

Classical Higher-Order Logic (HOL)

Automation and Selected Applications

Christoph Benzmüller

U Bamberg & FU Berlin

*"If we had it [a *characteristica universalis*], we should be able to reason in metaphysics and morals in much the same way as in geometry and analysis."*

(Leibniz, 1690; translation by Russell)

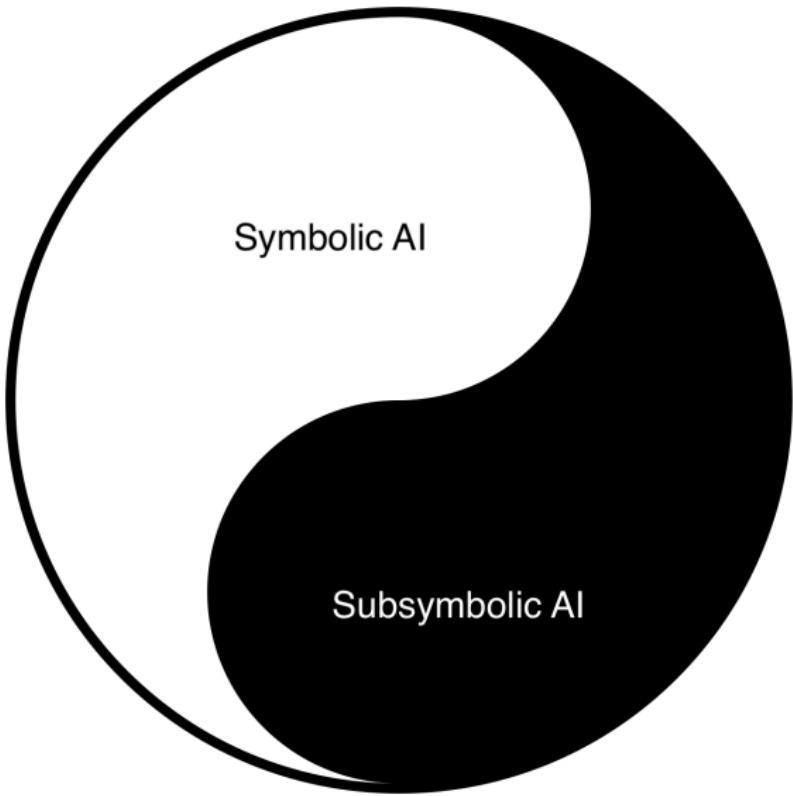
Overview

- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ Non-Classical Logics — Why Classical Logic is not enough
- ▶ Universal Logical Reasoning — Why Classical Logic is enough
- ▶ Application Direction: Ethico-Legal Governors for AI

Overview

- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ Non-Classical Logics — Why Classical Logic is not enough
- ▶ Universal Logical Reasoning — Why Classical Logic is enough
- ▶ Application Direction: Ethico-Legal Governors for AI

Yin and Yang of AI — Own Position

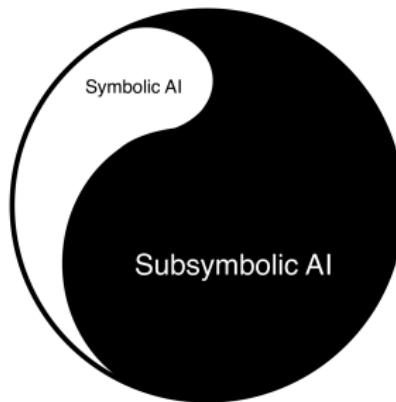


Yin and Yang of AI — Unhealthy Hype!

Examples of success

- ▶ ...
- ▶ SAT-Solver:
solution of open
maths problems
- ▶ ...

(symbolic AI)



Examples of success

- ▶ ...
- ▶ AlphaGo &
AlphaZero: world
champion Chess
and Go
- ▶ ...

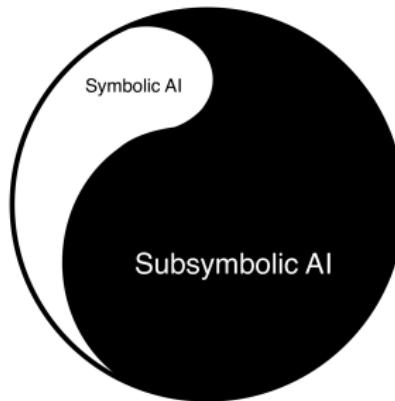
(subsymbolic KI)



Yin and Yang of AI — Unhealthy Hype!

Examples of success

- ▶ ...
 - ▶ SAT-Solver:
solution of open
maths problems
 - ▶ ...
- (symbolic AI)



Examples of success

- ▶ ...
 - ▶ AlphaGo &
AlphaZero: world
champion Chess
and Go
 - ▶ ...
- (subsymbolic KI)

nature

International weekly journal of science

[Home](#) | [News & Comment](#) | [Research](#) | [Careers & Jobs](#) | [Current Issue](#) | [Archive](#) | [Audio & Video](#) | [For Authors](#)

[Archive](#) > [Volume 534](#) > [Issue 7605](#) > [News](#) > [Article](#)

NATURE | NEWS

Two-hundred-terabyte maths proof is largest ever

A computer cracks the Boolean Pythagorean triples problem — but is it really maths?

Evelyn Lamb

26 May 2016



Another Interesting Story — Formal Logic in Maths

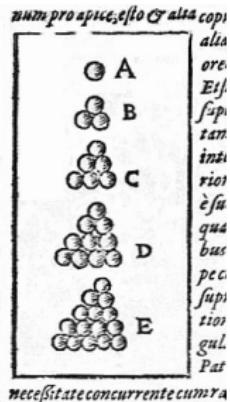


Kepler (1571-1630)



The most compact way of stacking balls of the same size in space is a pyramid.

$$V_{\text{pyramid}} = \frac{\pi}{\sqrt{18}} \approx 74\%$$



- ▶ Proved in 1998 by Hales: 300 page proof, with code and data
- ▶ Submitted to the Annals of Mathematics: referees gave up to verify it all
- ▶ **Flyspeck project (completed in 2014):**
 - ▶ A formal verification of the proof in HOL Light
 - ▶ 27,223 proved theorems, 228 definitions, **30 person-years**
- ▶ **Work of e.g. Cezary Kalyszik & Josef Urban:**
Automated Theorem Proving assisted by Machine Learning
Result: **more than 65% of the proofs can actually be fully automated**

Yin and Yang of AI — The Next (really) Big Thing?!

Causalities

Abstraction

Precise Reasoning

Explain-/Verifiability

...

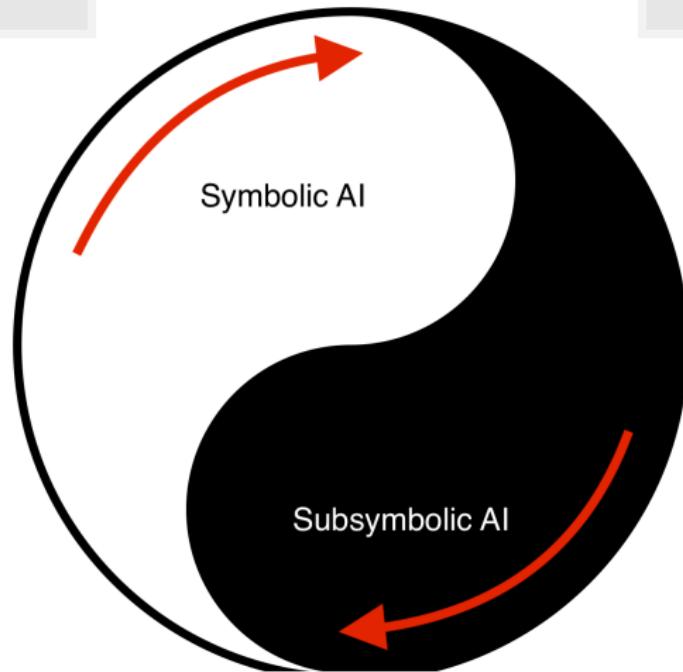
Correlations

Patterns

Robustness

Learning

...



Spectrum of Own Research & Interests

Since more than two decades

- ▶ Automation of HOL: Leo Provers & TPTP Infrastructure
- ▶ Proof Assistants: Interaction & Automation
- ▶ Traditional Applications: Formal Methods in Maths, CS/AI

More recently

- ▶ Universal Logical Reasoning in HOL
- ▶ Novel Applications: Computational Metaphysics
Foundations (e.g. Category Theory)
Ethics & Law

Overview

- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ Non-Classical Logics — Why Classical Logic is not enough
- ▶ Universal Logical Reasoning — Why Classical Logic is enough
- ▶ Application Direction: Ethico-Legal Governors for AI

Classical Higher-Order Logic (HOL)

Expressivity	FOL	HOL	Example
Quantification over			
- Individuals	✓	✓	$\forall \textcolor{teal}{X} p(f(\textcolor{teal}{X}))$
- Functions	✗	✓	$\forall F p(F(a))$
- Predicates/Sets/Relns	✗	✓	$\forall P P(f(a))$
Unnamed			
- Functions	✗	✓	$\lambda X X$
- Predicates	✗	✓	$\lambda X (X = X)$
Statements about			
- Funcs/Preds/Sets/Relns	✗	✓	<i>reflexive</i> \leftrightarrow
Powerful definitions	✗	✓	<i>reflexive</i> := $\lambda R \forall X (R X X)$

Classical Higher-Order Logic (HOL)

Expressivity	FOL	HOL	Example
Quantification over			
- Individuals	✓	✓	$\forall \textcolor{teal}{X} p(f(\textcolor{teal}{X}))$
- Functions	✗	✓	$\forall \textcolor{teal}{F} p(\textcolor{teal}{F}(a))$
- Predicates/Sets/Relns	✗	✓	$\forall \textcolor{teal}{P} P(f(a))$
Unnamed			
- Functions	✗	✓	$\lambda X X$
- Predicates	✗	✓	$\lambda X (X = X)$
Statements about			
- Funcs/Preds/Sets/Relns	✗	✓	<i>reflexive</i> \leftrightarrow
Powerful definitions	✗	✓	<i>reflexive</i> := $\lambda R \forall X (R X X)$

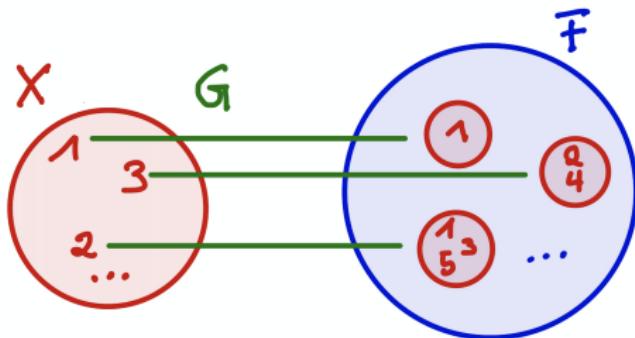
Classical Higher-Order Logic (HOL)

Expressivity	FOL	HOL	Example
Quantification over			
- Individuals	✓	✓	$\forall X_i p_{i \rightarrow o}(f_{i \rightarrow i}(X_i))$
- Functions	✗	✓	$\forall F_{i \rightarrow i} p_{i \rightarrow o}(F_{i \rightarrow o}(a_i))$
- Predicates/Sets/Relns	✗	✓	$\forall P_{i \rightarrow o} P_{i \rightarrow o}(f_{i \rightarrow i}(a_i))$
Unnamed			
- Functions	✗	✓	$\lambda X_i X_i$
- Predicates	✗	✓	$\lambda X_i (X = X)_i$
Statements about			
- Funcs/Preds/Sets/Relns	✗	✓	$\text{reflexive}_{(o \rightarrow o \rightarrow o) \rightarrow o} \leftrightarrow o \rightarrow o \rightarrow o$
Powerful definitions	✗	✓	$\text{reflexive}_{(\alpha \rightarrow \alpha \rightarrow o) \rightarrow o} :=$ $\lambda R_{(\alpha \rightarrow \alpha \rightarrow o)} \forall X_\alpha (R X X)$

Types: Prevent paradoxes and inconsistencies

HOL: Expressivity Matters — Surjective Cantor Theorem —

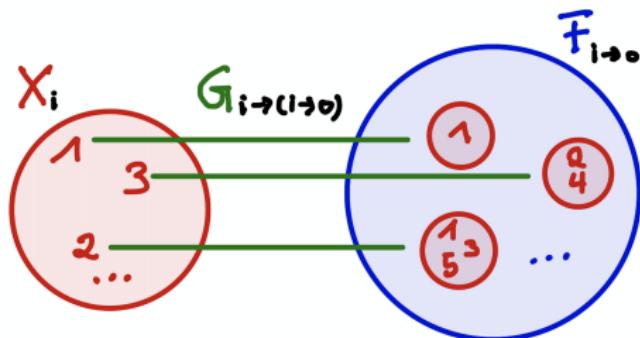
Theorem: *There is no surjective map from a set to its powerset.*



$$\nexists G \forall F \exists x Gx = F$$

HOL: Expressivity Matters — Surjective Cantor Theorem —

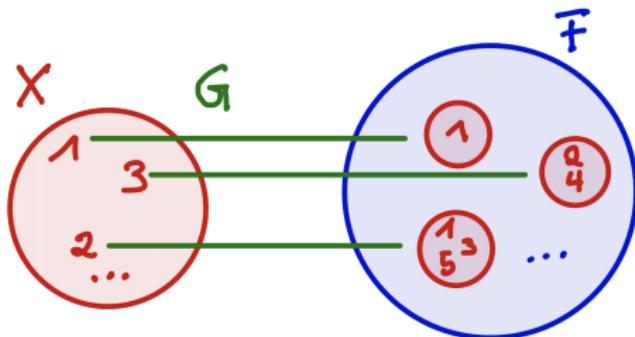
Theorem: There is no surjective map from a set to its powerset.



$$\exists G_{i \rightarrow (i \rightarrow 0)} \forall F_{i \rightarrow 0} \exists X_i \quad G X = F$$

HOL: Expressivity Matters — Surjective Cantor Theorem —

Theorem: There is no surjective *map* from a *set* to its *powerset*.



$$\neg \exists G \forall F \exists x Gx = F$$

Proof (by contradiction):

Assume there is a surjective *G*.

Consider the following choice for *F*: $F := \{X \mid \neg X \in (G X)\}$

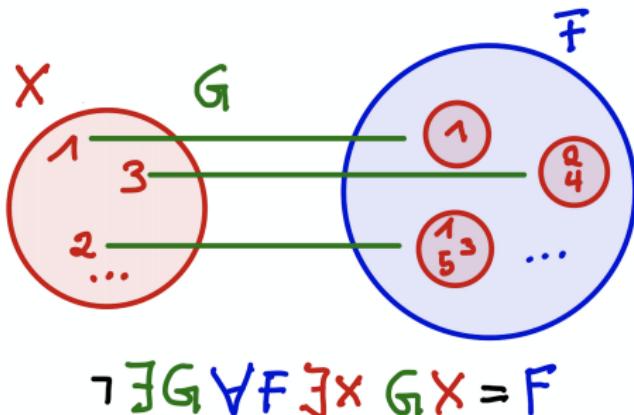
Then there is an *a* such that $(G a) = F$ (since *G* is surj. by ass.)

But then: $a \in (G a)$ iff $a \in F$ iff $\neg a \in (G a)$ (by def. of *F*)

Contradiction. \square

HOL: Expressivity Matters — Surjective Cantor Theorem —

Theorem: *There is no surjective map from a set to its powerset.*



HOL ATPs can solve this problem very efficiently

In our **Leo provers** this works by a combination of (heuristic) **guessing** and straightforward **resolution and HO unification**:

“ \neg ” in $F := \{X \mid \neg X \in (G X)\}$ is **guessed**, rest is straightforward

Further reading: [AndrewsEtAl., Automating higher-order logic, 1984]

HOL: Expressivity Matters — Boolos' Curious Inference —

1. $\forall n f(n, 1) = s(1)$
2. $\forall x f(1, s(x)) = s(s(f(1, x)))$
3. $\forall n \forall x f(s(n), s(x)) = f(n, f(s(n), x))$
4. $D(1)$
5. $\forall x D(x) \rightarrow D(s(x)))$
- ⋮
6. $D(f(s(s(s(s(1))))), s(s(s(s(1))))))$

[George Boolos, A curious inference, J.Phil.Log., 16:1-12, 1987]

Proof in FOL? — yes, but practically infeasible number of proof steps

Proof in HOL? — yes, on one single page

- ▶ Key: powerful lemmata as instances of comprehension axioms

See our interactive verification of Boolos's proof:

[BenzmüllerBrown, The curious inference of Boolos in MIZAR and OMEGA, 2007]

HOL: Expressivity Matters — Boolos' Curious Inference —

1. $\forall n f(n, 1) = s(1)$
2. $\forall x f(1, s(x)) = s(s(f(1, x)))$
3. $\forall n \forall x f(s(n), s(x)) = f(n, f(s(n), x))$
4. $D(1)$
5. $\forall x D(x) \rightarrow D(s(x)))$
- ⋮
6. $D(f(s(s(s(s(1))))), s(s(s(s(1))))))$

[George Boolos, A curious inference, J.Phil.Log., 16:1-12, 1987]

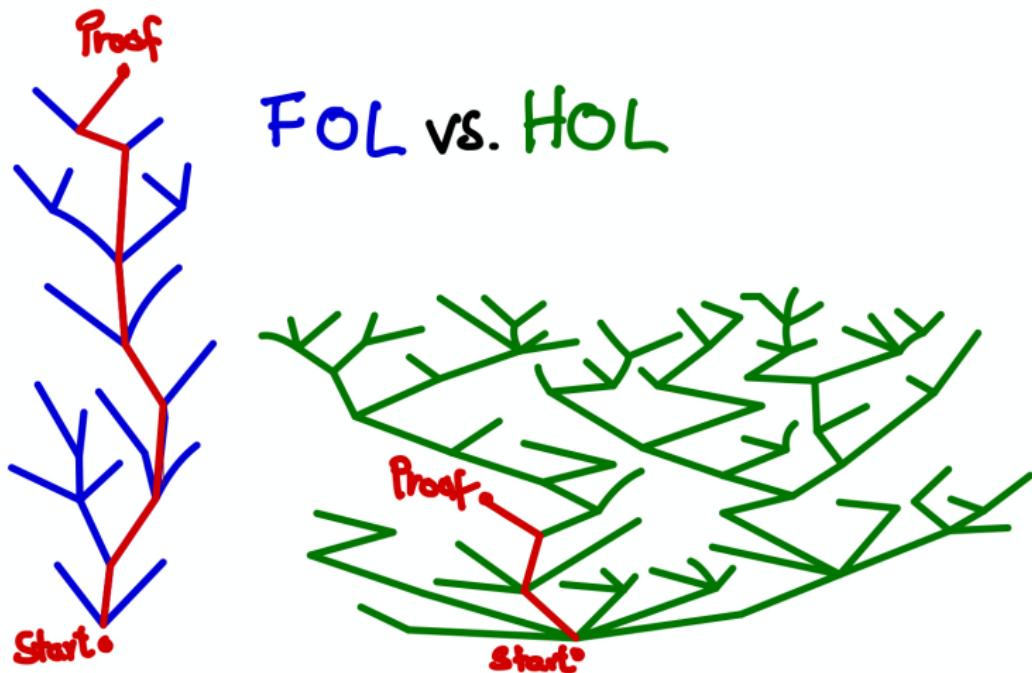
Proof in FOL? — yes, but practically infeasible number of proof steps

Proof in HOL? — yes, on one single page

- ▶ Key: powerful lemmata as instances of comprehension axioms

See our interactive verification of Boolos's proof:

[BenzmüllerBrown, The curious inference of Boolos in MIZAR and OMEGA, 2007]



Proofs in HOL can be short, elegant, intuitive

HOL: Sets and Relations

Elegant representations (omitting types) for operations on sets:

$\{x \mid p(x) \text{ and } q(x)\}$	represented as	$\lambda X ((p\ X) \wedge (q\ X))$
\in	represented as	$\lambda X \lambda M\ (M\ X)$
\cap	represented as	$\lambda M \lambda N\ (\lambda X\ (M\ X \wedge N\ X))$
\cup	represented as	$\lambda M \lambda N\ (\lambda X\ (M\ X \vee N\ X))$
\subseteq	represented as	$\lambda M \lambda N\ (\forall X\ (M\ X \rightarrow N\ X))$
...		

Advantage: Axiomatisation of sets can (usually) be avoided in HOL

Very useful, e.g., for semantic embeddings of other logics in HOL

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p\ X))\ t) \xrightarrow{\beta\text{-reduction}} (p\ t)$

HOL defined on Top of Simply Typed λ -Calculus

- add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(o \rightarrow o) \rightarrow o} \quad \text{(or only } =_{o \rightarrow o \rightarrow o})$$

- no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= \boxed{p_\alpha} \quad | \quad X_\alpha \quad | \quad (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \quad | \quad (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_o$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_o)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_o$ stands for $\neg \forall X_\alpha \neg s_o$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \quad | \quad \boxed{X_\alpha} \quad | \quad (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \quad | \quad (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \quad | \quad X_\alpha \quad | \quad (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \quad | \quad (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_o$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_o)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_o$ stands for $\neg \forall X_\alpha \neg s_o$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_o$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_o)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_o$ stands for $\neg \forall X_\alpha \neg s_o$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Simple Syntax

Simple Types:

$$\alpha, \beta ::= i \mid o \mid (\alpha \rightarrow \beta)$$

(we may add further base types; types are often not displayed)

Simply Typed λ -Calculus (with constants):

$$s, t ::= p_\alpha \mid X_\alpha \mid (\lambda X_\alpha s_\beta)_{\alpha \rightarrow \beta} \mid (s_{\alpha \rightarrow \beta} t_\alpha)_\beta$$

constants variables lambda abstraction application

abstraction and application interact, e.g.: $((\lambda X (p X)) t) \xrightarrow{\beta\text{-reduction}} (p t)$

HOL defined on Top of Simply Typed λ -Calculus

- ▶ add special constant symbols to signature, e.g.

$$\neg_{o \rightarrow o} \quad \vee_{o \rightarrow o \rightarrow o} \quad \Pi_{(\alpha \rightarrow o) \rightarrow o} \quad \text{(or only } =_{\alpha \rightarrow \alpha \rightarrow o})$$

- ▶ no binder besides λ needed: $\forall X_\alpha s_\alpha$ stands for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda X_\alpha s_\alpha)$
- ▶ $\perp, \top, \rightarrow, \leftrightarrow, \exists, =$ can be defined: e.g., $\exists X_\alpha s_\alpha$ stands for $\neg \forall X_\alpha \neg s_\alpha$

HOL is a language of terms. Terms of type o are called formulas.

HOL: Example Formula — Surjective Cantor Theorem —

Theorem: $\neg \exists G_{i \rightarrow (i \rightarrow o)} \forall F_{i \rightarrow o} \exists X_i G X = F$

represented in HOL (as defined on previous slide) as

$$\neg \neg \Pi(\lambda G_{i \rightarrow (i \rightarrow o)} \neg (\Pi(\lambda F_{i \rightarrow o} \neg \Pi(\lambda X_i \neg G X = F))))$$

This is e.g. also the *internal representation* in our Leo provers.

HOL: Example Formula — Surjective Cantor Theorem —

Theorem: $\neg \exists G_{i \rightarrow (i \rightarrow o)} \forall F_{i \rightarrow o} \exists X_i G X = F$

represented in HOL (as defined on previous slide) as

$$\neg \neg \Pi(\lambda G_{i \rightarrow (i \rightarrow o)} \neg (\Pi(\lambda F_{i \rightarrow o} \neg \Pi(\lambda X_i \neg G X = F))))$$

This is e.g. also the *internal representation* in our Leo provers.

However, *intuitive user representations* are supported in HOL provers (e.g. in the Isabelle/HOL system):

Lemma SurjectiveCantor: " $\neg (\exists G :: i \Rightarrow i \Rightarrow \text{bool}. \forall F. \exists X. G X = F)$ "

HOL: Example Formula — Surjective Cantor Theorem —

Theorem: $\neg \exists G_{i \rightarrow (i \rightarrow o)} \forall F_{i \rightarrow o} \exists X_i G X = F$

represented in HOL (as defined on previous slide) as

$$\neg \neg \Pi(\lambda G_{i \rightarrow (i \rightarrow o)} \neg (\Pi(\lambda F_{i \rightarrow o} \neg \Pi(\lambda X_i \neg G X = F))))$$

This is e.g. also the *internal representation* in our Leo provers.

However, *intuitive user representations* are supported in HOL provers (e.g. in the Isabelle/HOL system):

lemma SurjectiveCantor: " $\neg (\exists G :: i \Rightarrow i \Rightarrow \text{bool}. \forall F. \exists X. G X = F)$ "

```
proof
  assume 1: "\exists G :: i \Rightarrow i \Rightarrow \text{bool}. \forall F. \exists X. G X = F"
  obtain g :: "i \Rightarrow i \Rightarrow \text{bool}" where 2: "\forall F. \exists X. g X = F" using 1 by blast
  let ?F = "\lambda X. \neg g X X" (* choose F = {x | \neg(x \in (g x))} *)
  have 3: "\exists Y. ?F = g Y" using 2 by metis
  obtain a :: i where 4: "?F = g a" using 3 by blast
  have 5: "g a a \longleftrightarrow ?F a" using 4 by metis
  have 6: "g a a \longleftrightarrow \neg g a a" using 5 by metis
  show False using 6 by blast
qed
```

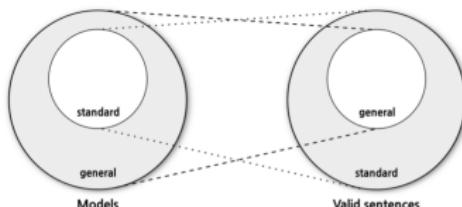
HOL: Well Understood Semantics

HOL with standard semantics:

HOL with Henkin's general semantics:

incomplete

semi-decidable & compact



more model structures ... fewer valid formulas

Important principles are still valid in Henkin's general models:

- ▶ Comprehension (type-restricted):
- ▶ Boolean Extensionality:
- ▶ Functional Extensionality:

$$\forall G \exists F \forall \bar{X}^n F \bar{X}^n = G$$

$$\forall P \forall Q ((P \leftrightarrow Q) \rightarrow P = Q)$$

$$\forall F \forall G ((\forall X F X = G X) \rightarrow F = G)$$

Note: Any “Henkin-valid” formula is also valid in standard semantics!

Suggested Reading

- ▶ Origin [Church, JSL, 1940]
- ▶ Henkin's general semantics: [Henkin, JSL, 1950] [Andrews, JSL, 1971, 1972]
- ▶ Extensionality & Intensionality [Benzmüller et al., ISI 2004] [Muskens, JSL, 2007]

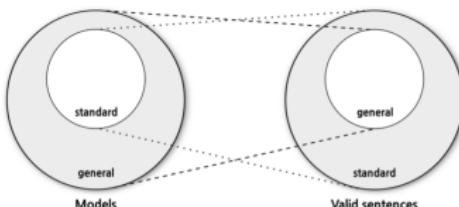
HOL: Well Understood Semantics

HOL with standard semantics:

HOL with Henkin's general semantics:

incomplete

semi-decidable & compact



more model structures ... fewer valid formulas

Important principles are still valid in Henkin's general models:

- ▶ Comprehension (type-restricted):
- ▶ Boolean Extensionality:
- ▶ Functional Extensionality:

$$\begin{aligned} \forall G \exists F \forall \bar{X}^n F \bar{X}^n &= G \\ \forall P \forall Q ((P \leftrightarrow Q) \rightarrow P = Q) \\ \forall F \forall G ((\forall X F X = G X) \rightarrow F = G) \end{aligned}$$

Note: Any “Henkin-valid” formula is also valid in standard semantics!

Suggested Reading

- ▶ Origin [Church, JSL, 1940]
- ▶ Henkin's general semantics: [Henkin, JSL, 1950] [Andrews, JSL, 1971, 1972]
- ▶ Extensionality & Intensionality [Benzmüller et al., ISI 2004] [Muskens, JSL, 2007]

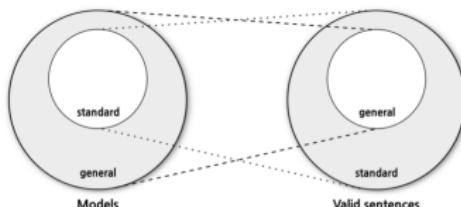
HOL: Well Understood Semantics

HOL with standard semantics:

HOL with Henkin's general semantics:

incomplete

semi-decidable & compact



more model structures ... fewer valid formulas

Important principles are still valid in Henkin's general models:

- ▶ Comprehension (type-restricted):
- ▶ Boolean Extensionality:
- ▶ Functional Extensionality:

$$\forall G \exists F \forall \bar{X}^n F \bar{X}^n = G$$

$$\forall P \forall Q ((P \leftrightarrow Q) \rightarrow P = Q)$$

$$\forall F \forall G ((\forall X F X = G X) \rightarrow F = G)$$

Note: Any “Henkin-valid” formula is also valid in standard semantics!

Suggested Reading

- ▶ Origin [Church, JSL, 1940]
- ▶ Henkin's general semantics: [Henkin, JSL, 1950] [Andrews, JSL, 1971, 1972]
- ▶ Extensionality & Intensionality [Benzmüller et al., ISI 2004] [Muskens, JSL, 2007]

HOL: Theorem Provers

Since 2008: TPTP Infrastructure for HOL ATP

(with Sutcliffe and others)

Increasing activities in HOL ATP since then:

TPS	CMU, US	1980s–2000s	Mating Method
LEO-I	U Saarbrücken, DE	1995–2006	Resolution
LEO-II	Cambridge U, UK	2007–	Paramodulation
Satallax	Saarbr./Paris-Saclay, FR	2010–	Tableaux
IsabelleP	Isabelle Team, DE/UK/NL	2010–	Misc
Leo-III	Berlin/Luxemb., DE/LU	2013–	Ordered Paramodulation
CoqATP	Coq Team, FR	2013–	Saturation Algorithm
agsyHOL	Chalmers, SE	2014–	Lazy Narrowing
HOLyHammer	Innsbr./Praha, AU/CZ	2014–	Misc
Zipperpin	Vrije U Amsterdam, NL	2017–	Superposition
CVC4	U Iowa, US	2019–	SMT Solver
Vampire	U Manchester, UK	2020–	Superposition
...

Model finders:

Nitpick, Nunchaku

Proof assistants: Isabelle, HOL4, HOL light, PVS, Coq, IMPS, Lean, Nuprl, ...

Survey articles:

[BenzmüllerMiller, Automation of Higher-Order Logic, HB of History Of Logic, 2014]

[BenzmüllerAndrews, Church's Type Theory, SEP, 2019]

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: \dots

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: $\text{undecidable \& infinitary}$ (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: \dots

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: $\text{undecidable \& infinitary}$ (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: ...

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: **undecidable & infinitary** (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: ...

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: $\text{undecidable \& infinitary}$ (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: ...

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: undecidable & infinitary (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: ...

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: **undecidable & infinitary** (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Signature contains: $f_{i \rightarrow i}$ $p_{i \rightarrow o}$

Conjecture: $\forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Negated Conjecture: $\neg \forall X_i \exists H_{i \rightarrow i}((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\exists X_i \forall H_{i \rightarrow i} \neg((p(H(fX))) \rightarrow (p(f(HX))))$

Clause normalisation: $\neg((p(H(fs))) \rightarrow (p(f(Hs))))$

Clause normalisation: ...

Clause normalisation: $c1 : (p(H(fs))) \quad c2 : \neg(p(f(Hs)))$

Resolution: $c3 : \perp \quad \text{if } (p(H(fs))) \stackrel{?}{=} (p(f(Hs)))$

Unification: $H \leftarrow \lambda Y_i Y$

Unification: $H \leftarrow \lambda Y_i (f Y)$

Unification: $H \leftarrow \lambda Y_i (f(f Y))$

Unification: $H \leftarrow \dots$

Unification in HOL: **undecidable & infinitary** (but often harmless in practice)

Unification in HOL mod. Boolean extensionality: $c1 : (p(a \wedge b)) \quad c2 : \neg(p(b \wedge a))$

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

HOL: Some Simple Examples (Resolution)

Conjecture: $\forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Negated Conjecture: $\neg \forall X_i \forall Y_i \exists R_{i \rightarrow i \rightarrow o} \neg(R X Y)$

Clause normalisation: $\exists X_i \exists Y_i \forall R_{i \rightarrow i \rightarrow o} \neg\neg(R X Y)$

Clause normalisation: $\neg\neg(R s t)$

Clause normalisation: $(R s t)$

Only clause: $c1 : (R s t)$

Primitive Substitution: $c2 : ((\lambda X_i \lambda Y_i \perp) s t)$ (i.e. $R \leftarrow$ empty relation)

β -normalisation: $c3 : \perp$

Obvious: clause normalisation generally needed over and over again

Remark: Primitive Substitution is key to the automation of Cantor's theorem

Primitive Substitution as Eureka Step: Example

Studies in Computational Metaphysics:

Gödel's (1970) Ontological Argument for Existence of God

Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics

Christoph Benzmüller*

Freie Universität Berlin & Stanford University
c.benzmueller@gmail.com

Bruno Woltzenlogel Paleo

Australian National University
bruno.wp@gmail.com

Primitive Substitution: is **key step** in LEO-II's proof of inconsistency!

(However, we also verified Dana Scott's variant and showed consistency)

Overview

- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ **Non-Classical Logics — Why Classical Logic is not enough**
- ▶ Universal Logical Reasoning — Why Classical Logic is enough
- ▶ Application Direction: Ethico-Legal Governors for AI

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”
True

True
“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

↔

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Why Classical Logic is not enough

The Problem with Modalities

Peter knows “New York is a city in the US”

“New York is a city in the US”

True

=

True

“Batman is a city in Turkey”

Peter knows “Batman is a city in Turkey”

Need for Modal Logics: Belief Time Obligation ...

Many other Non-Classical Logics (besides Modal Logics)

In general: combinations of non-classical logics are needed

Challenge Question: How to Tame the Expanding Logic Zoo?

Overview

- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ Non-Classical Logics — Why Classical Logic is not enough
- ▶ **Universal Logical Reasoning — Why Classical Logic is enough**
- ▶ Application Direction: Ethico-Legal Governors for AI

The Higher-Order Prover Leo-III

Authors

Alexander Steen  , Christoph Benzmüller

Authors and affiliations

Conference paper

First Online: 30 June 2018



Citations Downloads

Part of the [Lecture Notes in Computer Science](#) book series (LNCS, volume 10900)

Abstract

The automated theorem prover Leo-III for classical higher-order logic with Henkin semantics and choice is presented. Leo-III is based on extensional higher-order paramodulation and accepts every common TPTP dialect (FOF, TFF, THF), including their recent extensions to rank-1 polymorphism (TF1, TH1). In addition, the prover natively supports almost every normal higher-order modal logic. Leo-III cooperates with first-order reasoning tools using translations to many-sorted first-order logic and produces verifiable proof certificates. The prover is evaluated on heterogeneous benchmark sets.

How to Tame the Logic Zoo?



STUDIES IN LOGIC AND PRACTICAL REASONING

VOLUME 3

D.M. GABBAY / P. GARDENFORS / J. SIEKMANN / J. VAN BENTHEM / M. VARDI / J. WOODS

EDITORS

*Handbook of
Modal Logic*

How to Tame the Logic Zoo?

2 BASIC MODAL LOGIC

In this section we introduce the basic modal language and its relational semantics. We define basic modal syntax, introduce models and frames, and give the satisfaction definition. We then draw the reader's attention to the internal perspective that modal languages offer on relational structure, and explain why models and frames should be thought of as graphs. Following this we give the standard translation. This enables us to convert any basic modal formula into a first-order formula with one free variable. The standard translation is a bridge between the modal and classical worlds, a bridge that underlies much of the work of this chapter.

2.1 *First steps in relational semantics*

Suppose we have a set of proposition symbols (whose elements we typically write as p, q, r and so on) and a set of modality symbols (whose elements we typically write as $m, m', m'',$ and so on). The choice of PROP and MOD is called the *signature* (or *similarity type*) of the language; in what follows we'll tacitly assume that PROP is denumerably infinite, and we'll often work with signatures in which MOD contains only a single element. Given a signature, we define the *basic modal language* (over the signature) as follows:

$$\varphi ::= p \mid \top \mid \perp \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi \mid \langle m \rangle \varphi \mid [m] \varphi.$$

That is, a basic modal formula is either a proposition symbol, a boolean constant, a boolean combination of basic modal formulas, or (most interesting of all) a formula prefixed by a diamond

How to Tame the Logic Zoo?

2 BASIC MODAL LOGIC

In this section we introduce the basic modal language and its relational semantics. We define basic modal syntax, introduce models and frames, and give the satisfaction definition. We then draw the reader's attention to the internal perspective that modal languages offer on relational structure, and explain why models and frames should be thought of as graphs. Following this we give the standard translation. This enables us to convert any basic modal formula into a first-order formula with one free variable. The standard translation is a bridge between the modal and classical worlds, a bridge that underlies much of the work of this chapter.

2.1 First steps in relational semantics

Syntax

Metalanguage

WHAT FOLLOWS WE WILL tacitly assume that PROP is denumerably infinite, and we'll often work with signatures in which MOD contains only a single element. Given a signature, we define the *basic modal language* (over the signature) as follows:

$$\varphi ::= p \mid \top \mid \perp \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi \mid \langle m \rangle \varphi \mid [m] \varphi.$$

That is, a basic modal formula is either a proposition symbol, a boolean constant, a boolean combination of basic modal formulas, or (most interesting of all) a formula prefixed by a diamond

How to Tame the Logic Zoo?

A model (or Kripke model) \mathfrak{M} for the basic modal language (over some fixed signature) is a triple $\mathfrak{M} = (W, \{R^m\}_{m \in \text{MOD}}, V)$. Here W , the *domain*, is a non-empty set, whose elements we usually call *points*, but which, for reasons which will soon be clear, are sometimes called *states*, *times*, *situations*, *worlds* and other things besides. Each R^m in a model is a binary relation on W , and V is a function (the valuation) that assigns to each proposition symbol p in PROP a subset $V(p)$ of W ; think of $V(p)$ as the set of points in \mathfrak{M} where p is true. The first two components $(W, \{R^m\}_{m \in \text{MOD}})$ of \mathfrak{M} are called the *frame* underlying the model. If there is only one relation in the model, we typically write (W, R) for its frame, and (W, R, V) for the model itself. We encourage the reader to think of Kripke models as graphs (or to be slightly more precise, *directed graphs*, that is, graphs whose points are linked by directed arrows) and will shortly give some examples which show why this is helpful.

Suppose w is a point in a model $\mathfrak{M} = (W, \{R^m\}_{m \in \text{MOD}}, V)$. Then we inductively define the notion of a formula φ being *satisfied* (or *true*) in \mathfrak{M} at point w as follows (we omit some of the clauses for the booleans):

$\mathfrak{M}, w \models p$	iff	$w \in V(p)$,
$\mathfrak{M}, w \models \top$		always,
$\mathfrak{M}, w \models \perp$		never,
$\mathfrak{M}, w \models \neg\varphi$	iff	not $\mathfrak{M}, w \models \varphi$ (notation: $\mathfrak{M}, w \not\models \varphi$),
$\mathfrak{M}, w \models \varphi \wedge \psi$	iff	$\mathfrak{M}, w \models \varphi$ and $\mathfrak{M}, w \models \psi$,
$\mathfrak{M}, w \models \varphi \rightarrow \psi$	iff	$\mathfrak{M}, w \not\models \varphi$ or $\mathfrak{M}, w \models \psi$,
$\mathfrak{M}, w \models \langle m \rangle \varphi$	iff	for some $v \in W$ such that $R^m w v$ we have $\mathfrak{M}, v \models \varphi$,
$\mathfrak{M}, w \models [m] \varphi$	iff	for all $v \in W$ such that $R^m w v$ we have $\mathfrak{M}, v \models \varphi$.

How to Tame the Logic Zoo?

A model (or Kripke model) \mathfrak{M} for the basic modal language (over some fixed signature) is a triple $\mathfrak{M} = (W, \{R^m\}_{m \in \text{MOD}}, V)$. Here W , the *domain*, is a non-empty set, whose elements we usually call *points*, but which, for reasons which will soon be clear, are sometimes called *states*, *times*,

and V

$V(p)$

$(W, \{$

in the

encourage the reader to think of Kripke models as graphs (or to be slightly more precise, *directed graphs*, that is, graphs whose points are linked by directed arrows) and will shortly give some examples which show why this is helpful.

Suppose w is a point in a model $\mathfrak{M} = (W, \{R^m\}_{m \in \text{MOD}}, V)$. Then we inductively define the notion of a formula φ being *satisfied* (or *true*) in \mathfrak{M} at point w as follows (we omit some of the clauses for the booleans):

Semantics

$\mathfrak{M}, w \models p$	iff	$w \in V(p)$,
$\mathfrak{M}, w \models \top$		always,
$\mathfrak{M}, w \models \perp$		never,
$\mathfrak{M}, w \models \neg\varphi$	iff	not $\mathfrak{M}, w \models \varphi$ (notation: $\mathfrak{M}, w \not\models \varphi$),
$\mathfrak{M}, w \models \varphi \wedge \psi$	iff	$\mathfrak{M}, w \models \varphi$ and $\mathfrak{M}, w \models \psi$,
$\mathfrak{M}, w \models \varphi \rightarrow \psi$	iff	$\mathfrak{M}, w \not\models \varphi$ or $\mathfrak{M}, w \models \psi$,
$\mathfrak{M}, w \models \langle m \rangle \varphi$	iff	for some $v \in W$ such that $R^m w v$ we have $\mathfrak{M}, v \models \varphi$,
$\mathfrak{M}, w \models [m] \varphi$	iff	for all $v \in W$ such that $R^m w v$ we have $\mathfrak{M}, v \models \varphi$.

Universal Logical Reasoning

Approach: Shallow Semantic Embedding in HOL

L (target logic):

$s, t ::=$

HOL (meta-logic):

$s, t ::=$

Embedding of  in 

Signature of HOL (Constants and Logical Symbols):

 := 
 := 
 := 
 := 

Meta-logical notions:

 := 

Formulas of L are directly identified with terms HOL

Universal Logical Reasoning

Kripke Style Semantics

(propositional modal logic K)

$M, g, s \models P$ if and only if $s \in g(P)$

$M, g, s \models \neg \varphi$ if and only if $M, g, s \not\models \varphi$

$M, g, s \models \varphi \vee \psi$ if and only if $M, g, s \models \varphi$ or $M, g, s \models \psi$

$M, g, s \models \Box \varphi$ if and only if for all t with sRt we have $M, g, t \models \varphi$

Universal Logical Reasoning

Kripke Style Semantics

(propositional modal logic K)

$M, g, s \models P$ if and only if $s \in g(P)$

$M, g, s \models \neg \varphi$ if and only if $M, g, s \not\models \varphi$

$M, g, s \models \varphi \vee \psi$ if and only if $M, g, s \models \varphi$ or $M, g, s \models \psi$

$M, g, s \models \Box \varphi$ if and only if for all t with sRt we have $M, g, t \models \varphi$

Standard Translation for Propositional Fragment (encoded in HOL)

- ▶ $P = P_{i \rightarrow o}$
- ▶ $\neg = \lambda \varphi_{i \rightarrow o} \lambda w_i \neg(\varphi w)$
- ▶ $\vee = \lambda \varphi_{i \rightarrow o} \lambda \psi_{i \rightarrow o} \lambda w_i \varphi w \vee \psi w$
- ▶ $\Box = \lambda \varphi_{i \rightarrow o} \lambda w_i \forall v_i R w v \rightarrow \varphi v$

Validity

- ▶ $[\varphi_{i \rightarrow o}] = \forall w_i \varphi w$

Modal Logic (in fact, Hybrid Logic) as a Fragment of HOL

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{(\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px)}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvv \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxv))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvv \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvv \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{(\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px)}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvv \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxv))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvv \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvv \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{(\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px)}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvv \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxv))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvv \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvv \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{(\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px)}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x P x w))}^{\beta\text{-reduction}}] && \beta\text{-reduction} \\ &\equiv [\Box \overbrace{(\lambda \varphi \lambda w \Pi(\lambda v R w v \rightarrow \varphi v))(\lambda w \Pi(\lambda x P x w))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (R w v \rightarrow \forall x P x v)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (R w v \rightarrow \forall x P x v). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x P x w))}^{\beta\text{-reduction}}] && \beta\text{-reduction} \\ &\equiv [\Box \overbrace{(\lambda \varphi \lambda w \Pi(\lambda v R w v \rightarrow \varphi v))(\lambda w \Pi(\lambda x P x w))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (R w v \rightarrow \forall x P x v)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (R w v \rightarrow \forall x P x v). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvv \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxv))}^{\Box}] && \text{def. of } \Box \\ &\equiv [\lambda w \forall v (Rvv \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvv \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvw \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxw))}^{\Box}] && \text{def. of } \Box \\ &\equiv [\lambda w \forall v (Rvw \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvw \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvw \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvw \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvw \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvw \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvw \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvw \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning

Kripke Style Semantics

(adding quantifiers)

$M, g, s \models \forall x \varphi$ if and only if for all $d \in D$ we have $M, ([d/x]g), s \models \varphi$

Standard Translation extended for Quantifiers (and encoded in HOL)

- ▶ remember: $\forall x_\alpha s$ is shorthand for $\Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha s)$ —no binder needed!!!
- ▶ $\Pi = \lambda \Phi_{\alpha \rightarrow (i \rightarrow o)} \lambda w_i \Pi_{(\alpha \rightarrow o) \rightarrow o} (\lambda x_\alpha \Phi xw)$

Example (compositionality and λ -conversion at work; omitting types)

$$\begin{aligned} [\Box \forall x Px] &\equiv [\Box \overbrace{\Pi(\lambda x Px)}^{\forall x Px}] && \text{def. of } \forall \\ &\equiv [\Box \overbrace{((\lambda \Phi \lambda w \Pi(\lambda x \Phi xw))(\lambda x Px))}^{\Pi}] && \text{def. of } \Pi \\ &\equiv [\Box \overbrace{(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \beta\text{-reduction} \\ &\equiv [\overbrace{(\lambda \varphi \lambda w \Pi(\lambda v Rvw \rightarrow \varphi v))(\lambda w \Pi(\lambda x Pxw))}^{\square}] && \text{def. of } \square \\ &\equiv [\lambda w \forall v (Rvw \rightarrow \forall x Pxv)] && \beta\text{-reduction} \\ &\equiv \forall w \forall v (Rvw \rightarrow \forall x Pxv). && \text{def. of validity, } \beta\text{-reduction} \end{aligned}$$

Universal Logical Reasoning in HOL

```
1 theory QML_SSU imports Main
2 begin
3   typedecl i (* "type for possible worlds" *)
4   typedecl μ (* "type for individuals" *)
5   type_synonym σ = "(i=bool)"
6
7   abbreviation mnnot ("¬_"[52|53]) where "¬φ ≡ λw. ¬φ(w)"
8   abbreviation mnnotpred ("¬_"[52|53]) where "¬Φ ≡ λx. λw. ¬Φ(x)(w)"
9   abbreviation mand (infixr "∧"51) where "φ ∧ ψ ≡ λw. φ(w) ∧ψ(w)"
10  abbreviation mor (infixr "∨"50) where "φ ∨ ψ ≡ λw. φ(w) ∨ψ(w)"
11  abbreviation mimp (infixr "→"49) where "φ → ψ ≡ λw. φ(w) →ψ(w)"
12  abbreviation mequ (infixr "≡"48) where "φ ≡ ψ ≡ λw. φ(w) ≡ψ(w)"
13  abbreviation mall ("∀")
14  abbreviation malB (binder "∀"[8|9]) where "∀x. φ(x) ≡ ∀z"
15  abbreviation mexi ("∃")
16  abbreviation mexiB (binder "∃"[8|9]) where "∃x. φ(x) ≡ ∃z"
17  abbreviation mbox ("□")
18  abbreviation mdia ("◊")
19
20  abbreviation valid ("□"|"7|8) where "[p] ≡ ∀w. p w"
21
22 lemma "[□φ → φ]" sledgehammer by simp
23 lemma "[φ → ◊φ]" sledgehammer by simp
24 lemma "[□φ → □(□φ)]" sledgehammer by simp
25 end
```

Higher-order modal logic S5^U in HOL

With a few lines of formal HOL code:

- ▶ novel applications (here in metaphysics)
- ▶ excellent degree of automation (automation of Gödel's argument)
- ▶ while still being intuitive (readable by philosophers)

Further reading:

[ECAI 2014, IJCAI 2016, Open Philosophy 2019]

```
1 theory Scott_SSU imports QML_SSU
2 begin
3   consts P :: "(μ⇒σ)⇒σ" (* Positive *)
4   axiomatization where
5     A1a: "[∀Φ. P(Φ) → ¬P(Φ)]" and
6     A1b: "[∀Φ. ¬P(Φ) → P(Φ)]" and
7     A2: "[∀Φ. Φ(Φ) ∧ □(∀x. Φ(x) → Ψ(x)) → P(Ψ)]"
8   definition G where "G(x) = (∀Φ. P(Φ) → Φ(x))" (* Def. of God *)
9   axiomatization where
10    A3: "[P(G)]" and
11    A4: "[∀Φ. P(Φ) → □(P(Φ))]"
12   definition ess (infixr "ess" 85) where
13     "Φ ess x = Φ(x) ∧ (∀Ψ. Ψ(x) → □(∀y. Φ(y) → Ψ(y)))" (* Essence *)
14   definition NE where "NE(x) = (∀Φ. Φ ess x → □(∃Φ))" (* Nec. Existence *)
15   axiomatization where
16     A5: "[P(NE)]"
17
18 theorem T3: "[□(∃x. G(x))]" (* Necessarily, God ∃*)
19 sledgehammer [provers = remote leo2,verbose] (* LEO-II proof in 2,5sec *)
20 by (metis (lifting, full_types)
21      AlA AlB A2 A3 A4 A5 G_def NE_def ess_def)
22 end
```

Ontological argument in S5^U

Universal Logical Reasoning in HOL

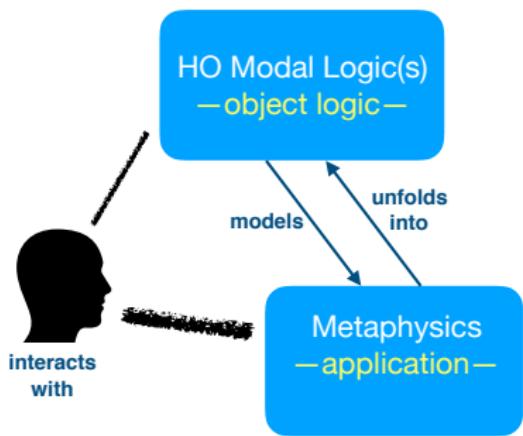


interacts
with

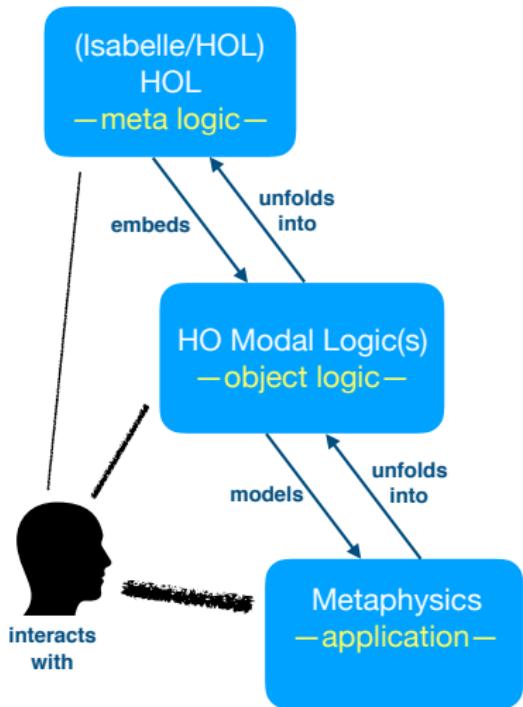


Metaphysics
—application—

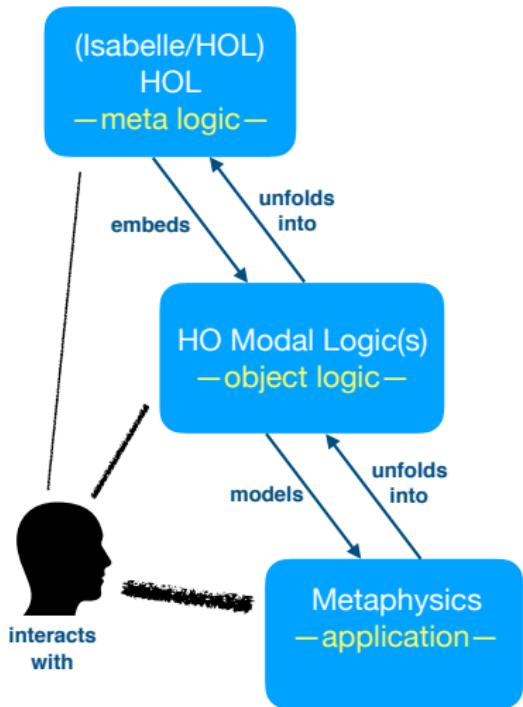
Universal Logical Reasoning in HOL



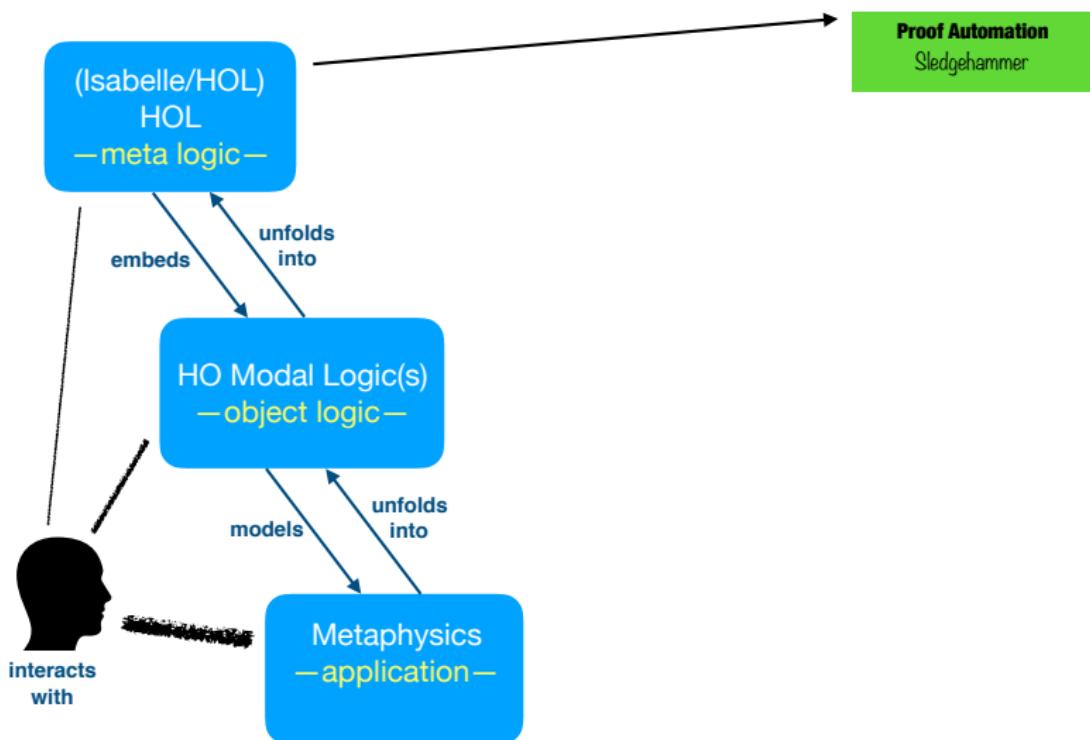
Universal Logical Reasoning in HOL



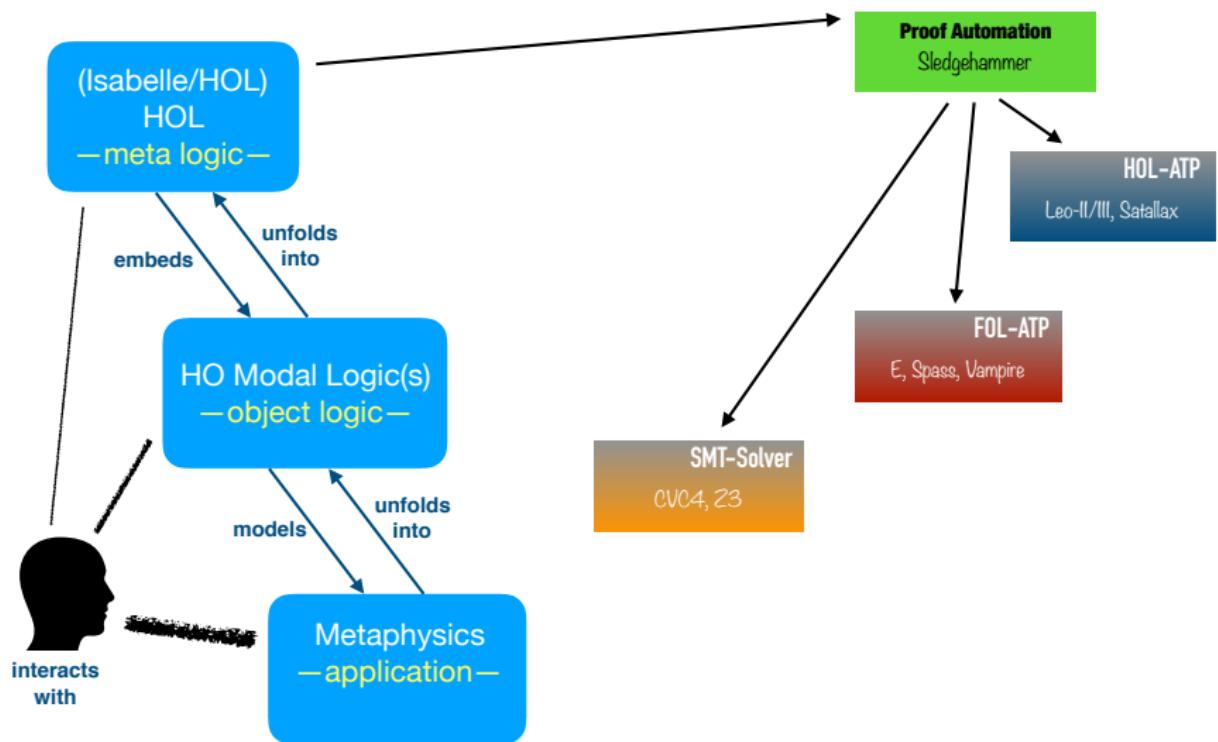
Universal Logical Reasoning in HOL



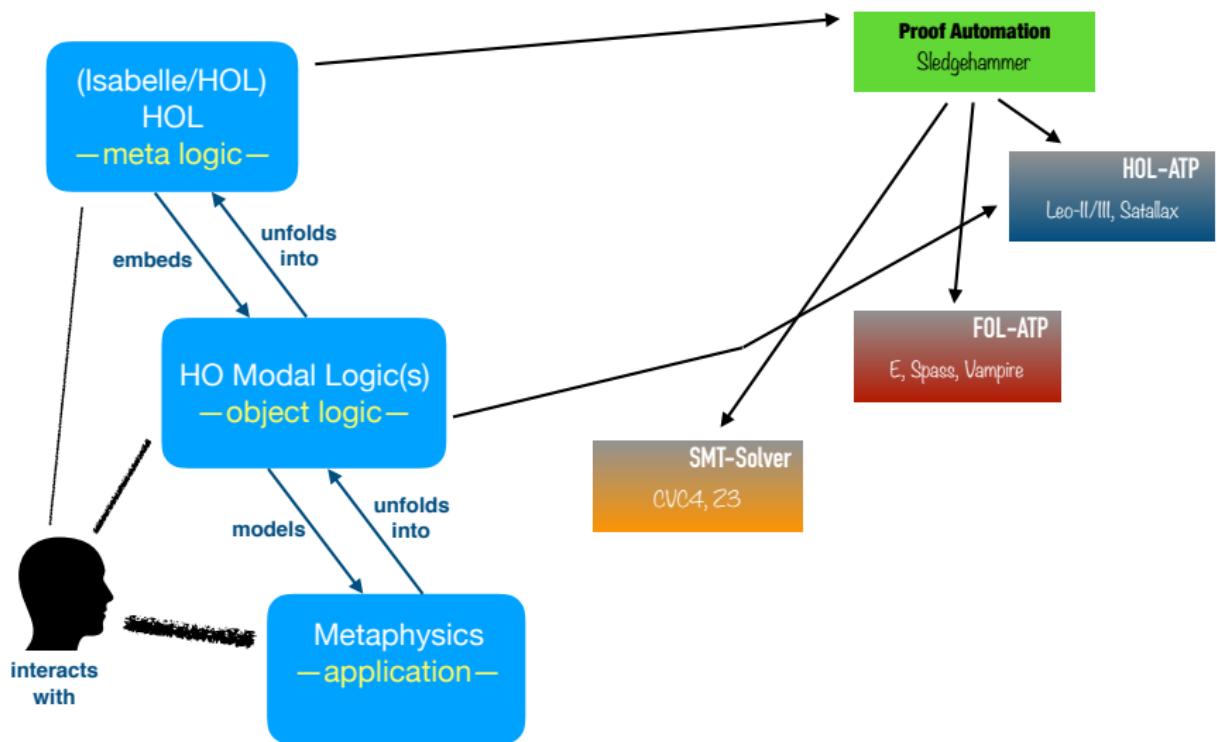
Universal Logical Reasoning in HOL



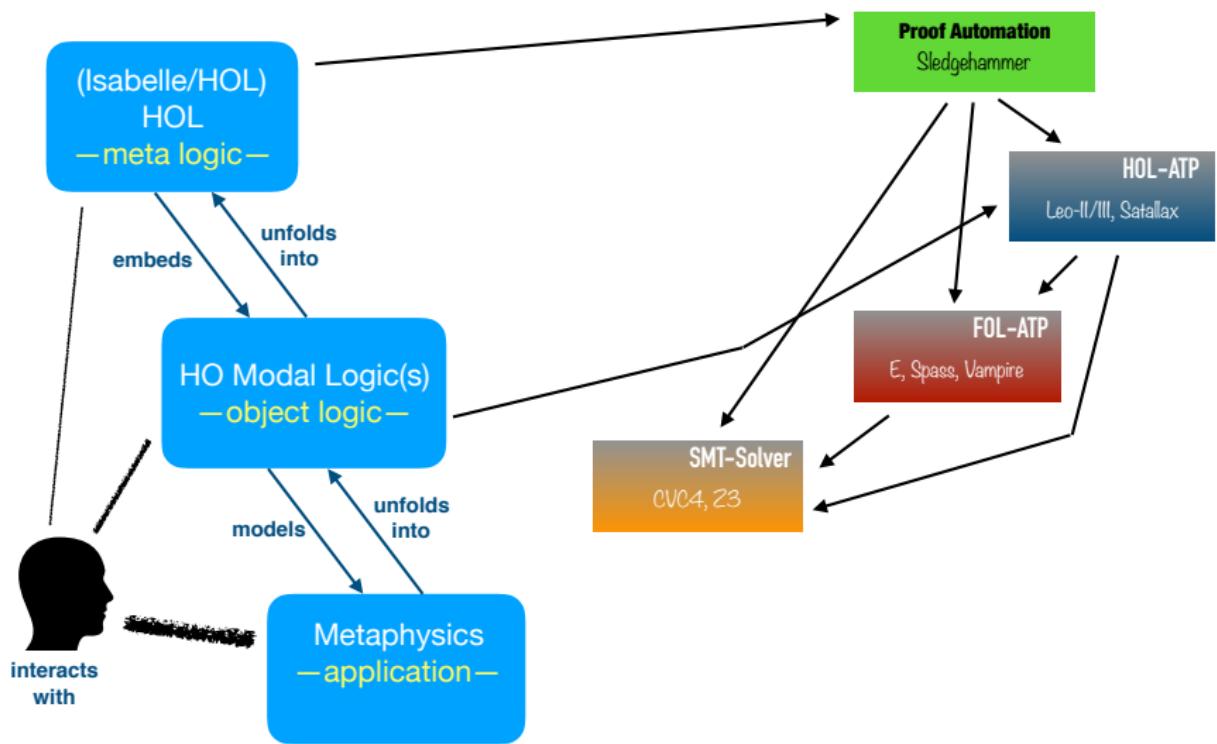
Universal Logical Reasoning in HOL



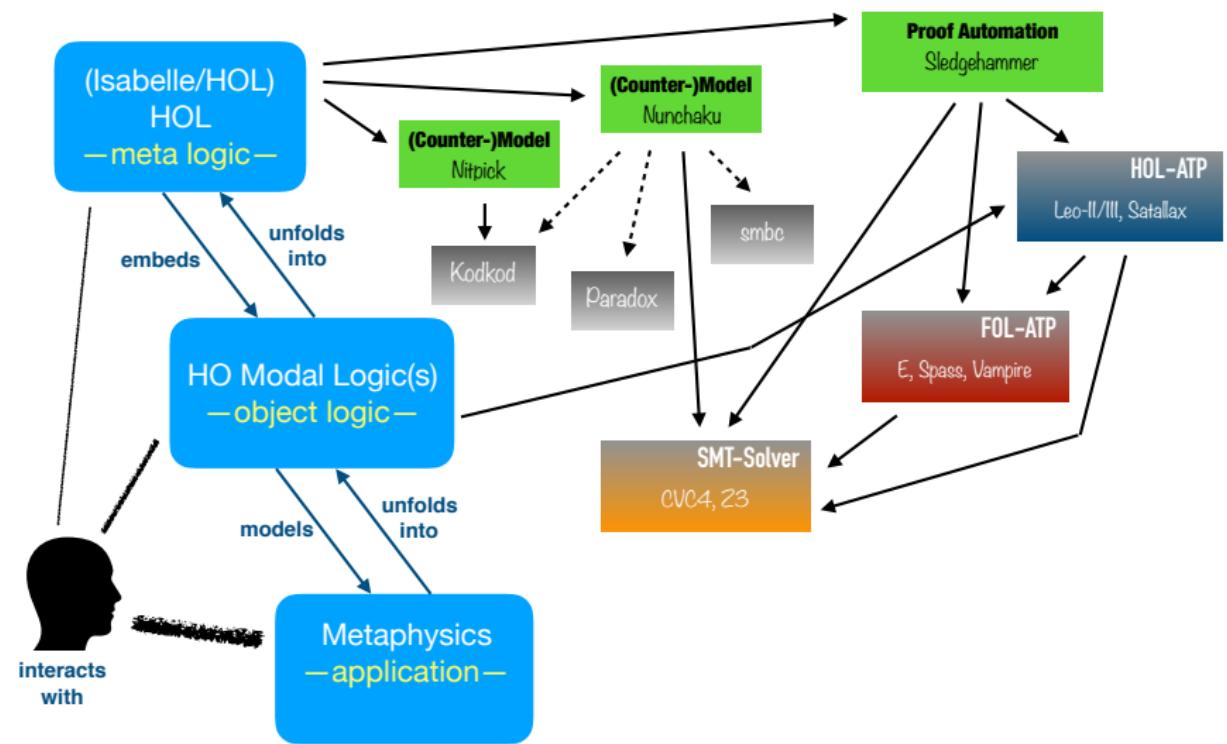
Universal Logical Reasoning in HOL



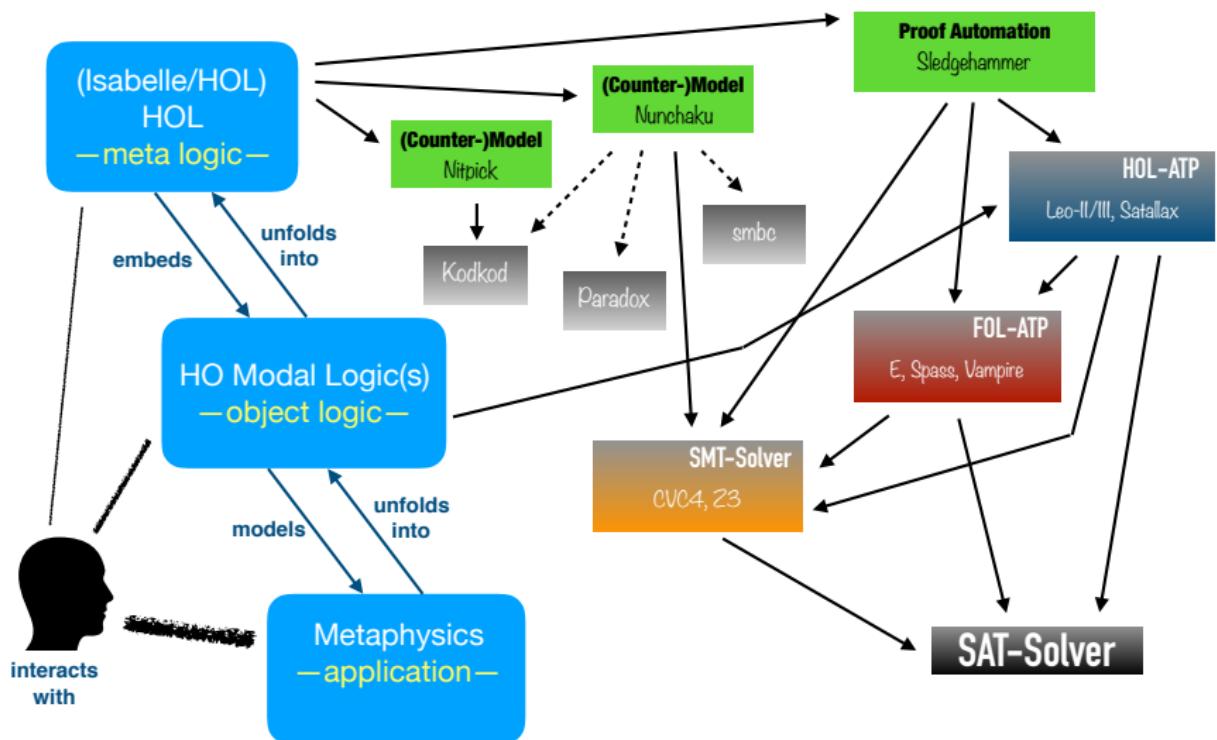
Universal Logical Reasoning in HOL



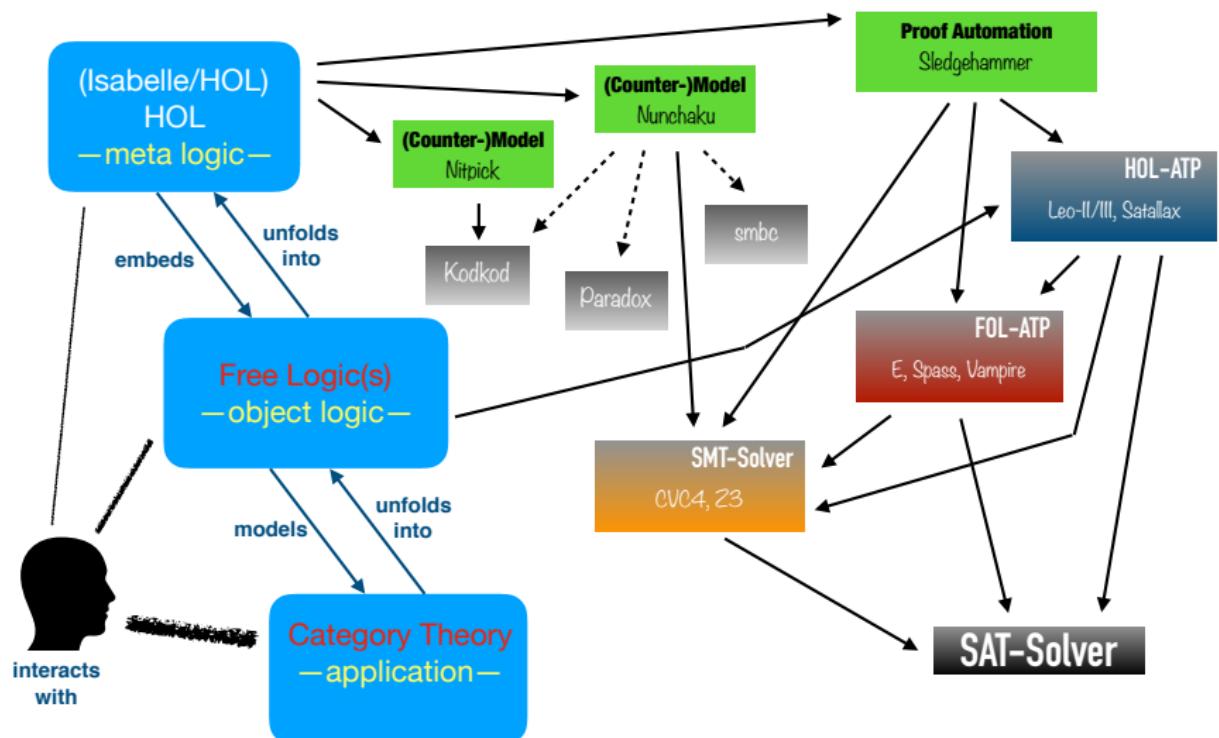
Universal Logical Reasoning in HOL



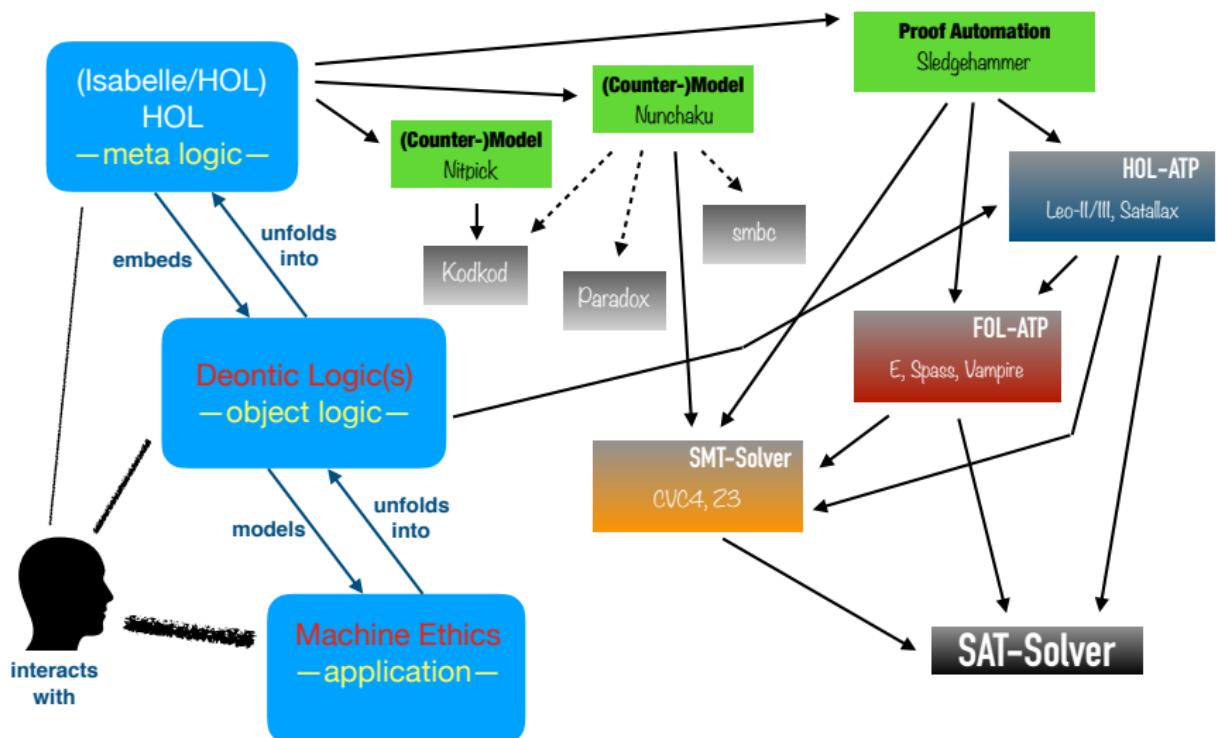
Universal Logical Reasoning in HOL



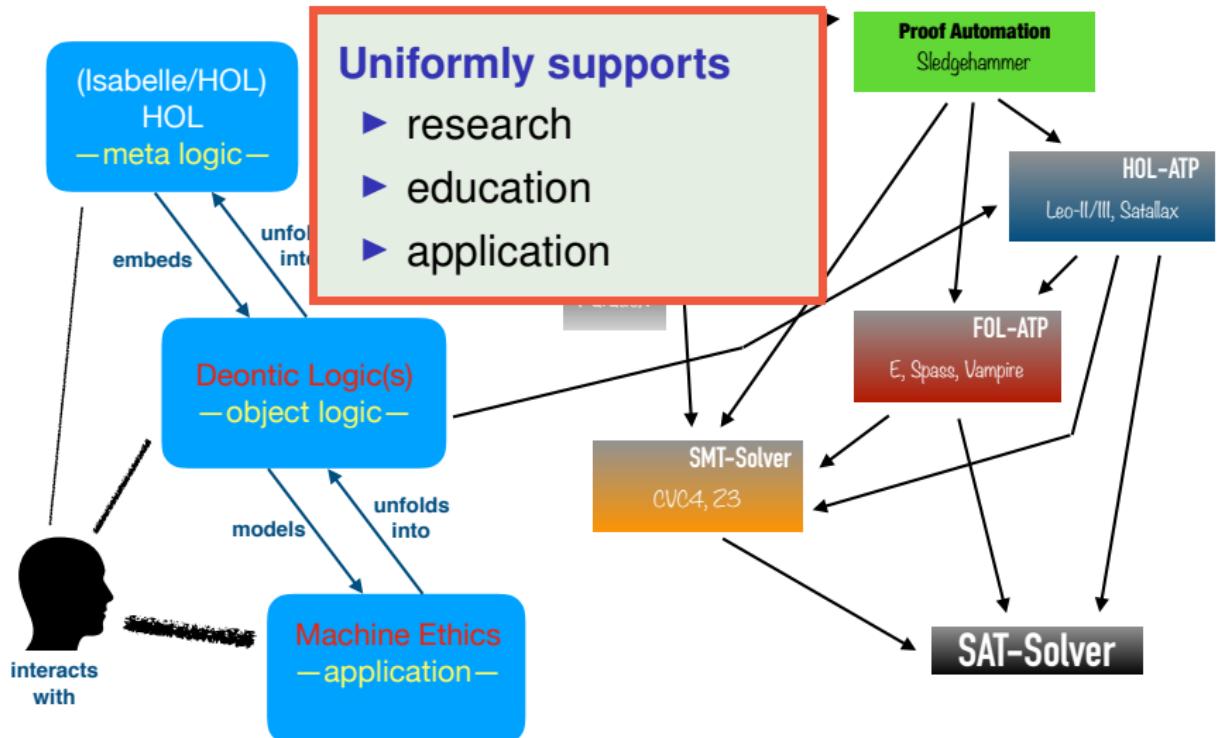
Universal Logical Reasoning in HOL



Universal Logical Reasoning in HOL



Universal Logical Reasoning in HOL



Overview

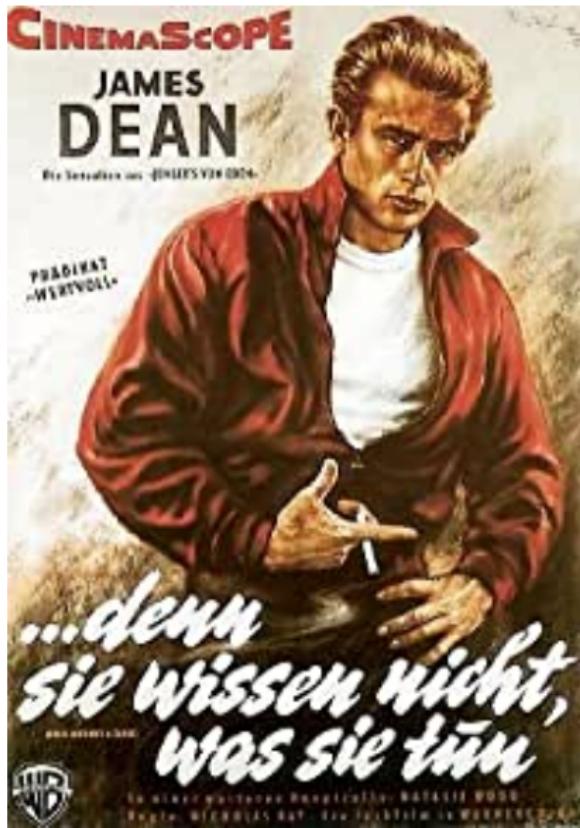
- ▶ Motivation & Own Position
- ▶ Classical Higher-Order Logic (HOL)
- ▶ Non-Classical Logics — Why Classical Logic is not enough
- ▶ Universal Logical Reasoning — Why Classical Logic is enough
- ▶ Application Direction: Ethico-Legal Governors for AI

How to create trust in AI? How to control?



English title: "Rebel Without a Cause" (1955)

How to create trust in AI? How to control?



English title: "Rebel Without a Cause"

- ▶ Do our current AI systems know what they are doing?
- ▶ Do we know what we are doing when we entrust such AI systems with increasingly critical decisions?
- ▶ Is normative directionlessness and unpredictability the core character of future AI systems?
- ▶ How to control such systems?

How to create trust in AI? How to control?



English title: "Rebel Without a Cause"

- ▶ Do our current AI systems know what they are doing?
- ▶ Do we know what we are doing when we entrust such AI systems with increasingly critical decisions?
- ▶ Is normative directionlessness and unpredictability the core character of future AI systems?
- ▶ How to control such systems?

How to create trust in AI? How to control?



English title: "Rebel Without a Cause"

- ▶ Do our current AI systems know what they are doing?
- ▶ Do we know what we are doing when we entrust such AI systems with increasingly critical decisions?
- ▶ Is normative directionlessness and unpredictability the core character of future AI systems?
- ▶ How to control such systems?

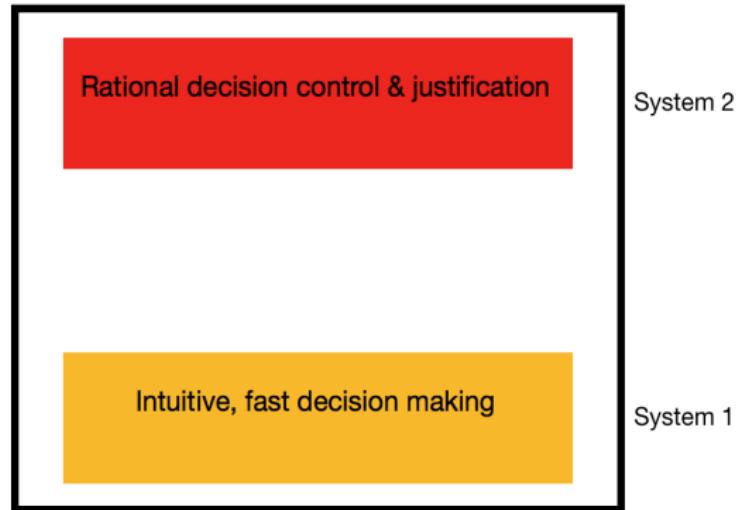
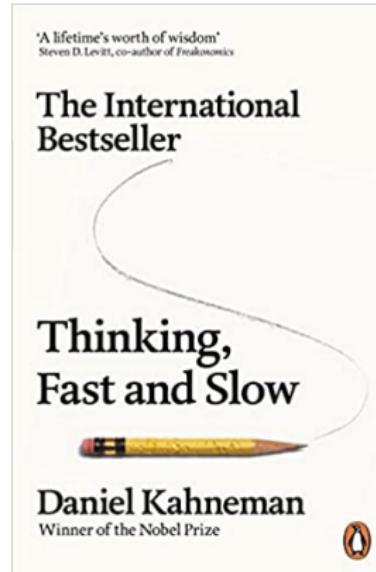
How to create trust in AI? How to control?



English title: "Rebel Without a Cause"

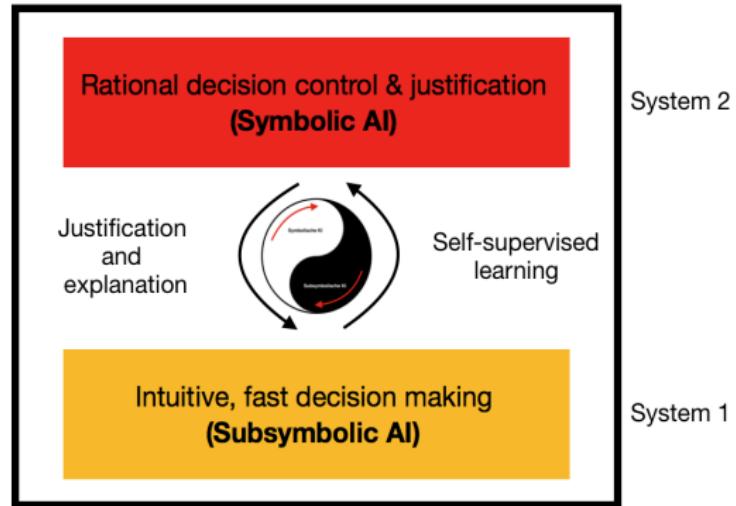
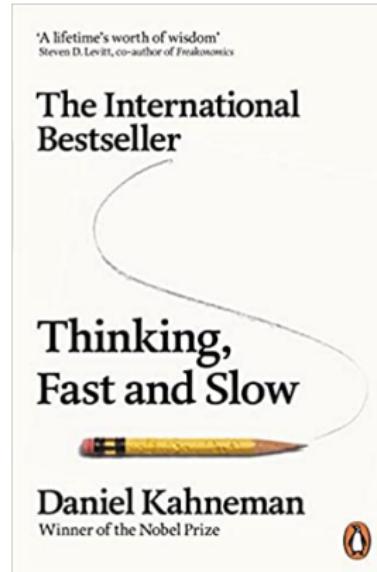
- ▶ Do our current AI systems know what they are doing?
- ▶ Do we know what we are doing when we entrust such AI systems with increasingly critical decisions?
- ▶ Is normative directionlessness and unpredictability the core character of future AI systems?
- ▶ How to control such systems?

How to create trust? How to control?



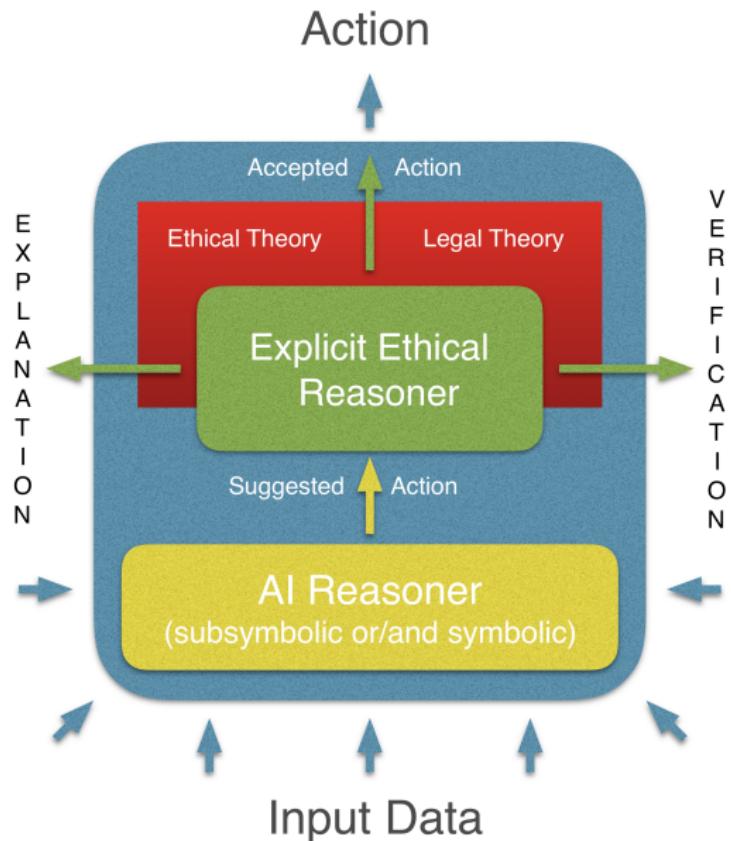
See e.g. also: **J. Haidt, The Emotional Dog and its Rational Tail, 2001**

How to create trust? How to control?

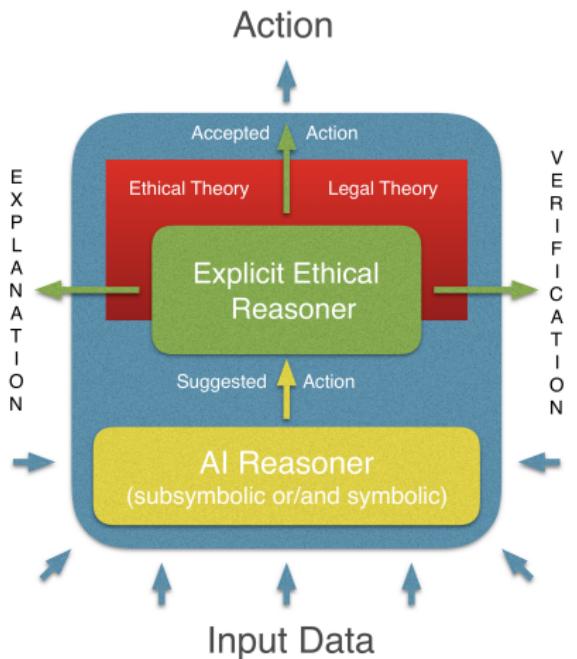


See e.g. also: **J. Haidt, The Emotional Dog and its Rational Tail, 2001**

How to control? Ethico-Legal Governance



How to control? Ethico-Legal Governance



Related Works

- ▶ Toward Ethical Robots
 - ▶ [ArkoudasEtAl., 2005]
- ▶ Artificial Moral Agents
 - ▶ [Wallach&Allen, 2008]
- ▶ Ethical Governors
 - ▶ [ArkinEtAl., 2009, 2012]
 - ▶ [Dennis&Fisher, 2017]
- ▶ Ethical Deliberation in ART
 - ▶ [Dignum, 2017]
- ▶ Programming Machine Ethics
 - ▶ [Pereira&Saptawijaya, 2016]

Addresses demands for explainability and verifiability:

⇒ but not at the level of AI black box systems!

Recent Research Focus: Experiments in expressive, symbolic KR&R in the fields of ethics and law



Artificial Intelligence
Volume 287, October 2020, 103348



Designing normative theories for ethical and legal reasoning: LogIKey framework, methodology, and tool support ★

Christoph Benzmüller , Xavier Parent , Leendert van der Torre

Show more ▾

<https://doi.org/10.1016/j.artint.2020.103348>

Get rights and content



Data in Brief
Volume 35, December 2020, 106409



Data Article
LogIKey workbench: Deontic logics, logic combinations and expressive ethical and legal reasoning (Isabelle/HOL dataset)

Christoph Benzmüller , Ali Farjami , David Fuenmayer , Paul Mender , Xavier Parent , Alexander Steen , Leendert van der Torre , Valeria Zafarzadeh

arXiv



Cornell University



Computer Science > Artificial Intelligence

arXiv:2006.12789 (cs)

[Submitted on 23 Jun 2020]

Encoding Legal Balancing: Automating an Abstract Ethico-Legal Value Ontology in Preference Logic

Christoph Benzmüller, David Fuenmayer, Bertram Lomfeld

EGAI 2020
G.D. Giacomo et al. (Eds.)
© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms
of the Creative Commons Attribution Non-Commercial License 4.0 (CC-BY-NC 4.0).
doi:10.3233/FALON200548

Normative Reasoning with Expressive Logic Combinations

David Fuenmayer and Christoph Benzmüller

Springer Link

Search

[International Conference on Logic and Argumentation](#)

[CLAR 2020: Logic and Argumentation](#) pp 104-115 | [Cite as](#)

Computer-Supported Analysis of Arguments in Climate Engineering

Authors

Authors and affiliations

David Fuenmayer , Christoph Benzmüller

SpringerLink

Search

Download book

Pacific Rim International Conference on Artificial Intelligence

[PRICAI 2019: PRICAI 2019: Trends in Artificial Intelligence](#) pp 418-432 | [Cite as](#)

Harnessing Higher-Order (Meta-)Logic to Represent and Reason with Complex Ethical Theories

Authors

Authors and affiliations

David Fuenmayer , Christoph Benzmüller

(some recent papers)



X. Parent



L. van der Torre



A. Farjami



D. Fuenmayer



A. Steen

Conclusion

Universal Logical Reasoning in HOL

- ▶ works surprisingly well
- ▶ enables many novel applications, including
- ▶ pluralistic ethico-legal reasoning (e.g. to control AI systems)

Required: further research and investment in

- ▶ logic embeddings in HOL
- ▶ automated theorem proving and model finding for HOL
- ▶ proof assistants for HOL
- ▶ education & training

Next envisioned challenge:

- ▶ ML (Intuition) $\xleftarrow{\text{Interaction}}$ Universal Logical Reasoning (Deliberation)

[German Conference on Artificial Intelligence \(Künstliche Intelligenz\)](#)

... KI 2020: [KI 2020: Advances in Artificial Intelligence pp 251-258](#) | [Cite as](#)

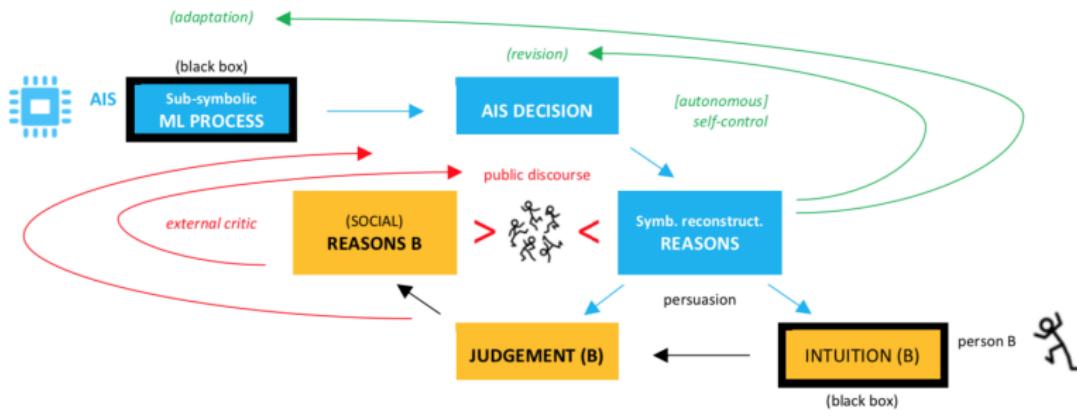
Reasonable Machines: A Research Manifesto

Authors

[Authors and affiliations](#)

Christoph Benzmüller  , Bertram Lomfeld

Artificial “Social Reasoning Model”



Building trust into AI systems through rational communication of »reasons«

Important:

- ▶ these reasons may actually be independent of the original motivational impulse to act
- ▶ should be possible to avoid opening the black-box, since this can increase vulnerability