A Logical Framework Perspective on Conservativity

Florian Rabe*[0000−0003−3040−3655]

Computer Science, University Erlangen-Nuremberg
florian.rabe@gmail.com

Abstract. Conservative extension is one of the most important concepts in formal logic, capturing the intuition when an extension does not substantially change the extended language or theory. Multiple non-equivalent definitions have emerged, including conceptually very different ones in proof and model theory.
We use a logical framework that allows stating these notions in a logic-independent way. This allows proving several meta-theorems that yield new intuitions about conservativity: The existence of the different notions of conservativity is neither a coincidence nor a defect: we recover them as canonical points on a spectrum of gradual refinement from syntax to semantics. Moreover, the model and proof-theoretical notions correspond to the well-known difference between admissible and derivable rules. Finally, we can formally capture that the completeness of a logic corresponds to the conservativity of its semantics. All results are intuitively simple but have previously not been stated rigorously and in full generality.

1 Introduction

Most abstractly, conservativity means to extend a formal system in a way that effectively does not change it. Because this subsumes adding definitions and theorems, it is central to both informal and formal mathematics. Two notions have been used 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 but not the other way around.

This paper develops both definitions in the context of the author's Mmt language [RK13,Rab17a] that combines theory morphisms [FGT92], proof-theoretical logical frameworks [Pfe01], and institutional model theory [GB92].

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

[1] The author has traced the model-theoretical definition back to [Hod93] but could not locate the first use of the (older) proof theoretical definition.

We substantially generalize the two notions in order to arrive at definitions that allow emphasizing the commonalities and differences of the two definitions. We prove various meta-theorems in full generality that can be cast as the following slogans:

- Syntactic, proof- and model-theoretical conservativity are canonical points on a spectrum of conservativity notions.
- That spectrum corresponds to the existing admissible-derivable distinction.
- Completeness of a logic is conservativity of the semantics.

Overview Sect. 2 recalls existing definitions of logics and conservativity, both abstractly in plain mathematical notations, and concretely using MMT and LF as a logical framework for formalizing logics. MMT is maturely implemented, but this is not essential for this paper; instead, we abstract from all idiosyncrasies of the implementation and use ordinary mathematical notations for the MMT concepts. Sect. 3 develops our definitions of conservativity in this framework and establishes their properties. Sect. 4 establishes the link to completeness. A preliminary version of this work was informally published as [Rab17b].

2 Logics

2.1 Abstract Logics

Abstractly, we can define an arbitrary logic as a tuple of

- syntax:
  - a category of the **theories** Σ and the **theory morphisms** $v : \Sigma \to \Sigma'$,
  - a set $\mathbf{Sen}(\Sigma)$ for each Σ holding the sentences, and a function $v(-) : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ for each v translating sentences along morphisms,
- proof theory: for every Σ resp. v
  - a calculus defining a set of **derivations** resp. a function $v(-)$ that translates derivations along morphisms,
  - a judgment $\vdash_\Sigma F$ for $F \in \mathbf{Sen}(\Sigma)$ whose derivations determine the **provable** sentences,
- model theory: for every Σ resp. v
  - a class of **models** $\mathbf{Mod}(\Sigma)$ resp. a function $\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ that reduces models against morphisms,
  - a **satisfaction** relation between models in $\mathbf{Mod}(\Sigma)$ and sentences in $\mathbf{Sen}(\Sigma)$,

with some coherence conditions between them that are not essential here. This abstract definition already allows stating some common definitions:

- a sentence is **valid** if it is satisfied by every model
- a logic is **sound** if provable sentences are valid, and **complete** if valid sentences are provable

Conservativity is usually defined for theory *extensions* $\Sigma \hookrightarrow \Sigma'$. That would require introducing a suitable notion of inclusion morphism. But actually it gets both simpler and more general if we use an arbitrary morphism instead:

Definition 1 (Conservative). *A morphism $v : \Sigma \to \Sigma'$ is*

- ***syntactically conservative*** *(SC) if it has a retraction, i.e., a morphism r such that $v; r = id_\Sigma$,*
- ***model-theoretically conservative*** *(MC) if for every Σ-model m there is a Σ'-model m' that reduces to m via v,*
- ***proof-theoretically conservative*** *(PC) if every Σ-sentence F is provable iff the Σ'-sentence $v(F)$ is provable.*

Intuitively, SC means that the syntax does not change substantially along v: anything expressible in Σ' could (via the retraction) already be expressed in Σ. That is a very strong condition, but it subsumes a lot of practically relevant cases including isomorphisms, extension with a definable constant, extension with a provable theorem, and almost all extensions with a new uninterpreted type, predicate, or function symbol.

In particular, if v is an extension, then sentence translation $v(-)$ is the identity function, and model reduction is a forgetful functor, and we recover the usual proof and model theoretical definitions. Then MC means that the extension does not change models substantially: any Σ-model can be extended to a Σ'-model. And PC means that the extension does not change provability: $F \in \mathbf{Sen}(\Sigma)$ is provable over Σ iff it is provable over Σ'.

2.2 The LF Type Theory

The above definitions, while simple and intuitive, *abstract* from the internal structure of the concepts: they do not capture how, typically, theories are sets of declarations, sentences are generated by a grammar, theorems by an inference system, and sentence translation by homomorphic extension, and how Σ-models provide interpretations for the declarations in Σ, and how satisfaction is defined by an inductive interpretation function.

To state our results, we will make these notions more *concrete* by capturing this internal structure. The framework presented in [Rab17a], which combines proof-theoretical concepts from LF [HHP93] with model-theoretical concepts from institutions [GB92], does that in a concise and still fairly general way. The basic idea is to represent

- theories/morphisms as certain LF-theories/morphisms,
- sentences and derivations as certain LF-expressions and sentence translation as LF-homomorphic extension,
- models as certain LF-morphisms, and model reduction as composition of LF-morphisms.

Therefore, we recap the type theory of LF

Theories and Expressions LF is a dependent type theory using

- a universe `type` of types
- dependent function types $\Pi_{x:A}\, B$ for types A, B (written $A \to B$ if x does not occur free in B)

– function formation $\lambda_{x:A}\, t$,
– function application $f\, a$,

An LF-theory Σ is a list of declarations, each of which is either
– a dependent type symbol $c : \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}\, \texttt{type}$, or
– a term symbol $c : A$ for some type A.
Each declaration must be well-typed relative to the preceding declarations. We write Σ, Σ' for the concatenation of two theories.

Relative to a theory Σ, 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 $A \equiv A'$ defined by the usual typing rules and α, β, and η-conversion. Typing and equality are decidable, and each term has a unique type up to type equality. We say that a type A is **inhabited** if there is a term $t : A$. We refer to [HHP93] for details.

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

$$
\begin{array}{ll@{\qquad}ll}
o & : \texttt{type} & \doteq & : i \to i \to o \\
i & : \texttt{type} & \forall & : (i \to o) \to o \\
\mathit{thm} & : o \to \texttt{type} & \exists & : (i \to o) \to o \\
\neg & : o \to o & \multicolumn{2}{l}{\mathit{mp} : \Pi_{F:o}\, \Pi_{G:o}\, \mathit{thm}\,(F \Rightarrow G) \to \mathit{thm}\, F \to \mathit{thm}\, G} \\
\wedge & : o \to o \to o & \multicolumn{2}{l}{\vdots} \\
\Rightarrow & : o \to o \to o &
\end{array}
$$

FOL-terms and sentences are represented as LF-terms over the theory *FOL* of type i and o, respectively. For example, 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 $\forall x.F(x)$. From now on, we will use the usual notations $F \wedge G$, $\forall x.F(x)$, etc. instead of the ones technically prescribed by the representation in LF.

The type symbol *thm* serves as the provability judgment. Terms of type $\mathit{thm}\, F_1 \to \ldots \mathit{thm}\, F_n \to \mathit{thm}\, F$ are the derivations of F using assumptions $F_1, \ldots, F_n$. In particular, F is provable if the type $\mathit{thm}\, F$ is inhabited.

We only give a single proof rule as an example here: 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 and Homomorphic Extension An LF-theory morphism $\sigma : \Sigma \to \Sigma'$ is a list of assignments, one for each declaration in Σ:
– for every Σ-type symbol $c : \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}\, \texttt{type}$, a type assignment

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

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

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

Every theory morphism σ extends to a homomorphic translation, which maps Σ-expressions E to Σ'-expressions $\sigma(E)$ by replacing every Σ-symbol with the respective assignment provided by σ. $\sigma(-)$ 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* to 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 $\in\colon i \to i \to o$ 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 $Elem : i \to \texttt{type}$ to *ZF* that is axiomatized in such a way that terms of type $Elem\, A$ correspond exactly to the elements of the set $A : i$. We also lift the equality predicate $\doteq$ on sets to the family of predicates $\Pi_A\, Elem\, A \to Elem\, A \to o$ on set-elements. For simplicity, we denote it by infix $\doteq$ as well. The complete definition of *ZF* can be found in [IR11].

We extend *ZF* with a theory Δ that axiomatizes the common structure of a FOL-model. This can be seen as a model of the empty FOL-theory. Thus, Δ is the theory of a non-empty set (the universe of the model), i.e., it contains the declarations $univ : i$ and $nonempty : thm\,(\exists x.x \in univ)$.

Then we define the semantics as a morphism $FOLZF : FOL \to ZF, \Delta$. The intuition of *FOLZF* is that its homomorphic extension represents the interpretation function that maps *FOL*-syntax to its *ZF*-semantics. It contains in particular the following assignments:

- $i \mapsto Elem\, univ$, i.e., terms are interpreted as elements of $univ$,
- $o \mapsto Elem\, bool$, i.e., formula are interpreted a Booleans,
- $thm \mapsto \lambda x : Elem\, bool.thm(x \doteq 1)$, i.e., provable sentences are interpreted as the Boolean 1.

For all connectives and quantifiers, *FOLZF* contains an assignment that captures the respective case of the inductive interpretation function. For example, $FOLZF(\wedge)$ is the binary conjunction of Booleans.

Because LF-morphisms preserve typing, $FOLZF(thm\, F) = thm(FOLZF(F) \doteq 1)$ means that if F is provable, i.e., $thm\, F$ is inhabited, *ZF* must allow proving $FOLZF(F) \doteq 1$. Consequently, after adding *FOLZF*-assignments to all proof rules in *FOL*, the morphism *FOLZF* proves the soundness of the semantics with each assignment to a proof rule representing a case in the soundness proof.

Category and Pushouts LF theories and theory morphisms form a category. The identity id_Σ uses the assignment $c \mapsto c$ for all Σ-symbols c. The composition $\sigma;\sigma'$ uses the assignment $c \mapsto \sigma'(E)$ whenever σ contains $c \mapsto E$. Moreover, this category has pushouts of inclusions as in

$$\begin{array}{ccc} Syn, \Sigma & \xrightarrow{sem^{\Sigma}} & Sem, sem(\Sigma) \\ \uparrow & & \uparrow \\ Syn & \xrightarrow{sem} & Sem \end{array}$$

Here $sem(\Sigma)$ is given by the declarations $c : sem^{\Sigma}(U)$ for every declaration $c : U$ in Σ (where we have merged the cases of type and term symbols into a single case $c : U$ for convenience). sem^{Σ} maps Syn-constants in the same way as sem (i.e., the rectangle commutes) and maps $c \mapsto c$ for each c declared in Σ.

Example 3 (FOL-Theories). The FOL-theory *Semigroup* is represented by the LF theory that includes *FOL* and adds the declarations

$$\begin{array}{ll} \circ & : i \to i \to i \\ associative & : thm\, \forall x.\forall y.\forall z.(x \circ y) \circ z \doteq x \circ (y \circ z) \end{array}$$

where we use the usual infix notation for $\circ$. The pushout of this theory along $FOLZF : FOL \to ZF, \Delta$ results in the theory $ZF, \Delta, FOLZF(Semigroup)$, which adds the declarations

$$\begin{array}{ll} \circ & : Elem\ univ \to Elem\ univ \to Elem\ univ \\ associative & : thm\, FOLZF^{Semigroup}\big(\forall x.\forall y.\forall z.(x \circ y) \circ z \doteq x \circ (y \circ z)\big) \doteq 1 \end{array}$$

The morphism $FOLZF^{Semigroup}$ (whose homomorphic extension in the theory above we have left unsimplified for simplicity) extends *FOLZF* with the assignments $\circ \mapsto \circ$ and $associative \mapsto associative$.

For a morphism $\sigma : Syn, \Sigma \to Syn, \Sigma'$ that is the identity on Syn, we write $sem(\sigma) : Sem, sem(\Sigma) \to Sem, sem(\Sigma')$ for the unique factorization through the pushout. It is obtained by applying the homomorphic extension of sem throughout σ. Combining the above definitions, $sem(-)$ becomes a functor from extensions of Syn to extensions of Sem.[2]

Example 4 (FOL-Theory Morphisms). The theory *Semigroup* admits an endomorphism $FlipSemigroup : Semigroup \to Semigroup$ that flips the arguments of the operation. It extends id_{FOL} with the assignments $\circ \mapsto \lambda_{x:i}\, \lambda_{y:i}\, y \circ x$ and $associative \mapsto P$ where the omitted proof P shows that the flipped operation is associative as well.

$FOLZF(FlipSemigroup)$ is the endomorphism of $ZF, \Delta, FOLZF(Semigroup)$ that extends $id_{ZF,\Delta}$ with the assignments $\circ \mapsto \lambda_{x:Elem\ univ}\, \lambda_{y:Elem\ univ}\, y \circ x$ and $associative \mapsto FOLZF^{Semigroup}(P)$.

[2] 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 Σ. This is discussed in [Rab17a] and not essential for our results.

2.3 Concrete Logics in LF

We can now give a more concrete definition of logic. It formally captures the intuition from Ex. 1 and 2, where we represented essential parts of FOL using theories and morphisms such that $FOLZF : FOL \to ZF, \Delta$.

Definition 2 (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 Syn′-expression $k : l(o) \to o'$.*

*We call k the **truth lifting**. It is uniquely determined by l.*

Definition 3 (Logic). *A logic is a 4-tuple forming a logical morphism $sem : Syn \to Sem, \Delta$.*

Example 5 (First-Order Logic). FOL from Ex. 1 is a logical theory where o and thm are the distinguished declarations.

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

Below we use one auxiliary definition that allows us to talk about inconsistency and classicality of arbitrary logical theories:

Definition 4. *For a type A in a logical theory, we abbreviate $\overline{A} := A \to \Pi_{F:o} thm\, F$. A logical theory is* classical *if it has a term of type $\Pi_{F:o} \overline{\overline{thm\, F}} \to thm\, F$.*

If A and $\overline{A}$ are inhabited, the logical theory is inconsistent because every sentence is provable. Thus, $\overline{thm\, F}$ captures the negation of F, and classicality captures double-negation elimination. Defining $\overline{A}$ for arbitrary types has the advantage that it works for any logical theory without any assumptions about which connectives are present.

Every concrete logic L in the sense of Def. 3 induces an abstract logic in the sense of Sect. 2.1. **In the remainder of this section**, we make these intuitions precise for a **fixed logic** $sem : Syn \to Sem, \Delta$ with truth lifting k.

The logical theory Syn represents the syntax and proof theory of L: sentences are the terms of type o, and derivations are the terms of type $thm\, F_1 \to \ldots \to thm\, F_n \to thm\, F$. To represent the theories of L, we use extensions of Syn:

Definition 5 (Syntax and Proofs). *A Syn-**theory** is an extension $Syn \hookrightarrow Syn, \Sigma$ of Syn.*

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

Consider a Syn-theory Σ:

- *A Σ-**sentence** is a Σ-term $F : o$.*
- *A Σ-**proof** of F is a Σ-term $p : thm\, F$.*
- *A Σ-**disproof** of F is a Σ-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 Σ to Σ' by applying the homomorphic extension $\sigma(-)$.

Usually, only certain *Syn*-theories represent actual theories of L.:

Example 6 (Legal FOL-Theories). The theories of first-order logic are those extensions of *FOL* that declare only function symbols $c : i \to \ldots \to i \to i$, predicate symbols $c : i \to \ldots \to i \to o$, or axioms $c : thm\, F$. The theory from Ex. 3 is one of those. We can disregard all other extensions of *FOL*.

[HR12] introduces a way to specify exactly which *Syn*-theories are desired. But for our purposes this is not essential.

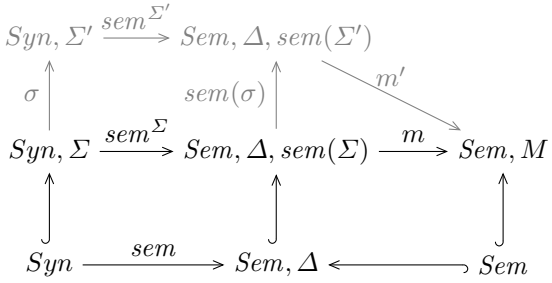
The logical theory *Sem* formalizes the ambient foundation of mathematics that was implicitly assumed in Def. 1, e.g., a formalization of *ZF* of set theory. Alternatively, we could use a formal foundation such as the ones underlying proof assistants like HOL or Coq. Δ is a *Sem*-theory that axiomatizes what it means to be a model of L. While this is just a non-empty set for FOL, it 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.

To lift these notions to arbitrary *Syn*-theories, we use the pushout of $Syn \hookrightarrow Syn, \Sigma$ along *sem*:

Definition 6 (Semantics and Models). *Consider a Syn-theory Σ.*

- *A Σ-**model** via sem is a Sem-theory morphism $m : \Delta, sem(\Sigma) \to M$ such that every Σ-sentence is either true or false in m.*
- *A Σ-sentence F is **true** resp. **false** in such a model m if Sem, M proves resp. disproves $k\,(sem^{\Sigma}(F))$.*

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



Note that F is true resp. false in m iff $(sem^{\Sigma}; m)(thm\, F)$ resp. $\overline{(sem^{\Sigma}; m)(thm\, F)}$ is inhabited.

Finally, yet another way to frame conservativity is consistency preservation:

Definition 7 (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.*

3 Conservative Theory Morphisms

3.1 Admissibility and Derivability

As a stepping stone, we define a general property of morphisms that generalizes the well-known concepts of admissible and derivable rules.

In the narrower, well-known sense, *derivable* and *admissible* both mean that a rule can be added to an inference system without changing the derivable sentences. The former is stronger by additionally requiring a derivation of the rule inside the proof system. Admissible-but-not-derivable rules include the deduction theorem in Hilbert calculi or the cut rule in sequent calculi. Proofs of derivability are usually straightforward: we simply exhibit the derivation. Proofs of admissibility can be very involved, often requiring an induction over all derivations. Derivability is also the more robust notion: Extending an inference system with new rules can never break the derivability of a rule but can break admissibility (because it adds a new case in the induction).

In our concrete logics, adding a rule to an inference system amounts to extending an LF-theory with a declaration. Following the idea of Def. 1, we can generalize to arbitrary morphisms:

Definition 8 (Admissible/Derivable Morphisms). *Consider a theory morphism $v : Syn \to Syn'$.*

*v is called **admissible** for a set J of Syn-types, if it reflects inhabitation of all types in J, i.e., if every type $A \in J$ is inhabited over Syn iff $v(A)$ is inhabited over Syn'.*
We simply say "T-admissible" if J is the image of the type-valued function T over Syn.

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

In the definition of admissibility, we write "iff", but only the right-to-left implication is special; the other direction always holds anyway.

The remainder collects some elementary closure properties that are helpful later but can be skipped on a first read:

Theorem 1 (Closure Properties). *Derivable and admissible morphisms have the following closure properties:*
- *The identity morphism is derivable and J-admissible for any J.*
- *A composition $v; w$ is derivable if v and w are. It is J-admissible if v is J-admissible and w is $v(J)$-admissible.*
- *If $v; w$ is derivable or J-admissible, then so is v.*
- *The union $\Sigma, \Sigma_0, \Sigma_1$ of derivable extensions Σ, Σ_0 and Σ, Σ_1 is derivable.*
- *The pushout of a derivable morphism is derivable.*

Proof. All proofs are straightforward.

The closure under pushouts captures the robustness of derivability: As seen in Def. 6, pushout along *sem* represents the interpretation of a language Syn in another language Sem. Such translations preserve derivability of Syn-morphisms. Admissibility of morphisms, however, is much more brittle and can be lost by pushouts along relatively simple morphisms.

3.2 Conservativity between Logics

If we consider morphisms between logical theories, Def. 8 subsumes the usual definition of admissible and derivable rules:

Example 7 (Admissible/Derivable Rules). Consider a proof theory represented as a logical theory Syn, and a rule represented as a type R over Syn. Then R is admissible as a rule in the usual sense if the extension morphism $Syn \hookrightarrow Syn, c : R$ is thm-admissible. Indeed, thm-admissibility means that over $Syn, c : R$ (i.e., Syn extended with the rule R) there is no inhabited type of the form $thm\, F$ over $Syn, c : R$ that is not also inhabited over Syn.

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 retraction $FOL, c : R \to FOL$ maps $c \mapsto \lambda_{F,G,H}\, \lambda_{p,q,r}\, mp\, G\, H\, (mp\, F\, (G \Rightarrow H)\, p\, q)\, r$.

Similarly, R is derivable as a rule in the usual sense if the extension morphism $Syn \hookrightarrow Syn, c : R$ is derivable. Indeed, a retraction consists only of an assignment $c \mapsto r$ such that $r : R$ over Syn, and such an r is exactly a derivation of R. For example, if we add the axioms of Hilbert-calculus to FOL, the type $R = (thm\, F \to thm\, G) \to thm\, F \Rightarrow G$ (the deduction theorem) is not inhabited and thus R is not derivable. Yet, adding $c : R$ does not make new sentences provable, i.e., R is thm-admissible.

For the important special case of extending a logical theory with some rules, this definition also captures the intuition that derivability is the extreme case of admissibility:

Theorem 2. *An extension $Syn \hookrightarrow Syn, Syn'$ where Σ adds no type symbols, is derivable iff it is admissible for all Syn-types.*

Proof. Assume there is a retraction r. For every Syn-type A, if there is Syn, Syn'-term $t : A$, then $r(t) : r(A)$ is the needed Syn-term. Thus, the extension is admissible for arbitrary types A.

Assume admissibility for every type A. We build the retraction r by induction on Σ. Assume we have $r : Syn, Syn_0 \to Syn$. For the next Syn'-declaration $c : A$, we instantiate admissibility with $r(A)$ to obtain a Syn-term a. Then $r,\ c \mapsto a$ is a retraction $Syn, Syn_0, c : A \to Syn$.

For *arbitrary* morphisms $v : Syn \to Syn'$, admissibility for all Syn-types is *not* the same as derivability. 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 $v(A)$.

3.3 Conservativity within a Logic

In this section, we fix a concrete logic $sem : Syn \to Sem, \Delta$. We want to study the conservativity of a Syn-morphism $\sigma : \Sigma \to \Sigma'$ as in the diagram below. Recall that pushout translates Σ and Σ' to $sem(\Sigma)$ resp. $sem(\Sigma')$, and (by the universal factorization through the pushout) σ to $sem(\sigma)$.

$$\begin{array}{ccc} Syn, \Sigma' & \xrightarrow{sem^{\Sigma'}} & Sem, \Delta, sem(\Sigma') \\ \uparrow{\scriptstyle \sigma} & & \uparrow{\scriptstyle sem(\sigma)} \\ Syn, \Sigma & \xrightarrow{sem^{\Sigma}} & Sem, \Delta, sem(\Sigma) \\ \uparrow & & \uparrow \\ Syn & \xrightarrow{sem} & Sem, \Delta \end{array}$$

Def. 8 allows characterizing syntactic and proof-theoretical conservativity directly:

Theorem 3 (Syntactically Conservative). *σ is SC iff it is derivable with a retraction that is also Syn-morphism.*

Proof. This follows immediately from the definitions.

Theorem 4 (Proof-Theoretically Conservative). *σ is PC iff it is thm-admissible.*

Proof. This follows immediately because a sentence F is provable iff $thm\, F$ is inhabited.

Characterizing model-theoretical conservativity is harder. It will sit between derivability and admissibility, and we split the characterization into two statements.

Theorem 5. *SC implies MC.*

Proof. By Thm. 1, if σ is derivable, then so is $sem(\sigma)$. Let r be its retraction. Now assume a Σ-model $m : Sem, \Delta, sem(\Sigma) \to Sem, M$. Then $m' = r; m$ is the needed Σ'-model. Indeed, the reduct of m' via σ is $sem(\sigma); m' = m$.

Theorem 6. *In a sound and complete logic, MC implies PC.*

Proof. Straightforward.

A more precise characterization of MC depends on subtle properties of Sem, the ambient foundation of mathematics that is implicitly assumed in Def. 1 but made explicit in Def. 6. If we formalize model-theoretical conservativity from Def. 1 relative to Sem, we obtain something like the Sem-sentence

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

Assume we have proved that sentence. To show that σ is derivable, is equivalent to exhibiting a function f such that

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

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 case $\mathit{Sem} = \mathit{ZF}$. Then we cannot necessarily exhibit f even if we add the axiom of choice to ZF. Alternatively, consider *Sem* to be a language with a Hilbert-style choice operator $\varepsilon : \Pi_{F:i\to\mathtt{prop}} \to i$, which chooses some element that satisfies F provided that such an element exists. Then we can define f as the LF-expression

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

Such a choice operator is a very strong language feature and therefore adding it to a formal system is often avoided. It exists, e.g., in higher-order logic [Gor88] and in the set theory underlying Mizar [TB85]. In some constructive foundations, it is definable.

The following theorem makes this precise:

Theorem 7. *Assume that Sem*
- *adequately formalizes the ambient foundation that is implicitly used in Def. 1*
- *can encode record types*
- *can define a function f such that $F(m, f(m))$ whenever it can prove "for all m exists m' such that $F(m, m')$"*

Then $\sigma : \Sigma \to \Sigma'$ is MC if the Sem-morphism $\mathit{sem}(\sigma)$ is SC.

Proof. The right-to-left direction is Thm. 5.

For the left-to-right 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 $\mathit{sem}(\sigma)$ as follows. First we package the declarations in $\Delta, \mathit{sem}(\Sigma)$ into a *Sem*-term m by using a record type. Then r maps every symbol c of $\mathit{sem}(\Sigma')$ to the term that selects the component c from $f(m)$.

3.4 The Conservativity Spectrum

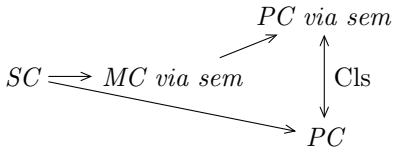
The derivability of σ implies the derivability of $\mathit{sem}(\sigma)$, but the reverse is not true in general. That inspires the following generalizations:

Definition 9. *A Syn-morphism $\sigma : \Sigma \to \Sigma'$ is called*
- *PC via sem: if $\mathit{sem}(\sigma)$ is $\mathit{sem}^{\Sigma}(\mathit{thm})$-admissible,*
- *MC via sem: if $\mathit{sem}(\sigma)$ is derivable with a retraction that is a Sem-morphism.*

The trivial case of conservativity via id_{Syn} yields the following: SC is MC via id_{Syn}, and (by Thm. 4) PC is PC via id_{Syn}. PC via *sem* expresses that semantic proofs (i.e. proofs about an arbitrary model carried out in the ambient foundation *Sem*) exhibit the typical conservativity property. And Thm. 7 relates MC to MC via *sem*.

The following theorem establishes a spectrum in which SC/derivable and PC/admissible form the extreme points and MC via *sem* sits in the middle:

Theorem 8. *Let* Cls *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*:*



Proof. MC via *sem* implies PC via *sem*: This is immediate because the derivability of $sem(\sigma)$ implies its admissibility for all types.

SC implies PC: As above for the special case $sem = id_{Syn}$.

SC implies MC via *sem*: This is the preservation of derivability along pushout from Thm. 1.

PC via *sem* implies PC if Cls: Consider a Σ-sentence F such that there is a Σ'-proof $p' : \sigma(thm\, F)$. We need to exhibit a Σ-proof $p : thm\, F$.

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

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

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

We obtain p from q as follows. If Σ is inconsistent, p exists trivially. So assume it is consistent. Consider $\Sigma^* = \Sigma, a : \overline{thm\, F}$. If Σ^* is inconsistent, we obtain a Σ-term of type $\overline{\overline{thm\, F}}$ and classicality of *Syn* yields the needed term p. We conclude the proof by showing that Σ^* 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.

PC implies PC via *sem* if Cls: Consider a Σ-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 Σ is inconsistent, this is trivial. So assume it is consistent. Then PC implies that Σ' is consistent, too. Now as in the previous case, we use the consistency of Σ' and Cls to obtain from q' a Σ'-term $p' : v(thm\, F)$. Then PC yields a Σ-term $p : thm\, F$, and we can put $q = sem^{\Sigma}(p)$.

SC captures the situation where the syntax of the logic itself can express the proof of conservativity: as a theory morphism that retracts σ. Therefore, if σ satisfies SC, σ satisfies any other reasonable definition of conservativity, i.e., SC

is the minimal/strongest reasonable definition. In particular, the retraction of σ yields both a proof transformation from Σ' to Σ, which shows that SC implies PC, and a model reduction from Σ to Σ', which helps showing that SC implies MC via *sem*.[3]

PC can be seen as a dual to SC—not satisfying PC captures the situation where the syntax of the logic itself can express a counter-example to conservativity: as a Σ-sentence that is a Σ'-theorem but not a Σ-theorem. Therefore, if σ satisfies any reasonable definition of conservativity, it should satisfy PC, i.e., PC is the maximal/weakest reasonable choice.

MC via *sem* sits in between SC and PC. 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 Σ-models to Σ'-models. Because the semantics is usually more expressive than the syntax, it is not surprising that MC via *sem* is weaker than SC. Moreover, because there may be multiple different ways to give the semantics of a logic, it is not surprising that MC depends on the choice of the model theory.

MC-via is preserved along refinement of the semantics: MC via *sem* implies MC via $sem; sem'$. SC is the case MC via id_{Syn} where Syn is not refined at all. The more we refine Syn by translating it along sem_1, $sem_1; sem_2, \ldots$, the weaker MC-via becomes. Intuitively, in the limit, MC via $sem_1; sem_2; \ldots$ coincides with PC. This happens, e.g., if we use maximal consistent sets of sentences as the models.

In particular, if the logic is classical and all sem_i preserve consistency, we have the following simple picture of the spectrum:

$$\begin{array}{c} \text{SC} = \text{MC via } id_{Syn} \\ \downarrow \\ \text{MC via } sem_1 \\ \downarrow \\ \text{MC via } sem_1; sem_2 \\ \downarrow \\ \vdots \\ \downarrow \\ \text{PC} = \text{MC via models are maximal consistent sets of sentences} \end{array}$$

4 Conservativity and Completeness

Finally, we can understand completeness as a special case of conservativity:

Theorem 9. *The logic $sem : Syn \to Sem, \Delta$ is complete iff all sem^{Σ} are thm-admissible.*

[3] The corresponding observation for the framework of institutions was previously made in [SML04].

Proof. [Rab17a] already proves that a sentence F holds in all Σ-models iff $Sem, \Delta, sem(\Sigma)$ has a term of type $sem^{\Sigma}(thm\, F)$. Due to admissibility, such a term exists iff Σ has a term of type $thm\, F$.

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.

One might hope for a stronger theorem where completeness already holds whenever *sem* is *thm*-admissible. For that, we have to ask if the admissibility of *sem* implies the admissibility of sem^{Σ}. That feels true, but it is unfortunately not always the case. We develop a sufficient criterion now:

Definition 10. *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 8. *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 10. *Assume a logic where sem is thm-admissible.*

If Syn can abstract over Σ, then sem^{Σ} 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 counterexample, 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 PC.

5 Conclusion

We investigated the various notions of conservativity in a meta-logical framework. We found a spectrum consisting of syntactic $\to$ model-theoretical via *sem* $\to$ proof-theoretical conservativity. Here the end points correspond to derivability and admissibility.

The left end point, syntactical conservativity, can be seen as the special case of initial semantics, where the syntax is used as its own semantics. The middle points form a lattice of refinements where the syntax is gradually translated in a sound and complete way into more expressive languages. With each refinement of the semantics, a weaker notion of model-theoretical conservativity arises. In the limit, these collapse into the right end point of proof-theoretical conservativity: its maximally refined model theory can be seen as models given by maximal consistent sets of sentences.

This harmonically resolves the tension between the competing notions of conservativity. In a related 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

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

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

Gor88. M. Gordon. HOL: A Proof Generating System for Higher-Order Logic. In G. Birtwistle and P. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis*, pages 73–128. Kluwer-Academic Publishers, 1988.

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.

Hod93. W. Hodges. *Model Theory*. Cambridge University Press, 1993.

HR12. F. Horozal and F. Rabe. Representing Categories of Theories in a Proof-Theoretical Logical Framework. In *Workshop on Algebraic Development Techniques*, 2012.

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.

Rab17a. F. Rabe. How to Identify, Translate, and Combine Logics? *Journal of Logic and Computation*, 27(6):1753–1798, 2017.

Rab17b. F. Rabe. The MMT Perspective on Conservativity. In S. Alves and R. Wassermann, editors, *Logical and Semantic Frameworks, with Applications*, pages 17–33, 2017.

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

SML04. 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.

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.