# An Isabelle/HOL Framework for Synthetic Completeness Proofs

**CPP 2025** - 2025-01-21

Asta Halkjær From, University of Copenhagen

# Motivation

- We have a new and exciting logic

$$A \vdash \varphi$$

- We want to prove completeness

$$\Gamma \vDash \varphi \Longrightarrow \Gamma \vdash \varphi$$

where $\Gamma \vdash \varphi$ means $A \vdash \varphi$ for some list of elements $A$ from $\Gamma$

- We want the computer to help us prove it

??

# Strategy

- Cf. R. M. Smullyan [1]
- Assume $\varphi$ is valid under $\Gamma$ but has no derivation

$$\Gamma \vDash \varphi, \quad \Gamma \nvdash \varphi$$

- Then $\neg\varphi, \Gamma$ is *consistent*

$$\neg\varphi, \Gamma \nvdash \bot$$

- *Lindenbaum's lemma* gives us a *maximal consistent, witnessed set* $S$
  - $\Gamma \subseteq S$
  - $\varphi \notin S \implies \varphi, S$ inconsistent
  - $\exists\varphi \in S \implies \varphi(t) \in S$, for some $t$ (whatever $\exists\varphi$ means)

## Strategy (ii)

- We prove a *truth lemma* for such MCSs $S$

$$\varphi \in S \iff S \vDash \varphi$$

- But $\neg\varphi, \Gamma \subseteq S$, so

$$S \vDash \neg\varphi, \Gamma$$

- This contradicts the initial assumption

$$\Gamma \vDash \varphi$$

- So we must have a derivation

$$\Gamma \vdash \varphi$$

# Desiderata

- Prove Lindenbaum's lemma *once*
  - subsets preserve consistency,
  - inconsistencies are finite,
  - witnesses preserve consistency
- *Reflect* proof rules in MCSs
  - $A \vdash \top \Longrightarrow \top \in S$
- Split the truth lemma in syntactic and semantics parts
  - MCSs are *saturated*
    - $(\exists \varphi) \in S \Longleftrightarrow \exists x.\ (\varphi(x) \in S)$
  - Saturated sets are models
    - Proved by Isabelle/HOL automation

Lindenbaum

# Lindenbaum's Lemma

- Transfinite proof adapted from C. C. Chang and H. J. Keisler [2]
  - Generalise from first-order logic
  - Formalize in higher-order logic instead of set theory
  - Without expanding the language (now a type)
  - Formalization inspired by S. Berghofer [3]
- Abstract consistency predicates over formulas `‹'a›`:

```
locale MCS_Base =
  fixes consistent :: ‹'a set ⇒ bool›
  assumes consistent_hereditary:
      ‹⋀S S'. consistent S ⟹ S' ⊆ S ⟹ consistent S'›
    and inconsistent_finite:
      ‹⋀S. ¬ consistent S ⟹ ∃S' ⊆ S. finite S' ∧ ¬ consistent S'›
```

# Witnesses

- Abstract `witness` and `params` functions over parameters `‹'i›`:

```
locale MCS_Witness = MCS_Base consistent
  for consistent :: ‹'a set ⇒ bool› +

  fixes witness :: ‹'a ⇒ 'a set ⇒ 'a set›
    and params :: ‹'a ⇒ 'i set›

  assumes finite_params:
      ‹⋀p. finite (params p)›
    and finite_witness_params:
      ‹⋀p S. finite (⋃q ∈ witness p S. params q)›
    and consistent_witness:
      ‹⋀p S. consistent ({p} ∪ S)
        ⟹ infinite (UNIV - (⋃q ∈ S. params q))
        ⟹ consistent ({p} ∪ S ∪ witness p S)›
```

# Maximal Consistent, Witnessed Sets

- Every consistent formula is included

```
definition maximal :: ‹'a set ⇒ bool› where
  ‹maximal S ≡ ∀p. consistent ({p} ∪ S) ⟶ p ∈ S›
```

- Every included formula has its witnesses included

```
definition witnessed :: ‹'a set ⇒ bool› where
  ‹witnessed S ≡ ∀p ∈ S. ∃S'. witness p S' ⊆ S›
```

- Convenient abbreviation

```
abbreviation MCS :: ‹'a set ⇒ bool› where
  ‹MCS S ≡ consistent S ∧ maximal S ∧ witnessed S›
```

# Lindenbaum Extension

- Extend a consistent set by formulas one-by-one
  - Assume an infinite cardinal order on the formulas

```
fixes r :: ‹'a rel›
assumes Cinfinite_r: ‹Cinfinite r›
```

- Add formula (and witnesses) exactly when consistency is preserved

```
definition extend :: ‹'a set ⇒ 'a ⇒ 'a set› where
  ‹extend S a ≡ worecZSL r S extendS extendL a›

  ‹extendS a prev ≡
    if consistent ({a} ∪ prev) then {a} ∪ prev ∪ witness a prev else prev›

  ‹extendL rec a ≡ ⋃b ∈ underS r a. rec b›
```

# Properties

- The union is an MCS

```
definition Extend :: ‹'a set ⇒ 'a set› where
  ‹Extend S ≡ ⋃a ∈ Field r. extend S a›
```

- Proofs follow C. C. Chang and H. J. Keisler [2]
  - Inconsistencies are finite
  - Assume at least as many unused parameters as formulas

```
theorem MCS_Extend:
  assumes ‹consistent S› ‹|UNIV :: 'a set| ≤o |UNIV - paramss S|›
  shows ‹MCS (Extend S)›
```

- Sublocales give us the witness-free cases for free

```
sublocale MCS_No_Witness_UNIV ⊆ MCS_Witness_UNIV consistent ‹λ_ _. {}› ‹λ_. {}›
```

# Example (i)

- Given a natural deduction system for first-order logic:

```
inductive Calculus :: ‹('f, 'p) fm list ⇒ ('f, 'p) fm ⇒ bool› (...) where
  Assm: ‹p ∈ set A ⟹ A ⊢∃ p›
| FlsE: ‹A ⊢∃ ⊥ ⟹ A ⊢∃ p›
| ImpI: ‹p # A ⊢∃ q ⟹ A ⊢∃ p ⟶ q›
| ImpE: ‹A ⊢∃ p ⟶ q ⟹ A ⊢∃ p ⟹ A ⊢∃ q›
| ExiI: ‹A ⊢∃ ⟨t⟩p ⟹ A ⊢∃ ∃p›
| ExiE: ‹A ⊢∃ ∃p ⟹ a ∉ params (set (p # q # A)) ⟹ ⟨⋆a⟩p # A ⊢∃ q ⟹ A ⊢∃ q›
| Clas: ‹(p ⟶ q) # A ⊢∃ p ⟹ A ⊢∃ p›
```

- Consistency means falsity is underivable:

```
definition consistent :: ‹('f, 'p) fm set ⇒ bool› where
  ‹consistent S ≡ ∀A. set A ⊆ S → ¬ A ⊢∃ ⊥›
```

# Example (ii)

- Existentials are witnessed by fresh parameters:

```
fun witness :: ‹('f, 'p) fm ⇒ ('f, 'p) fm set ⇒ ('f, 'p) fm set› where
  ‹witness (∃p) S = (let a = SOME a. a ∉ params ({p} ∪ S) in {⟨⋆a⟩p})›
| ‹witness _ _ = {}›
```

- Building an MCS requires proving six goals (only no. 5 non-trivial):

```
goal (6 subgoals):
 1. ‹⋀S S'. consistent S ⟹ S' ⊆ S ⟹ consistent S'›
 2. ‹⋀S. ¬ consistent S ⟹ ∃S'⊆S. finite S' ∧ ¬ consistent S'›
 3. ‹⋀p. finite (params_fm p)›
 4. ‹⋀p S. finite (params (witness p S))›
 5. ‹⋀p S. consistent ({p} ∪ S) ⟹ infinite (UNIV - params S)
      ⟹ consistent ({p} ∪ S ∪ witness p S)›
 6. ‹infinite UNIV›
```

# Reflection

# MCSs and Proof Rules

- The proof system is deliberately abstract
  - Let us make some assumptions
- Say we can derive and rearrange assumptions

```
fixes derive :: ‹'fm list ⇒ 'fm ⇒ bool› (infix ‹⊢› 50)
assumes derive_assm: ‹⋀A p. p ∈ set A ⟹ A ⊢ p›
    and derive_set: ‹⋀A B r. A ⊢ r ⟹ set A = set B ⟹ B ⊢ r›
```

- Say consistency means something is underivable

```
assumes consistent_underivable:
  ‹⋀S. consistent S ⟷ (∀A. set A ⊆ S → (∃q. ¬ A ⊢ q))›
```

# Absence

- This tells us about MCS *absence*:
  - A formula is excluded exactly when inclusion is *explosive*

```
theorem MCS_explode:
  assumes ‹consistent S› ‹maximal S›
  shows ‹p ∉ S ⟷ (∃A. set A ⊆ S ∧ (∀q. p # A ⊢ q))›
```

- if $p \notin S$,
  - $p$, $S$ is inconsistent by maximality,
  - assuming $p$ would be explosive. ↯
- if having $p$ is explosive,
  - $p$, $S$ is inconsistent
  - $p \in S$ would contradict assumption. ↯

# Presence

- Say we also have a general cut principle:

```
assumes derive_cut: ‹⋀A B p q. A ⊢ p ⟹ p # B ⊢ q ⟹ A @ B ⊢ q›
```

- This tells us about MCS *presence*:
  - A formula is included exactly when it can be derived

```
theorem MCS_derive:
  assumes ‹consistent S› ‹maximal S›
  shows ‹p ∈ S ⟷ (∃A. set A ⊆ S ∧ A ⊢ p)›
```

- if $p \in S$, we derive it as an assumption
- if $A \vdash p$, then $p \in S$ follows by maximality, given $p$, $S$ consistent
  - assume $p$, $B$ explosive ($B$ from $S$), then $A$, $B$ explosive by cut,
  - but $A$, $B$ are from consistent $S$. ↯

# Reflecting Falsity

- Assume a concrete piece of syntax with a concrete proof rule:

```
fixes bot :: 'fm (‹⊥›)
assumes botE: ‹⋀A p. A ⊢ ⊥ ⟹ A ⊢ p›
```

- Then we can mirror this in any MCS:

```
corollary MCS_botE:
  assumes ‹consistent S› ‹maximal S›
    and ‹⊥ ∈ S›
  shows ‹p ∈ S›
```

- If $\bot \in S$,
  - $A \vdash \bot$ by *presence*,
  - $A \vdash p$ by the elimination rule,
  - $p \in S$ by *presence*.

# Reflecting Negation

- Assume more concrete syntax and proof rules:

```
  fixes not :: ‹'fm ⇒ 'fm› (‹¬›)
  assumes notI: ‹⋀A p. p # A ⊢ ⊥ ⟹ A ⊢ ¬ p›
      and notE: ‹⋀A p. A ⊢ p ⟹ A ⊢ ¬ p ⟹ A ⊢ ⊥›
```

- We can mirror these too:

```
corollary MCS_not_xor:
  assumes ‹consistent S› ‹maximal S›
  shows ‹p ∈ S ⟷ ¬ p ∉ S›
```

- if $p \in S$ and $\neg p \in S$, then, as above, $\bot \in S$ by the elimination rule. ↯
- if $\neg p \notin S$ and $p \notin S$, then $p, A$ explosive by *absence*, so $p, A \vdash \bot$,
  - $A \vdash \neg p$, by the introduction rule,
  - $\neg p \in S$, by *presence*. ↯

# Reflecting Quantifiers

- Assume an existential quantifier with a method of instantiation:

```
fixes exi :: ‹'fm ⇒ 'fm› (‹∃›)
  and inst :: ‹'t ⇒ 'fm ⇒ 'fm› (‹⟨_⟩›)
assumes
  exi_witness: ‹⋀S S' p. MCS S ⟹ witness (∃p) S' ⊆ S ⟹ ∃t. ⟨t⟩p ∈ S› and
  exiI: ‹⋀A p t. A ⊢ ⟨t⟩p ⟹ A ⊢ ∃p›
```

- We can lift this too:

```
corollary MCS_exi:
  assumes ‹consistent S› ‹maximal S› ‹witnessed S›
  shows ‹∃p ∈ S ⟷ (∃t. ⟨t⟩p ∈ S)›
```

- if $\exists p \in S$, then the Lindenbaum construction gives us $p(t) \in S$.
- if $p(t) \in S$, then *presence* + introduction rule gives us $\exists p \in S$.

# Example

- With *absence* and *presence* we can move between proof system and MCS
  - The framework provides some existing results
  - Users can prove their own in the same way
- Flexible enough for hybrid logic global modality $\mathbf{A}$
  - $M, - \models \mathbf{A}p \longleftrightarrow \forall w.\ M, w \models p$
  - Calculus labels formulas with world-naming nominals $(i, p)$
  - Treat as a universal quantifier
    - Formation preserves context: `‹λ(i, p). (i, A p)›`
    - Instantiation switches world: `‹λk (i, p). (k, p)›`
  - Reflection yields $(i, \mathbf{A}p) \in S \longleftrightarrow \forall k.\ ((k, p) \in S)$

# Truth Lemma

# Generalized Semantics

- We need a syntax-independent handle on sub-formulas

**Take your semantics**

$$M \vDash (\varphi \wedge \psi) \longleftrightarrow M \vDash \varphi \text{ and } M \vDash \psi$$

**Abstract the recursion**

$$M \llbracket \vDash \rrbracket (\varphi \wedge \psi) \longleftrightarrow M \vDash \varphi \text{ and } M \vDash \psi$$

**Plug in arbitrary relation**

$$M \llbracket \mathcal{R} \rrbracket (\varphi \wedge \psi) \longleftrightarrow \mathcal{R}(M, \varphi) \text{ and } \mathcal{R}(M, \psi)$$

- Now $- \llbracket \mathcal{R} \rrbracket -$ applies $\mathcal{R}$ with respect to the semantics

# Calculating Saturated Set Clauses

- Add our MCS $S$ to the mix:

$$\mathcal{R}(S)(-, \varphi) \equiv \varphi \in S$$

- We can write the truth lemma clause abstractly:

$$(\varphi \wedge \psi) \in S \longleftrightarrow \varphi \in S \text{ and } \psi \in S$$
$$\mathcal{R}(S)(M, \varphi \wedge \psi) \longleftrightarrow \mathcal{R}(S)(M, \varphi) \text{ and } \mathcal{R}(S)(M, \psi)$$
$$\mathcal{R}(S)(M, \varphi \wedge \psi) \longleftrightarrow M \, [\![\mathcal{R}(S)]\!] \, (\varphi \wedge \psi)$$

- In general:

$$\mathcal{R}(S)(M, \varphi) \longleftrightarrow M \, [\![\mathcal{R}(S)]\!] \, \varphi$$

# What is this $M$ Doing Here

- This is too naive

$$\mathcal{R}(S)(-,\varphi) \equiv \varphi \in S$$

- For first-order logic
  - Account for *quantifiers*
  - $\mathcal{R}(S)(M,\varphi) \equiv M(\varphi) \in S$
  - $\forall p \in S \longleftrightarrow \forall t\,.\ (p(t) \in S)$ (rather than ... $\forall t\,.\ (p \in S)$)
- For hybrid logic
  - Account for *current world*
  - $\mathcal{R}(S)(M,\varphi) \equiv (M_i,\varphi) \in S$
  - $(i, \Diamond p) \in S \longleftrightarrow \exists k\,.\ ((k,p) \in S)$ (rather than ... $\exists k\,.\ ((i,p) \in S)$)

# Abstract Saturation

- In the abstract

```
locale Truth_Base =
  fixes semics :: ‹'model ⇒ ('model ⇒ 'fm ⇒ bool) ⇒ 'fm ⇒ bool› (‹(_ ⟦_⟧ _)›)
    and semantics :: ‹'model ⇒ 'fm ⇒ bool› (infix ‹⊨› 50)
    and 𝕄 :: ‹'a set ⇒ 'model set›
    and ℛ :: ‹'a set ⇒ 'model ⇒ 'fm ⇒ bool›
```

- The generalization is faithful

```
  assumes semics_semantics: ‹M ⊨ p ⟷ M ⟦(⊨)⟧ p›
```

- The equation from before (allowing multiple models per MCS)

```
abbreviation saturated :: ‹'a set ⇒ bool› where
  ‹saturated S ≡ ∀p. ∀M ∈ 𝕄(S). M ⟦ℛ(S)⟧ p ⟷ ℛ(S) M p›
```

# Carving the Truth Lemma in Two

- Saturated sets model their members and MCS are saturated

```
assumes saturated_semantics:
    ‹⋀S M p. saturated S ⟹ M ∈ 𝕄(S) ⟹ M ⊨ p ⟷ ℛ(S) M p›
  and MCS_saturated:
    ‹⋀S. MCS S ⟹ saturated S›
```

- Therefore, MCSs model their members

```
theorem truth_lemma:
  assumes ‹MCS S› ‹M ∈ 𝕄(S)›
  shows ‹M ⊨ p ⟷ ℛ(S) M p›
```

- Isabelle/HOL generates the concrete proof obligations
  - Factors out semantic part for the automation
  - Leaves the syntactic part related to reflection

# All Together Now

- First-order logic
  - Saturation asks: $(\exists p) \in S \longleftrightarrow \exists t.\ (p(t) \in S)$
  - Reflection gave: $(\exists p) \in S \longleftrightarrow \exists t.\ (p(t) \in S)$
- Global modality
  - Saturation asks: $([i], \mathbf{A}p) \in S \longleftrightarrow \forall k.\ (([k], p) \in S)$
  - Reflection gave: $(i, \mathbf{A}p) \in S \longleftrightarrow \forall k.\ ((k, p) \in S)$
    - **sledgehammer** deals with equivalence classes
- Satisfaction operator
  - Saturation asks: $([i], @_k p) \in S \longleftrightarrow (([k], p) \in S)$
  - Reflection gave ??
    - **sledgehammer** finds a proof with *absence*, *presence* and proof rules

# In Summary

- Framework helps:
  - Build MCSs
  - Reflect proof system in MCSs
  - Carve up the truth lemma
- Artifact [4] includes strong completeness for
  - Propositional sequent calculus and tableau
  - Axiomatic modal logic
  - Natural deduction systems for first-order logic and hybrid logic
- Limitations
  - Compactness
  - To go beyond MCSs: general cut

# Related and Future Work

- J. C. Blanchette, A. Popescu, and D. Traytel [5] mechanize an analytic framework
  - abstract derivation trees instead of MCSs
  - more operational perspective with possible prover generation
- M. Petria [6] uses category theory to abstract over syntax and semantics
  - assume compactness and cut, and build MCSs
  - not mechanized
- M. Fitting [7] uses *consistency properties* to abstract over proof systems
  - This specifies the *reflected proof rules* up front
  - S. Berghofer [3] mechanized completeness for a concrete first-order logic
  - I am working on a generalization using Smullyan's uniform notation [1]
    - works for strong hybrid logic

# Abridged Bibliography


[1] R. M. Smullyan, *First-Order Logic*. Mineola, New York: Dover Publications, 1995.

[2] C. C. Chang and H. J. Keisler, *Model theory, Third Edition*, vol. 73. in Studies in logic and the foundations of mathematics, vol. 73. North-Holland, 1992.

[3] S. Berghofer, “First-Order Logic According to Fitting,” *Archive of Formal Proofs*, Aug. 2007.

[4] A. H. From, “Formalization for An Isabelle/HOL Framework for Synthetic Completeness Proofs .” [Online]. Available: https://doi.org/10.5281/zenodo.14278854

[5] J. C. Blanchette, A. Popescu, and D. Traytel, “Soundness and Completeness Proofs by Coinductive Methods,” *Journal of Automated Reasoning*, vol. 58, no. 1, pp. 149–179, 2017, doi: 11.1007/s10817-016-9391-3.

[6] M. Petria, “An Institutional Version of Gödel's Completeness Theorem,” in *Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Proceedings*, T. Mossakowski, U. Montanari, and M. Haveraaen, Eds., in Lecture Notes in Computer Science, vol. 4624. Springer, 2007, pp. 409–424. doi: 10.1007/978-3-540-73859-6_28.

[7] M. Fitting, *First-Order Logic and Automated Theorem Proving, Second Edition*. in Graduate Texts in Computer Science. Springer, 1996. doi: 10.1007/978-1-4612-2360-3.