The MMT Perspective on Conservativity

Florian Rabe

Jacobs University Bremen

September 2017

Basic Definitions

For the purposes of this talk, a logic L consists of

- theories T
- T-formulas $F : prop$
- T-proofs and provability judgment $\vdash_T F$
- T-models M and satisfaction $M \models_T F$

Theories are lists of symbol declarations.

types, functions, predicates, axioms, proof rules, rewrite rules, ...

$S \hookrightarrow T$ is a theory **extension** if S-declarations $\subseteq$ T-declarations

Two Conflicting Definitions of Conservativity

Intuition: $S \hookrightarrow T$ is **conservative** if T-semantics does not substantially differ from S-semantics.

e.g., T adds only definitions, theorems, admissible rules, ...

Problem: How to define that rigorously?

Two answers for "When is $S \hookrightarrow T$ conservative?":

- **proof theorist**: for any T-proof of S-formula F, there is an S-proof of F proof retraction
  mentions only proofs, no models
- **model theorist**: for any S-model M, there is a T-model M' that agrees with M on S-symbols model extension
  mentions only models, no proofs

Tension between the Definitions

Are proofs or models primary for semantics?

practical, social, and philosophical difference

Not so unusual — compare "When is F a theorem?"

- proof theorist: if there is a proof of F
- model theorist: if F holds in all models
- both are equivalent via soundness/completeness

achieved by fine-tuning the definitions

Ideally, model and proof-theoretical conservativity also equivalent.

(they aren't)

Relating the Definitions

Theorem:

If L is sound and complete, then
model-conservative implies proof-conservative.

But not the other way around.

causes confusion at best, conflict at worst

Motivation

Vision: UniFormal

a universal framework for the formal representation of knowledge

- integrate all domains
  model theory, proof theory, computation, mathematics, ...
- be independent of foundational languages
  logics, programming languages, foundations of mathematics, ...
- build generic, reusable implementations
  type checker, module system, library manager, IDE, ...

My (evolving) solution: MMT

- a uniformal knowledge representation framework
  developed since 2006, $\sim 100,000$ loc, ~ 500 pages of publications
- allows foundation-independent solutions
  module system, type reconstruction, theorem proving, ...
  IDE, search, build system, library, ...

`http://uniformal.github.io/`

Foundation-Independent Development

Foundation-specific workflow (almost all systems)

1. choose foundation
   type theories, set theories, first-order logics, higher-order logics, ...
2. implement kernel
3. develop support algorithms, tools reconstruction, proving, IDE, ...
4. build library

Foundation-independent workflow (MMT)

1. MMT provides generic kernel
   no built-in bias towards any foundation
2. develop generic support on top of MMT
3. flexibly customize MMT for desired foundation(s)
4. build multi-foundation universal library

Advantages of Foundation-Independence

- Avoids segregation into mutually incompatible systems
- Allows maximally general results

  meta-theorems, algorithms, formalizations
- Separation of concerns between
  - foundation developers
  - support service developers: search, axiom selection, ...
  - application developers: IDE, proof assistant, ...
- Rapid prototyping for logic systems
- Allows evolving and experimenting with foundations

But how much can be done foundation-independently?

surprisingly much — this talk: conservativity

Logical Frameworks and Syntax

Logical framework LF in MMT

```
theory LF {
    type
    Pi       # Π V1 . 2                    name[:type][#notation]
    arrow    # 1 → 2
    lambda   # λ V1 . 2
    apply    # 1 2
}
```

Logics in MMT/LF

```
theory Logic: LF {
    prop : type
    ded  : prop → type # ⊢ 1                   judgments-as-types
}
theory FOLSyn: LF {
  include Logic
  term        : type                    higher-order abstract syntax
  forall      : (term → prop) → prop #  ∀ V1 . 2
}
```

Proof Theory

FOLSyn from previous slide:

```
theory FOLSyn : LF {
  include Logic
  term        : type
  forall      : (term → prop) → prop  #  ∀ V1 . 2
}
```

Proof-theory = syntax + calculus

```
theory FOL : LF {
  include FOLSyn
                                                rules are constants
  forallIntro  : ΠF:term→prop .
                    (Πx:term.⊢ (F x)) → ⊢ ∀(λx:term.F x)
  forallElim   : ΠF:term→prop .
                    ⊢ ∀(λx:term.F x) → Πx:term.⊢ (F x)
}
```

Domain Theories

FOLSyn from previous slide:

```
theory FOLSyn: LF {
  include Logic
  term        : type
  forall      : (term → prop) → prop #  ∀ V1 . 2
}
```

Algebraic theories in MMT/LF/FOL:

```
theory Magma: FOL {
  comp : term → term → term # 1 ∘ 2
}
theory SemiGroup: FO {
  include Magma
  associative : ⊢ ∀ x,y,z. (x ∘ y) ∘ z = x ∘ (y ∘ z)
}
```

MMT Theory Morphisms (highly simplified)

An MMT **theory** is a list of declarations $c[: E]$, where E is an expression using the previous symbols.

An MMT theory **morphism** $m : S \to T$ maps every S-symbol to a T-expression such that
if $\vdash_S A : B$ then $\vdash_T m(A) : m(B)$ preservation of typing/truth

Model Theory

Universe = set theory, category theory, programming languages, ...

```
theory ZFC: LF {
  set   : type
  prop  : type
  in    : set → set → prop # 1 ∈ 2
  equal : set → set → prop # 1 = 2
  ded   : prop → type # ⊢ 1
  ...
  bool  : set = {0,1}
  ...
}
```

Interpretation = theory morphism from syntax+calculus to semantics

```
morphism FOLMod : FOL → ZFC {
  prop ↦ bool
  ded  ↦ λx∈bool.x=1
  ...                        (proof rules mapped to their soundness proofs)
}
```

Individual Models

FOLMod from previous slide:

```
morphism FOLMod : FOL → ZFC {
  prop ↦ bool
  ded  ↦ λx∈bool . x=1
  ...
}
```

Integer addition as a model of SemiGroup:

```
morphism IntegerAddition : SemiGroup → ZFC {
  include FOLMod
  term  ↦ ℤ
  comp  ↦ +
  assoc ↦ ... (proof that + is associative)
}
```

Derivable and Admissible Rules

Consider an extension $S \hookrightarrow T$ in MMT.

Example: $T = S,\ cut : R$
S is a cut-free sequent calculus and R is the cut rule.

We say that $S \hookrightarrow T$ is

- **derivable** if
  - example case: there is a term $r : R$ over S
  - general case: there is a retraction morphism $r : T \to S$
- $\vdash$-**admissible** if $\vdash F$ is inhabited over S whenever it is inhabited over T

Conservative as a Special Case of Derivable/Admissible

$$\begin{tikzcd}
FOL \arrow[r, "FOLMod"] \arrow[d, hook] & ZFC \arrow[d, hook] \\
S \arrow[r, "FOLMod * S"] \arrow[d, hook] & FOLMod(S) \arrow[d, hook] \arrow[u, bend right, "M \in Mod(S)"'] \\
T \arrow[r, "FOLMod * T"] & FOLMod(T)
\end{tikzcd}$$

$FOLMod(S)$

- pushout of $L \hookrightarrow S$ along $FOLMod$
- obtained by homomorphic translation of S-declarations

Conservative as a Special Case of Derivable/Admissible

$$\begin{tikzcd}
FOL \arrow[rr, "FOLMod"] \arrow[d, hook] & & ZFC \arrow[d, hook] \\
\color{blue}{S} \arrow[r, "FOLMod * S"] \arrow[d, hook, blue] & FOLMod(S) \arrow[d, hook] \arrow[r, phantom] & FOLMod(S) \arrow[u, bend right, "M \in Mod(S)"'] \\
\color{blue}{T} \arrow[r, "FOLMod * T"] & FOLMod(T) &
\end{tikzcd}$$

Theorem: $S \hookrightarrow T$ is

- proof-conservative iff $S \hookrightarrow T$ is $\vdash$-admissible
- model-conservative iff $FOLMod(S) \hookrightarrow FOLMod(T)$ is derivable

Different Kinds of Conservativity

General case: 4 notions of conservativity

$$\begin{tikzcd} S \arrow[r, "FOLMod * S"] \arrow[d, hook] & FOLMod(S) \arrow[d, hook] \\ T \arrow[r, "FOLMod * T"] & FOLMod(T) \end{tikzcd}$$

- $S \hookrightarrow T$ is $\vdash$-admissible — proof-conservative
- $S \hookrightarrow T$ is derivable
- $FOLMod(S) \hookrightarrow FOLMod(T)$ is derivable — model-conservative
- $FOLMod(S) \hookrightarrow FOLMod(T)$ is $\vdash$-admissible

Relating the Different Kinds of Conservativity

- $S \hookrightarrow T$ is derivable
  - syntax has witness for conservativity
  - minimal/strongest reasonable definition
- $S \hookrightarrow T$ is $\vdash$-admissible — proof-conservative
  - syntax has no counter-example for conservativity
  - maximal/weakest reasonable definition
- $FOLMod(S) \hookrightarrow FOLMod(T)$ is derivable — model-conservative
  - semantics has witness for conservativity
  - in between the above
- $FOLMod(S) \hookrightarrow FOLMod(T)$ is $\vdash$-admissible
  - equivalent to proof-conservative for sound and complete logics

Conservativity under Refinement of Semantics

Refinement chain of multiple interpretations, e.g.,

$$\begin{tikzcd}
ModalLogic \arrow[r, "p"] \arrow[d, hook] & FOL \arrow[r, "q"] \arrow[d, hook] & HOL \arrow[r, "r"] \arrow[d, hook] & ZFC \arrow[d, hook] \\
S \arrow[r] \arrow[d, hook] & p(S) \arrow[r] \arrow[d, hook] & q(p(S)) \arrow[r] \arrow[d, hook] & r(q(p(S))) \arrow[d, hook] \\
T \arrow[r] & p(T) \arrow[r] & q(p(T)) \arrow[r] & r(q(p(T)))
\end{tikzcd}$$

At each step, 2 notions of conservativity of $S \hookrightarrow T$:

- using $\vdash$-admissibility:
  - all notions equivalent for sound+complete interpretations
  - strongest possible notion proof-conservativity (absolute)
- Using derivability: model-conservativity relative to semantics
  - notions grow weaker as semantics is more refined
  - converges to proof-conservativity for increasing refinements

Summary

- MMT: foundation-independent framework for formal systems
  maximally general conceptualizations, theorems, implementations
- Allows resolving conflict between notions of conservativity
  results apply to arbitrary logic defined in arbitrary logical framework
- Proof-conservativity
  - corresponds to $\vdash$-admissiblity of rules
  - weakest possible notion
- Model-conservativity
  - corresponds to derivability of rules
  - relative to chosen model theory
  - strongest possible notion if applied to initial semantics
  - grows weaker as semantics is more refined
  - converges against proof-conservativity

Terms

Abstract syntax

contexts	Γ	$::=$	$(x[:E][=E])^*$
terms	E	$::=$	
constants		$\mid$	c
variables		$\mid$	x
complex terms		$\mid$	$c(\Gamma; E^*)$

Complex term examples

- typical operators: Γ empty e.g., $\texttt{apply}(\cdot; f, a)$ for $(f\ a)$
- typical binders: Γ and $\vec{E}$ have length 1 e.g., $\texttt{lambda}(x:A; t)$ for $\lambda x:A.t$

Judgments relative to a theory T that declares the constants c

$\Gamma \vdash_T t : E$	t has type E
$\Gamma \vdash_T E = E'$	E and E' are equal
$\Gamma \vdash_T \_ : E$	E is inhabitable