Completeness of Second-Order Logic for Henkin Semantics

Second Bachelor Seminar Talk

Mark Koch
Advisor: Dominik Kirst
Supervisor: Gert Smolka

July 5, 2021

Saarland University, Programming Systems Lab

Recap

Recap

Second-order logic is incomplete

i.e. there is no deduction system that is complete, sound and enumerable

Recap

Second-order logic is incomplete

i.e. there is no deduction system that is complete, sound and enumerable

$$\frac{A[\uparrow^n_p] \vdash_2 \varphi}{A \vdash_2 \dot{\forall}^n_p\, \varphi}\ \mathsf{AI_p} \qquad \frac{A \vdash_2 \dot{\forall}^n_p\, \varphi}{A \vdash_2 \varphi[P]}\ \mathsf{AE_p}$$

$$\frac{A \vdash_2 \varphi[P]}{A \vdash_2 \dot{\exists}^n_p\, \varphi}\ \mathsf{EI_p} \qquad \frac{A \vdash_2 \dot{\exists}^n_p\, \varphi \qquad A[\uparrow^n_p], \varphi \vdash_2 \psi[\uparrow^n_p]}{A \vdash_2 \psi}\ \mathsf{EE_p}$$

Recap

Second-order logic is incomplete

i.e. there is no deduction system that is complete, sound and enumerable

$$\frac{A[\uparrow^n_p] \vdash_2 \varphi}{A \vdash_2 \dot\forall^n_p\, \varphi}\ \mathrm{AI_p} \qquad \frac{A \vdash_2 \dot\forall^n_p\, \varphi}{A \vdash_2 \varphi[P]}\ \mathrm{AE_p}$$

$$\frac{A \vdash_2 \varphi[P]}{A \vdash_2 \dot\exists^n_p\, \varphi}\ \mathrm{EI_p} \qquad \frac{A \vdash_2 \dot\exists^n_p\, \varphi \qquad A[\uparrow^n_p], \varphi \vdash_2 \psi[\uparrow^n_p]}{A \vdash_2 \psi}\ \mathrm{EE_p}$$

$$\frac{}{A \vdash_2 \dot\exists^n_p\, P.\, \dot\forall x_1 ... x_n.\, P(x_1, ..., x_2) \dot\leftrightarrow \varphi[\uparrow^n_p]}\ \mathrm{Compr}_p$$

Second-order logic is incomplete

i.e. there is no deduction system that is complete, sound and enumerable

$$\frac{A[\uparrow^n_p] \vdash_2 \varphi}{A \vdash_2 \dot{\forall}^n_p\, \varphi}\ \mathrm{AI_p} \qquad \frac{A \vdash_2 \dot{\forall}^n_p\, \varphi}{A \vdash_2 \varphi[P]}\ \mathrm{AE_p}$$

$$\frac{A \vdash_2 \varphi[P]}{A \vdash_2 \dot{\exists}^n_p\, \varphi}\ \mathrm{EI_p} \qquad \frac{A \vdash_2 \dot{\exists}^n_p\, \varphi \qquad A[\uparrow^n_p], \varphi \vdash_2 \psi[\uparrow^n_p]}{A \vdash_2 \psi}\ \mathrm{EE_p}$$

$$\frac{}{A \vdash_2 \dot{\exists}^n_p\, P.\, \dot{\forall} x_1 \dots x_n.\, P(x_1, \dots, x_2) \dot{\leftrightarrow} \varphi[\uparrow^n_p]}\ \mathrm{Compr}_p$$

$\vdash_2$ is incomplete, i.e. $\neg \forall A\, \varphi.\, A \vDash_2 \varphi \rightarrow A \vdash_2 \varphi$.

Recap

Second-order logic is incomplete

i.e. there is no deduction system that is complete, sound and enumerable (for standard semantics)

$$\frac{A[\uparrow^n_p] \vdash_2 \varphi}{A \vdash_2 \dot\forall^n_p \varphi}\ \mathrm{AI_p} \qquad \frac{A \vdash_2 \dot\forall^n_p \varphi}{A \vdash_2 \varphi[P]}\ \mathrm{AE_p}$$

$$\frac{A \vdash_2 \varphi[P]}{A \vdash_2 \dot\exists^n_p \varphi}\ \mathrm{EI_p} \qquad \frac{A \vdash_2 \dot\exists^n_p \varphi \qquad A[\uparrow^n_p], \varphi \vdash_2 \psi[\uparrow^n_p]}{A \vdash_2 \psi}\ \mathrm{EE_p}$$

$$\frac{}{A \vdash_2 \dot\exists^n_p P.\, \dot\forall x_1 \dots x_n.\, P(x_1, \dots, x_2) \dot\leftrightarrow \varphi[\uparrow^n_p]}\ \mathrm{Compr}_p$$

$\vdash_2$ is incomplete (for standard semantics), i.e. $\neg\forall A\, \varphi.\, A \vDash_2 \varphi \rightarrow A \vdash_2 \varphi$.

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}^n_p P.\, \varphi \;\sim$ There exists a predicate P such that φ holds.

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}_p^n P.\,\varphi \sim$ There exists a predicate P such that φ holds.

Now: $\dot{\exists}_p^n P.\,\varphi \sim$ There exists a predicate P in $\mathbb{P}_n$ such that φ holds.

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}^n_p P.\,\varphi \;\sim\;$ There exists a predicate P such that φ holds.

Now: $\dot{\exists}^n_p P.\,\varphi \;\sim\;$ There exists a predicate P in $\mathbb{P}_n$ such that φ holds.

Definition (Henkin Semantics)

A Henkin model $\mathcal{H}$ specifies a set of relations $\mathbb{P}_n : (D^n \to \text{Prop}) \to \text{Prop}$

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}^n_p\, P.\, \varphi \;\sim$ There exists a predicate P such that φ holds.

Now: $\dot{\exists}^n_p\, P.\, \varphi \;\sim$ There exists a predicate P in $\mathbb{P}_n$ such that φ holds.

Definition (Henkin Semantics)

A Henkin model $\mathcal{H}$ specifies a set of relations $\mathbb{P}_n : (D^n \to \text{Prop}) \to \text{Prop}$ that constrain the predicates that are quantified over, i.e

$$\mathcal{H} \vDash_\rho \dot{\exists}^n_p\, \varphi \;:=\; \exists P^{D^n \to \text{Prop}}.\, \mathbb{P}_n\, P \wedge \mathcal{H} \vDash_{P \cdot \rho} \varphi.$$

Henkin Semantics

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}^n_p P.\,\varphi \;\sim$ There exists a predicate P such that φ holds.

Now: $\dot{\exists}^n_p P.\,\varphi \;\sim$ There exists a predicate P in $\mathbb{P}_n$ such that φ holds.

Definition (Henkin Semantics)

A Henkin model $\mathcal{H}$ specifies a set of relations $\mathbb{P}_n : (D^n \to \mathrm{Prop}) \to \mathrm{Prop}$ that constrain the predicates that are quantified over, i.e

$$\mathcal{H} \vDash_\rho \dot{\exists}^n_p \varphi := \exists P^{D^n \to \mathrm{Prop}}.\, \mathbb{P}_n\, P \wedge \mathcal{H} \vDash_{P.\rho} \varphi.$$

$\mathbb{P}_n$ should satisfy comprehension, i.e. it must at least contain all second-order definable properties.

However, $\vdash_2$ is complete if one switches to Henkin semantics!

Before: $\dot{\exists}^n_p P.\,\varphi \sim$ There exists a predicate P such that φ holds.

Now: $\dot{\exists}^n_p P.\,\varphi \sim$ There exists a predicate P in $\mathbb{P}_n$ such that φ holds.

Definition (Henkin Semantics)

A Henkin model $\mathcal{H}$ specifies a set of relations $\mathbb{P}_n : (D^n \to \mathsf{Prop}) \to \mathsf{Prop}$ that constrain the predicates that are quantified over, i.e

$$\mathcal{H} \vDash_\rho \dot{\exists}^n_p \varphi := \exists P^{D^n \to \mathsf{Prop}}.\, \mathbb{P}_n\, P \land \mathcal{H} \vDash_{P \cdot \rho} \varphi.$$

$\mathbb{P}_n$ should satisfy comprehension, i.e. it must at least contain all second-order definable properties.

Functions are constrained in the same way via a relation $\mathbb{F}_n$.

Henkin Semantics

- A Henkin model is equivalent to a standard model if $\mathbb{F}$ and $\mathbb{P}$ contain everything.

Henkin Semantics

- A Henkin model is equivalent to a standard model if $\mathbb{F}$ and $\mathbb{P}$ contain everything.
- Henkin semantics allow to recover much of the first-order model theory. We are most interested in completeness.

Henkin Semantics

- A Henkin model is equivalent to a standard model if $\mathbb{F}$ and $\mathbb{P}$ contain everything.
- Henkin semantics allow to recover much of the first-order model theory. We are most interested in completeness.
- The usual Henkin style completeness proof would work [Shapiro, 1991], but we want to use a different approach:

Henkin Semantics

- A Henkin model is equivalent to a standard model if $\mathbb{F}$ and $\mathbb{P}$ contain everything.
- Henkin semantics allow to recover much of the first-order model theory. We are most interested in completeness.
- The usual Henkin style completeness proof would work [Shapiro, 1991], but we want to use a different approach:

> SOL with Henkin semantics reduces to (mono-sorted) FOL.

Translation

$$\forall x.\, \exists^2_p\, P.\, P(x, x)$$

Translation

$$\forall x.\, \exists^2_p P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathrm{predApp}_2(p, x, x)$$

Translation

$$\forall x.\, \exists_p^2\, P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathrm{predApp}_2(p, x, x)$$

Mono-sorted:

$$\forall x.\, \mathrm{isIndi}(x) \rightarrow \exists p.\, \mathrm{isPred}_2(p) \wedge \mathrm{predApp}_2(p, x, x)$$

Translation

$$\forall x.\, \exists^2_p\, P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathrm{predApp}_2(p, x, x)$$

Mono-sorted:

$$\forall x.\, \mathrm{isIndi}(x) \rightarrow \exists p.\, \mathrm{isPred}_2(p) \land \mathrm{predApp}_2(p, x, x)$$

- “Tedious but routine job” to verify mono-sorted reduction for deduction system according to textbook [Van Dalen, 1994].

Translation

$$\forall x.\, \exists^2_p\, P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathsf{predApp}_2(p, x, x)$$

Mono-sorted:

$$\forall x.\, \mathsf{isIndi}(x) \to \exists p.\, \mathsf{isPred}_2(p) \land \mathsf{predApp}_2(p, x, x)$$

- "Tedious but routine job" to verify mono-sorted reduction for deduction system according to textbook [Van Dalen, 1994].
- More difficult than it seems. Nour and Raffalli "do not know how to end his proof" [Nour and Raffalli, 2003].

Translation

$$\forall x.\, \exists^2_p P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathrm{predApp}_2(p, x, x)$$

Mono-sorted:

$$\cancel{\forall x.\, \mathrm{isIndi}(x) \to \exists p.\, \mathrm{isPred}_2(p) \wedge \mathrm{predApp}_2(p, x, x)}$$

- "Tedious but routine job" to verify mono-sorted reduction for deduction system according to textbook [Van Dalen, 1994].
- More difficult than it seems. Nour and Raffalli "do not know how to end his proof" [Nour and Raffalli, 2003]. They propose a simpler reduction:

$$\forall x.\, \exists p.\, \mathrm{predApp}_2(p, x, x)$$

Translation

$$\forall x.\, \exists_p^2 P.\, P(x, x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathrm{predApp}_2(p, x, x)$$

Mono-sorted:

$$\sout{\forall x.\, \mathrm{isIndi}(x) \to \exists p.\, \mathrm{isPred}_2(p) \wedge \mathrm{predApp}_2(p, x, x)}$$

- "Tedious but routine job" to verify mono-sorted reduction for deduction system according to textbook [Van Dalen, 1994].
- More difficult than it seems. Nour and Raffalli "do not know how to end his proof" [Nour and Raffalli, 2003]. They propose a simpler reduction:

$$\forall x.\, \exists p.\, \mathrm{predApp}_2(p, x, x)$$

$\Rightarrow$ x and p represent individual, function, and predicate **at the same time**!

Translation

$$\forall x.\, \exists^2_p P.\, P(x,x)$$

Many-sorted (easy):

$$\forall x^{\mathcal{I}}.\, \exists p^{\mathcal{P}_2}.\, \mathsf{predApp}_2(p,x,x)$$

Mono-sorted:

$$\sout{\forall x.\, \mathsf{isIndi}(x) \to \exists p.\, \mathsf{isPred}_2(p) \wedge \mathsf{predApp}_2(p,x,x)}$$

- "Tedious but routine job" to verify mono-sorted reduction for deduction system according to textbook [Van Dalen, 1994].
- More difficult than it seems. Nour and Raffalli "do not know how to end his proof" [Nour and Raffalli, 2003]. They propose a simpler reduction:

$$\forall x.\, \exists p.\, \mathsf{predApp}_2(p,x,x)$$

$\Rightarrow$ x and p represent individual, function, and predicate **at the same time**!

Define translation function $\_^\star : \mathsf{form}_2(\Sigma) \to \mathsf{form}_1(\Sigma_+)$.

Henkin to First-Order Model

Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:

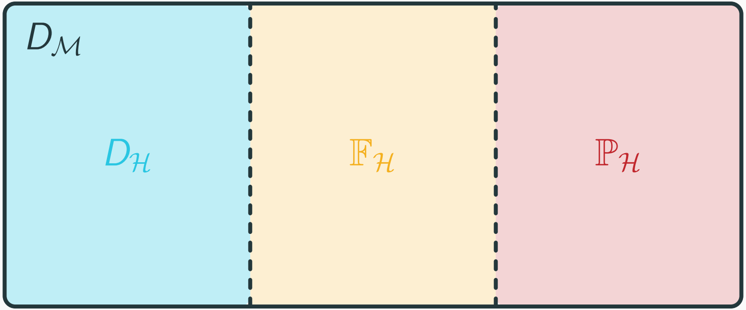
$D_{\mathcal{H}}$

$\mathbb{F}_{\mathcal{H}}$

$\mathbb{P}_{\mathcal{H}}$

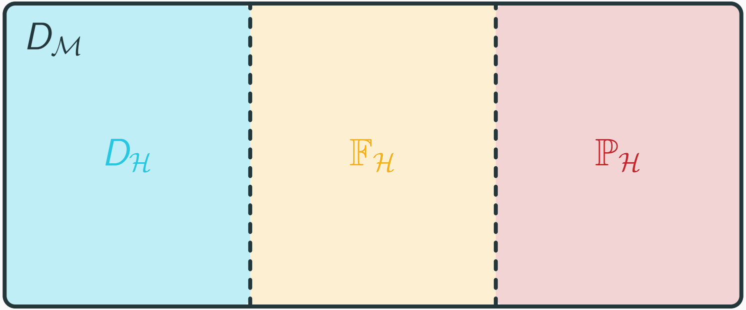
Henkin to First-Order Model

Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



Henkin to First-Order Model

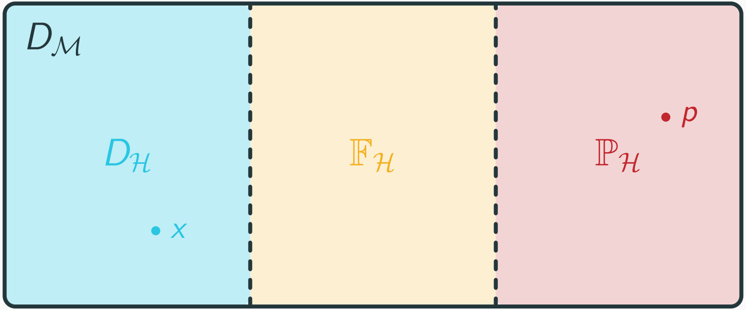
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) \;:=$$

Henkin to First-Order Model

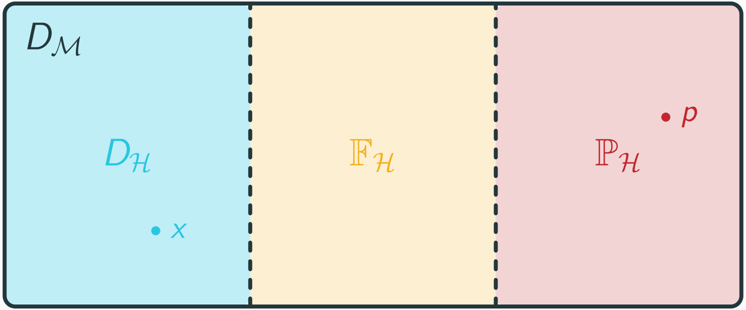
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\mathrm{predApp}_2^{\mathcal{M}}(p, x, x) \ :=$$

Henkin to First-Order Model

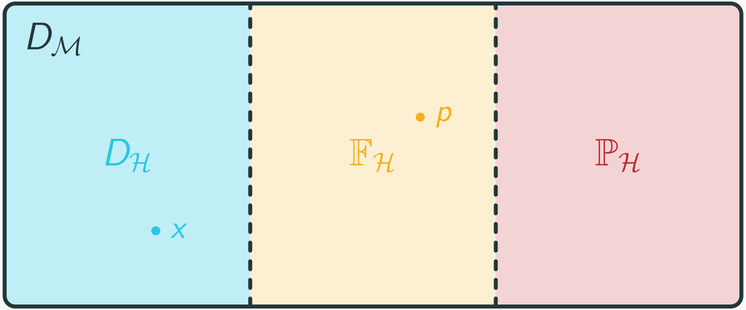
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) \; := \; p(x, x)$$

Henkin to First-Order Model

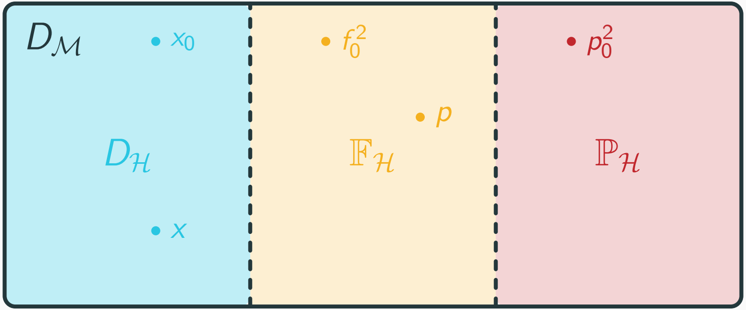
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) \; :=$$

Henkin to First-Order Model

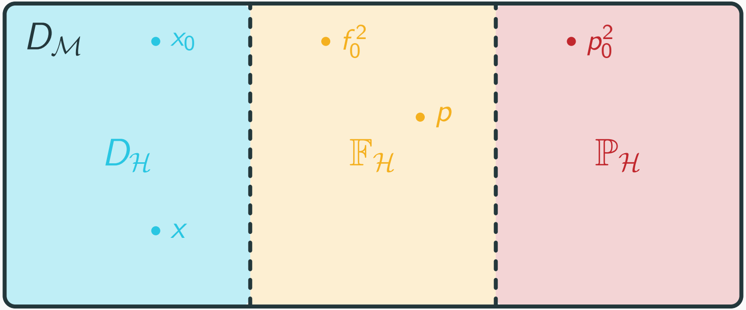
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) \ :=$$

Henkin to First-Order Model

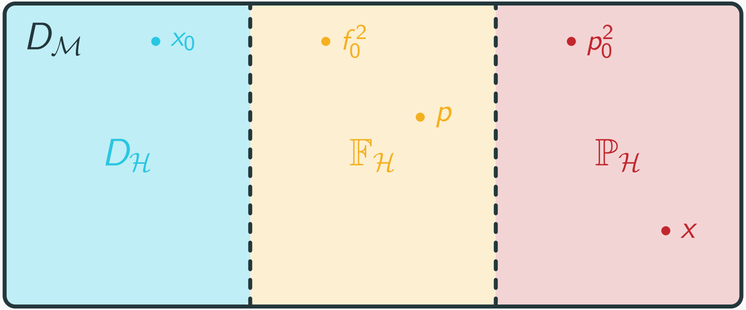
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) := p_0^2(x, x)$$

Henkin to First-Order Model

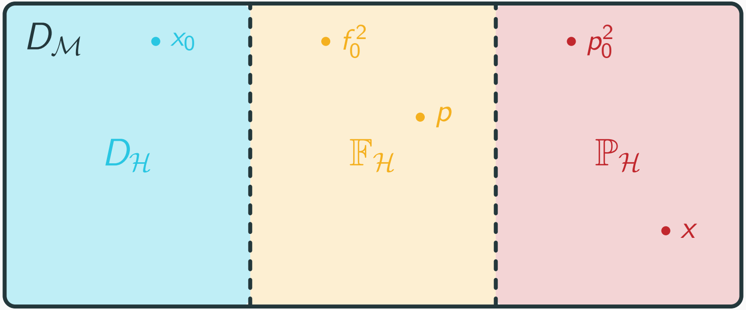
Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$$\text{predApp}_2^{\mathcal{M}}(p, x, x) \;:=\; p_0^2(x_0, x_0)$$

Henkin to First-Order Model

Convert Henkin model $\mathcal{H}$ to first-order model $\mathcal{M}$:



$\mathcal{H} \vDash \varphi \;\leftrightarrow\; \mathcal{M} \vDash \varphi^\star$ for all closed φ

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n \; P \;\; := \;\; \exists p. \forall x_1 ... x_n. \, P\,(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}\,(p, x_1, ..., x_2)$

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n\ P \ := \ \exists p. \forall x_1 ... x_n. P\,(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}\,(p, x_1, ..., x_2)$

  For standard semantics, every predicate would need to be included. But we have no guarantee that $\mathcal{M}$ contains all predicates.

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n \; P \;\; := \;\; \exists p. \forall x_1 ... x_n. \, P\,(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}\,(p, x_1, ..., x_2)$

  For standard semantics, every predicate would need to be included. But we have no guarantee that $\mathcal{M}$ contains all predicates.
- $\mathbb{P}_n$ must have comprehension.

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n\ P \ := \ \exists p. \forall x_1 ... x_n. P(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}(p, x_1, ..., x_2)$

  For standard semantics, every predicate would need to be included. But we have no guarantee that $\mathcal{M}$ contains all predicates.
- $\mathbb{P}_n$ must have comprehension. This holds if $\mathcal{M}$ has comprehension.

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n\ P \ := \ \exists p. \forall x_1 ... x_n. P(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}(p, x_1, ..., x_2)$

  For standard semantics, every predicate would need to be included. But we have no guarantee that $\mathcal{M}$ contains all predicates.
- $\mathbb{P}_n$ must have comprehension. This holds if $\mathcal{M}$ has comprehension.
  $\Rightarrow$ Encode this requirement in a theory $\mathcal{C}$.

First-Order to Henkin Model

Convert first-order model $\mathcal{M}$ to Henkin model $\mathcal{H}$:

- $D_{\mathcal{H}} := D_{\mathcal{M}}$
- $\mathbb{P}_n\ P \ := \ \exists p. \forall x_1 ... x_n. P(x_1, ..., x_n) \leftrightarrow \text{predApp}_n^{\mathcal{M}}(p, x_1, ..., x_2)$

  For standard semantics, every predicate would need to be included. But we have no guarantee that $\mathcal{M}$ contains all predicates.
- $\mathbb{P}_n$ must have comprehension. This holds if $\mathcal{M}$ has comprehension.
  $\Rightarrow$ Encode this requirement in a theory $\mathcal{C}$.

$$\mathcal{M} \vDash \mathcal{C} \ \rightarrow \ (\mathcal{H} \vDash \varphi \ \leftrightarrow \ \mathcal{M} \vDash \varphi^{\star}) \quad \text{for all closed } \varphi$$

Reduction

Theorem

We can reduce Henkin validity to first-order validity. For closed second-order formulas φ and theories $\mathcal{T}$ it holds that

$$\mathcal{T} \vDash_2 \varphi \;\leftrightarrow\; \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star.$$

Reduction

Theorem

We can reduce Henkin validity to first-order validity. For closed second-order formulas φ and theories $\mathcal{T}$ it holds that

$$\mathcal{T} \vDash_2 \varphi \;\leftrightarrow\; \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star.$$

This suffices to show that there exists a sound, complete and enumerable deduction system for SOL. Simply define

$$\mathcal{T} \vdash_2' \varphi \;:=\; \mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star$$

Reduction

Theorem

We can reduce Henkin validity to first-order validity. For closed second-order formulas φ and theories $\mathcal{T}$ it holds that

$$\mathcal{T} \vDash_2 \varphi \;\leftrightarrow\; \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star.$$

This suffices to show that there exists a sound, complete and enumerable deduction system for SOL. Simply define

$$\mathcal{T} \vdash_2' \varphi \;:=\; \mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star$$

But we want to show our ND system $\vdash_2$ complete. This is the hard part:

$$\mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star \;\rightarrow\; \mathcal{T} \vdash_2 \varphi$$

Reduction

Theorem

We can reduce Henkin validity to first-order validity. For closed second-order formulas φ and theories $\mathcal{T}$ it holds that

$$\mathcal{T} \vDash_2 \varphi \ \leftrightarrow\ \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star.$$

This suffices to show that there exists a sound, complete and enumerable deduction system for SOL. Simply define

$$\mathcal{T} \vdash_2' \varphi \ :=\ \mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star$$

But we want to show our ND system $\vdash_2$ complete. This is the hard part:

$$\mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star \ \rightarrow\ \mathcal{T} \vdash_2 \varphi$$

From this point on, we only work in the SOL fragment without function quantifiers and variables!

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathsf{form}_1(\Sigma_+) \to \mathsf{form}_2(\Sigma)$.

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathsf{form}_1(\Sigma_+) \to \mathsf{form}_2(\Sigma)$. For example

$$\forall x.\, \mathsf{predApp}_0(x) \,\dot{\wedge}\, \mathsf{predApp}_1(x, x)$$

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathrm{form}_1(\Sigma_+) \to \mathrm{form}_2(\Sigma)$. For example

$$(\forall x.\, \mathrm{predApp}_0(x) \mathbin{\dot\wedge} \mathrm{predApp}_1(x, x))^{\diamond}$$

$$||$$

$$x_p^0 \mathbin{\dot\wedge} x_p^1(x_i)$$

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathrm{form}_1(\Sigma_+) \to \mathrm{form}_2(\Sigma)$. For example

$$(\forall x.\ \mathrm{predApp}_0(x) \mathbin{\dot\wedge} \mathrm{predApp}_1(x, x))^{\diamond}$$

$$||$$

$$\forall x_i.\ \forall^0_p\, x^0_p.\ \forall^1_p\, x^1_p.\ x^0_p \mathbin{\dot\wedge} x^1_p(x_i)$$

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathrm{form}_1(\Sigma_+) \to \mathrm{form}_2(\Sigma)$. For example

$$(\forall x.\ \mathrm{predApp}_0(x) \dot{\wedge} \mathrm{predApp}_1(x, x))^{\diamond}$$

$$||$$

$$\forall x_i.\ \forall^0_p\, x^0_p.\ \forall^1_p\, x^1_p.\ x^0_p \dot{\wedge} x^1_p(x_i)$$

$$(\mathrm{predApp}_1(f(x), y))^{\diamond} =$$

Backwards Translation

Define a backwards translation $\_^{\diamond} : \mathrm{form}_1(\Sigma_+) \to \mathrm{form}_2(\Sigma_{\mathrm{err}})$. For example

$$(\forall x.\, \mathrm{predApp}_0(x) \dot{\wedge} \mathrm{predApp}_1(x, x))^{\diamond}$$

$$||$$

$$\forall x_i.\, \forall_p^0\, x_p^0.\, \forall_p^1\, x_p^1.\, x_p^0 \;\dot{\wedge}\; x_p^1(x_i)$$

$$(\mathrm{predApp}_1(f(x), y))^{\diamond} = \mathrm{Err}_1(y_i)$$

Special error symbol if first argument is not a variable

Completeness

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^\diamond \vdash_2 \;\varphi^\diamond$

Completeness

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^{\diamond} \vdash_2 \varphi^{\diamond}$
2. $\vdash_2 \varphi^{\star\diamond} \mathrel{\dot{\leftrightarrow}} \varphi$

Completeness

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^\diamond \vdash_2 \;\; \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \dot{\leftrightarrow} \varphi$

$\mathcal{T} \vDash_2 \varphi$

Completeness

Lemma

1. $A \vdash_1 \varphi \ \rightarrow \ A^\diamond \vdash_2 \ \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \dot{\leftrightarrow} \varphi$

$$\mathcal{T} \vDash_2 \varphi \quad \longleftrightarrow \quad \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star$$

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^{\diamond} \vdash_2 \varphi^{\diamond}$
2. $\vdash_2 \varphi^{\star\diamond} \mathrel{\dot{\leftrightarrow}} \varphi$

$$\mathcal{T} \vDash_2 \varphi \;\longleftrightarrow\; \mathcal{T}^{\star}, \mathcal{C} \vDash_1 \varphi^{\star}$$

FOL Completeness
[Forster et al., 2021]

$\downarrow$ MP/LEM

$$\mathcal{T}^{\star}, \mathcal{C} \vdash_1 \varphi^{\star}$$

Completeness

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^{\diamond} \vdash_2 \varphi^{\diamond}$
2. $\vdash_2 \varphi^{\star\diamond} \mathrel{\dot{\leftrightarrow}} \varphi$

$$
\begin{array}{ccc}
\mathcal{T} \vDash_2 \varphi & \longleftrightarrow & \mathcal{T}^{\star}, \mathcal{C} \vDash_1 \varphi^{\star} \\
 & \text{FOL Completeness [Forster et al., 2021]} \;\Big\downarrow\; \text{MP/LEM} & \\
 & & \mathcal{T}^{\star}, \mathcal{C} \vdash_1 \varphi^{\star} \xrightarrow{(1)} \mathcal{T}^{\star\diamond}, \mathcal{C}^{\diamond} \vdash_2 \varphi^{\star\diamond}
\end{array}
$$

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^\diamond \vdash_2^{\mathsf{err}} \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \dot{\leftrightarrow} \varphi$

$$\mathcal{T} \vDash_2 \varphi \longleftrightarrow \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star$$

FOL Completeness [Forster et al., 2021]

MP/LEM

$$\mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star \xrightarrow{(1)} \mathcal{T}^{\star\diamond}, \mathcal{C}^\diamond \vdash_2^{\mathsf{err}} \varphi^{\star\diamond}$$

Error symbol can occur in this derivation!

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^\diamond \vdash_2^{\mathsf{err}} \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \mathrel{\dot\leftrightarrow} \varphi$

$\mathcal{T} \vDash_2 \varphi \longleftrightarrow \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star$

FOL Completeness [Forster et al., 2021] ↓ MP/LEM

$\mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star \xrightarrow{(1)} \mathcal{T}^{\star\diamond}, \mathcal{C}^\diamond \vdash_2^{\mathsf{err}} \varphi^{\star\diamond}$

Error symbol can occur in this derivation!

Remove error symbol using comprehension ↓

$\mathcal{T}^{\star\diamond}, \mathcal{C}^\diamond \vdash_2 \varphi^{\star\diamond}$

Completeness

Lemma

1. $A \vdash_1 \varphi \;\to\; A^\diamond \vdash_2^{\mathsf{err}} \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \dot\leftrightarrow \varphi$

$\mathcal{T} \vDash_2 \varphi \longleftrightarrow \mathcal{T}^\star, \mathcal{C} \vDash_1 \varphi^\star$

FOL Completeness [Forster et al., 2021]

MP/LEM

$\downarrow$

Error symbol can occur in this derivation!

$\mathcal{T}^\star, \mathcal{C} \vdash_1 \varphi^\star \xrightarrow{(1)} \mathcal{T}^{\star\diamond}, \mathcal{C}^\diamond \vdash_2^{\mathsf{err}} \varphi^{\star\diamond}$

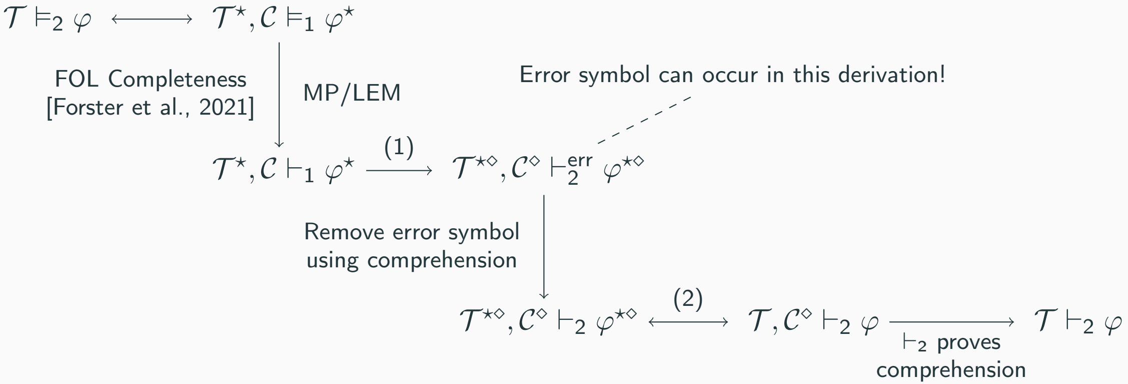
Remove error symbol using comprehension

$\downarrow$

$\mathcal{T}^{\star\diamond}, \mathcal{C}^\diamond \vdash_2 \varphi^{\star\diamond} \xleftrightarrow{(2)} \mathcal{T}, \mathcal{C}^\diamond \vdash_2 \varphi$

Lemma

1. $A \vdash_1 \varphi \;\rightarrow\; A^{\diamond} \vdash_2^{\mathrm{err}} \varphi^{\diamond}$
2. $\vdash_2 \varphi^{\star\diamond} \dot{\leftrightarrow} \varphi$



Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \to \mathcal{T} \vdash_2 \varphi.$$

Conclusion

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \rightarrow \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.

Conclusion

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \to \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.
- Deductive part fairly tedious to mechanize (not finished yet)

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \to \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.
- Deductive part fairly tedious to mechanize (not finished yet)

Overall, the Bachelor's project contributes the first mechanization of SOL, including:

Conclusion

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \rightarrow \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.
- Deductive part fairly tedious to mechanize (not finished yet)

Overall, the Bachelor's project contributes the first mechanization of SOL, including:

- Categoricity of PA_2

Conclusion

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \to \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.
- Deductive part fairly tedious to mechanize (not finished yet)

Overall, the Bachelor's project contributes the first mechanization of SOL, including:

- Categoricity of PA_2
- Undecidability and incompleteness for standard semantics

Conclusion

Theorem (Completeness)

For closed φ and $\mathcal{T}$ without function quantifiers and variables it holds that

$$\mathcal{T} \vDash_2 \varphi \rightarrow \mathcal{T} \vdash_2 \varphi.$$

- Semantic reduction straightforward. Also allows to obtain Compactness, Löwenheim-Skolem, etc. from FOL.
- Deductive part fairly tedious to mechanize (not finished yet)

Overall, the Bachelor's project contributes the first mechanization of SOL, including:

- Categoricity of PA_2
- Undecidability and incompleteness for standard semantics
- Reduction to mono-sorted FOL and completeness for Henkin semantics

Forster, Y., Kirst, D., and Wehr, D. (2021).
Completeness theorems for first-order logic analysed in constructive type theory: Extended version.
Journal of Logic and Computation, 31(1):112–151.

Nour, K. and Raffalli, C. (2003).
Simple proof of the completeness theorem for second-order classical and intuitionistic logic by reduction to first-order mono-sorted logic.
Theoretical computer science, 308(1-3):227–237.

Shapiro, S. (1991).
***Foundations without foundationalism: A case for second-order logic*, volume 17.**
Clarendon Press.

Van Dalen, D. (1994).
***Logic and structure*, volume 3.**
Springer.

Coq mechanization

Overview	LOC
Utility	300
Syntax & Substitutions	900
Tarski Semantics	1000
Deduction System	900
PA & Categoricity	1200
Undec. & Incompleteness	400
Henkin Semantics	200
FOL Reduction	1300
Total	6200