# Classical Higher-Order Logic (HOL)
## Automation and Selected Applications

**Christoph Benzmüller**

U Bamberg (& FU Berlin)

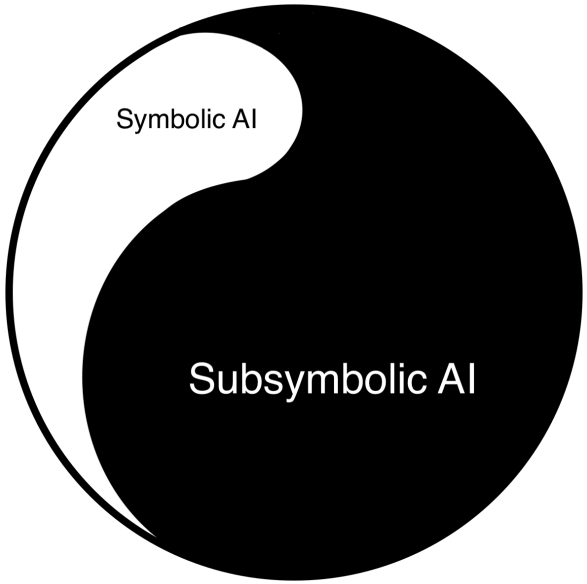
*"If we had it [a characteristica universalis], we should be able to reason in metaphysics and morals in much the same way as in geometry and analysis."* (Leibniz, 1690; translation by Russell)

# Yin and Yang of AI — Unhealthy Hype!

## Examples of success

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

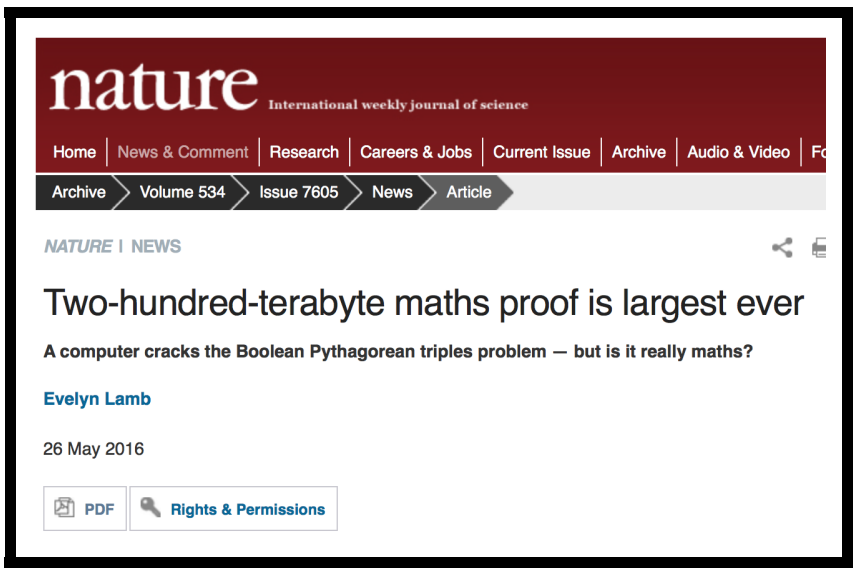
**(symbolic AI)**




## Examples of success

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

**(subsymbolic KI)**

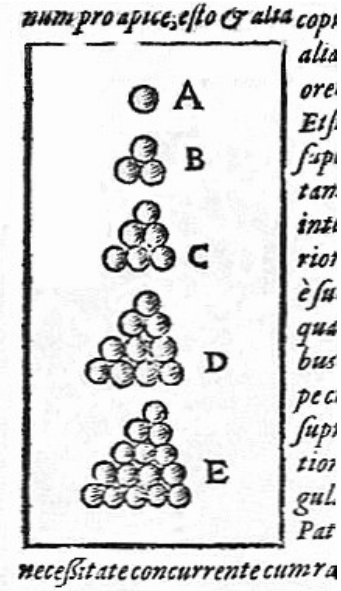



# Another Interesting Story — Formal Logic in Maths

Kepler (1571-1630)

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

$$V\_ = \frac{\pi}{\sqrt{18}} \approx 74\%$$

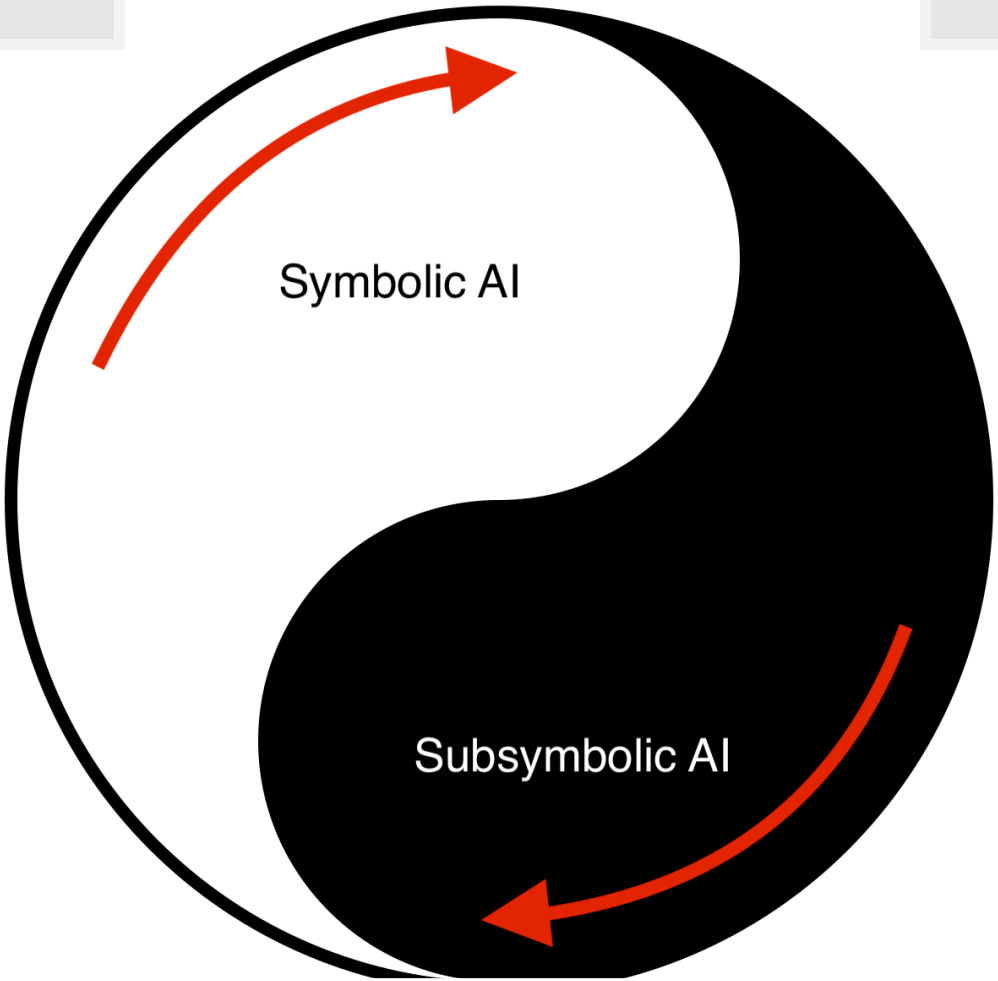



- Proved in 1998 by Hales: 300 page proof, with code and data
- Submitted to the Annals of Mathematics: referees gave up to verify it all
- **Flyspeck project (completed in 2014):**
  - A formal verification of the proof in HOL Light
  - 27,223 proved theorems, 228 definitions, 30 person-years
- **Work 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 | ✓ | ✓ | $\forall X\ p(f(X))$ |
| - Functions | - | ✓ | $\forall F\ p(F(a))$ |
| - Predicates/Sets/Rels | - | ✓ | $\forall P\ P(f(a))$ |
| Unnamed | | | |
| - Functions | - | ✓ | $\lambda X\ X$ |
| - Predicates | - | ✓ | $\lambda X\ (X = X)$ |
| Statements about | | | |
| - Funcs/Preds/Sets/Rels | - | ✓ | $reflexive \leftrightarrow$ |
| Powerful definitions | - | ✓ | $reflexive := \lambda R\ \forall X\ (R\ X\ X)$ |

# Classical Higher-Order Logic (HOL)

| Expressivity | FOL | HOL | Example |
|---|---|---|---|
| Quantification over | | | |
| - Individuals | ✓ | ✓ | $\forall X\ p(f(X))$ |
| - Functions | - | ✓ | $\forall F\ p(F(a))$ |
| - Predicates/Sets/Rels | - | ✓ | $\forall P\ P(f(a))$ |
| Unnamed | | | |
| - Functions | - | ✓ | $\lambda X\ X$ |
| - Predicates | - | ✓ | $\lambda X\ (X = X)$ |
| Statements about | | | |
| - Funcs/Preds/Sets/Rels | - | ✓ | $\mathit{reflexive} \leftrightarrow$ |
| Powerful definitions | - | ✓ | $\mathit{reflexive} := \lambda R\ \forall X\ (R\ X\ X)$ |

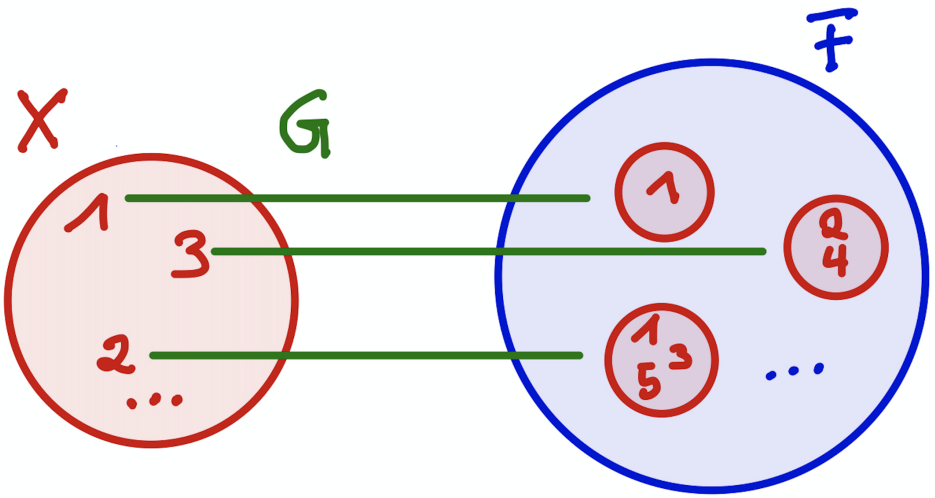
# Classical Higher-Order Logic (HOL)

| Expressivity | FOL | HOL | Example |
|---|---|---|---|
| Quantification over | | | |
| - Individuals | ✓ | ✓ | $\forall X_i \; p_{i\to o}(f_{i\to i}(X_i))$ |
| - Functions | - | ✓ | $\forall F_{i\to i} \; p_{i\to o}(F_{i\to o}(a_i))$ |
| - Predicates/Sets/Rels | - | ✓ | $\forall P_{i\to o} \; P_{i\to o}(f_{i\to i}(a_i))$ |
| Unnamed | | | |
| - Functions | - | ✓ | $\lambda X_i \; X_i$ |
| - Predicates | - | ✓ | $\lambda X_i \; (X = X)_i$ |
| Statements about | | | |
| - Funcs/Preds/Sets/Rels | - | ✓ | $reflexive_{(o\to o\to o)\to o} \leftrightarrow_{o\to o\to o}$ |
| Powerful definitions | - | ✓ | $reflexive_{(\alpha\to\alpha\to o)\to o} := \lambda R_{(\alpha\to\alpha\to o)} \; \forall X_\alpha \; (R \; X \; X)$ |

Types: Prevent paradoxes and inconsistencies

# HOL: Expressivity Matters — Surjective Cantor Theorem —

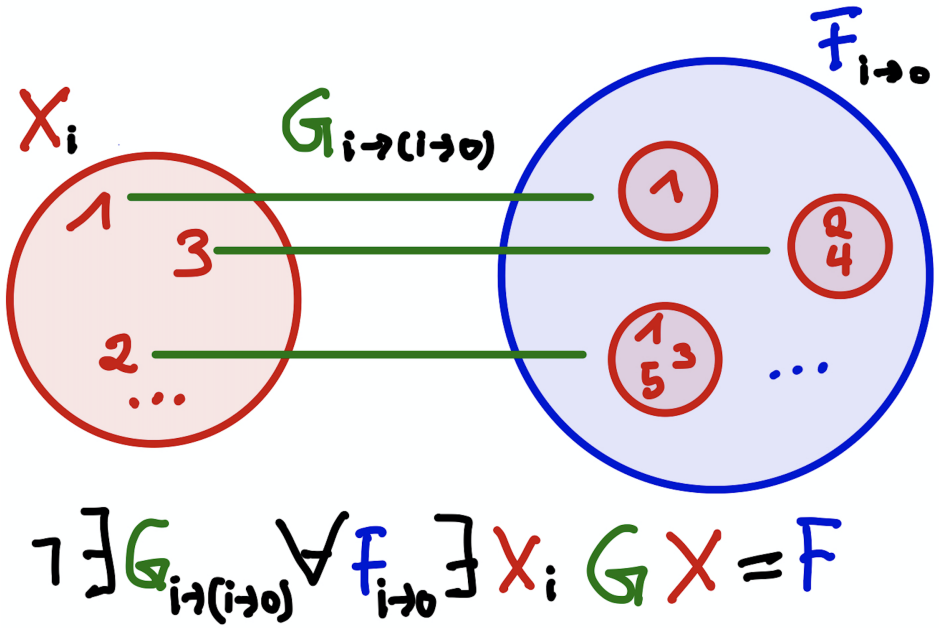
**Theorem:** *There is no surjective map from a set to its powerset.*




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

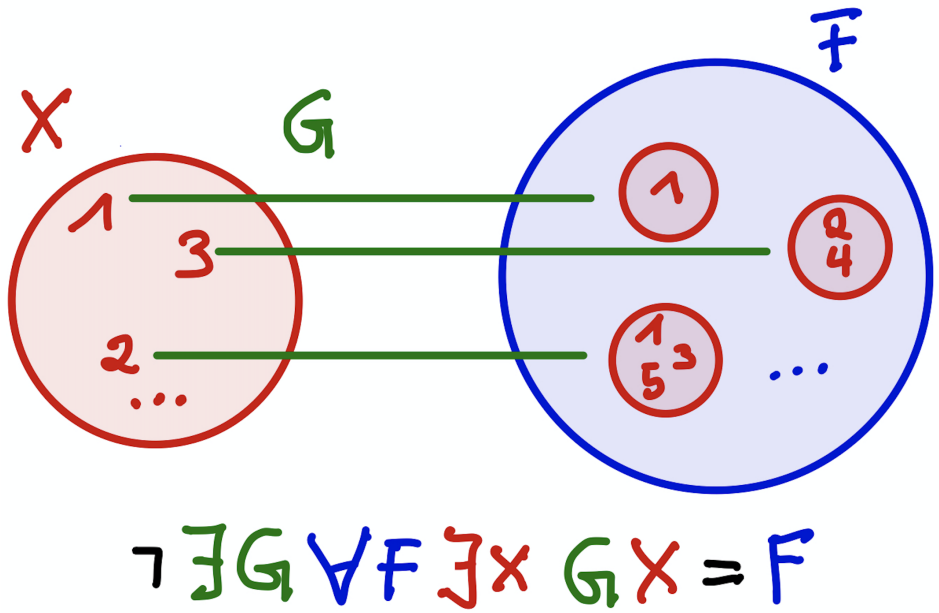
# HOL: Expressivity Matters — Surjective Cantor Theorem —

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



# HOL: Expressivity Matters — Surjective Cantor Theorem —

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




**Proof (by contradiction):**

Assume there is a surjective $G$.

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

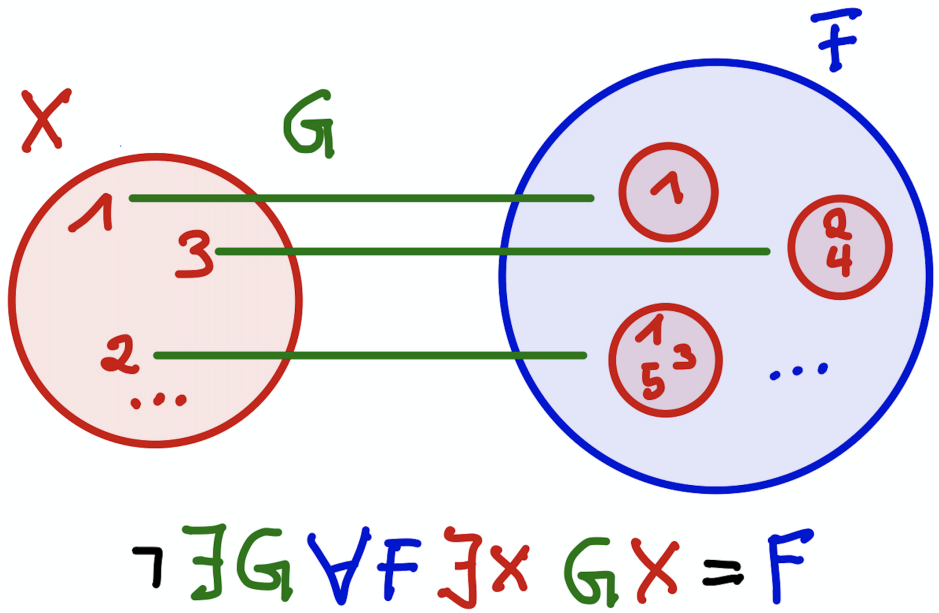
Then there is an $a$ such that $(G\ a) = F$ (since $G$ is surj. by ass.)

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

Contradiction. □

# 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**:

"¬" 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' Curius 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. $\forall x\, D(x) \to D(s(x))$

   $\therefore$
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' Curius 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. $\forall x\, D(x) \to D(s(x))$

   $\therefore$
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

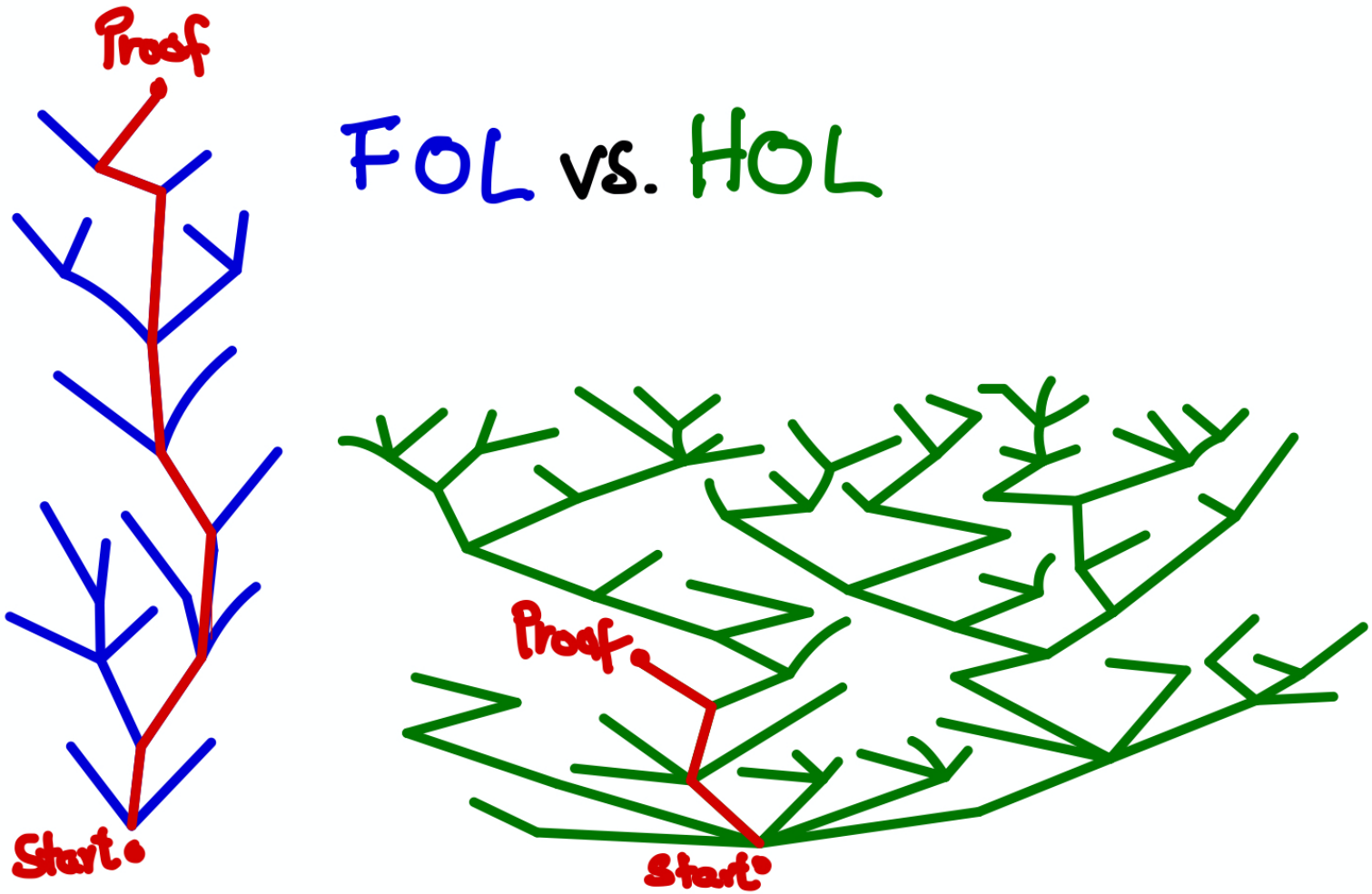
**Proof in HOL?** — yes, on **one single page**

- 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' Curius Inference —




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 | $\lambda M \lambda N\ (\forall X\ (M\ X \rightarrow N\ X))$ |
| | $\ldots$ | |

**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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: $\alpha, \beta \quad ::= \quad i \mid o \mid (\alpha \to \beta)$

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

Simply Typed $\lambda$-Calculus (with constants):

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

constants variables lambda abstraction application

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

HOL defined on Top of Simply Typed $\lambda$-Calculus

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

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

- no binder besides $\lambda$ needed: $\forall X_\alpha\ s_o$ stands for $\Pi_{(\alpha\to o)\to o}(\lambda X_\alpha\ s_o)$
- $\bot, \top, \to, \leftrightarrow, \exists, =$ 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: Example Formula — Surjective Cantor Theorem —

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

represented in HOL (as defined on previous slide) as

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

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

# HOL: Example Formula — Surjective Cantor Theorem —

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

represented in HOL (as defined on previous slide) as

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

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: "¬(∃G::i⇒i⇒bool.∀F.∃X. G X = F)"
```

# HOL: Example Formula — Surjective Cantor Theorem —

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

represented in HOL (as defined on previous slide) as

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

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):

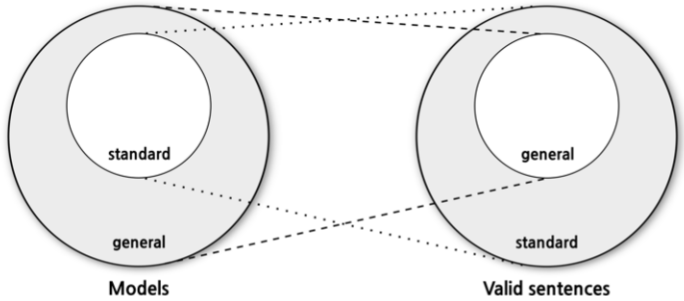
```
lemma SurjectiveCantor: "¬(∃G::i⇒i⇒bool.∀F.∃X. G X = F)"

proof
  assume  1: "∃G::i⇒i⇒bool.∀F.∃X. G X = F"
  obtain  g::"i⇒i⇒bool" where 2: "∀F.∃X. g X = F" using 1 by blast
  let ?F = "λX. ¬ g X X" (* choose F = {x|¬(x∈(g x))} *)
  have    3: "∃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 ⟷ ?F a" using 4 by metis
  have    6: "g a a ⟷ ¬g a a" using 5 by metis
  show False using 6 by blast
qed
```

# HOL: Well Understood Semantics

HOL with standard semantics: incomplete

HOL with Henkin's general semantics: semi-decidable & compact




more model structures . . . fewer valid formulas

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

- Comprehension (type-restricted): $\forall G\, \exists F\, \forall \overline{X^n}\, F\overline{X^n} = G$
- Boolean Extensionality: $\forall P\, \forall Q\, ((P \leftrightarrow Q) \rightarrow P = Q)$
- Functional Extensionality: $\forall F\, \forall G\, ((\forall X\, F\, X = G\, X) \rightarrow F = G)$

**Note: Any "Henkin-valid" formula is also valid in standard semantics!**

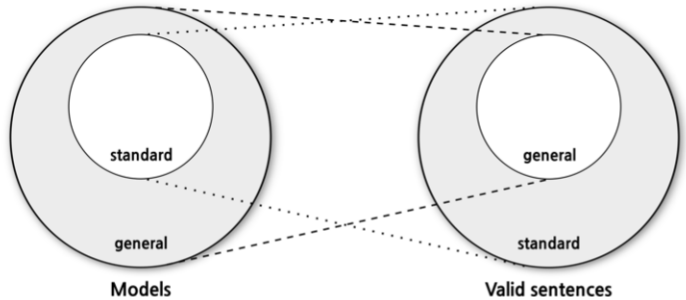
Suggested Reading

- Origin [Church, JSL, 1940]
- Henkin's general semantics: [Henkin, JSL, 1950] [Andrews, JSL, 1971, 1972]
- Extensionality&Intensionality: [BenzmüllerEtAl, JSL, 2004] [Muskens, JSL, 2007]

# HOL: Well Understood Semantics

HOL with standard semantics: incomplete

HOL with Henkin's general semantics: semi-decidable & compact




more model structures . . . fewer valid formulas

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

- Comprehension (type-restricted): $\forall G\, \exists F\, \forall \overline{X^n}\, F\overline{X^n} = G$
- Boolean Extensionality: $\forall P\, \forall Q\, ((P \leftrightarrow Q) \rightarrow P = Q)$
- Functional Extensionality: $\forall F\, \forall G\, ((\forall X\, F\ X = G\ X) \rightarrow F = G)$

**Note: Any "Henkin-valid" formula is also valid in standard semantics!**

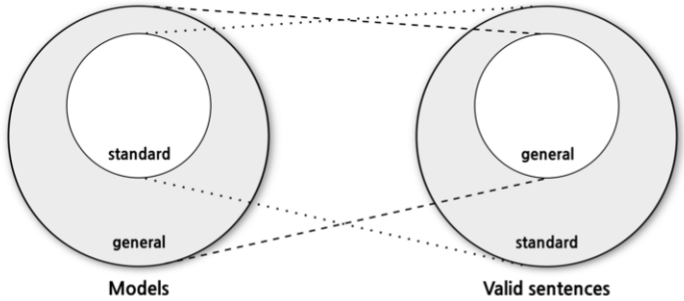
Suggested Reading

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

# HOL: Well Understood Semantics

HOL with standard semantics: incomplete

HOL with Henkin's general semantics: semi-decidable & compact




more model structures . . . fewer valid formulas

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

- Comprehension (type-restricted): $\forall G\, \exists F\, \forall \overline{X^n}\, F\overline{X^n} = G$
- Boolean Extensionality: $\forall P\, \forall Q\, ((P \leftrightarrow Q) \rightarrow P = Q)$
- Functional Extensionality: $\forall F\, \forall G\, ((\forall X\, F\ X = G\ X) \rightarrow F = G)$

**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: [BenzmüllerEtAl, JSL, 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: $f_{i\to i}$ $p_{i\to o}$

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: $\ldots$
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

Conjecture: $\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Negated Conjecture: $\neg\forall X_i \exists H_{i\to i}((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$

Clause normalisation: $\exists X_i \forall H_{i\to i}\ \neg((p\ (H\ (f\ X))) \to (p\ (f\ (H\ X))))$
Clause normalisation: $\neg((p\ (H\ (f\ s))) \to (p\ (f\ (H\ s))))$
Clause normalisation: ...
Clause normalisation: $c1 : (p\ (H\ (f\ s)))$ $c2 : \neg(p\ (f\ (H\ s)))$

Resolution: $c3 : \bot$ if $(p\ (H\ (f\ s))) \stackrel{?}{=} (p\ (f\ (H\ s)))$

Unification: $H \longleftarrow \lambda Y_i\ Y$
Unification: $H \longleftarrow \lambda Y_i\ (f\ Y)$
Unification: $H \longleftarrow \lambda Y_i\ (f\ (f\ Y))$
Unification: $H \longleftarrow \ldots$

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

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

# HOL: Some Simple Examples (Resolution)

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

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

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

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

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

$\beta$-normalisation: $c3 : \bot$

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

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

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

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

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

$\beta$-normalisation: $c3 : \bot$

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

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

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

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

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

$\beta$-normalisation: $c3 : \bot$

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

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

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

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

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

$\beta$-normalisation: $c3 : \bot$

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: | $\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ |
| Negated Conjecture: | $\neg\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ |
| Clause normalisation: | $\exists X_i \exists Y_i \forall R_{i\to i\to o}\ \neg\neg(R\ X\ Y)$ |
| Clause normalisation: | $\neg\neg(R\ s\ t)$ |
| Clause normalisation: | $(R\ s\ t)$ |
| Only clause: | $c1 : (R\ s\ t)$ |
| Primitive Substitution: | $c2 : ((\lambda X_i \lambda Y_i\ \bot)\ s\ t)$ (i.e. $R \longleftarrow$ empty relation) |
| $\beta$-normalisation: | $c3 : \bot$ |

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: | $\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ | |
| Negated Conjecture: | $\neg\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ | |
| Clause normalisation: | $\exists X_i \exists Y_i \forall R_{i\to i\to o}\ \neg\neg(R\ X\ Y)$ | |
| Clause normalisation: | $\neg\neg(R\ s\ t)$ | |
| Clause normalisation: | $(R\ s\ t)$ | |
| Only clause: | $c1 : (R\ s\ t)$ | |
| Primitive Substitution: | $c2 : ((\lambda X_i \lambda Y_i\ \bot)\ s\ t)$ | (i.e. $R \longleftarrow$ empty relation) |
| $\beta$-normalisation: | $c3 : \bot$ | |

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)

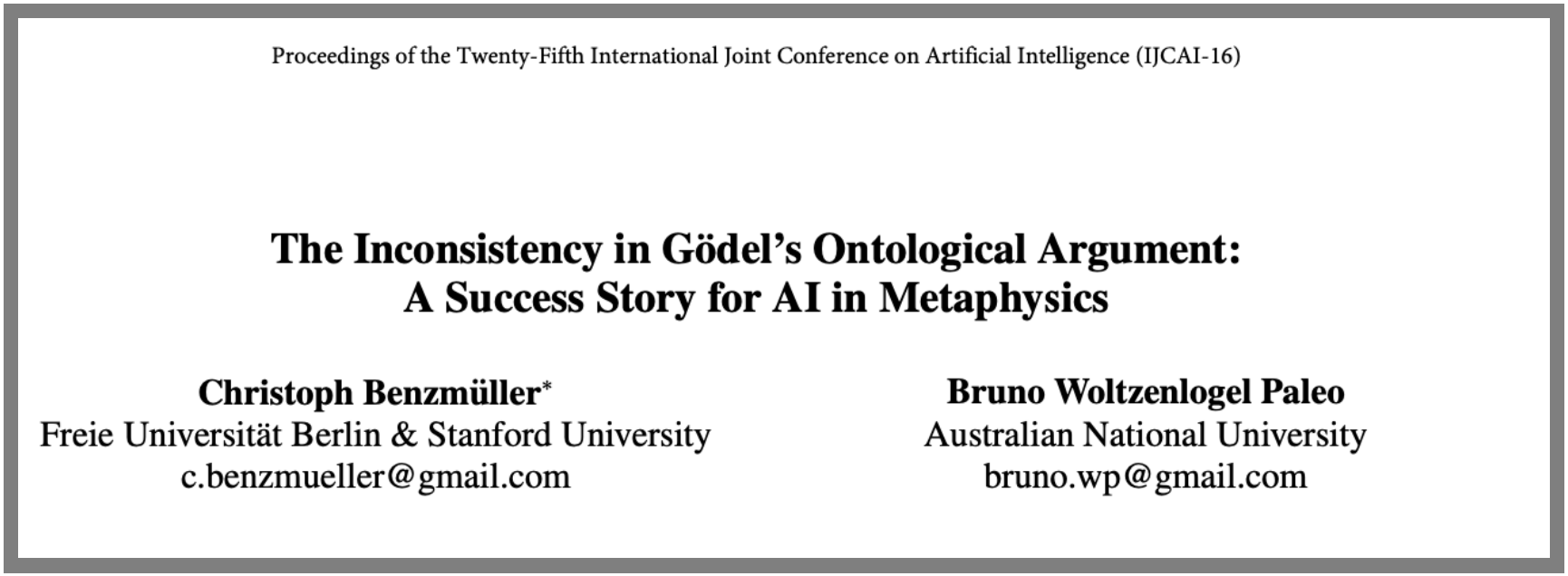
| | |
|---|---|
| Conjecture: | $\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ |
| Negated Conjecture: | $\neg\forall X_i \forall Y_i \exists R_{i\to i\to o}\ \neg(R\ X\ Y)$ |
| Clause normalisation: | $\exists X_i \exists Y_i \forall R_{i\to i\to o}\ \neg\neg(R\ X\ Y)$ |
| Clause normalisation: | $\neg\neg(R\ s\ t)$ |
| Clause normalisation: | $(R\ s\ t)$ |
| Only clause: | $c1 : (R\ s\ t)$ |
| Primitive Substitution: | $c2 : ((\lambda X_i \lambda Y_i\ \bot)\ s\ t)$ (i.e. $R \longleftarrow$ empty relation) |
| $\beta$-normalisation: | $c3 : \bot$ |

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"

$$\underbrace{\text{"New York is a city in the US"}}_{True}$$

$$\underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey"

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True}$$

$$\underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey"

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True}$$

$$\overbrace{\text{"Batman is a city in Turkey"}}^{True}$$

Peter knows "Batman is a city in Turkey"

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} \leftrightarrow \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey"

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{“New York is a city in the US”}}_{True} = \underbrace{\text{“Batman is a city in Turkey”}}_{True}$$

Peter knows "Batman is a city in Turkey"

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} = \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey""

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} = \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey""

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} = \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey""

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} = \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey""

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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"

$$\underbrace{\text{"New York is a city in the US"}}_{True} = \underbrace{\text{"Batman is a city in Turkey"}}_{True}$$

Peter knows "Batman is a city in Turkey""

**Need for Modal Logics:** $\Box_{\text{Belief}}$ $\Box_{\text{Time}}$ $\Box_{\text{Obligation}}$ $\Box_{\ldots}$

**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

International Joint Conference on Automated Reasoning
IJCAR 2018: Automated Reasoning pp 108-116 | Cite as

# The Higher-Order Prover Leo-III

Authors | Authors and affiliations

Alexander Steen, Christoph Benzmüller

Conference paper
**First Online:** 30 June 2018

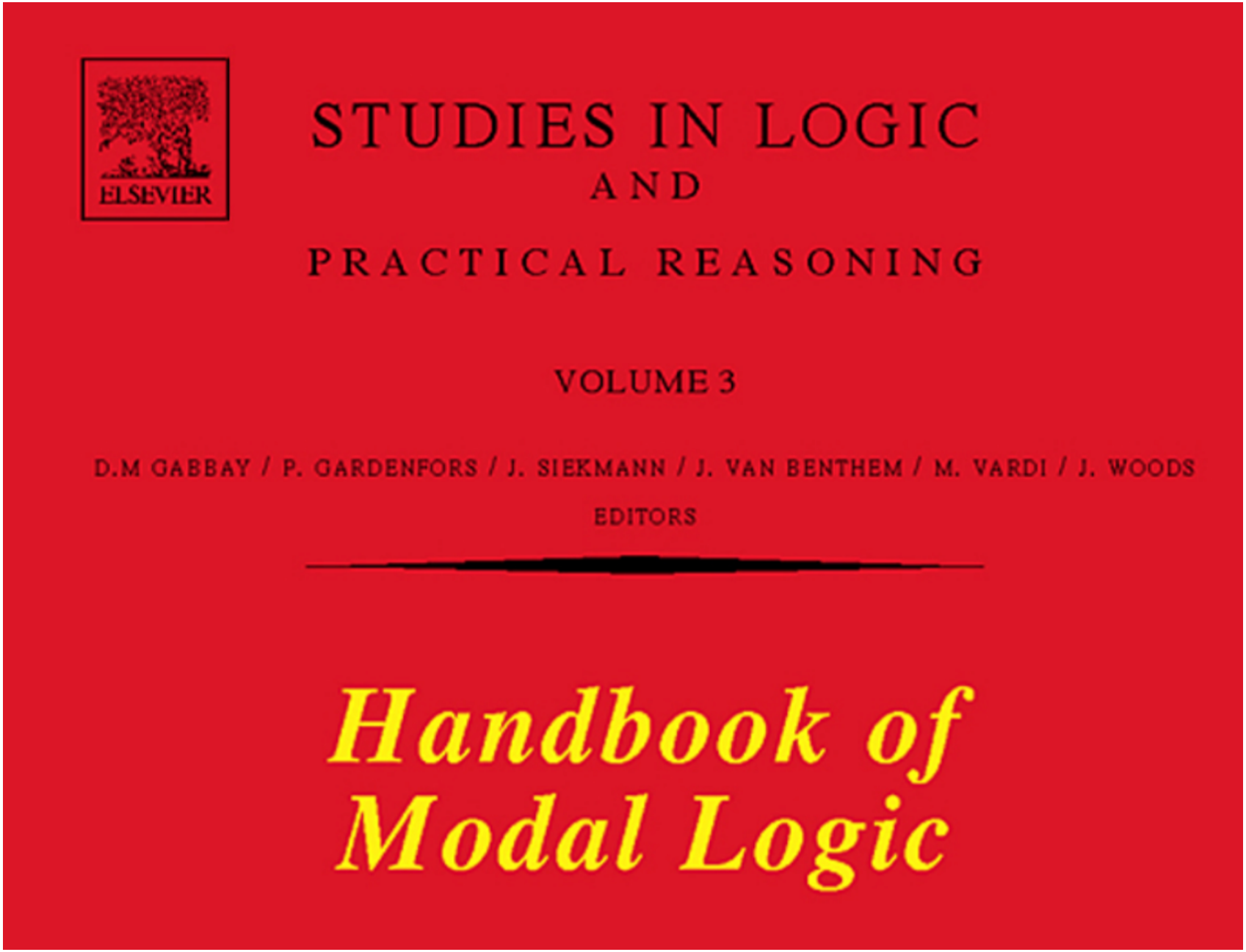


Part of the Lecture Notes in Computer Science book series (LNCS, volume 10900)

## 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?



# 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:

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

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**

ose elements we typically write as $p$, $q$, $r$ and
ents we typically write as $m$, $m'$, $m''$, and so
*nature* (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:

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

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 $\varphi$ 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, |
| $\mathfrak{M}, w \models \bot$ | | 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 $R^m wv$ we have $\mathfrak{M}, v \models \varphi$, |
| $\mathfrak{M}, w \models [m] \varphi$ | iff | for all $v \in W$ such that $R^m wv$ 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* in a model is a binary relation on $W$, and V position symbol $p$ in PROP a subset $V(p)$ $p$ is true. The first two components $(W, \{$ model. If there is only one relation in the $(W, R, V)$ 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.

**Metalanguage**

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 $\varphi$ 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, |
| $\mathfrak{M}, w \models \bot$ | | 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 $R^m wv$ we have $\mathfrak{M}, v \models \varphi$, |
| $\mathfrak{M}, w \models [m]\varphi$ | iff | for all $v \in W$ such that $R^m wv$ we have $\mathfrak{M}, v \models \varphi$. |

# Universal Logical Reasoning

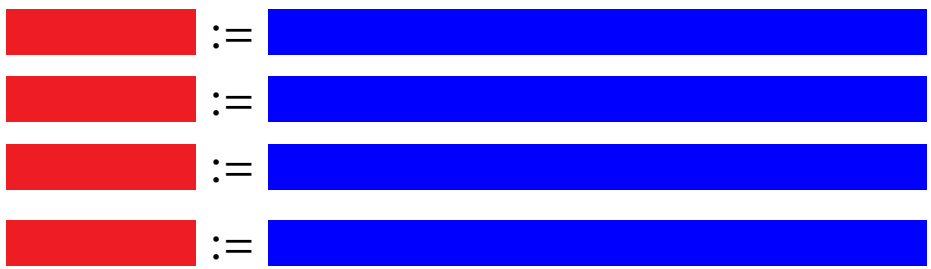
**Approach:** Shallow Semantic Embedding in HOL

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)**

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

**Validity**

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

**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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

$$[\Box\forall x\ Px] \equiv [\Box \overbrace{\Pi(\lambda x\ Px)}^{\forall x\ Px}] \quad \text{def. of } \forall$$

$$\equiv [\Box \overbrace{((\lambda\Phi\ \lambda w\ \Pi(\lambda x\ \Phi x w))(\lambda x\ Px))}^{\Pi}] \quad \text{def. of } \Pi$$

$$\equiv [\Box(\lambda w\ \Pi(\lambda x\ Pxw))] \quad \beta\text{-reduction}$$

$$\equiv [\overbrace{(\lambda\varphi\ \lambda w\ \Pi(\lambda v\ Rwv \to \varphi v))(\lambda w\ \Pi(\lambda x\ Pxw))}^{\Box}] \quad \text{def. of } \Box$$

$$\equiv [\lambda w\ \forall v\ (Rwv \to \forall x\ Pxv)] \quad \beta\text{-reduction}$$

$$\equiv \forall w\ \forall v\ (Rwv \to \forall x\ Pxv). \quad \text{def. of validity, } \beta\text{-reduction}$$

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

$$
\begin{array}{lll}
[\Box\forall x\ Px] & \equiv & [\Box\overbrace{\Pi(\lambda x\ Px)}^{\forall x\ Px}] \qquad \text{def. of } \forall \\
& \equiv & [\Box(\overbrace{(\lambda\Phi\ \lambda w\ \Pi(\lambda x\ \Phi x w))}^{\Pi}(\lambda x\ Px))] \qquad \text{def. of } \Pi \\
& \equiv & [\Box(\lambda w\ \Pi(\lambda x\ Pxw))] \qquad \beta\text{-reduction} \\
& \equiv & [\overbrace{(\lambda\varphi\ \lambda w\ \Pi(\lambda v\ Rwv \to \varphi v))}^{\Box}(\lambda w\ \Pi(\lambda x\ Pxw))] \qquad \text{def. of } \Box \\
& \equiv & [\lambda w\ \forall v\ (Rwv \to \forall x\ Pxv)] \qquad \beta\text{-reduction} \\
& \equiv & \forall w\ \forall v\ (Rwv \to \forall x\ Pxv). \qquad \text{def. of validity, } \beta\text{-reduction}
\end{array}
$$

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

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

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

$$
\begin{array}{lll}
[\Box\forall x\ Px] & \equiv & [\Box\overbrace{\Pi(\lambda x\ Px)}^{\forall x\ Px}] \qquad \text{def. of } \forall \\
& \equiv & [\Box\overbrace{((\lambda\Phi\ \lambda w\ \Pi(\lambda x\ \Phi x w))(\lambda x\ Px))}^{\Pi}] \qquad \text{def. of } \Pi \\
& \equiv & [\Box(\lambda w\ \Pi(\lambda x\ Pxw))] \qquad \beta\text{-reduction} \\
& \equiv & [\overbrace{(\lambda\varphi\ \lambda w\ \Pi(\lambda v\ Rwv \to \varphi v))(\lambda w\ \Pi(\lambda x\ Pxw))}^{\Box}] \qquad \text{def. of } \Box \\
& \equiv & [\lambda w\ \forall v\ (Rwv \to \forall x\ Pxv)] \qquad \beta\text{-reduction} \\
& \equiv & \forall w\ \forall v\ (Rwv \to \forall x\ Pxv). \qquad \text{def. of validity, } \beta\text{-reduction}
\end{array}
$$

# 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 $\Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ s)$ —no binder needed!!!
- $\Pi = \lambda\Phi_{\alpha\to(i\to o)}\ \lambda w_i\ \Pi_{(\alpha\to o)\to o}(\lambda x_\alpha\ \Phi x w)$

**Example (compositionality and $\lambda$-conversion at work; omitting types)**

$$
\begin{array}{lll}
[\Box\forall x\ Px] & \equiv & [\Box\overbrace{\Pi(\lambda x\ Px)}^{\forall x\ Px}] \qquad \text{def. of } \forall \\
& \equiv & [\Box\overbrace{((\lambda\Phi\ \lambda w\ \Pi(\lambda x\ \Phi x w))(\lambda x\ Px))}^{\Pi}] \qquad \text{def. of } \Pi \\
& \equiv & [\Box(\lambda w\ \Pi(\lambda x\ Pxw))] \qquad \beta\text{-reduction} \\
& \equiv & [\overbrace{(\lambda\varphi\ \lambda w\ \Pi(\lambda v\ Rwv \to \varphi v))(\lambda w\ \Pi(\lambda x\ Pxw))}^{\Box}] \qquad \text{def. of } \Box \\
& \equiv & [\lambda w\ \forall v\ (Rwv \to \forall x\ Pxv)] \qquad \beta\text{-reduction} \\
& \equiv & \forall w\ \forall v\ (Rwv \to \forall x\ Pxv). \qquad \text{def. of validity, } \beta\text{-reduction}
\end{array}
$$

# Universal Logical Reasoning in HOL

```
theory QML_S5U imports Main
begin
 typedecl i (* "type for possible worlds" *)
 typedecl μ (* "type for individuals" *)
 type_synonym σ = "(i⇒bool)"

 abbreviation mnot ("¬_"[52]53)     where "¬φ ≡ λw. ¬φ(w)"
 abbreviation mnotpred ("¬_"[52]53) where "¬Φ ≡ λx.λw. ¬Φ(x)(w)"
 abbreviation mand (infixr"∧"51)    where "φ∧ψ ≡ λw. φ(w)∧ψ(w)"
 abbreviation mor  (infixr"∨"50)    where "φ∨ψ ≡ λw. φ(w)∨ψ(w)"
 abbreviation mimp (infixr"→"49)    where "φ→ψ ≡ λw. φ(w)⟶ψ(w)"
 abbreviation mequ (infixr"≡"48)    where "φ≡ψ ≡ λw. φ(w)⟷ψ(w)"
 abbreviation mall ("∀")            where "∀Φ ≡ λw.∀x. Φ(x)(w)"
 abbreviation mallB (binder"∀"[8]9) where "∀x. φ(x) ≡ ∀φ"
 abbreviation mexi ("∃")            where "∃Φ ≡ λw.∃x. Φ(x)(w)"
 abbreviation mexiB (binder"∃"[8]9) where "∃x. φ(x) ≡ ∃φ"
 abbreviation mbox ("□")            where "□φ ≡ λw.∀v. φ(v)"
 abbreviation mdia ("◇")            where "◇φ ≡ λw.∃v. φ(v)"

 abbreviation valid ("⌊_⌋"[7]8)     where "⌊p⌋ ≡ ∀w. p w"

lemma "⌊□φ → φ⌋" sledgehammer by simp
lemma "⌊□φ → ◇φ⌋" sledgehammer by simp
lemma "⌊□φ → □(□φ)⌋" sledgehammer by simp
end
```

Higher-order modal logic $S5^U$ in HOL

```
theory Scott_S5U imports QML_S5U
begin
 consts P :: "(μ⇒σ)⇒σ"                                  (* Positive *)
 axiomatization where
  A1a: "⌊∀Φ. P(¬Φ) → ¬P(Φ)⌋" and
  A1b: "⌊∀Φ. ¬P(Φ) → P(¬Φ)⌋" and
  A2:  "⌊∀Φ Ψ. P(Φ) ∧ □(∀x. Φ(x) → Ψ(x)) → P(Ψ)⌋"
 definition G where "G(x) = (∀Φ. P(Φ) → Φ(x))"          (* Def. of God *)
 axiomatization where
  A3: "⌊P(G)⌋" and
  A4: "⌊∀Φ. P(Φ) → □(P(Φ))⌋"
 definition ess (infixr "ess" 85) where
  "Φ ess x = Φ(x) ∧ (∀Ψ. Ψ(x) → □(∀y. Φ(y) → Ψ(y)))"     (* Essence *)
 definition NE where "NE(x) = (∀Φ. Φ ess x → □(∃ Φ))"(* Nec. Existence *)
 axiomatization where
  A5:  "⌊P(NE)⌋"

 theorem T3: "⌊□(∃x. G(x))⌋"                          (* Necessarily, God ∃*)
  sledgehammer [provers = remote_leo2,verbose] (* LEO-II proof in 2,5sec *)
  by (metis (lifting, full_types)
     A1a A1b A2 A3 A4 A5 G_def NE_def ess_def)
end
```

Ontological argument in $S5^U$

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)

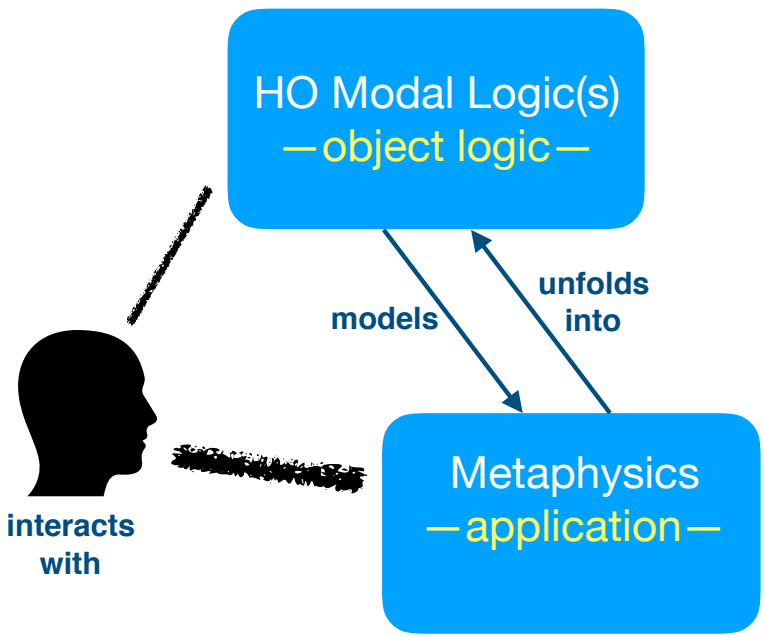
Further reading: [ECAI 2014, IJCAI 2016, Open Philosophy 2019]

# Universal Logical Reasoning in HOL

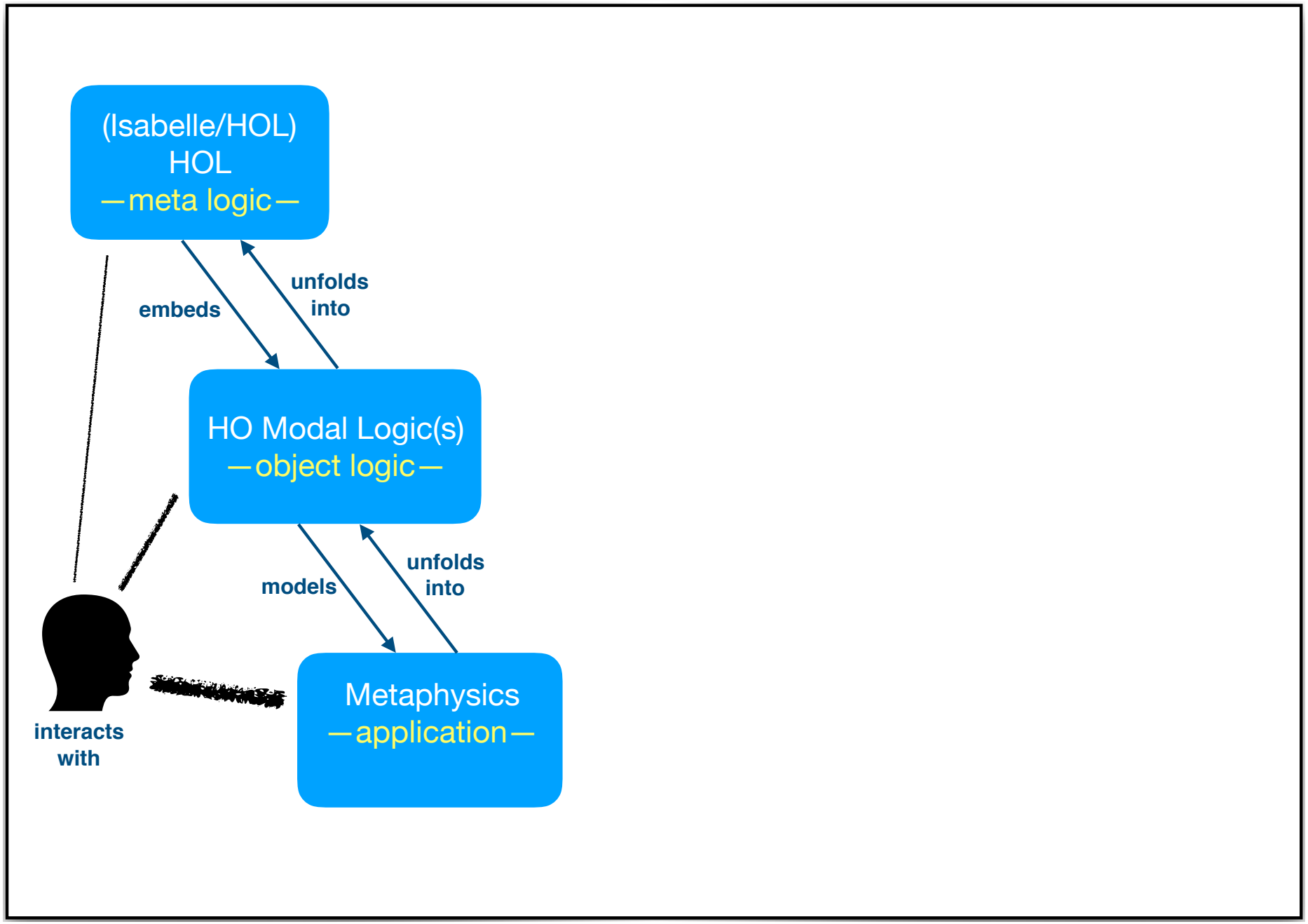
interacts with

Metaphysics
—application—

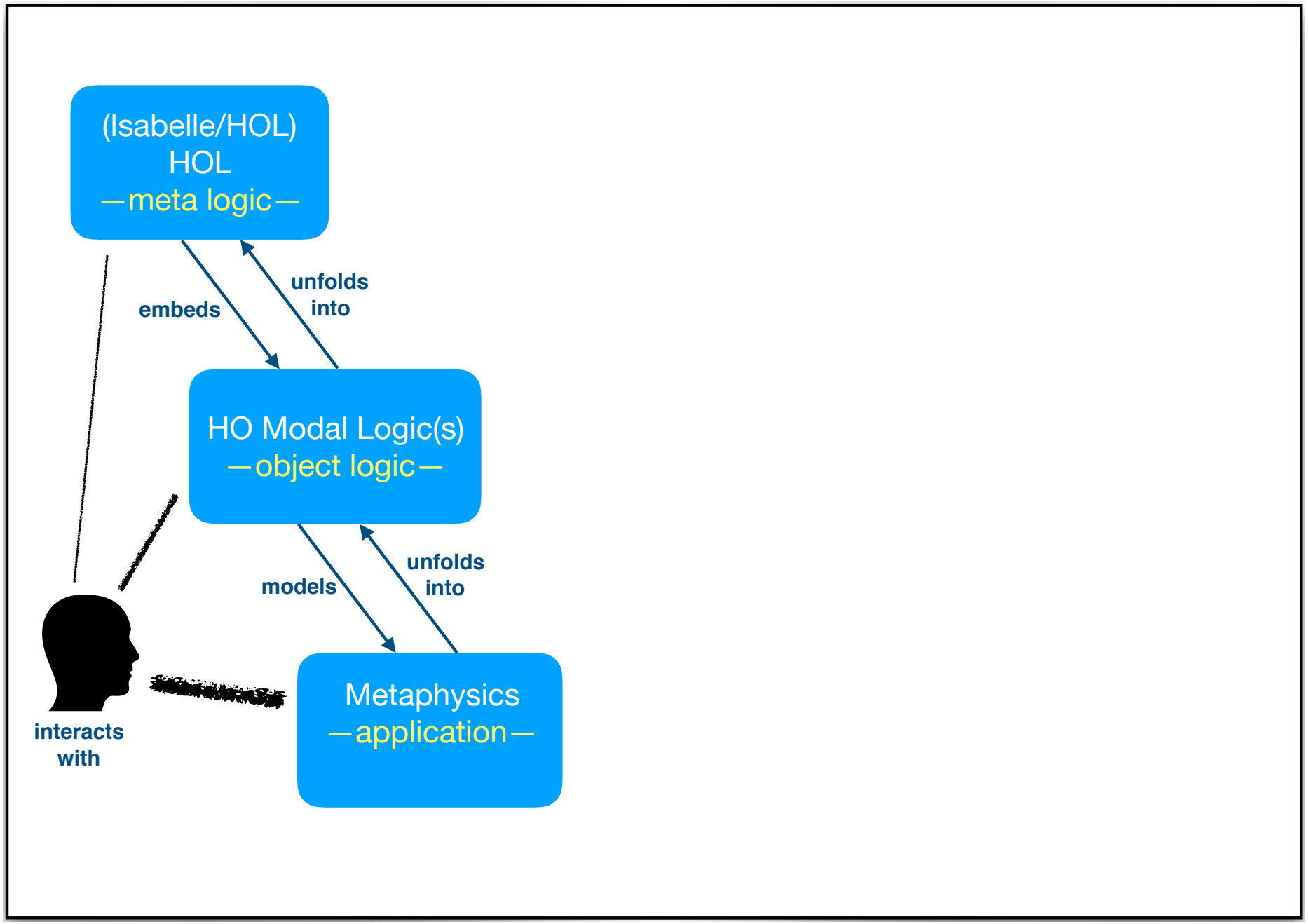
# Universal Logical Reasoning in HOL



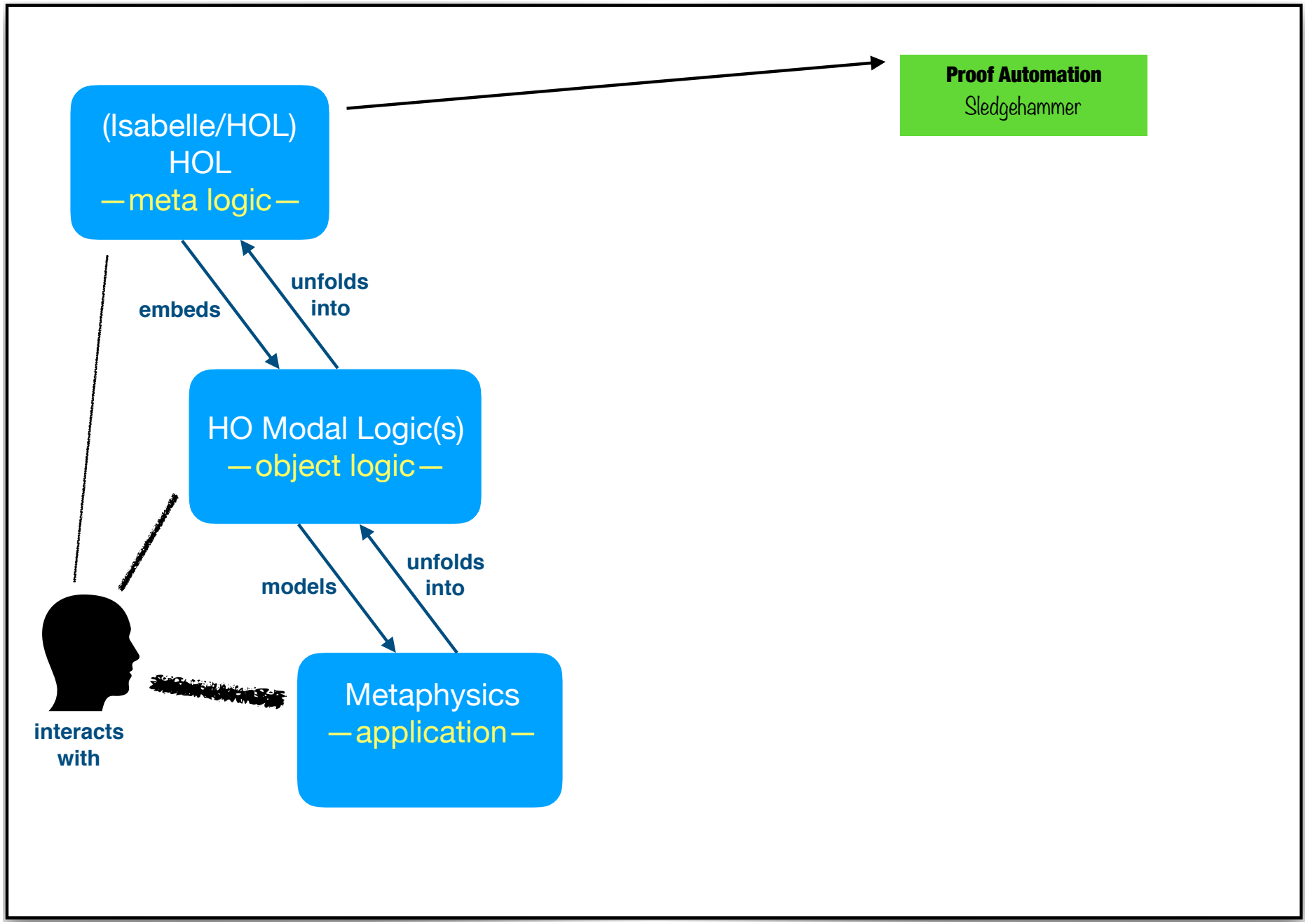
# Universal Logical Reasoning in HOL



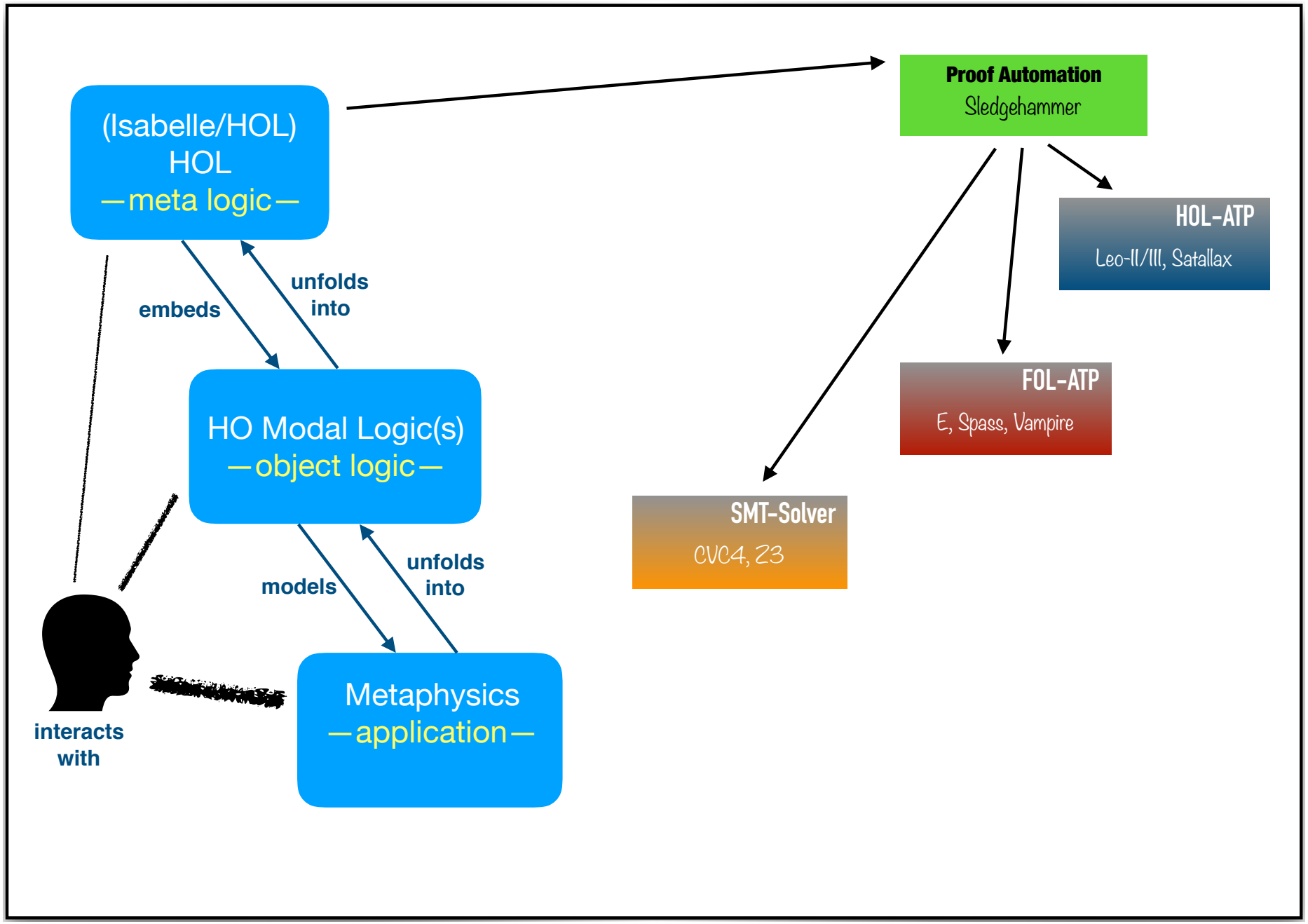
# Universal Logical Reasoning in HOL



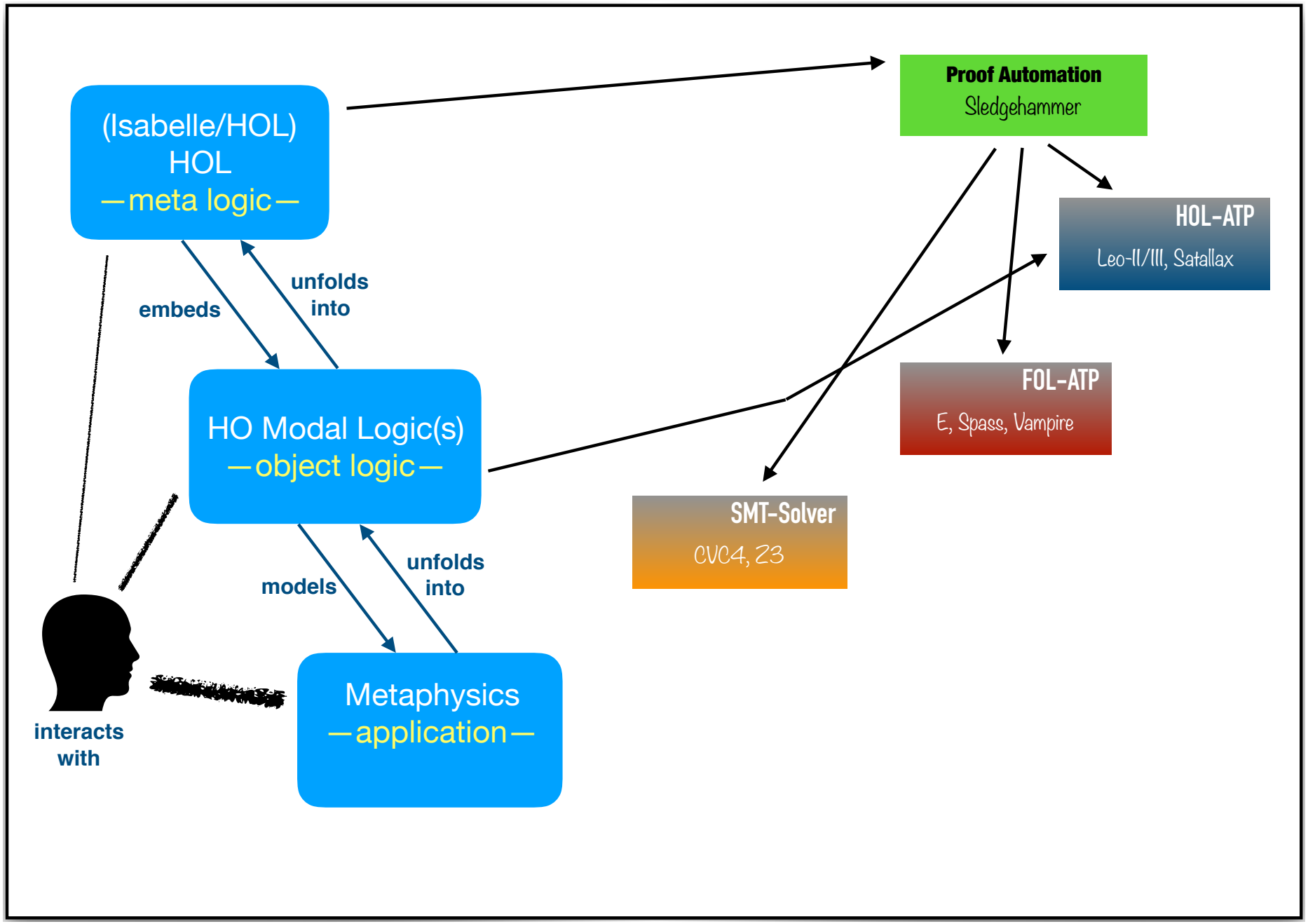
# Universal Logical Reasoning in HOL



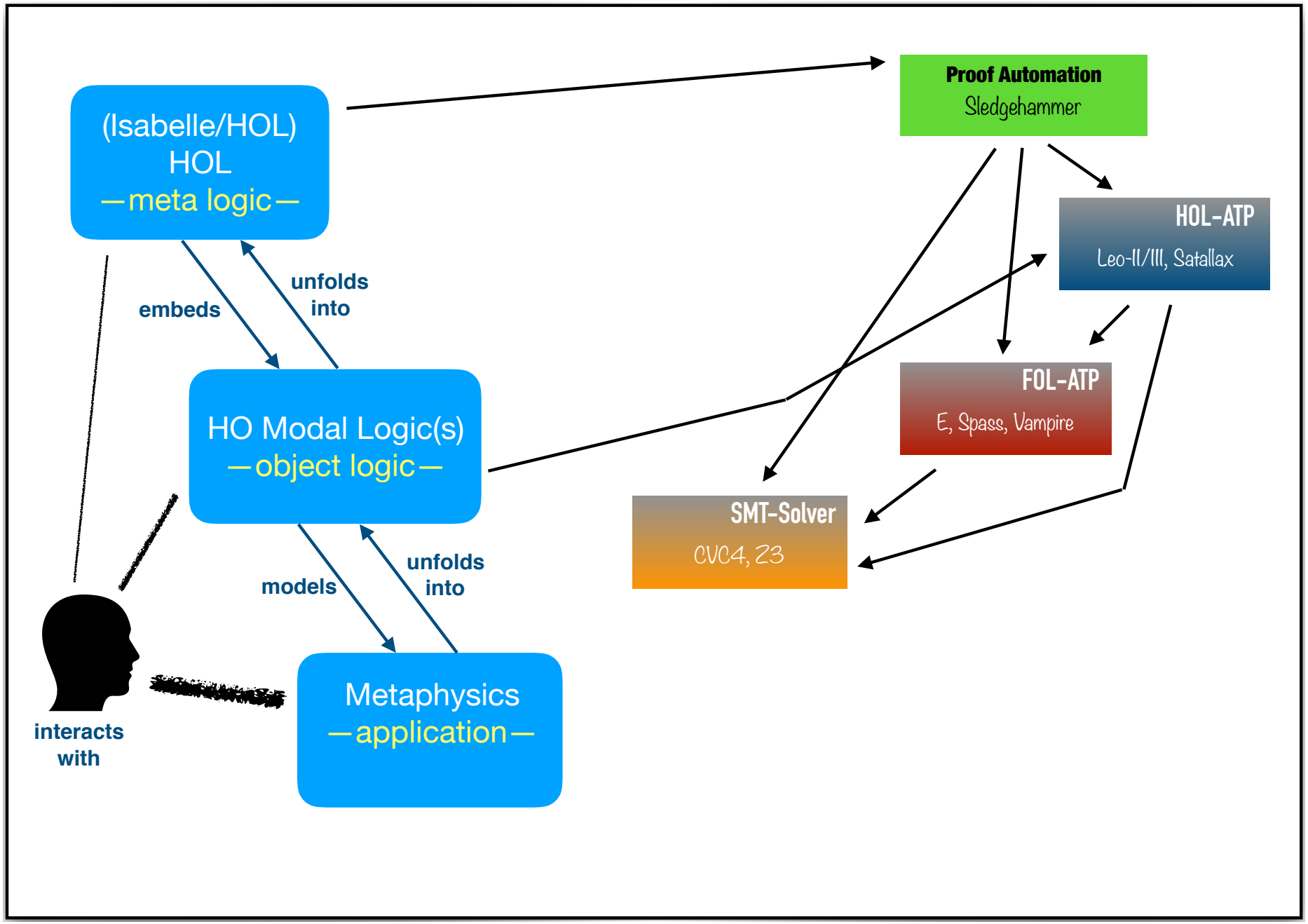
# Universal Logical Reasoning in HOL



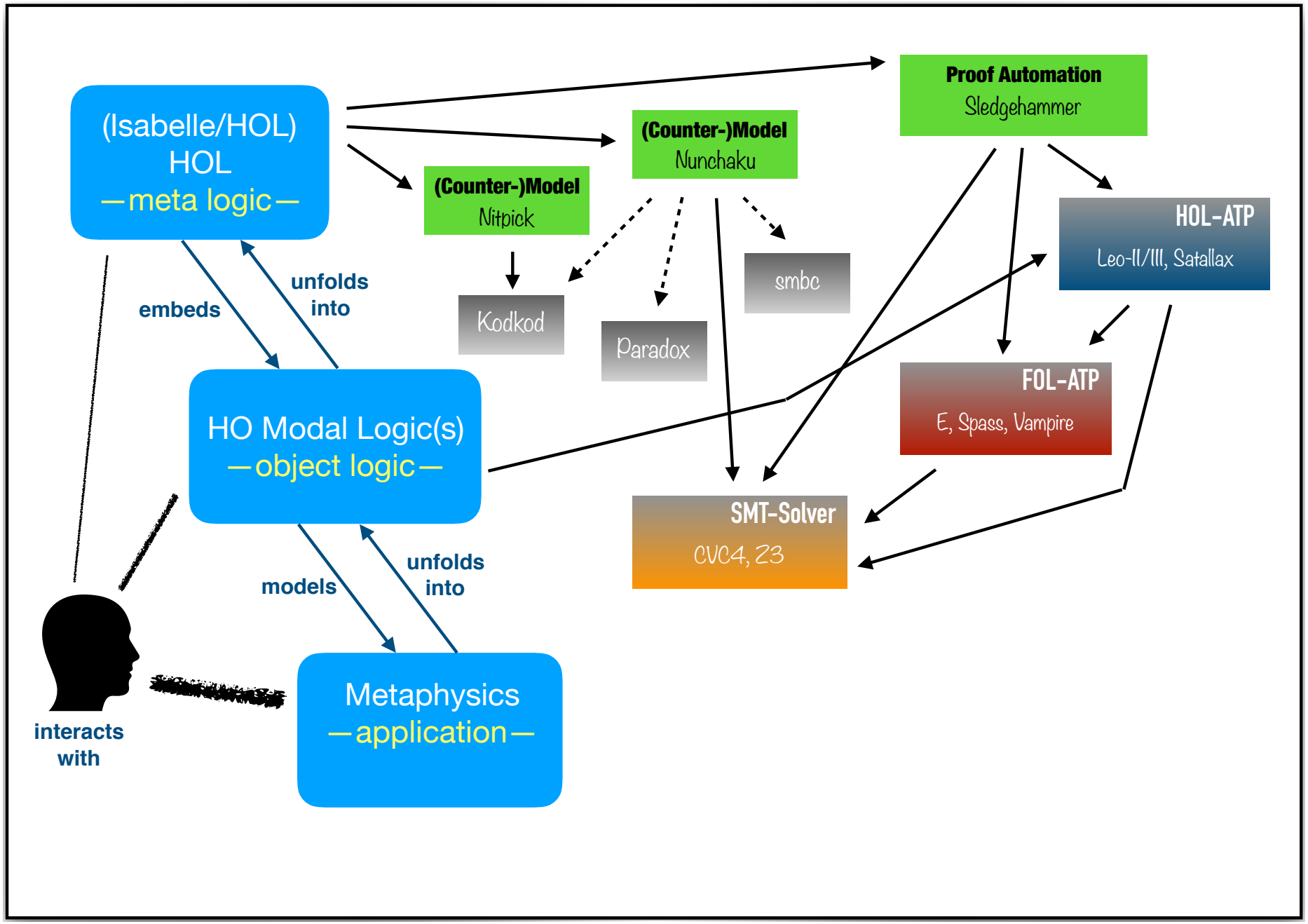
# Universal Logical Reasoning in HOL



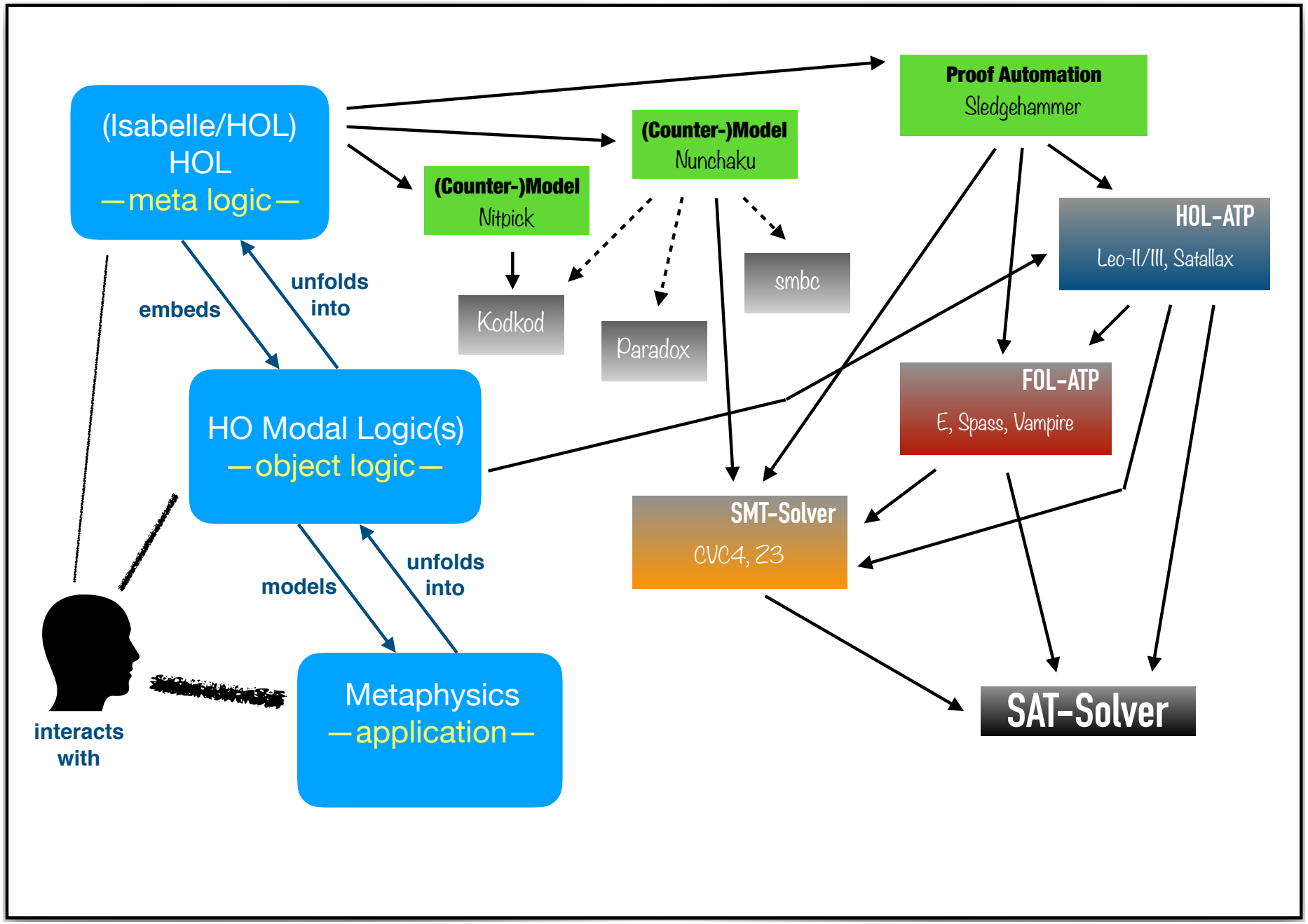
# Universal Logical Reasoning in HOL



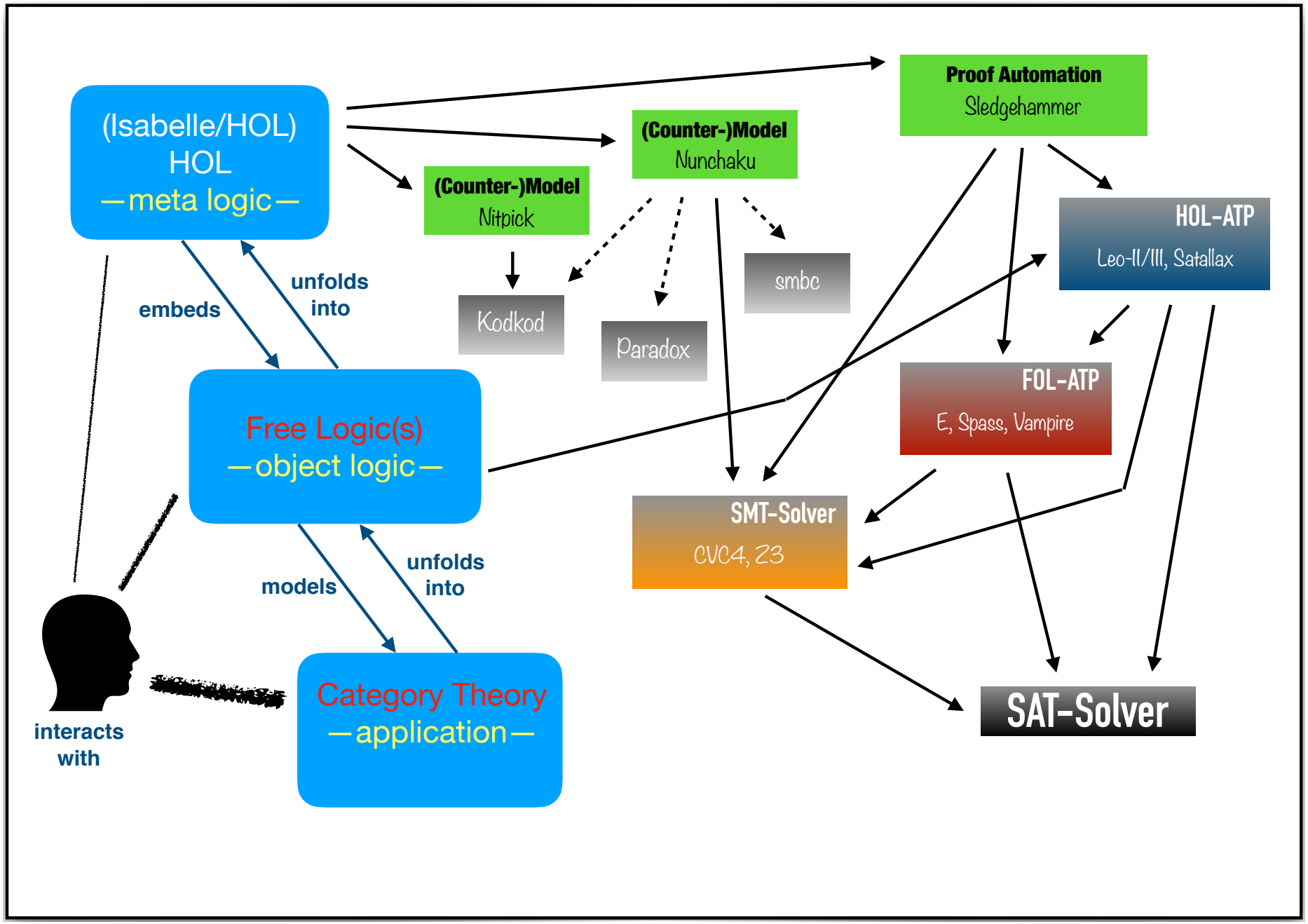
# Universal Logical Reasoning in HOL



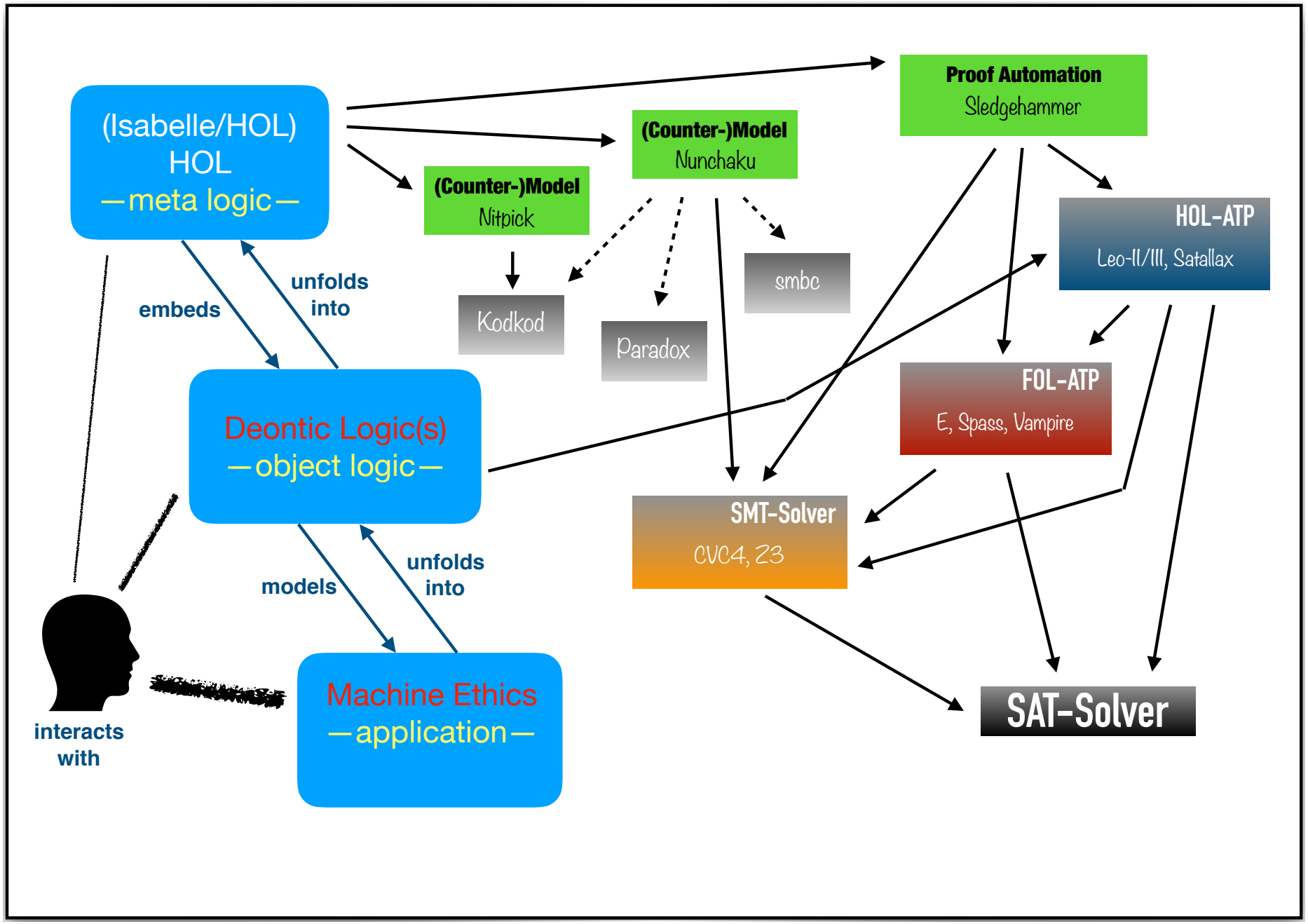
# Universal Logical Reasoning in HOL



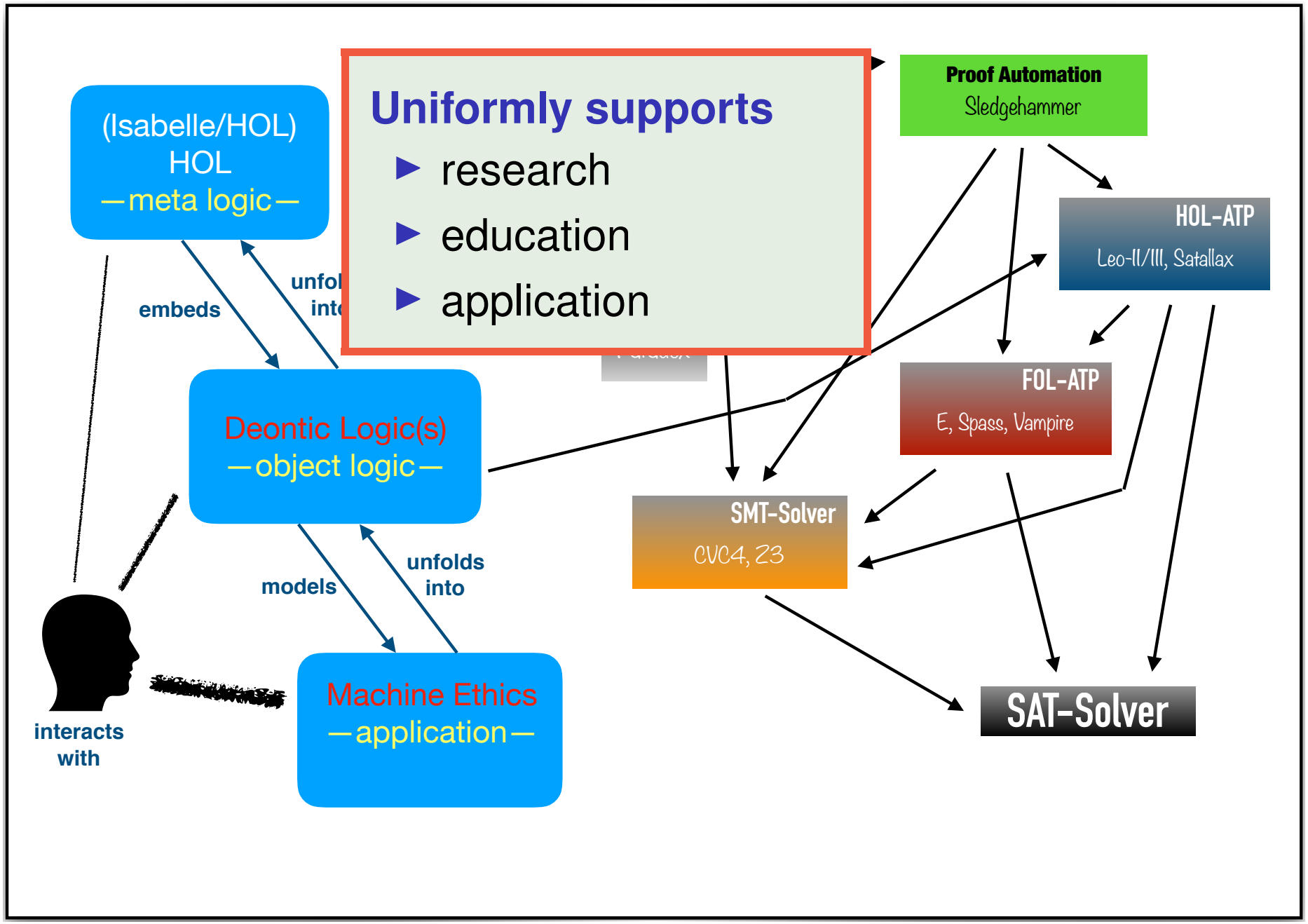
# Universal Logical Reasoning in HOL



# 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”

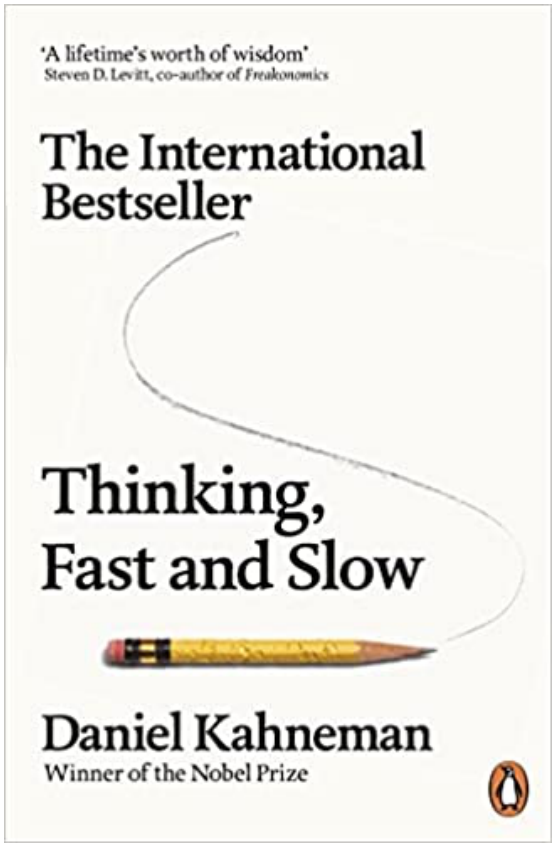
- Do our current AI systems know what they are doing?
- Do we know what we are doing when we entrust such AI systems with increasingly critical decisions?
- Is normative directionlessness and unpredictability the core character of future AI systems?
- How to control such systems?

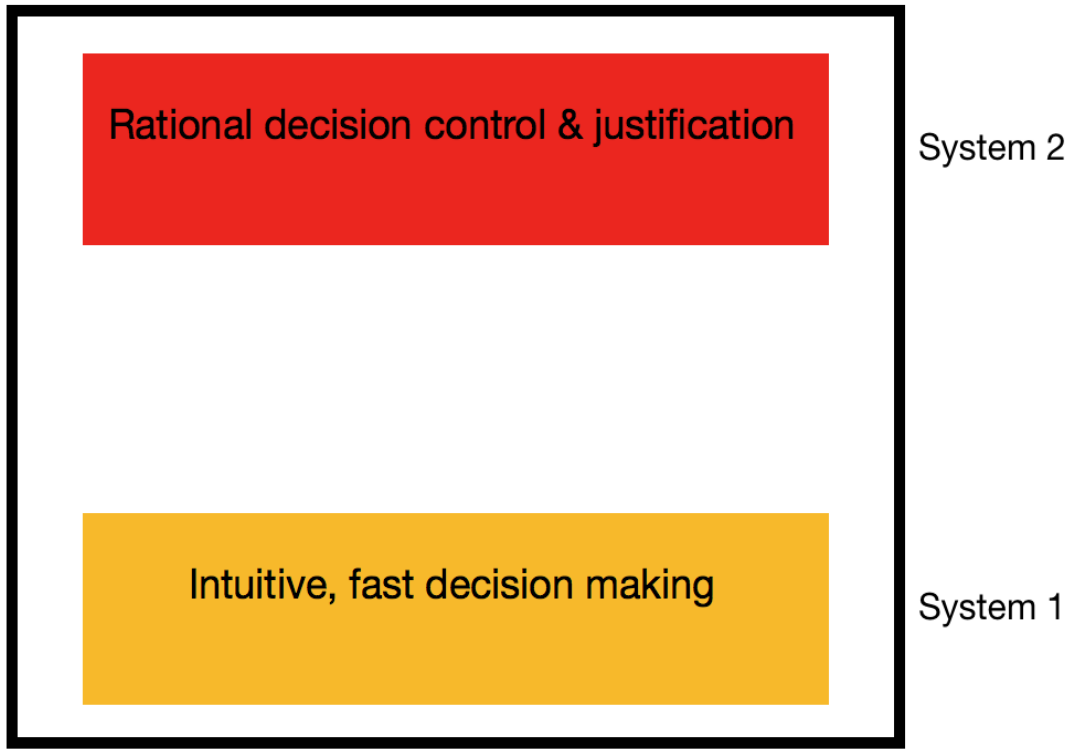
# How to create trust in AI? How to control?


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?

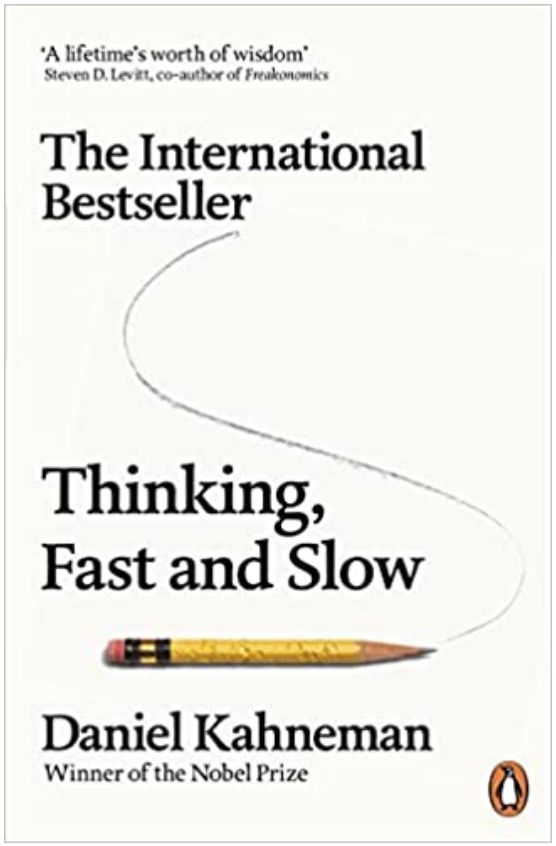
# How to create trust? How to control?

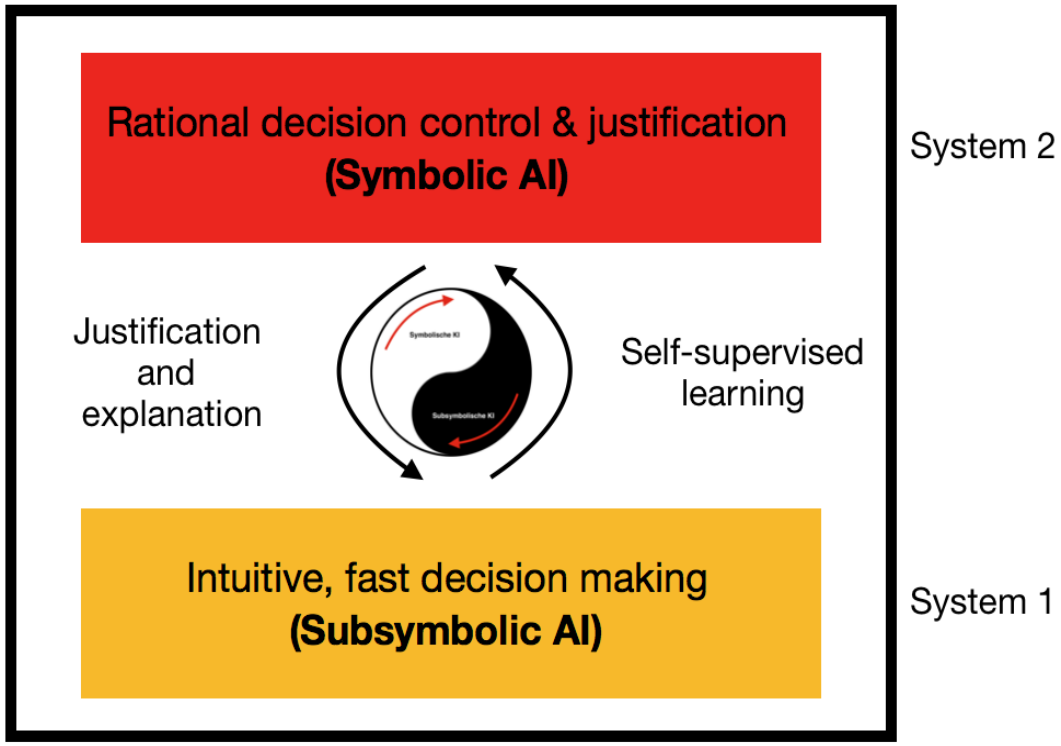



See e.g. also: **J. Haidt, The Emotional Dog and its Rational Tail, 2001**

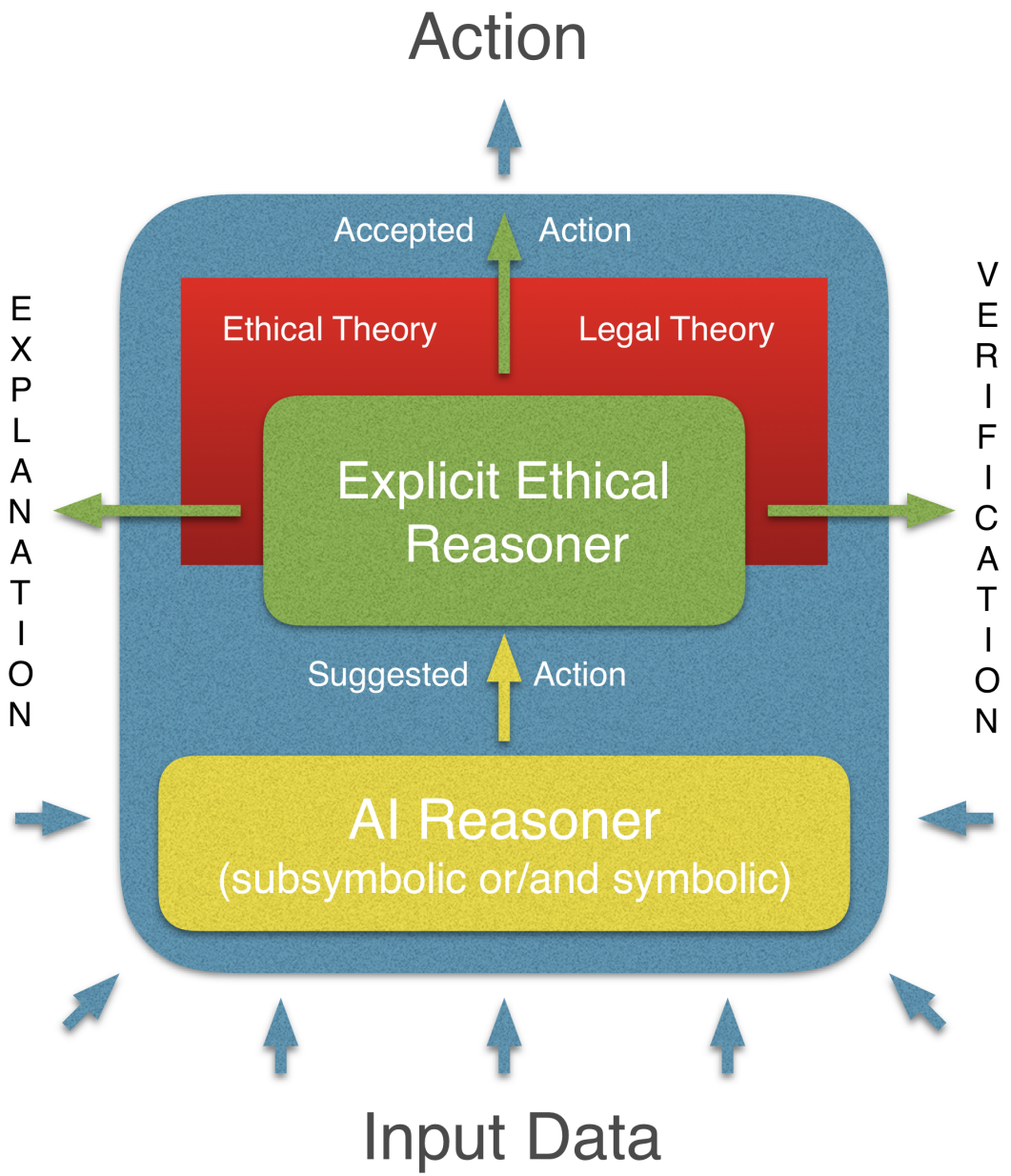
# How to create trust? How to control?



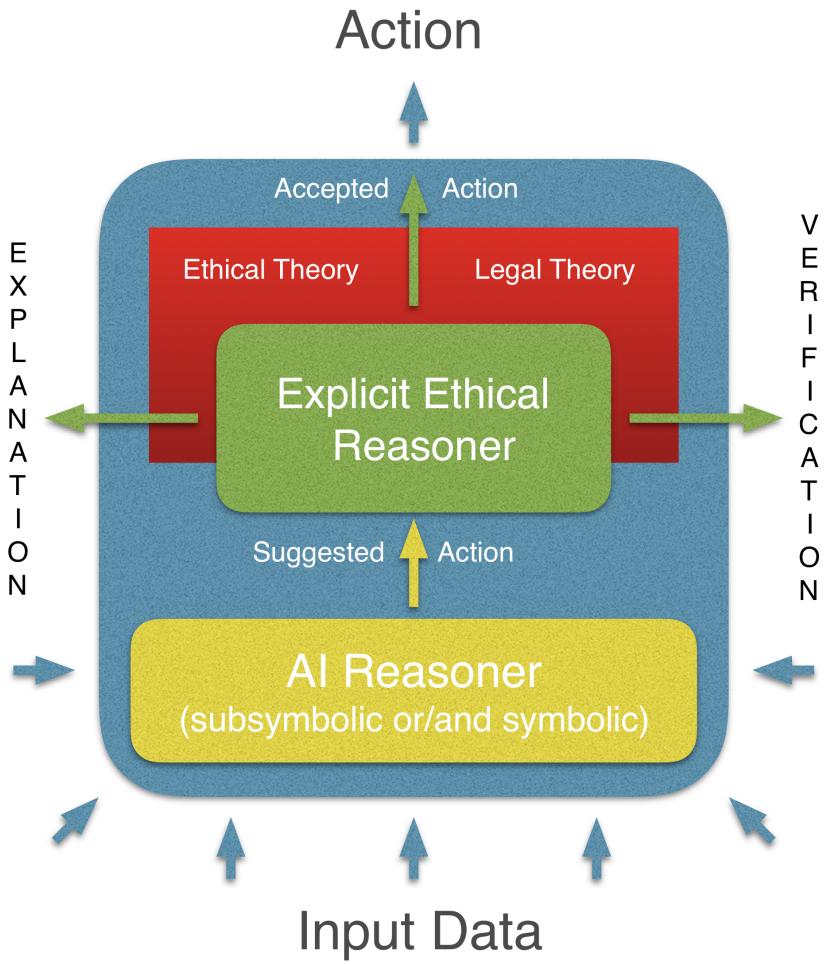




See e.g. also: **J. Haidt, The Emotional Dog and its Rational Tail, 2001**

# 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

ELSEVIER

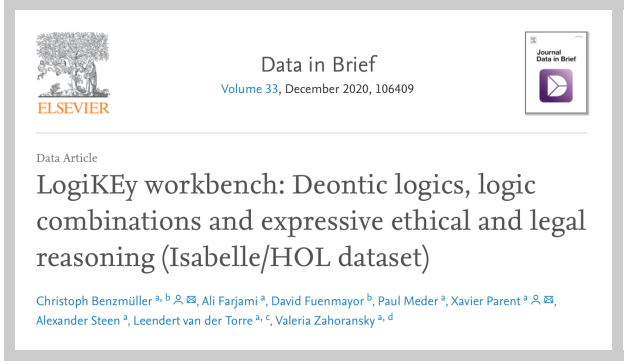
Artificial Intelligence
Volume 287, October 2020, 103348

Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support ☆

Christoph Benzmüller [b, a], Xavier Parent [a], Leendert van der Torre [a, c]

Show more

https://doi.org/10.1016/j.artint.2020.103348 Get rights and content



ELSEVIER

Data in Brief
Volume 33, December 2020, 106409

Data Article

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

Christoph Benzmüller [a, b], Ali Farjami [a], David Fuenmayor [b], Paul Meder [a], Xavier Parent [a], Alexander Steen [a], Leendert van der Torre [a, c], Valeria Zahoransky [a, d]

arXiv Cornell University

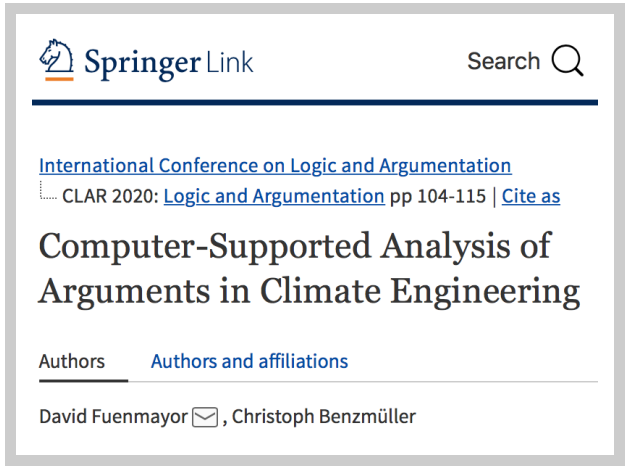
Computer Science > Artificial Intelligence

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 Fuenmayor, Bertram Lomfeld



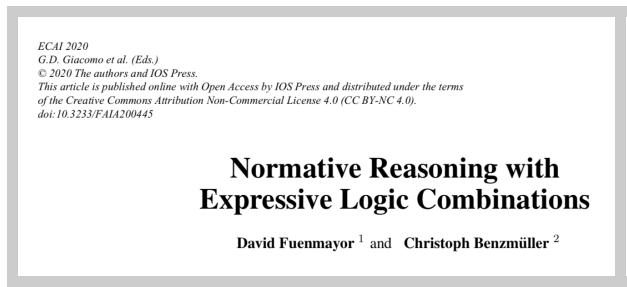
Springer Link Search

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 Fuenmayor, Christoph Benzmüller



ECAI 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/FAIA200445

Normative Reasoning with Expressive Logic Combinations

David Fuenmayor [1] and Christoph Benzmüller [2]

Springer Link 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 Fuenmayor, Christoph Benzmüller

(some recent papers)

X. Parent

L. van der Torre

A. Farjami

D. Fuenmayor

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) $\overset{\text{Interaction}}{\longleftrightarrow}$ Universal Logical Reasoning (Deliberation)

German Conference on Artificial Intelligence (Künstliche Intelligenz)

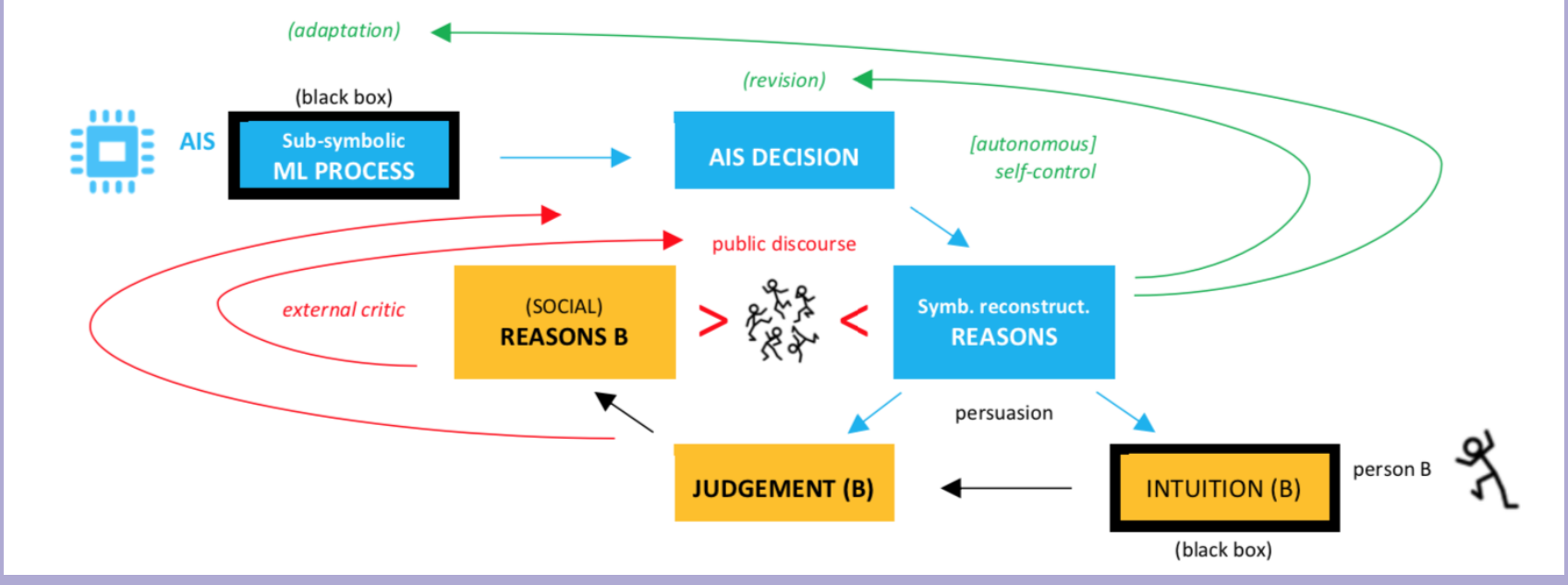
KI 2020: KI 2020: Advances in Artificial Intelligence pp 251-258 | Cite as

# Reasonable Machines: A Research Manifesto



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