# The MMT Perspective on Conservativity*

**Florian Rabe**[1]

1 **Jacobs University Bremen, Germany**
f.rabe@jacobs-university.de

## Abstract

Conservative extensions are one of the most important concepts in formal logic, capturing the intuition when an extension does not substantially change the extended theory. Two conceptually different definitions have emerged in proof and model theory, respectively. Unfortunately these are conceptually very different and not equivalent.

We develop both notions in the MMT framework, which allows for an elegant uniform treatment of proof and model theory. In MMT, the difference between the two definitions becomes less fundamental. Moreover, we see that the existence of two different notions of conservativity is neither a coincidence nor a defect: it becomes a special case of the well-known difference between admissible and derivable reasoning principles.

Moreover, we are able to relate conservativity to the completeness of a logic, thus adding another connection between proof and model theory.

**1998 ACM Subject Classification** F.4.1

**Keywords and phrases** conservative extension, logical framework, admissible, derivable, completeness

**Digital Object Identifier** 10.4230/LIPIcs...

## 1 Introduction

### Motivation

Conservativity is at the heart of mathematics and logics. Most abstractly, it means to extend a formal system in a way that effectively does not change it. Because this is what happens when adding definitions and theorems, it has received substantial attention in the area of formal logic.

Not surprisingly, different concrete definitions of the abstract intuition have been studied. Two notions have been used most widely, one based on proof theory and one based on model theory.[1] Due to the different philosophical backgrounds, it is not surprising that these notions are conceptually very different. This is not unusual: for example, the notion of *theorem* also has two very different definitions in proof and model theory. However, contrary to theorems, the two definitions of conservativity are not equivalent even in the presence of a sound and complete calculus. (The model theoretical one implies the proof theoretical one.) Consequently, this may lead to confusion and possibly even contention among researchers.[2]

* This work was supported by DFG under grant RA-18723-1.

[1] The author has traced the model-theoretical definition back to [7] but could not locate the first use of the (older) proof theoretical definition.

[2] In fact, this paper was motivated by the author's impression that this seems to be the case.

## Contribution

This paper contributes to the discussion by developing both definitions in the context of the author's MMT framework [13, 12]. MMT uses logical frameworks [10] as formal meta-logics in which to represent logics. In doing so, it integrates frameworks with a proof theoretical background such as LF [6] with model theoretical frameworks such as institutions [4]. This lets MMT elegantly represent both proof and model theoretical concepts uniformly [8, 1, 12]. Moreover, MMT is systematically designed to be logic-independent. Thus we can use it to give rigorous definitions of logical concepts (such as proofs and models or logic translations) in full generality. These properties make MMT well-suited to study the two notions of conservativity.

Our most important result is that we are able to relate the two notions of conservativity to the concepts of admissible and derivable rules. At the MMT-level, proof and model theoretical conservativity end up being special cases of admissibility and derivability, respectively. This provides an elegant understanding of both the similarities and the differences of the two competing definitions.

MMT also lets us elegantly capture the subtlety that the model theoretic definition actually constitutes a family of different definitions—one each for every model theory that is used. We naturally encounter a certain syntactic model theory that induces the strictest reasonable notion of conservativity, with every refinement of the model theory leading to a more lenient notion. Taking these refinements to the extreme, we find the proof theoretic definition as the most lenient reasonable notion.

Furthermore, we are able to cast completeness of a logic as a special case of admissibility as well, thus creating an appealing connection between conservativity and completeness.

## Overview

Sect. 2 recalls the existing definitions of proof and model theoretical conservativity, and we summarize the necessary preliminaries about MMT in Sect. 3. Sect. 4 develops our definitions and establishes their properties.

# 2 Existing Definitions of Conservativity

Conservativity can be defined for an arbitrary logic. To state the definitions in full generality, we can use a framework like institutions [4]. However, the precise abstract definition is not essential for our purposes. Therefore, we only assume a very lightweight definition to make the paper more accessible. A logic consists of

- a category of theories,
- a set of sentences $\mathbf{Sen}(\Sigma)$ for each theory $\Sigma$ and a sentence translation $v(-) : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ for every theory morphism $v : \Sigma \to \Sigma'$,
- a provability predicate giving the provable subset of $\mathbf{Sen}(\Sigma)$,
- a class of models $\mathbf{Mod}(\Sigma)$ for each theory $\Sigma$ and a model reduction function $\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ for every theory morphism $v : \Sigma \to \Sigma'$,
- a satisfaction relation between models in $\mathbf{Mod}(\Sigma)$ and sentences in $\mathbf{Sen}(\Sigma)$,

with some coherence conditions between them. Detailed definitions based on institutions are given in, e.g., in [3, 11].

Relative to such a logic, soundness and completeness can be defined in the usual way by relating the provable sentences to those that are satisfied by all models. Moreover, we can state the usual definitions of conservativity:


Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

▶ **Definition 1** (Proof-Theoretically Conservative). A morphism $v : \Sigma \to \Sigma'$ is conservative if every $\Sigma$-sentence $F$ is provable iff the $\Sigma'$-sentence $v(F)$ is provable.

For the special case of an extension $v : \Sigma \hookrightarrow \Sigma'$, this says that the larger theory does not prove any $\Sigma$-sentences that were not already provable in $\Sigma$.

▶ **Definition 2** (Model-Theoretically Conservative). A morphism $v : \Sigma \to \Sigma'$ is conservative if for every $\Sigma$-model $m$ there is a $\Sigma'$-model $m'$ that reduces to $m$ via $v$.

For the special case of an extension, this says that every model of the smaller theory can be extended to a model of the larger theory.

As indicated before, these definitions are not equivalent:

▶ **Theorem 3.** *If $v$ is conservative in the model-theoretical sense and the logic is sound and complete, then $v$ is conservative in the proof-theoretical sense.*

The converse is not true in general.

# 3 Logics in the MMT-Framework

This section is a summary of the basic definitions of logics in MMT as given in [12]. [12] uses an arbitrary logical framework that is itself defined in MMT. All our results in this paper generalize easily to this general case. However, to make this paper more accessible (and shorter), we use a single, fixed logical framework. Concretely, we use LF.

## 3.1 Logical Framework

### The Logical Framework LF

LF [6] is a dependent type theory using the following concepts and notations:

- a universe `type` of types
- dependent function types $\Pi_{x:A}\, B$, which are written $A \to B$ if $x$ does not occur free in $B$
- dependent function abstraction $\lambda_{x:A}\, t$,
- function application $f\, a$,
- $\beta$-reduction and $\eta$-conversion.

We omit the well-known typing rules.

### Theories

LF-theories $\Sigma$ are sets of declarations, which are one of the following

- type declarations $c : \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}\, \mathtt{type}$
- term declarations $c : A$ for some type $A$

Relative to a theory $\Sigma$, we have the set of all types $A$ and the set of all terms $t$. These are subject to the typing judgment $t : A$ and the equality judgments $t \equiv t'$ and $A \equiv A'$. (The latter equality judgment is needed because terms may appear in types.) Typing and equality of terms and types are decidable. We say that a type $A$ is **inhabited** if there is a term $t : A$.

▶ **Example 4** (Syntax of First-Order Logic). We define first-order logic FOL as the following LF theory *FOL*:

$$
\begin{array}{lll}
o & : & \texttt{type} \\
i & : & \texttt{type} \\
thm & : & o \to \texttt{type} \\
\neg & : & o \to o \\
\wedge & : & o \to o \to o \\
\Rightarrow & : & o \to o \to o
\end{array}
\qquad
\begin{array}{lll}
\doteq & : & i \to i \to o \\
\forall & : & (i \to o) \to o \\
\exists & : & (i \to o) \to o \\
mp & : & \Pi_{F:o}\,\Pi_{G:o}\, thm\,(F \Rightarrow G) \to thm\,F \to thm\,G \\
\vdots & &
\end{array}
$$

FOL-terms and sentences are represented as LF-terms over the theory *FOL* of type $i$ and $o$, respectively. We use currying to represent functions with multiple arguments as unary functions. We will use the usual infix notations where applicable, e.g. the term $(\wedge\, F)\, G$ represents the sentence $F \wedge G$. Binders are represented using higher-order abstract syntax: the term $\forall(\lambda x : i.F(x))$ represents the sentence $\forall x.F(x)$. In future examples, we will use the usual notations instead of the ones technically prescribed by our encoding in LF.

The type constructor *thm* serves as the truth judgment. Proof-theoretically, we use a Curry-Howard representation of proofs as terms, i.e., the terms $p : thm\, F$ are the proofs of $F$; model-theoretically, we will below treat $F$ as true if the type $thm\, F$ is inhabited.

We only give a single proof rule as an example: The modus ponens rule *mp* takes two formulas $F$ and $G$, a proof of $F \Rightarrow G$ and a proof of $F$ and returns a proof of $G$. All natural deduction rules of first-order logic can be written as LF declarations in this style.

### Theory Morphisms

LF-theory morphisms $\sigma : \Sigma \to \Sigma'$ are sets of assignments, one for each declaration in $\Sigma$:

- for every $\Sigma$-type declaration $c : \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}\, \texttt{type}$, a type assignment

  $$a \mapsto \lambda_{x_1:\sigma(A_1)} \ldots \lambda_{x_n:\sigma(A_n)}\, B$$

  for some $\Sigma'$-type $B$ with free variables $x_1, \ldots, x_n$.
- for every $\Sigma$-term declaration $c : A$, a term assignment $c \mapsto t$ for a $\Sigma'$-term $t : \sigma(A)$

where $\sigma(-)$ is the homomorphic extension of $\sigma$ defined below.

Every theory morphism $\sigma$ extends to a homomorphic translation $\sigma(-)$, which maps $\Sigma$-terms and types to $\Sigma'$-terms and types. $\sigma(-)$ preserves typing and equality, e.g., if $t : A$ holds over $\Sigma$, then $\sigma(t) : \sigma(A)$ holds over $\Sigma'$. If particular, if $A$ is inhabited over $\Sigma$, then $\sigma(A)$ is inhabited over $\Sigma'$.

▶ **Example 5** (Semantics of First-Order Logic)**.** We sketch a morphism *FOLZF* from *FOL* a theory *ZF* for axiomatic set theory. The intuition behind *FOLZF* is that it is the interpretation function that maps *FOL*-terms and *FOL*-formulas to their denotations.

*ZF* is an extension of *FOL* that declares the binary predicate $\in : i \to i \to o$ and adds the axioms of set theory. Besides the usual set theoretical operations, *ZF* defines in particular the 2-element set $bool : i$ of Booleans containing the elements $0 : i$ and $1 : i$. Moreover, we add a type constructor $Elem : i \to \texttt{type}$ such that terms of type $Elem\, A$ represent the elements of the set $A : i$. The complete definition of *ZF* can be found in [9].

We extend *ZF* with a theory $\Delta$ that axiomatizes a basic FOL-model: $\Delta$ contains the declarations $univ : i$ and $nonempty : thm\,(\exists x.x \in univ)$ which describe a non-empty set.

Then we define the semantics as the morphism $FOLZF : FOL \to ZF, \Delta$, which maps in particular

- $FOLZF(i) = Elem\, univ$, i.e., *univ* is an arbitrary non-empty set representing the universe of the model and terms are interpreted as elements of *univ*,
- $FOLZF(o) = Elem\, bool$, i.e., every formula is interpreted as a boolean truth value,

- $FOLZF(thm) = \lambda x : Elem\, bool.thm(x \doteq 1)$, i.e., $FOLZF(thm\, F)$ is inhabited iff $FOLZF(F)$ is provably equal to the truth value 1.

All connectives can now be mapped in the usual way. For example, $FOLZF(\wedge)$ is the binary conjunction of Booleans. All proof rules can be mapped as well—each assignment of a proof rule represents a case in the soundness proof.

Ultimately, the typing preservation of LF-morphism guarantees soundness: every $FOL$-proof $P : thm\, F$ gives rise to a $ZF$-proof $FOLZF(P) : thm\,(FOLZF(F) \doteq 1)$, i.e., the usual soundness theorem.

### Pushouts

LF theories and theory morphisms form a category. Moreover, this category has pushouts along inclusions, which we write as

$$\begin{tikzcd}
Syn, \Sigma \arrow[r, "sem^{\Sigma}"] & Sem, sem(\Sigma) \\
Syn \arrow[u, hook] \arrow[r, "sem"] & Sem \arrow[u, hook]
\end{tikzcd}$$

$sem(\Sigma)$ can be seen as the homomorphic translation of $\Sigma$: It contains the declaration $c : sem^{\Sigma}(A)$ for every declaration $c : A$ in $\Sigma$. Here $sem^{\Sigma}$ maps $Syn$-constants like $sem$, and it maps each $c : A$ in $\Sigma$ to the corresponding $c$ in $sem(\Sigma)$.

For a morphism $v : Syn, \Sigma \to Syn, \Sigma'$ that is the identity on $Syn$, we write $sem(v) : Sem, sem(\Sigma) \to Sem, sem(\Sigma')$ for the unique factorization through the pushout. Thus, $sem(-)$ is a functor from extensions of $Syn$ to extensions of $Sem$.

Technically, there are some subtleties here because the above construction of the pushout is not always well-defined—there is a problem if the same name is declared in both $Sem$ and $\Sigma$. This is discussed in [12] and not essential for the results in this paper.

## 3.2 Logics

In MMT, we can define logics easily by abstracting from the intuitions presented in Ex. 4 and 5:

▶ **Definition 6** (Logical Theories). A **logical theory** $Syn$ is an LF-theory with distinguished declarations $o : \texttt{type}$ and $thm : o \to \texttt{type}$.

Consider two logical theories $Syn$ (with $o$ and $thm$) and $Syn'$ (with $o'$ and $thm'$). A **logical morphism** is an LF-morphism $l : Syn \to Syn'$ such that $l(thm\, x) = thm'(k\, x)$ for some expression $k : l(o) \to o'$. ($k$ is uniquely determined if it exists.)

▶ **Definition 7** (Logic). A logic is a 4-tuple forming a logical morphism $sem : Syn \to Sem, \Delta$.

▶ **Example 8** (First-Order Logic). $FOL$ from Ex. 4 is a logical theory where $o$ and $thm$ are the distinguished declarations.

$FOLZF : FOL \to ZF, \Delta$ from Ex. 5 is a logical morphism with $k\, x = x \doteq 1$.

Every logic in the sense of Def. 7 induces a logic in the sense of Sect. 2. Here the intuitions behind $Syn$, $Sem$, $\Delta$, and $sem$ are as follows:

- The logical theory $Syn$ represents the syntax and proof theory: sentences are the terms of type $o$, and proofs are the terms of type $thm\, F$.


Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- The logical theory *Sem* represents the semantic foundation, e.g., an ambient set theory like *ZF*.
- $\Delta$ extends *Sem* with the axiomatization of a basic model. For FOL, $\Delta$ is very simple—the theory of a non-empty set. But $\Delta$ can be arbitrarily complex, e.g., the theory of a category for categorical models or the theory of a Kripke frame for Kripke models.
- The logical morphism *sem* describes the interpretation of the syntax and proofs in an arbitrary model.

In the remainder of this section, we make these intuitions precise for a fixed logic $sem : Syn \to Sem, \Delta$.

▶ **Definition 9** (Abstract Negation)**.** For a type $A$ in a logical theory, we abbreviate $\overline{A} := A \to \Pi_{F:o} F$.

A logical theory is *classical* if it has a term of type $classical : \Pi_{F:o} \overline{\overline{thm\, F}} \to thm\, F$.

The type $\overline{A}$ represents a negation of $A$ in the sense that if $A$ is inhabited, the theory is inconsistent because every formula is provable. Thus, classical logics are the one that have double-negation elimination.

▶ **Definition 10** (Syntax and Proofs)**.** A ***Syn*-theory** is an extension $Syn \hookrightarrow Syn, \Sigma$ of *Syn*.

A ***Syn*-theory morphism** is a morphism $\sigma : Syn, \Sigma \to Syn, \Sigma'$ satisfying $\sigma|_{Syn} = id_{Syn}$.

Consider a *Syn*-theory $\Sigma$:
- A **$\Sigma$-sentence** is a $\Sigma$-term $F : o$.
- A **$\Sigma$-proof** of $F$ is a $\Sigma$-term $p : thm\, F$.
- A **$\Sigma$-disproof** of $F$ is a $\Sigma$-term of type $\overline{thm\, F}$.
- $F$ is **(dis)provable** if there is a (dis)proof of $F$.

A *Syn*-theory morphism $\sigma : \Sigma \to \Sigma'$ translates sentences and proofs over $\Sigma$ to $\Sigma'$ by applying the homomorphic extension $\sigma(-)$.

▶ **Definition 11** (Semantics and Models)**.** Consider a *Syn*-theory $\Sigma$, and a $\Sigma$-sentence $F$. Then:
1. A **$\Sigma$-model** via *sem* is a *Sem*-theory morphism $m : Sem, \Delta, sem(\Sigma) \to Sem, M$ such that $m|_{Sem} = id_{Sem}$.
2. $F$ is **true** resp. **false** in such a model $m$ if the type $m(sem^{\Sigma}(thm\, F))$ resp. $\overline{m(sem^{\Sigma}(thm\, F))}$ is inhabited in the theory in $Sem, M$.

A *Syn*-theory morphism $\sigma : \Sigma \to \Sigma'$ reduces a model $m$ of $\Sigma'$ to the model $m \circ sem(\sigma)$ of $\Sigma$.

$$\begin{tikzcd}
Syn, \Sigma \arrow[r, "sem^{\Sigma}"] & Sem, \Delta, sem(\Sigma) \arrow[r, "m"] & Sem, M \\
Syn \arrow[u, hook] \arrow[r, "sem"] & Sem, \Delta \arrow[u, hook] & Sem \arrow[u, hook] \arrow[l, hook']
\end{tikzcd}$$

Note that a model $m$ must be the identity on *Sem* and interpret the declarations in $\Delta$ and in $sem(\Sigma)$ as values in *Sem*. That is the reason why we need two different theories for *Sem* and $\Delta$ when defining a logic.

▶ **Definition 12** (Consistency)**.** An *Syn*-theory $\Sigma$ is **consistent** if there is no $\Sigma$-sentence that is both provable and disprovable.

A logical morphism $sem : Syn \to Sem$ **preserves consistency** if $sem(\Sigma)$ is consistent whenever $\Sigma$ has at least one sentence and is consistent.

One of the main theorems of [12] is the following:

▶ **Theorem 13** (Soundness/Completeness). *Every logic* $sem : Syn \to Sem, \Delta$ *is sound in the sense that provable* $\Sigma$*-sentences are true in all* $\Sigma$*-models via* $sem$.

*Moreover, if* $Syn$ *is classical and* $sem$ *preserves consistency, the logic is also complete in the dual sense.*

# 4 Conservative Morphisms

We will now develop definitions of conservativity for arbitrary logics defined in the MMT framework. A key insight is to define general notions of derivability and admissibility first.

## 4.1 Derivable and Admissible Rules

*Derivable* and *admissible* are well-known concepts in the study of inference systems. Both mean that a rule can be added to an inference system without changing its essence. In MMT, we can give a very general precise definition:

▶ **Definition 14** (Admissible, Derivable). Consider a theory $Syn$.

For a set $J$ of $Syn$-types, a type $R$ is called $J$-**admissible** if every type in $J$ is inhabited over $Syn$ iff it is inhabited over $Syn, r : R$.

A type $R$ is called **derivable** if it is admissible for the set of all $Syn$-types.

For $J$-admissibility, the left-to-right implication always holds: If there is a $Syn$-term $t : A$, then $t$ is also a $Syn, r : R$-term. Only the right-to-left implication is special.

▶ **Theorem 15.** *A* $Syn$*-type* $R$ *is derivable iff there is a* $Syn$*-term* $P : R$.

*In particular, we have a morphism* $id_{Syn}, r \mapsto P$ *from* $Syn, r : R$ *to* $Syn$.

**Proof.** Assume there is a term $P : R$. We can always replace $r$ with $P$ in any term over $Syn, r : R$, thus proving $J$-admissibility.

Assume admissibility for every type $T$. We obtain $P$ by instantiating with $T = R$. ◀

Thus, if a rule (represented by the type $R$) is derivable, we have a derivation for it (represented by the term $P$), and adding it to the inference system (the declaration $r : R$) just adds an abbreviation (the name $r$) for a derivation that already existed. For example, let $Syn = FOL$ and $R = \Pi_{F,G,H}\, thm\,(F \Rightarrow G \Rightarrow H) \to thm\, F \to thm\, G \to thm\, H$. $R$ is a derivable rule: We can derive it applying modus ponens twice, and the term $P : R$ is given by $\lambda_{F,G,H}\, \lambda_{p,q,r}\, mp\, G\, H\, (mp\, F\, (G \Rightarrow H)\, p\, q)\, r$.

A $J$-admissible rule may create substantively new derivations as long as it does not create new $J$-terms. The most important special case arises when $Syn$ is a logical theory and $J$ is the set of all types of the form $thm\, F$: then $J$-admissibility of a rule $R$ means that no new formula $F$ becomes provable if $R$ is added to $Syn$.

Derivable rules are admissible but not vice versa. Admissible-but-not-derivable rules are of great importance in the meta-theory of logics. Examples are the deduction theorem in Hilbert calculi and the cut rule in sequent calculi.

Proofs of derivability are usually straightforward: we just have to exhibit the derivation $P$. Proofs of admissibility, however, are usually very involved, often requiring an induction over all $Syn$-terms. Moreover, derivability is the more robust notion: Extending $Syn$ in any way can never break the derivability of $R$ (because $P : R$ remains a well-formed term), but it can break admissibility (because it adds a new case that has to be considered in the inductive proof).


Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the sequel, we often need to talk about admissibility where the set $J$ a certain form. Therefore, we define:

▶ **Definition 16.** For a logical theory and a type constructor $thm : o \to \mathtt{type}$, we say *thm*-**admissible** if we mean admissible for the set of all types of the form $thm\,F$.

## 4.2 Derivable and Admissible Morphisms

The previous section formalized the existing concepts of derivable and admissible in MMT. Now we use the MMT formalism to generalize the definitions to arbitrary theory extensions.

▶ **Definition 17** (Admissible/Derivable Extensions). Consider a theory extension $Syn \hookrightarrow Syn'$.

It is called $J$-**admissible** if every type in $J$ is inhabited over $Syn$ iff it is inhabited over $Syn'$.

It is called **derivable** if it is admissible for the set of all $Syn$-types.

▶ **Theorem 18.** *An extension $Syn \hookrightarrow Syn'$ is derivable iff it has a retraction, i.e., if there is a morphism $P : Syn' \to Syn$ such that $P|_{Syn} = id_{Syn}$.*

**Proof.** Let $Syn' = Syn, \Sigma$.

Assume there is a morphism $P$. We can always replace any $\Sigma$-symbol $c$ with $P(c)$ in any $Syn'$-term, thus proving $J$-admissibility.

Assume admissibility for every type $T$. We build the morphism $P$ by induction on $\Sigma$. Assume we have $P : Syn, \Sigma_0 \to Syn$. For the next $\Sigma$-declaration $c : A$, we instantiate admissibility with $T = P(A)$ to obtain a $Syn$-term $p$. Then $P, c \mapsto p$ is a morphism $Syn, \Sigma_0, c : A \to Syn$. ◀

Theory extensions are the most important special case when studying conservativity. But it is easy to generalize the concepts to arbitrary morphisms. This has practical importance because it allows considering, e.g., renamings or isomorphisms in addition to extensions:

▶ **Definition 19** (Admissible/Derivable Morphisms). Consider a theory morphism $v : Syn \to Syn'$.

$v$ is called $J$-**admissible** if every type $A$ in $J$ is inhabited over $Syn$ iff $v(A)$ is inhabited over $Syn'$.

$v$ is called **derivable** if it has a retraction, i.e., if there is a morphism $P : Syn' \to Syn$ such that $P \circ v = id_{Syn}$.

▶ **Theorem 20.** *If $v : Syn \to Syn'$ is derivable, then it is admissible for all $Syn$-types.*

**Proof.** Applying the retraction of $v$ yields a $Syn$-term for every $Syn'$-term. ◀

For arbitrary morphisms, admissibility for all $Syn$-types is not the same anymore as having a retraction. The problem is that quantifying over all $Syn$-types $A$ is not strong enough to quantify over all $Syn'$-types because not every $Syn'$-type is of the form $v(A)$. Therefore, we have to choose one of the two notions as the appropriate generalization of derivability. In Def. 19, we choose the retraction property because it better captures the intuition that all $Syn'$-declarations can be derived in $Syn$.

Finally we establish some basic closure properties.

▶ **Theorem 21** (Closure Properties). *Derivable and admissible morphisms have the following closure properties:*


Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- *Identity*
  - *The identity morphism is derivable.*
  - *The identity morphism is $J$-admissible for any $J$.*
- *Composition*
  - *If $v$ and $w$ are derivable, then so is $w \circ v$.*
  - *If $v$ is $J$-admissible and $w$ is $v(J)$-admissible, then $w \circ v$ is $J$-admissible.*
- *Decomposition*
  - *If $w \circ v$ is derivable, then so is $v$.*
  - *If $w \circ v$ is $J$-admissible, then so is $v$.*

*Additionally, derivable morphisms have the following closure properties:*
- *Union: The union $\Sigma, \Sigma_0, \Sigma_1$ of derivable extensions $\Sigma, \Sigma_0$ and $\Sigma, \Sigma_1$ is derivable.*
- *Pushout: The pushout of a derivable morphism is derivable.*

**Proof.** All proofs are straightforward. ◀

The closure under pushouts formally captures the robustness of derivability: It is preserved under translations of the domain theory. Admissibility of morphisms, however, is brittle: It can be broken by relatively minor changes to the domain theory.

## 4.3 Conservative Morphisms

In this section, we fix a logic $sem : Syn \to Sem, \Delta$. We want to study the conservativity of a $Syn$ morphism $v : \Sigma \to \Sigma'$. The following diagram describes our basic situation.

$$\begin{tikzcd}
& Syn, \Sigma' \arrow[r, "sem^{\Sigma'}"] & Sem, \Delta, sem(\Sigma') & \\
& Syn, \Sigma \arrow[u, "v"] \arrow[r, "sem^{\Sigma}"] & Sem, \Delta, sem(\Sigma) \arrow[u, "sem(v)"'] & \\
Syn \arrow[uur, hook] \arrow[ur, hook] \arrow[rrr, "sem"] & & & Sem, \Delta \arrow[ul, hook] \arrow[uul, hook]
\end{tikzcd}$$

▶ **Definition 22** (Proof-Theoretically Conservative)**.** A $Syn$-morphism $v : \Sigma \to \Sigma'$ is called **proof-conservative** via $sem$ if $sem(v)$ is $sem^{\Sigma}(thm)$-admissible.

$v$ is simply called **proof-conservative** if it is proof-conservative via $id_{Syn}$.

Intuitively, $v$ is proof-conservative via $sem$ if semantic proofs (i.e. proofs carried out in the ambient foundation $Sem$ that talk about truth in an arbitrary model) exhibit the typical conservativity property when moving along $v$.

For every logical theory $Syn$, we have the trivial semantics $id_{Syn}$. The special case of being proof-conservative via $id_{Syn}$ immediately yields the usual proof-theoretical notion from Def. 1:

▶ **Theorem 23.** *$v : \Sigma \to \Sigma'$ is proof-conservative iff the following holds: every $\Sigma$-sentence $F$ is $\Sigma$-provable iff $v(F)$ is $\Sigma'$-provable.*

*In particular, an extension $\Sigma \hookrightarrow \Sigma'$ is proof-conservative if every $\Sigma$-sentence is $\Sigma$-provable iff it is $\Sigma'$-provable.*

**Proof.** This follows easily because if $sem = id_{Syn}$, then $sem(-)$ is the identity, and in particular $sem(v) = v$. ◀

▶ **Definition 24** (Model-Theoretically Conservative). A $Syn$-morphism $v : \Sigma \to \Sigma'$ is **model-conservative** via $sem$ if $sem(v)$ is derivable, i.e., if there is a retraction $r$ as in the commutative diagram below.

$$\begin{tikzcd}
Syn, \Sigma' \arrow[r, "sem^{\Sigma'}"] & Sem, \Delta, sem(\Sigma') \arrow[r, "r"] & Sem, \Delta, sem(\Sigma) \\
Syn, \Sigma \arrow[u, "v"] \arrow[r, "sem^{\Sigma}"] & Sem, \Delta, sem(\Sigma) \arrow[u, "sem(v)"] \arrow[ur, "id"] &
\end{tikzcd}$$

$v$ is simply called **model-conservative** if it is model-conservative via $id_{Syn}$, i.e., if $v$ has a retraction.

The question whether our notion of model-conservativity is equivalent to the usual one from Def. 2, is tricky. We establish one direction first:

▶ **Theorem 25.** *If $v : \Sigma \to \Sigma'$ is model-conservative via $sem$, then every $\Sigma$-model $m$ via $sem$ can be expanded to a $\Sigma'$-model $m'$ via $sem$ that reduces to $m$.*

**Proof.** We put $m' = m \circ r$. The reduct of $m'$ via $v$ is $m' \circ sem(v)$, which is equal to $m$ because $r \circ sem(v) = id$.

$$\begin{tikzcd}
Syn, \Sigma' \arrow[r, "sem^{\Sigma'}"] & Sem, \Delta, sem(\Sigma') \arrow[r, "r"] & Sem, \Delta, sem(\Sigma) \arrow[d, "m"] \\
Syn, \Sigma \arrow[u, "v"] \arrow[r, "sem^{\Sigma}"] & Sem, \Delta, sem(\Sigma) \arrow[u, "sem(v)"] \arrow[ur, "id"] & Sem, M
\end{tikzcd}$$

◀

The other direction of the equivalence depends on subtle properties of the theory $Sem$. For example, consider the case where $sem$ and $Sem$ formalize the usual set-theoretical semantics in some ambient set theory. If we formalize Def. 2, we obtain something like the $Sem$-sentence $U$ given by

$$\forall m \in \mathbf{Mod}(\Sigma).\, \exists m' \in \mathbf{Mod}(\Sigma').\, reduct(v, m') = m$$

Our definition, on the other hand, is equivalent to exhibiting a function $f$ such that

$$\forall m \in \mathbf{Mod}(\Sigma).\, f(m) \in \mathbf{Mod}(\Sigma') \wedge reduct(v, f(m)) = m$$

This is subtly stronger because it requires actually giving $f$, which in particular requires $f$ to be definable as a $Sem$-expression.

For example, consider the most direct formalization of set theory using a $FOL$-theory $ZF$ with a single predicate symbol $\in$. This $FOL$-theory has no function symbols at all and therefore cannot give any function $f$ even if it can prove $U$ and has the axiom of choice. However, assume a variant of $ZF$ that has a choice operator $\varepsilon : \Pi_{F:i \to \mathtt{prop}} \to (thm\, \exists x.F(x)) \to i$, which chooses some element that satisfies $F$ provided that such an element exists. Then we can define $f$ as the LF-expression

$$\lambda_m \,.\, \varepsilon \left(\lambda_{m'}\ m' \in \mathbf{Mod}(\Sigma') \wedge reduct(v, m') = m\right) \left(\forall_E\, P\, m\right)$$

where $\forall_E$ is the elimination rule that instantiates $P : \mathit{thm}\, U$ with $m$ to show the existence of the needed $m'$.

Note that the axiom of choice would not be sufficient here. It would only allow proving the existence of $f$ but not choose a term for it.

Such choice operators are relatively strong features of axiomatic set theories. However, in practical foundations of mathematics that are used in proof assistants, they are very common. Examples include higher-order logic [5] and the set theory underlying Mizar [15]. For constructive foundations such as the calculus of constructions underlying Coq [2], the choice operator is trivial because the only way to prove $U$ in the first place is to exhibit $f$.

We summarize the above analysis in the following theorem:

▶ **Theorem 26.** *Assume that Sem adequately formalizes the ambient foundation of mathematics that is implicitly used in Def. 2.*

*Moreover, assume that whenever Sem can prove a statement of the form "for all $m$ exists $m'$ such that $F(m, m')$", it can also define an LF-function $f$ such that $F(m, f(m))$.*

*Then $v : \Sigma \to \Sigma'$ is model-conservative via sem iff it is conservative in sense of Def. 2.*

**Proof.** The left-to-right direction is proved by Thm. 25. For the right-to-left direction, assume that for every $\Sigma$-model $m$ there is a $\Sigma'$-model $m'$ that reduces to $m$. Using the assumptions about *Sem*, that yields a function $f$ that maps $\Sigma$-models to $\Sigma'$-models.

We construct the needed retraction $r$ of $v$ as follows. First we package the declarations in $\Delta, \mathit{sem}(\Sigma)$ into a *Sem*-term $m$. The details of this packaging depend on how *Sem* defines models. For example, if *Sem* defines models using record types, the packaging just constructs a record.

Second, $r$ maps every symbol $c$ of $\mathit{sem}(\Sigma')$ to the term that selects the component $c$ from $f(m)$. Again the details of this selection depend on *Sem*. For example, if *Sem* defines models using record types, the selection is just the projection of the field $c$. ◀

## 4.4 Relating the Notions of Conservativity

For a logic $\mathit{sem} : \mathit{Syn} \to \mathit{Sem}, \Delta$, Def. 22 and 24 yield four different notions of conservativity. We fix a *Syn*-morphism $v : \Sigma \to \Sigma'$ and abbreviate as follows:

(PS) $v$ is proof-conservative via *sem*.
(P) $v$ is proof-conservative.
(MS) $v$ is model-conservative via *sem*.
(M) $v$ model-conservative.

By instantiating Thm. 21, we immediately obtain several closure properties for all four notions.

Recall that (M) means that $v$ has a retraction, and that (P) and (MS) correspond to the notions of conservativity from Def. 1 and 2

The following theorem shows how the four properties relate to each other:

▶ **Theorem 27.** *Let* C *be the assumption that Syn is classical and that sem preserves consistency. Then we have the following graph of implications where $A \xrightarrow{L} B$ means that $L$ and $A$ imply $B$:*

$$\begin{tikzcd}
\mathrm{M} \arrow[r] \arrow[d] & \mathrm{MS} \arrow[d] \\
\mathrm{P} \arrow[r, leftrightarrow, "\mathrm{C}"] & \mathrm{PS}
\end{tikzcd}$$

**Proof.** (MS) implies (PS): This is a special case of Thm. 18.

(M) implies (P): This is the special case of (MS) implies (PS) for $sem = id_{Syn}$.

(M) implies (MS): (M) yields a retraction $r : \Sigma' \to \Sigma$. We obtain the retraction $r^+$ that establishes (MS) as $sem(r)$.

$$\begin{tikzcd}
Syn, \Sigma & Syn, \Sigma' \arrow[l, "r"'] \arrow[r, "sem^{\Sigma'}"] & Sem, \Delta, sem(\Sigma') \arrow[r, "r^+"] & Sem, \Delta, sem(\Sigma) \\
& Syn, \Sigma \arrow[ul, "id"'] \arrow[u, "v"] \arrow[r, "sem^{\Sigma}"] & Sem, \Delta, sem(\Sigma) \arrow[u, "sem(v)"] \arrow[ur, "id"] &
\end{tikzcd}$$

(PS) implies (P) if (C): Consider a $\Sigma$-sentence $F$ such that there is a $\Sigma'$-proof $P' : v(thm\, F)$. We need to exhibit a $\Sigma$-proof $P : thm\, F$.

First, applying (PS) to the term $sem^{\Sigma'}(P')$, whose type is

$$sem^{\Sigma'}(v(thm\, F)) = sem(v)(sem^{\Sigma}(thm\, F)),$$

yields a $sem(\Sigma)$-term $Q$ of type $sem^{\Sigma}(thm\, F)$.

The remainder of the proof uses (C) to obtain $P$ from $Q$. If $\Sigma$ is inconsistent, $P$ exists trivially. So assume it is consistent. Consider $\Sigma^* = \Sigma, a : \overline{thm\, F}$. If $\Sigma^*$ is inconsistent, we obtain a $\Sigma$-term of type $\overline{\overline{thm\, F}}$ and classicality of $Syn$ yields the needed term $P$.

We conclude the proof by showing that $\Sigma^*$ is indeed inconsistent. Because $sem$ preserves consistency, it suffices to show that $Sem, \Delta, sem(\Sigma^*)$ is inconsistent. That follows from the terms $a : sem^{\Sigma}(\overline{thm\, F})$ and $Q$.

(P) implies (PS) if (C): Consider a $\Sigma$-sentence $F$ such that there is a $sem(\Sigma')$-proof $Q' : sem(v)(sem^{\Sigma}(thm\, F))$. We need to exhibit a $sem(\Sigma)$-proof $Q : sem^{\Sigma}(thm\, F)$.

If $\Sigma$ is inconsistent, this is trivial. So assume it is consistent. Then (P) implies that $\Sigma'$ is consistent, too. Now as in the case (PS) implies (P), we use the consistency of $\Sigma'$ and (C) to obtain from $Q'$ a $\Sigma'$-term $P' : v(thm\, F)$. Then (P) yields a $\Sigma$-term $P : thm\, F$, and we can put $Q = sem^{\Sigma}(P)$. ◀

(M) captures the situation where the syntax of the logic itself can express the proof of conservativity: as a theory morphism that retracts $v$. Therefore, if $v$ satisfies (M), $v$ satisfies any other reasonable definition of conservativity, i.e., (M) is the minimal/strongest reasonable definition. In particular, the retraction of $v$ yields both a proof transformation from $\Sigma'$ to $\Sigma$, which shows that (M) implies (P), and a model reduction from $\Sigma$ to $\Sigma'$, which helps showing that (M) implies (MS).[3]

One might think that (M) is too strong in practice. But it is actually very common as we see in Ex. 28 below.

(P) can be seen as a dual to (M)—not satisfying (P) captures the situation where the syntax of the logic itself can express a counter-example to conservativity: as a $\Sigma$-sentence that is a $\Sigma'$-theorem but not a $\Sigma$-theorem. Therefore, if $v$ satisfies any reasonable definition of conservativity, it should satisfy (P), i.e., (P) is the maximal/weakest reasonable choice.

(MS) sits in between (M) and (P). It captures the situation where the semantics (but not necessarily the syntax) of the logic can express the proof of conservativity: as a model transformation that expands $\Sigma$-models to $\Sigma'$-models. Because the semantics is usually more expressive than the syntax, it is not surprising that (MS) is weaker than (M). Moreover,

[3] The corresponding observation for the framework of institutions was previously made in [14].

because there may be multiple different ways to give the semantics of a logic, it is not surprising that (MS) is less canonical than (M) or (P), i.e., that (MS) depends on the choice of the model theory.

▶ **Example 28.** (M) subsumes a wide variety of important morphisms including:

- Isomorphisms.
- Extension with a definable constant. If we add a constant $c : A$ as an abbreviation for some $\Sigma$-term $t : A$, then $\Sigma, c : A$ is derivable using the morphism that maps $c$ to $t$.
- Extension with a provable theorem. Adding an axiom $a : \mathit{thm}\,F$ if $F$ has a proof $P$ is just a special case of the previous case.
- Extension with a new type. Let *Syn* have a declaration $tp : \mathtt{type}$ such that terms $A : tp$ represent types of the logic. Adding a new type $c : tp$ is almost always derivable because we just have to map $c$ to some existing $\Sigma$-type. The only exception is when $\Sigma$ is the empty theory of a logic without any built-in types. In that case, (P) and (MS) hold but not (M).
- Extension with a new predicate symbol $p : A_1 \to \ldots \to A_n \to o$. This is essentially always derivable because we can map $p$ to $\lambda_{x_1,\ldots,x_n} F$ for some formula $F$. The only exception is contrived, namely when $\Sigma$ has no sentences, in which case (P) and (MS) hold but not (M).
- Extension with a new function symbol $f : A_1 \to \ldots \to A_n \to A$. This is derivable whenever $\Sigma$ has a term of type $A$ in context $x_1 : A_1, \ldots, x_n : A_n$, which is often the case. If there is no such term, (M) fails; (P) and (MS) still hold unless *Syn* allows empty types.
- Compositions, unions, and pushouts of morphisms that satisfy (M).

[12] describes how logic translations and semantics become formally the same thing in MMT. For example, we can have logical morphisms $Syn \xrightarrow{t} Syn' \xrightarrow{s} Sem \xrightarrow{r} Sem'$ representing a logic translation $t$, a semantics $s$, and a refinement $r$ of the semantic foundation. Note that if $r \circ s \circ t$ preserves consistency, so do $s \circ t$ and $t$. Let us write (M), (Mt), (Mts), (Mtsr) for model-conservativity via $id_{Syn}$, $t$, $s \circ t$, and $r \circ s \circ t$, respectively. Then we have (M) implies (Mt) implies (Mts) implies (Mtsr) implies (P).

Thus, the more we refine *Syn*, the weaker model-conservativity becomes. Eventually, if we refine further and further, model-conservative and proof-conservative may eventually coincide.[4] Thus, we can think of (P) as an extreme case of (MS).

## 4.5 Conservativity and Completeness

Via admissibility, we can unify the concepts of conservative morphism and complete semantics:

▶ **Theorem 29.** *The logic $sem : Syn \to Sem, \Delta$ is complete iff all $sem^\Sigma$ are thm-admissible.*

**Proof.** [12] already proves that a sentence $F$ holds in all $\Sigma$-models iff $Sem, \Delta, sem(\Sigma)$ has a term of type $sem^\Sigma(\mathit{thm}\,F)$. Due to admissibility, such a term exists iff $\Sigma$ has a term of type $\mathit{thm}\,F$. ◀

Of course, a logic is also complete if the morphisms are derivable. However, because *Sem* is usually stronger than *Syn*, they are virtually never derivable in practice.

[4] This happens, e.g., if we use maximal consistent sets of sentences as the models.

One might hope for a stronger theorem where completeness already holds whenever *sem* is admissible. For that, we have to ask if the admissibility of *sem* implies the admissibility of $sem^\Sigma$. This is not always the case, and we develop a sufficient criterion now:

▶ **Definition 30.** We say that $Syn$ can abstract over the declaration $c : A$ if for every for $\Sigma, c : A$-sentence $F$, there is a $\Sigma$-sentence $\forall_{c:A} F$ such that $\forall_{c:A} F$ is $\Sigma$-provable iff $F$ is $\Sigma, c : A$-provable.

We write $\forall_\Gamma F$ when we iterate this construction for all declarations in $\Gamma$ that occur in a $\Sigma, \Gamma$-sentence $F$.

We speak of abstracting over theories when we can abstract over every declaration that is allowed in a theory.

The intuition behind $\forall_\Gamma F$ is to universally quantify over the declarations in $\Gamma$. Thus, abstracting over theories means that $Syn$ has universal quantification over all concepts that may be declared in theories. Note that most logics can quantify over axioms by using implication, e.g., in $FOL$ we can put $\forall_{a:thm\, G} F := G \Rightarrow F$.

▶ **Example 31.** $FOL$-theories may declare function and predicate symbols. But $FOL$ can only universally quantify over variables. Therefore, it cannot abstract over theories.

Higher-order logic (HOL) with a single base type can declare typed constants. Because HOL can quantify over variables of all types, it can abstract over theories. However, the variant of HOL that allows theories to introduce additional base types cannot abstract over theories because HOL cannot quantify over type variables. For the same reason typed FOL cannot abstract over theories.

Type theories with universe hierarchies (such as the calculus of constructions) can usually quantify over all types. Therefore, they can abstract over theories.

First-order set theory allows its theories to declare sets and elements of sets. It can quantify over both and thus over theories.

Languages that allow axiom schemata, e.g., polymorphic axioms in HOL, usually cannot abstract over them.

▶ **Theorem 32.** *Assume a logic where sem is thm-admissible.*

*If $Syn$ can abstract over $\Sigma$, then $sem^\Sigma$ is thm-admissible. In particular, the logic is complete if $Syn$ can abstract over all theories.*

**Proof.** The proofs are straightforward. ◀

The requirement that $Syn$ can abstract over theories is needed because admissibility of *sem* is a very weak notion: it talks only about sentences over the empty $Syn$-theory. Abstracting over theories makes sure that every relevant statement can be coded as a sentence over the empty theory. As a counter-example, consider FOL without equality and without constants for truth and falsity: then the empty theory happens to have no sentences at all so that any morphism out of $Syn$ is already proof-conservative.

## 5 Conclusion

We investigated the various notions of conservativity in the MMT framework. We were able to recover the existing notions of model-theoretical (MC) and proof-theoretical (PC) conservativity as special cases of admissibility and derivability.

We saw that these two notions should always be discussed together with a third one, the retractability of an extension (R). Using MMT, it naturally emerged that (R) and (PC) are the strongest and weakest extremes, whereas (MC) sits anywhere in between depending on how far the model theory refines the syntax. Moreover, (R) arises as the special case where the initial model theory is used (where the models are theory morphisms), and (P) arises as the special cases where a maximally refined model theory is used (e.g., where the models are maximal consistent sets of sentences). This harmonically resolves the tension between the two competing notions of conservativity.

In a second result, we showed how the conservativity of a model theory (seen as a translation of the syntax) corresponds to the completeness of the logic.

## References


1 M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, F. Rabe, and K. Sojakova. Towards Logical Frameworks in the Heterogeneous Tool Set Hets. In T. Mossakowski and H. Kreowski, editors, *Recent Trends in Algebraic Development Techniques 2010*, pages 139–159. Springer, 2012.

2 Coq Development Team. The Coq Proof Assistant: Reference Manual. Technical report, INRIA, 2015.

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

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

5 M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

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

7 W. Hodges. *Model Theory*. Cambridge University Press, 1993.

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

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

10 F. Pfenning. Logical frameworks. In J. Robinson and A. Voronkov, editors, *Handbook of automated reasoning*, pages 1063–1147. Elsevier, 2001.

11 F. Rabe. A Logical Framework Combining Model and Proof Theory. *Mathematical Structures in Computer Science*, 23(5):945–1001, 2013.

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

13 F. Rabe and M. Kohlhase. A Scalable Module System. *Information and Computation*, 230(1):1–54, 2013.

14 L. Schröder, T. Mossakowski, and C. Lüth. Type Class Polymorphism in an Institutional Framework. In J. Fiadeiro, P. Mosses, and F. Orejas, editors, *Recent Trends in Algebraic Development Techniques*, pages 234–251. Springer, 2004.

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