiva, V., Rabe, F., and Schröder, L. (2007). An Institutional View on Categorical Logic. *International Journal of Software and Informatics*, 1(1):129–152.

Goguen, J. and Rosu, G. (2002). Institution morphisms. *Formal Aspects of Computing*, 13:274–307.

Goguen, J., Thatcher, J., and Wagner, E. (1978). An initial algebra approach to the specification, correctness and implementation of abstract data types. In Yeh, R., editor, *Current Trends in Programming Methodology*, volume 4, pages 80–149. Prentice Hall.

Gordon, M. (1988). HOL: A Proof Generating System for Higher-Order Logic. In Birtwistle, G. and Subrahmanyam, P., editors, *VLSI Specification, Verification and Synthesis*, pages 73–128. Kluwer-Academic Publishers.

Haftmann, F. and Wenzel, M. (2006). Constructive Type Classes in Isabelle. In Altenkirch, T. and McBride, C., editors, *TYPES conference*, pages 160–174. Springer.

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

Harper, R., Sannella, D., and Tarlecki, A. (1994). Structured presentations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160.

Harrison, J. (1996). HOL Light: A Tutorial Introduction. In *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*, pages 265–269. Springer.

Henkin, L. (1950). Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91.

Hilbert, D. (1926). Über das Unendliche. *Mathematische Annalen*, 95:161–90.

Horozal, F. and Rabe, F. (2011). Representing Model Theory in a Type-Theoretical Logical Framework. *Theoretical Computer Science*, 412(37):4919–4945.

Howard, W. (1980). The formulas-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press.

Iancu, M. and Rabe, F. (2011). Formalizing Foundations of Mathematics. *Mathematical Structures in Computer Science*, 21(4):883–911.

Keller, C. and Werner, B. (2010). Importing HOL Light into Coq. In Kaufmann, M. and Paulson, L., editors, *Interactive Theorem Proving*, pages 307–322. Springer.

Kohlhase, M., Rabe, F., and Zholudev, V. (2010). Towards MKM in the Large: Modular Representation and Scalable Software Architecture. In Autexier, S., Calmet, J., Delahaye, D., Ion, P., Rideau, L., Rioboo, R., and Sexton, A., editors, *Intelligent Computer Mathematics*, volume 6167 of *Lecture Notes in Computer Science*, pages 370–384. Springer.

Krauss, A. and Schropp, A. (2010). A Mechanized Translation from Higher-Order Logic to Set Theory. In Kaufmann, M. and Paulson, L., editors, *Interactive Theorem Proving*, pages 323–338. Springer.

Lambek, J. and Scott, P. (1986). *Introduction to Higher-Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press.

Lawvere, F. (1963). *Functional Semantics of Algebraic Theories*. PhD thesis, Columbia University.

Lawvere, W. (1969). Adjointness in Foundations. *Dialectica*, 23(3–4):281–296.

Mac Lane, S. (1998). *Categories for the working mathematician*. Springer.

Martí-Oliet, N. and Meseguer, J. (1996). Rewriting Logic as a Logical and Semantic Framework. In *Rewriting Logic and its Applications*, volume 4 of *Electronic Notes in Theoretical Computer Science*, pages 352–358.

Martin-Löf, P. (1974). An Intuitionistic Theory of Types: Predicative Part. In *Proceedings of the '73 Logic Colloquium*, pages 73–118. North-Holland.

Martin-Löf, P. (1996). On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):3–10.

McLaughlin, S. (2006). An Interpretation of Isabelle/HOL in HOL Light. In Shankar, N. and Furbach, U., editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*. Springer.

Meseguer, J. (1989). General logics. In Ebbinghaus, H.-D. et al., editors, *Proceedings, Logic Colloquium, 1987*, pages 275–329. North-Holland.

Mossakowski, T. (2005). Heterogeneous Specification and the Heterogeneous Tool Set. Habilitation thesis, see http://www.informatik.uni-bremen.de/~till/.

Mossakowski, T., Goguen, J., Diaconescu, R., and Tarlecki, A. (2005). What is a logic? In Béziau, J., editor, *Logica Universalis*, pages 113–133. Birkhäuser Verlag.

Mossakowski, T., Maeder, C., and Lüttich, K. (2007). The Heterogeneous Tool Set. In O. Grumberg and M. Huth, editor, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522.

Mossakowski, T., Tarlecki, A., and Pawlowski, W. (1997). Combining and Representing Logical Systems. In Moggi, E. and Rosolini, G., editors, *Category Theory and Computer Science*, pages 177–196. Springer.

Naumov, P., Stehr, M., and Meseguer, J. (2001). The HOL/NuPRL proof translator - a practical approach to formal interoperability. In *14th International Conference on Theorem Proving in Higher Order Logics*. Springer.

Nipkow, T., Paulson, L., and Wenzel, M. (2002). *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer.

Norell, U. (2005). The Agda WiKi. http://wiki.portal.chalmers.se/agda.

Obua, S. and Skalberg, S. (2006). Importing HOL into Isabelle/HOL. In Shankar, N. and Furbach, U., editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*. Springer.

Owre, S., Rushby, J., and Shankar, N. (1992). PVS: A Prototype Verification System. In Kapur, D., editor, *11th International Conference on Automated Deduction (CADE)*, pages 748–752. Springer.

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

Pfenning, F. (2000). Structural cut elimination: I. intuitionistic and classical logic. *Information and Computation*, 157(1-2):84–141.

Pfenning, F. (2001). Logical frameworks. In *Handbook of automated reasoning*, pages 1063–1147. Elsevier.

Pfenning, F. and Schürmann, C. (1999). System description: Twelf - a meta-logical framework for deductive systems. *Lecture Notes in Computer Science*, 1632:202–206.

Pfenning, F., Schürmann, C., Kohlhase, M., Shankar, N., and Owre., S. (2003). The Logosphere Project. http://www.logosphere.org/.

Pientka, B. and Dunfield, J. (2010). A Framework for Programming and Reasoning with Deductive Systems (System description). In *International Joint Conference on Automated Reasoning*. To appear.

Pitts, A. (2000). Categorical Logic. In Abramsky, S., Gabbay, D., and Maibaum, T., editors, *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*, chapter 2, pages 39–128. Oxford University Press.

Poswolsky, A. and Schürmann, C. (2008). System Description: Delphin - A Functional Programming Language for Deductive Systems. In Abel, A. and Urban, C., editors, *International Workshop on Logical Frameworks and Metalanguages: Theory and Practice*, pages 135–141. ENTCS.

Rabe, F. (2008). *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen. see http://kwarc.info/frabe/Research/phdthesis.pdf.

Rabe, F. and Kohlhase, M. (2011). A Scalable Module System. see http://arxiv.org/abs/1105.0548.

Rabe, F. and Schürmann, C. (2009). A Practical Module System for LF. In Cheney, J. and Felty, A., editors, *Proceedings of the Workshop on Logical Frameworks: Meta-Theory and Practice (LFMTP)*, volume LFMTP'09 of *ACM International Conference Proceeding Series*, pages 40–48. ACM Press.

Robinson, A. (1950). On the application of symbolic logic to algebra. In *Proceedings of*

the International Congress of Mathematicians, pages 686–694. American Mathematical Society.

Seely, R. (1984). Locally cartesian closed categories and type theory. *Math. Proc. Cambridge Philos. Soc.*, 95:33–48.

Smullyan, R. (1995). *First-Order Logic*. Dover, second corrected edition.

Sojakova, K. (2010). Mechanically Verifying Logic Translations. Master's thesis, Jacobs University Bremen.

Tarlecki, A. (1996). Moving between logical systems. In Haveraaen, M., Owe, O., and Dahl, O.-J., editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer Verlag.

Tarski, A. (1933). Pojęcie prawdy w językach nauk dedukcyjnych. *Prace Towarzystwa Naukowego Warszawskiego Wydział III Nauk Matematyczno-Fizycznych*, 34. English title: The concept of truth in the languages of the deductive sciences.

Tarski, A. and Vaught, R. (1956). Arithmetical extensions of relational systems. *Compositio Mathematica*, 13:81–102.

Trybulec, A. and Blair, H. (1985). Computer Assisted Reasoning with MIZAR. In Joshi, A., editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28.

Whitehead, A. and Russell, B. (1913). *Principia Mathematica*. Cambridge University Press.

Zermelo, E. (1908). Untersuchungen über die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281. English title: Investigations in the foundations of set theory I.