The MMT Perspective on Conservativity

Florian Rabe

Jacobs University Bremen

Abstract. Conservative extensions are one of the most important concepts in formal logic, capturing the intuition that 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, and we see that it 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.

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 does not effectively 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. Due to the different philosophical backgrounds, it is not surprising that these notions are conceptually 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.[1]

Contribution This paper contributes to the discussion by developing both definitions in the context of the author's MMT framework [RK13,Rab14]. MMT uses logical frameworks [Pfe01] as formal meta-logics in which to represent logics. It combines logical frameworks, which have a proof theoretical background, with institutions [GB92], which have a model theoretical background. This lets

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

MMT elegantly represent both proof and model theoretical concepts uniformly [HR11,CHK$^+$12,Rab14]. Moreover, MMT is systematically designed to be logic-independent. Thus we can use MMT 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.

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 recall 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 their properties.

2 Existing Definitions of Conservativity

Sect. 3 has introduced the basic concepts of an arbitrary logic. If we abstract away the concrete underpinning of LF, we obtain a very general definition of logic. The precise abstract definition is not essential for our purposes as long as we have the following:

- a category of theories,
- a set of sentences $\mathbf{Sen}(\Sigma)$ for each theory Σ 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 Σ 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 [Dia06,Rab13].

Relative to such a logic, soundness and completeness can be defined in the usual way. Moreover, we can state the usual definitions of conservativity:

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

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

Theorem 1. *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 [Rab14]. However, MMT uses an arbitrary logical framework. While all our results in this paper generalize easily to arbitrary frameworks in the MMT sense, it is more instructive (and necessary for brevity) to use only a single framework. Therefore, we will use LF [HHP93].

3.1 Logical Framework

The Logical Framework LF LF is a dependent type theory using the following concepts and notations:

- a universe $\mathtt{type}$ 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 $f\, a$,
- β-reduction and η-conversion.

Theories LF-theories Σ are sets of declarations, which are

- 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 Σ, we have the set of all types A and terms t, which are subject to the typing judgment $t : A$. Typing and equality of terms and types are decidable. We say that a type A is inhabited if there is a term $t : A$. In particular, a formula F is provable if $\mathit{thm}\, F$ is inhabited.

Example 1 (Syntax of First-Order Logic). We define first-order logic *FOL* as the following LF theory:

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

This uses currying to represent functions with multiple arguments, and we will use the usual infix notation where applicable. For example, the expression $(\wedge\, F)\, G$ represents the sentence $F \wedge G$. Binders are represented using higher-order abstract syntax: the expression $\forall(\lambda x : i.F(x))$ 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.

We only give a single proof rule of FOL as an example: The modus ponens rule 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 FOL can be written in this style.

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

- type assignments $c \mapsto \lambda_{x_1:\sigma(A_1)} \ldots \lambda_{x_n:\sigma(A_n)} B$ for a Σ-constant $c : \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}\, \mathtt{type}$ and a Σ'-type B
- term assignments $c \mapsto t$ for a Σ-constant $c : A$ and a Σ'-term $t : \sigma(A)$

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

Example 2 (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 expressions to their denotation.

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. Moreover, we add a type constructor $Elem : i \to \mathtt{type}$ such that terms of type $Elem\, A$ represent the elements of the set $A : i$. The complete definition of *ZF* can be found in [IR11].

We extend *ZF* with a theory Δ that axiomatizes a basic FOL-model: Δ 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. Moreover, all proof rules can be mapped as well—each map of a proof rule represents a case in the soundness proof. Ultimately, the typing preservation of LF-morphism guarantees that 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 Σ: It contains the declaration $c : sem^{\Sigma}(A)$ for every declaration $c : A$ in Σ. sem^{Σ} maps Syn-constants like sem, and it maps each $c : A$ in Σ 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 regarding the functoriality laws, which are discussed in [Rab14].)

3.2 Logics

Definition 3 (Logical Theories). *A **logical theory** Syn is an LF-theory with distinguished constants $o : \texttt{type}$ 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 4 (Logic). *A logic is a 4-tuple forming a logical morphism $sem : Syn \to Sem, \Delta$.*

Example 3 (First-Order Logic). FOL from Ex. 1 is a logical theory where o and $thm\,x$ are the distinguished constants.

$FOLZF : FOL \to ZF, \Delta$ from Ex. 2 is a logical morphism with $k\,x = x \doteq 1$.

Every logic in the sense of Def. 4 induces a logic in the sense of Sect. 2. Here the intuitions behind Syn, Sem, Δ, 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 $thm\,F$.

- The logical theory *Sem* represents the semantic foundation, e.g., an ambient set theory like *ZF*.
- $\mathit{\Delta}$ extends *Sem* with the axiomatization of a basic model. For FOL, $\mathit{\Delta}$ is very simple—the theory of a non-empty set. But $\mathit{\Delta}$ 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 $\mathit{sem} : \mathit{Syn} \to \mathit{Sem}, \mathit{\Delta}$.

Definition 5 (Abstract Negation). *For a type A in a logical theory, we abbreviate $\overline{A} := A \to \mathit{\Pi}_{F:o} F$.*

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

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 6 (Syntax and Proofs). *A Syn-**theory** is an extension $\mathit{Syn} \hookrightarrow \mathit{Syn}, \mathit{\Sigma}$ of Syn.*

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

Consider a Syn-theory $\mathit{\Sigma}$:

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

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

Definition 7 (Semantics and Models). *Consider a Syn-theory $\mathit{\Sigma}$, and a $\mathit{\Sigma}$-sentence F. Then:*

1. *A $\mathit{\Sigma}$-**model** via sem is a Sem-theory morphism $m : \mathit{Sem}, \mathit{\Delta}, \mathit{sem}(\mathit{\Sigma}) \to \mathit{Sem}, M$ such that $m|_{\mathit{Sem}} = \mathit{id}_{\mathit{Sem}}$.*
2. *F is **true (false)** in such an m if there is a Sem, M-term of type $m(\mathit{sem}^{\mathit{\Sigma}}(\mathit{thm}\, F))$ (or of type $\overline{m(\mathit{sem}^{\mathit{\Sigma}}(\mathit{thm}\, F))}$, respectively).*

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

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

Note that a model m must be the identity on Sem and interpret the declarations in Δ and in $sem(\Sigma)$ as values in Sem. That is the reason why Δ must be separated from Sem.

Definition 8 (Consistency). *An Syn-theory Σ is* ***consistent*** *if there is no Σ-sentence that is both provable and disprovable.*

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

Theorem 2 (Soundness/Completeness). *Every logic $sem : Syn \to Sem, \Delta$ is sound in the sense that provable Σ-sentences are true in all models via sem. 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 9 (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 3. *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 (the type R) is derivable, we have a derivation for it (the term P), and adding it to the inference system (the declaration $r : R$) just adds an abbreviation (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. The term P 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. For example, if Syn is a logical theory, J is usually the set of all types $thm\,F$. Then admissibility of R means that no new formula F becomes provable if R is added.

Derivable rules 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).

In this paper, Syn is always a logical theory, and we define an abbreviation:

Definition 10. *For a logical theory with distinguished constant thm, we say thm-**admissible** if we mean 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 11 (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 4. *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 Σ-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 Σ. Assume we have $P : Syn, \Sigma_0 \to Syn$. For the next Σ-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. 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 12 (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 5. *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. 12, 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 6 (Closure Properties). *Derivable and admissible morphisms have the following closure properties:*

- *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 Σ, Σ_0 and Σ, Σ_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.

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

Definition 13 (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}.*

The special case of being proof-conservative via id_{Syn} immediately yields the usual proof-theoretical notion from Def. 1:

Theorem 7. *$v : \Sigma \to \Sigma'$ is proof-conservative iff every Σ-sentence F is Σ-provable iff $v(F)$ is Σ'-provable.*

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

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

Definition 14 (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.*

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

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 8. *If $v : \Sigma \to \Sigma'$ is model-conservative via sem, then every Σ-model m via sem can be expanded to a Σ'-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 $r \circ sem(v) = id$.

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

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. However, assume a variant of *ZF* that has a choice operator $\varepsilon : \Pi_{F:i\to\mathtt{prop}} \to (\mathit{thm}\, \exists x.F(x)) \to i$, which chooses some element that satisfies F provided that such an element exists. Then we can define f as the LF-expression

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

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 [GM93] and the set theory underlying Mizar [TB85]. For constructive foundations such as the calculus of constructions underlying Coq [Coq15], 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 9. *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. 8. For the right-to-left direction, assume that for every Σ-model m there is a Σ'-model m' that reduces to m. Using the assumptions about *Sem*, that yields a function f that maps Σ-models to Σ'-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. 13 and 14 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. 6, we immediately obtain several closure properties for all four notions. The following theorem shows how the four properties relate to each other:

Theorem 10. *Let (C) be the assumption that Sem 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*:*

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

Proof. (MS) implies (PS): This is a special case of Thm. 4.

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

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

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

(PS) implies (P) if (C): Consider a Σ-sentence F such that there is a Σ'-proof $P' : v(\mathit{thm}\, F)$. We need to exhibit a Σ-proof $P : \mathit{thm}\, F$. If Σ is inconsistent, this is trivial. So assume it is consistent. Now consider $\Sigma^* = \Sigma, a : \overline{\mathit{thm}\, F}$. If Σ^* is inconsistent, we obtain a Σ-term of type $\overline{\overline{\mathit{thm}\, F}}$ and classicality of *Sem* yields the needed term P.

We conclude the proof by showing indirectly that Σ^* is inconsistent. Assume it is consistent. Then preservation of consistency yields that $\Theta = \mathit{Sem}, \Delta, \mathit{sem}(\Sigma^*)$ is consistent, too. Θ has a constant $a : \mathit{sem}^{\Sigma}(\overline{\mathit{thm}\, F})$. But applying (PS) to the term $\mathit{sem}^{\Sigma'}(P')$, whose type is $\mathit{sem}^{\Sigma'}(v(\mathit{thm}\, F)) = \mathit{sem}(v)(\mathit{sem}^{\Sigma}(\mathit{thm}\, F))$, yields a Θ-term of type $\mathit{sem}^{\Sigma}(\mathit{thm}\, F)$. Together with a, this term yields an inconsistency in Θ, which yields the needed contradiction.

Because (P) and (MS) are the notions from Def. 1 and 2, one might think that (M) is not very important. The fact that (M) is stronger than both (P) and (MS) also hints at that. However, (M) is very common in practice:

Example 4. (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 Σ-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 $\mathit{tp} : \mathtt{type}$ such that terms $A : \mathit{tp}$ represent types of the logic. Adding a new type $c : \mathit{tp}$ is almost always derivable because we just have to map c to some existing Σ-type. The only exception is when Σ 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 Σ 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 Σ 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).

4.5 Conservativity and Completeness

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

Theorem 11. *The logic $\mathit{sem} : \mathit{Syn} \to \mathit{Sem}, \Delta$ is complete iff all sem^Σ are thm-admissible.*

Proof. [Rab14] already proves that a sentence F holds in all Σ-models iff $\mathit{Sem}, \Delta, \mathit{sem}(\Sigma)$ has a term of type $\mathit{sem}^\Sigma(\mathit{thm}\, F)$. Due to admissibility, such a term exists iff Σ has a term of type $\mathit{thm}\, F$.

Of course, a logic is also complete if the morphism are derivable. However, because *Sem* is usually stronger than *Syn*, they are practically never derivable.

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

Definition 15. *We say that Syn can abstract over the declaration $c : A$ if for every for $\Sigma, c : A$-sentence F, there is a Σ-sentence $\forall_{c:A} F$ such that $\forall_{c:A} F$ is Σ-provable iff F is $\Sigma, c : A$-provable.*

We write $\forall_\Gamma F$ when we iterate this construction for all declarations in Γ that occur in a Σ, Γ-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 Γ. 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 5. 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 Thus, 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 12. *Assume a logic where sem is thm-admissible.*

If Syn can abstract over Σ, 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 necessary because admissibility is such a 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 or constants for truth and falsity: then the empty theory happens to have no sentences at all so that any morphism out of *Syn* is proof-conservative.

References

CHK$^+$12. 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.

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

Dia06. R. Diaconescu. Proof systems for institutional logic. *Journal of Logic and Computation*, 16(3):339–357, 2006.

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

GM93. M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

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

HR11. F. Horozal and F. Rabe. Representing Model Theory in a Type-Theoretical Logical Framework. *Theoretical Computer Science*, 412(37):4919–4945, 2011.

IR11. M. Iancu and F. Rabe. Formalizing Foundations of Mathematics. *Mathematical Structures in Computer Science*, 21(4):883–911, 2011.

Pfe01. F. Pfenning. Logical frameworks. In J. Robinson and A. Voronkov, editors, *Handbook of automated reasoning*, pages 1063–1147. Elsevier, 2001.

Rab13. F. Rabe. A Logical Framework Combining Model and Proof Theory. *Mathematical Structures in Computer Science*, 23(5):945–1001, 2013.

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

RK13. F. Rabe and M. Kohlhase. A Scalable Module System. *Information and Computation*, 230(1):1–54, 2013.

TB85. 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.