

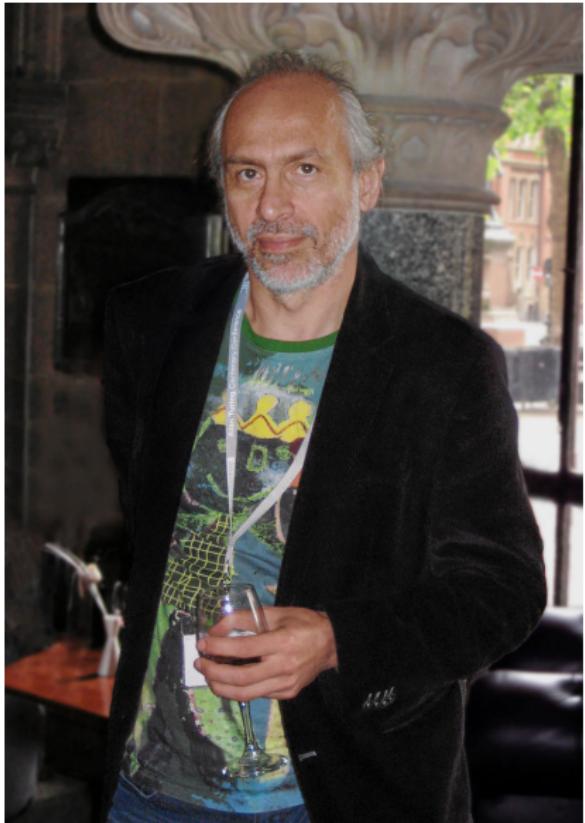
Higher-Order Automated Theorem Proving

– Why has there been such a Resistance? –

C. Benzmüller
FU Berlin & U Luxembourg

ANDREI-60, Tbilisi, Georgia, May 22, 2019

Andrei 60 — Happy Birthday!



And thank you! Competition is only fun if there are strong athletes to chase.



forallP.Leo III

with A. Steen, M. Wisniewski

What is Leo-III?

- ▶ ATP for classical HOL based on extensional higher-order paramodulation
- ▶ integrates SMT- & FO-ATPs
- ▶ supports all TPTP syntax dialects
- ▶ native support for **more than 120 modal logics** (normal quantified modal logics)
- ▶ including native support for **quantified SLD and DDL**
- ▶ Website: <http://page.mi.fu-berlin.de/lex/leo3/>

Recent Independent Evaluation:

- ▶ GRUNGE: A Grand Unified ATP Challenge [arXiv:1903.02539], accepted for CADE



forallP.Leo III

with A. Steen, M. Wisniewski

What is Leo-III?

- ▶ ATP for classical HOL based on extensional higher-order paramodulation
- ▶ integrates SMT- & FO-ATPs
- ▶ supports all TPTP syntax dialects
- ▶ native support for **more than 120 modal logics** (normal quantified modal logics)
- ▶ including native support for **quantified SLD and DDL**
- ▶ Website: <http://page.mi.fu-berlin.de/lex/leo3/>

Recent Independent Evaluation:

- ▶ GRUNGE: A Grand Unified ATP Challenge [arXiv:1903.02539], accepted for CADE



forallP.Leo III

with A. Steen, M. Wisniewski

What is Leo-III?

- ▶ ATP for classical HOL based on extensional higher-order paramodulation
- ▶ integrates SMT- & FO-ATPs
- ▶ supports all TPTP syntax dialects
- ▶ native support for **more than 120 modal logics** (normal quantified modal logics)
- ▶ including native support for **quantified SLD and DDL**
- ▶ Website: <http://page.mi.fu-berlin.de/lex/leo3/>

Recent Independent Evaluation:

- ▶ GRUNGE: A Grand Unified ATP Challenge [arXiv:1903.02539], accepted for CADE

GRUNGE: A Grand Unified ATP Challenge

Chad E. Brown, Thibault Gauthier, Cezary Kaliszyk, Geoff Sutcliffe, Josef Urban

(Submitted on 6 Mar 2019)

This paper describes a large set of related theorem proving problems obtained by translating theorems from the HOL4 standard library into multiple logical formalisms. The formalisms are in higher-order logic (with and without type variables) and first-order logic (possibly with multiple types, and possibly with type variables). The resultant problem sets allow us to run automated theorem provers that support different logical formats on corresponding problems, and compare their performances. This also results in a new "grand unified" large theory benchmark that emulates the ITP/ATP hammer setting, where systems and metasystems can use multiple ATP formalisms in complementary ways, and jointly learn from the accumulated knowledge.

System	TH1-I	TH0-I	TH0-II	TF1-I	TF0-I	TF0-II	FOF-I	FOF-II	Union
agsyHOL	1374	1187							1605
Beagle				2007	2047				2531
cocATP	899	599							1000
CSE_E						4251	3102		4480
CVC4				4851	3991				5252
E				4277	3622	4618	3844		5118
HOLyHammer	5059								5059
iProver						2778	2894		3355
iProverMo'				2435	1639				2699
LEO-II	2579	1923							3213
Leo-III	6668	5018	3485	3458	4032	3421			7062
Metis						2353	474		2356
Princess				3646	2138				3849
Prover9						2894	1742		3128
Satallax	2207	1292							2494
SPASS						2850	3349		3821
Vampire				4837	4693	4008	4928		5929
Zipperp'n	2252	2161	3771	3099	2576				4203
Union	6824	5209	3771	4608	5732	5073	5165	5108	7377

Table 2. Number of theorems proved, out of 12140.

What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

Why Lambda's should be your friends?

Why Lambda's should be your friends?

$\{x | \text{green}(x) \wedge \text{alien}(x)\}$

corresponds to

$\lambda x_\alpha. \text{green}_{\alpha \rightarrow o} x_\alpha \wedge \text{alien}_{\alpha \rightarrow o} x_\alpha$

∪ **is abbreviation for** $\lambda \varphi_{\alpha \rightarrow o}. \lambda \psi_{\alpha \rightarrow o}. (\lambda x_\alpha. \varphi_{\alpha \rightarrow o} x_\alpha \vee \psi_{\alpha \rightarrow o} x_\alpha)$

∩ **is abbreviation for** $\lambda \varphi_{\alpha \rightarrow o}. \lambda \psi_{\alpha \rightarrow o}. (\lambda x_\alpha. \varphi_{\alpha \rightarrow o} x_\alpha \wedge \psi_{\alpha \rightarrow o} x_\alpha)$

∅ **is abbreviation for** $\lambda x_\alpha. \perp$

etc.

Why Lambda's should be your friends?

One important aspect:

$$\exists N_{\overline{\alpha^n} \rightarrow \beta} \forall \overline{z^n} N(z^1, \dots, z^n) = B_\beta$$

versus

$$\lambda \overline{z^n}. B_\beta$$

And another one:

$$\forall x_\alpha. P(x)$$

versus

$$\Pi(\lambda x_\alpha. P x)$$

And there is more: ...

Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$$

$$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$$

$$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$$

$$\forall B, C. (B \cup C = C \cup B)$$

$$\forall B, C. (B \cap C = C \cap B)$$

$$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$$

$$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$$

Proof Goal:

$$\forall B, C, D.$$

$$B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

% SPASS---3.0

% Problem : SET171+3

% SPASS beiseite: Ran out of time.

% E---0.999

% Problem : SET171+3

% Failure: Resource limit exceeded
(time)

% Vampire---9.0

% Problem : SET171+3

% Result : Theorem 68.6s

Performance: LEO-II + E

Eureka --- Thanks to Corina!

Total Reasoning Time: 0.03s

LEO-II (Proof Found!)



Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$$

$$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$$

$$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$$

$$\forall B, C. (B \cup C = C \cup B)$$

$$\forall B, C. (B \cap C = C \cap B)$$

$$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$$

$$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$$

Proof Goal:

$$\forall B, C, D.$$

$$B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

% SPASS---3.0

% Problem : SET171+3

% SPASS beiseite: Ran out of time.

% E---0.999

% Problem : SET171+3

% Failure: Resource limit exceeded
(time)

% Vampire---9.0

% Problem : SET171+3

% Result : Theorem 68.6s

Performance: LEO-II + E

Eureka --- Thanks to Corina!

Total Reasoning Time: 0.03s

LEO-II (Proof Found!)



Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$$

$$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$$

$$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$$

$$\forall B, C. (B \cup C = C \cup B)$$

$$\forall B, C. (B \cap C = C \cap B)$$

$$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$$

$$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$$

Proof Goal:

$$\forall B, C, D.$$

$$B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

```
% SPASS---3.0
% Problem : SET171+3
% SPASS beiseite: Ran out of time.

% E---0.999
% Problem : SET171+3
% Failure: Resource limit exceeded
% (time)

% Vampire---9.0
% Problem : SET171+3
% Result : Theorem 68.6s
```

Performance: LEO-II + E

Eureka --- Thanks to Corina!

Total Reasoning Time: 0.03s

LEO-II (Proof Found!)



Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$$

$$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$$

$$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$$

$$\forall B, C. (B \cup C = C \cup B)$$

$$\forall B, C. (B \cap C = C \cap B)$$

$$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$$

$$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$$

Proof Goal:

$$\forall B, C, D.$$

$$B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

% SPASS---3.0

% Problem : SET171+3

% SPASS beiseite: Ran out of time.

% E---0.999

% Problem : SET171+3

% Failure: Resource limit exceeded
(time)

% Vampire---9.0

% Problem : SET171+3

% Result : Theorem 68.6s

Performance: LEO-II + E

Eureka --- Thanks to Corina!

Total Reasoning Time: 0.03s

LEO-II (Proof Found!)



$$I = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} s(x, y) \cos\left(\frac{n\pi x}{L}\right) \cos\left(\frac{m\pi y}{W}\right)$$

Results

Problem	Vamp. 9.0	LEO+Vamp.	LEO-II+E
014+4	114.5	2.60	0.300
017+1	1.0	5.05	0.059
066+1	—	3.73	0.029
067+1	4.6	0.10	0.040
076+1	51.3	0.97	0.031
086+1	0.1	0.01	0.028
096+1	5.9	7.29	0.033
143+3	0.1	0.31	0.034
171+3	68.6	0.38	0.030
580+3	0.0	0.23	0.078
601+3	1.6	1.18	0.089
606+3	0.1	0.27	0.033
607+3	1.2	0.26	0.036
609+3	145.2	0.49	0.039
611+3	0.3	4.00	0.125
612+3	111.9	0.46	0.030
614+3	3.7	0.41	0.060
615+3	103.9	0.47	0.035
623+3	—	2.27	0.282
624+3	3.8	3.29	0.047
630+3	0.1	0.05	0.025
640+3	1.1	0.01	0.033
646+3	84.4	0.01	0.032
647+3	98.2	0.12	0.037

Problem	Vamp. 9.0	LEO+Vamp.	LEO-II+E
648+3	98.2	0.12	0.037
649+3	117.5	0.25	0.037
651+3	117.5	0.09	0.029
657+3	146.6	0.01	0.028
669+3	83.1	0.20	0.041
670+3	—	0.14	0.067
671+3	214.9	0.47	0.038
672+3	—	0.23	0.034
673+3	217.1	0.47	0.042
680+3	146.3	2.38	0.035
683+3	0.3	0.27	0.053
684+3	—	3.39	0.039
716+4	—	0.40	0.033
724+4	—	1.91	0.032
741+4	—	3.70	0.042
747+4	—	1.18	0.028
752+4	—	516.00	0.086
753+4	—	1.64	0.037
764+4	0.1	0.01	0.032

Vamp. 9.0: 2.80GHz, 1GB memory, 600s time limit
LEO+Vamp.: 2.40GHz, 4GB memory, 120s time limit
LEO-II+E: 1.60GHz, 1GB memory, 60s time limit

Test: May 2019

```
%-----  
% File      : SET171+3 : TPTP v7.2.0. Released v2.2.0.  
% Domain    : Set Theory  
% Problem   : Union distributes over intersection  
%             ((B union (C intersection D)) = ((B union C) intersection (B union C))  
  
% END OF SYSTEM OUTPUT  
% RESULT: SOT_7huDyR - E---2.3 says Timeout - CPU = 60.04 WC = 60.17  
% OUTPUT: SOT_7huDyR - E---2.3 says None - CPU = 0 WC = 0  
  
% END OF SYSTEM OUTPUT  
% RESULT: SOT_7huDyR - SPASS---3.9 says Timeout - CPU = 60.03 WC = 60.16  
% OUTPUT: SOT_7huDyR - SPASS---3.9 says None - CPU = 0 WC = 0  
  
% END OF SYSTEM OUTPUT  
% RESULT: SOT_7huDyR - Vampire---4.3 says Theorem - CPU = 2.29 WC = 2.26  
% OUTPUT: SOT_7huDyR - Vampire---4.3 says Refutation - CPU = 2.29 WC = 2.26  
%------
```

Test: May 2019

```
%-----  
% File      : SET623+3 : TPTP v7.2.0. Released v2.2.0.  
% Domain    : Set Theory  
% Problem   : Associativity of symmetric difference  
%             (X symDiff Y) symDiff Z = X symDiff (Y symDiff Z)  
  
% END OF SYSTEM OUTPUT  
% RESULT: SET623+3 - E---2.3 says Timeout - CPU = 60.07 WC = 60.20  
% OUTPUT: SET623+3 - E---2.3 says None - CPU = 0 WC = 0  
  
% END OF SYSTEM OUTPUT  
% RESULT: SET623+3 - SPASS---3.9 says Timeout - CPU = 60.04 WC = 60.15  
% OUTPUT: SET623+3 - SPASS---3.9 says None - CPU = 0 WC = 0  
  
% END OF SYSTEM OUTPUT  
% RESULT: SET623+3 - Vampire---4.3 says Timeout - CPU = 60.04 WC = 58.06  
% OUTPUT: SET623+3 - Vampire---4.3 says None - CPU = 0 WC = 0  
%------
```

Expressivity Matters — Cantor's Theorem — Primitive Substitution

Cantor's theorem: The set of all subsets of A, that is, the power set of A, has a strictly greater cardinality than A itself.

In HOL Cantor's theorem (surjective version) can be encoded as

$$\neg \exists G_{i \rightarrow (i \rightarrow o)} \forall F_{i \rightarrow o} \exists X_i. GX = F$$

HO ATPs can solve this problem very efficiently.

Solution includes the detection and application of the diagonalisation argument.

In the Leo (and TPS) provers this works with **primitive substitution**:

$$F \leftarrow \lambda X_i. \neg H_{i \rightarrow (i \rightarrow o)} XX$$

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

Expressivity Matters — Boolos' Example — More on Comprehension

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

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. $D(f(s(s(s(s(1))))), s(s(s(s(1))))))$

Induction proof: from (4) and (5), we get $\forall x D(x)$, hence $D(f(s(s(s(s(1))))), s(s(s(s(1))))))$ by \forall -elimination.

But induction is not given, hence the first order proof consists of brute force modus ponens applications: infeasible number of steps ($2^{(2^{\dots^2})}$ with 64K '2s')

Expressivity Matters — Boolos' Example — More on Comprehension

[Boolos, A curious inference, J. Philosophical Logic, 16:1-12, 1987]

Comprehension axioms

$$\exists N_{\overline{\alpha^n} \rightarrow \beta} \forall \overline{z^n} N(z^1, \dots, z^n) = B_\beta$$

Can be avoided: use λ -binding construct to denote N

Instances of comprehension axioms in Boolos' proof:

$$\exists N \forall z N(z) \leftrightarrow (\forall X X(1) \wedge \forall y (X(y) \rightarrow X(s(y))) \rightarrow X(z))$$

$$\exists E \forall z E(z) \leftrightarrow (N(z) \wedge D(z))$$

Central idea: “assume the induction principle holds for number z – corresponding to $N(z)$ – then we can show for any predicate X a property $X(z)$ by induction.”

The proof employs the following lemmata:

Lemma 1:

$$N(1), \forall y (N(y) \rightarrow N(s(y))), N(s(s(s(1))))$$

$$E(1), \forall y (E(y) \rightarrow E(s(y))), E(s(1))$$

Lemma 2: $\forall n N(n) \rightarrow \forall x (N(x) \rightarrow E(f(n, x)))$

The theorem itself is then an easy application of the two lemmata.

Expressivity Matters — Boolos' Example — More on Comprehension

¹By the comprehension principle of second order logic, $\exists N \forall z(Nz \leftrightarrow \forall X[X1 \& \forall y(Yx \rightarrow Xsy) \rightarrow Xz])$, ²and then for some N , $\exists E \forall z(Ez \leftrightarrow Nz \& Dz)$.

³**Lemma 1:** $\frac{3.1}{3}N1$; $\frac{3.2}{3}\forall y(Ny \rightarrow Nsy)$; $\frac{3.3}{3}Nssss1$; $\frac{3.4}{3}E1$; $\frac{3.5}{3}\forall y(Ey \rightarrow Esy)$; $\frac{3.6}{3}Es1$;

⁴**Lemma 2:** $\forall n(Nn \rightarrow (\forall x(Nx \rightarrow Efnx)))$

Proof: ^{4.1}By comprehension, $\exists M \forall n(Mn \leftrightarrow \forall x(Nx \rightarrow Efnx))$. ^{4.2}We want $\forall n(Nn \rightarrow Mn)$. ^{4.3}Enough to show $\frac{4.3.1}{4.3}M1$ and $\frac{4.3.2}{4.3}\forall n(Mn \rightarrow Msn)$, for then if $\frac{4.4}{4.4}Nn$, $\frac{4.5}{4.5}Mn$.

^{4.3.1}**M1:** ^{4.3.1.1}Want $\forall x(Nx \rightarrow Ef1x)$. ^{4.3.1.2}By comprehension, $\exists Q \forall x(Qx \leftrightarrow Ef1x)$.

^{4.3.1.3}Want $\forall x(Nx \rightarrow Qx)$. ^{4.3.1.4}Enough to show $\frac{4.3.1.4.1}{4.3.1.4.1}Q1$ and $\frac{4.3.1.4.2}{4.3.1.4.2}\forall x(Qx \rightarrow Qsx)$.

^{4.3.1.4.1}**Q1:** ^{4.3.1.4.1.1}Want $Ef11$. ^{4.3.1.4.1.2}But $f11 = s1$ by (1) and ^{4.3.1.4.1.3} $Es1$ by Lemma 1.

^{4.3.1.4.2} $\forall x(Qx \rightarrow Qsx)$: ^{4.3.1.4.2.1}Suppose Qx , ^{4.3.1.4.2.2}i.e. $Ef1x$. ^{4.3.1.4.2.3}By (2) $f1sx = ssf1x$; ^{4.3.1.4.2.4}by Lemma 1 twice, $Ef1sx$. ^{4.3.1.4.2.5}Thus Qsx and ^{4.3.1.4.2.6} $M1$.

^{4.3.2} $\forall n(Mn \rightarrow Msn)$: ^{4.3.2.1}Suppose Mn , ^{4.3.2.2}i.e. $\forall x(Nx \rightarrow Efnx)$.

^{4.3.2.3}Want Msn , ^{4.3.2.4}i.e. $\forall x(Nx \rightarrow Efsnx)$. ^{4.3.2.5}By comprehension, $\exists P \forall x(Px \leftrightarrow Efsnx)$. ^{4.3.2.6}Want $\forall x(Nx \rightarrow Px)$. ^{4.3.2.7}Enough to show ^{4.3.2.7.1} $P1$ and ^{4.3.2.7.2} $\forall x(Px \rightarrow Psx)$.

^{4.3.2.7.1}**P1:** ^{4.3.2.7.1.1}Want $Efsn1$. ^{4.3.2.7.1.2}But $fsn1 = s1$ by (1) and ^{4.3.2.7.1.3} $Es1$ by Lemma 1.

^{4.3.2.7.2} $\forall x(Px \rightarrow Psx)$: ^{4.3.2.7.2.1}Suppose Px , ^{4.3.2.7.2.2}i.e. $Efsnx$; ^{4.3.2.7.2.3}thus $Nfsnx$. ^{4.3.2.7.2.4}Want $Efsnsx$. ^{4.3.2.7.2.5}Since $Nfsnx$ and Mn , $Efnfsnx$.

^{4.3.2.7.2.6}But by (3) $fnnfsnx = fsnsx$; ^{4.3.2.7.2.7}thus $Efsnsx$.

⁵By Lemma 1, $Nssss1$. ⁶By Lemma 2, $Efssss1ssss1$. ⁷Thus, $Dfssss1ssss1$, as desired.

Formalization in OMEGA and Mizar:

[BenzmüllerBrown, The curious inference of Boolos in MIZAR and OMEGA, Studies in Logic, Grammar, and Rhetoric, volume 10(23), pp. 299-388, 2007.]

Earlier paper: [BenzmüllerKerber, A Lost Proof, 2001].

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

<u>Ontologischer Beweis</u>		<u>FEB 10, 1970</u>
$P(\varphi)$	φ is positive	($\varphi \in P$)
At 1	$P(\varphi), P(\psi) \vdash P(\varphi \wedge \psi)$	At 2 $P(\varphi) \vdash P(\neg \varphi)$
φ	$\varphi(x) = (\varphi)[P(\varphi) \supset \varphi(x)]$	(God)
φ	$\varphi \text{ Emx} = (\psi)[\psi(x) \supset N(y)[\varphi(y) \supset \psi(y)]]$	(Emaxy x)
$P \supset_N q$	$= N(p \supset q)$	Necessity
At 2	$P(\varphi) \supset N P(\varphi)$	{ because it follows $\neg \varphi \supset N \neg \varphi$ } from the nature of the property
Th.	$G(x) \supset G \text{ Em. } x$	
Df.	$E(x) \equiv (\varphi)[\varphi \text{ Emx} \supset N \exists x \varphi(x)]$	necessary Existence
At 3	$P(E)$	
Th.	$G(x) \supset N(\exists y) G(y)$	
hence	$(\exists x) G(x) \supset N(\exists y) G(y)$	
"	$M(\exists x) G(x) \supset M N(\exists y) G(y)$	
"	$\supset N(\exists y) G(y)$	$M =$ permuting
any two instances of x are mech. equivalent		
exclusive or " and for any number of numerants		

$M(x) G(x)$: means all pos. prop. w.r.t. com-
 patible
 This is true because of:
 At 4: $P(\varphi), \varphi \supset \psi \vdash P(\psi)$ which impl.
 $\begin{cases} x=x & \text{is positive} \\ x \neq x & \text{is negative} \end{cases}$
 But if a system S of pos. prop. were incons.
 It would mean, that the num.prop. s (which
 is positive) would be $x \neq x$

Positive means positive in the moral aesth.
 sense (independently of the accidental structure of
 the world). Only ~~the~~ ⁱⁿ the at. time. It me-
 ans also "affirmation" as opposed to "privation"
 (or crushing of privation). This supports the pl. part
 $\neg \varphi$ is not $(\exists x) N \neg \varphi(x)$ otherwise $\varphi(x) \supset x \neq x$
 hence $x \neq x$ (positive) $\neg \varphi(x) \supset$ negating At.
 or the explosion of $\varphi(x)$
 i.e. the formal form in terms of elem. prop. contains a
 Member without negation.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

Ontologischer Beweis

Feb 10, 1970

$P(\varphi)$ φ is positive ($\Leftrightarrow \varphi \in P$)

At 1 $P(\varphi), P(\psi) \vdash P(\varphi \wedge \psi)$ • At 2 $P(\varphi) \vdash P(\neg \varphi)$

P1 $G(x) \equiv (\varphi)[P(\varphi) \supset \varphi(x)]$ (God)

P2 $\varphi \text{ Emx} \equiv (\psi)[\forall x(\varphi(x) \supset N(y)[\varphi(y) \supset \psi(y)])]$ (Existence)

$P \supset_N q = N(p \supset q)$ Necessity

At 2 $\begin{array}{l} P(\varphi) \supset N P(\varphi) \\ \sim P(\varphi) \supset N \sim P(\varphi) \end{array} \left. \begin{array}{l} \text{because it follows} \\ \text{from the nature of the} \\ \text{property} \end{array} \right\}$

Th. $G(x) \supset \text{Em. } x$

Df. $E(x) \equiv (\varphi)[\varphi \text{ Emx} \supset N \exists x \varphi(x)]$ necessary Existence

At 3 $P(E)$

Th. $G(x) \supset N(\exists y) G(y)$

thus $(\exists x) G(x) \supset N(\exists y) G(y)$

" $M(\exists x) G(x) \supset M N(\exists y) G(y)$

" $\supset N(\exists y) G(y)$ Mi = pernitens

any two instances of x are nec. equivalent

exclusive or * and for any number of numerants

$M(\exists x) G(x)$ means "all
possible This is:
At 4: $P(\varphi), \varphi \supset \psi$
~~True~~ $\left\{ \begin{array}{l} x=x \text{ is pr} \\ x \neq x \text{ is } \end{array} \right.$
But if a system S is
it would mean, that
(positive) would be $x \neq x$



Positive means positive in the moral aesthetic sense (independently of the accidental structure of the world). Only ~~the~~ the at time. It may also mean "affirmation" as opposed to "privation" (or crushing privation). This supports the plausibility

\supset of φ positive $\supset (\chi) N \sim P(\chi) \supset$ otherwise $\varphi(x) \supset x \neq x$
hence $x \neq x$ positive $\supset x = x$ negative At
or the opposite of φ At

dog
i.e. the formal form in terms of elem. prop. contains a member without negation.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

Ontologischer Beweis Feb 10, 1970

$P(\varphi)$ φ is positive ($\Leftrightarrow \varphi \in P$)

At 1 $P(\varphi), P(\varphi) \supset P(\varphi \wedge \psi) \vdash P(\varphi \wedge \psi)$ At 2 $P(\varphi) \supset P(\neg \varphi)$

P1 $G(x) \equiv (\varphi)[P(\varphi) \supset \varphi(x)]$ (God)

P2 $\varphi \text{ Emx} \equiv (\psi)[\psi(x) \supset N(y)[\varphi(y) \supset \psi(y)]]$ (Emx \neq x)

$P \supset_N q = N(p \supset q)$ Necessity

At 2 $\begin{array}{l} P(\varphi) \supset N(P(\varphi)) \\ \sim P(\varphi) \supset N \sim P(\varphi) \end{array} \left. \begin{array}{l} \text{because it follows} \\ \text{from the nature of the} \\ \text{property} \end{array} \right\}$

Th. $G(x) \supset G \text{ Em. } x$

Df. $E(x) \equiv (\varphi)[\varphi \text{ Emx} \supset N \exists x \varphi(x)]$ necessary Existence

At 3 $P(E)$

Th. $G(x) \supset N(\exists y) G(y)$

 hence $(\exists x) G(x) \supset N(\exists y) G(y)$

 " $M(\exists x) G(x) \supset M N(\exists y) G(y)$

 " $\supset N(\exists y) G(y)$ M = permuting

any two instances of x are nec. equivalent

exclusive or and for any number of numerants

$M(\exists x) G(x)$ means "the system
is possible" This is:

At 4: $P(\varphi), \varphi \supset \psi$

~~$\exists x \{ x=x \text{ is pr.}$~~

~~$\exists x \{ x \neq x \text{ is pr.}\}$~~

But if a system S is possible,
it would mean, that the num.prop. S (which
is positive) would be $x \neq x$



Positive means positive in the moral aesthetic sense (independently of the accidental structure of the world). Only ~~in the art. time~~ It may also mean "affirmation" as opposed to "privatization (or creation of privation)." This supports the plausibility of the argument.

$\exists x \varphi \text{ positive } \supset (x) N \sim \varphi(x) \text{ Otherwise } \varphi(x) \supset x \neq x$
hence $x \neq x$ (positive) $\supset x=x$ (negative)
or the opposite of $\varphi(x) \supset x \neq x$

dog
i.e. the formal form in terms of elem. prop. contains a member without negation.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

Ontologischer Beweis FEB 10, 1970

$P(\varphi)$ φ is positive ($\Leftrightarrow \varphi \in P$)

At 1 $P(\varphi), P(\psi) \vdash P(\varphi \wedge \psi)$ At 2 $P(\varphi) \vdash P(\neg \varphi)$

Pl $G(x) \equiv (\varphi)[P(\varphi) \Rightarrow \varphi(x)]$ (Good)

P2 $\varphi \text{ Emx} \equiv (\psi)[\psi(x) \supset N(y)[\varphi(y) \supset \psi(y)]]$ (Emx = x)

$\neg \varphi = N(\varphi, \neg \varphi)$ Necessity

At 2 $\begin{cases} P(\varphi) \supset N(P(\varphi)) \\ \neg \varphi \supset N \neg P(\varphi) \end{cases}$ because it follows from the nature of the property

Th. $G(x) \supset G \text{ Emx}$

Df. $E(x) \equiv (\varphi)[\varphi \text{ Emx} \supset N \exists x \varphi(x)]$ necessary Existence

At 3 $P(E)$

Th. $G(x) \supset N(\forall y) G(y)$

$M(x) F(x)$: means all pos. prop. w.r.t. com-
patible
This is true because of:
At 4: $P(\varphi), \varphi \supset \psi \vdash P(\psi)$ which impl.
True { $x=x$ is positive
False { $x \neq x$ is negative
But if a system S of pos. prop. were incons.
It would mean, that the num.prop. S (which is positive) would be $x \neq x$

Positive means positive in the moral aesth.
sense. (independently of the accidental structure of
the world). (Only ~~in the art. time~~ It may
also mean "attribution" as opposed to "privation"
(or contains negation). This is more subtle and

(Main) Difference between Gödel and Scott: Def. of "Essence (Ess.)"

- Gödel:** Property E is Ess. of x iff all of x's properties are nec. entailed by E.
- Scott:** Property E is Ess. of x iff **x has E** and all of x's properties are nec. entailed by E.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

The screenshot shows the Isabelle/HOL proof assistant interface with the file `Scott_SSU.thy` open. The code defines a theory `Scott_SSU` that imports `QML_SSU`. It begins with a `begin` keyword, followed by a `consts` declaration for `P`, an `axiomatization` block, and several axioms (A1 through A5) and definitions (D1 through D5). Axiom A1 states that $\neg P(\Phi) \leftrightarrow \neg P(\neg\Phi)$. Axiom A2 states that $\forall\Phi\forall\Psi. (P(\Phi) \wedge \square(\forall x. \Phi(x) \rightarrow \Psi(x))) \rightarrow P(\Psi)$. Definition D1 gives $G(x) = (\forall\Phi. P(\Phi) \rightarrow \Phi(x))$. Definition D2 gives $\text{ess}(\Phi) = \Phi(x) \wedge (\forall\Psi. \Psi(x) \rightarrow \square(\forall y. \Phi(y) \rightarrow \Psi(y)))$. Definition D3 gives $\text{NE}(x) = (\forall\Phi. \Phi \text{ ess } x \rightarrow \square(\exists x. \Phi(x)))$. Axiom A5 states that $P(\text{NE})$. Below this, a theorem is stated in German: `theorem (* Notwendigerweise, existiert Gott *)`.

```
theory Scott_SSU imports QML_SSU
begin
  consts P :: "(μ ⇒ σ) ⇒ σ"
  axiomatization where
    A1: "[| ∀Φ. P(¬Φ) ↔ ¬P(Φ)|]" and
    A2: "[| ∀Φ. ∀Ψ. (P(Φ) ∧ □(∀x. Φ(x) → Ψ(x))) → P(Ψ)|]"
    definition G where
      D1: "G(x) = (forall Φ. P(Φ) → Φ(x))"
      axiomatization where
        A3: "[| P(G)|]" and
        A4: "[| ∀Φ. P(Φ) → □(P(Φ))|]"
        definition ess (infixr "ess" 85) where
          D2: "Φ ess x = Φ(x) ∧ (forall Ψ. Ψ(x) → □(forall y. Φ(y) → Ψ(y)))"
        definition NE where
          D3: "NE(x) = (forall Φ. Φ ess x → □(exists x. Φ(x)))"
        axiomatization where
          A5: "[| P(NE)|]"

  theorem (* Notwendigerweise, existiert Gott *)
```

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

```

((SV8@)@SV3)=$false) | (((p@(^SX0@mu,SX1:$i) : $false))@SV3)=$true)),inference(prim_subst,[status(thm)],{66:[bind(SV11,$thf(^[SV23@mu,SV24:$i] : $false))]})).
thf(84,plain,!([SV22:(mu:$i$)=SV3:$i;SV23:$i;SV24:(mu:$i$)=0]): (((SV0@)(!($k2_S33$0@)(^*[SX0@mu,SX1:$i] : (~ ((SV22@SX0$@SX1))@SV3))=true) | ((p@($WB$@SV3)=$false) | ((p@(^LSX0@mu,SX1:$i) : (~ ((SV22@SX0$@SX1))@SV3))=true))@SV3)=$true))@SV3)=$true),inference(prim_subst,[status(thm)],{66:[bind(SV11,$thf(^[SV20@mu,SV21:$i] : (~ ((SV22@SV20$@SV21))))]}).
thf(85,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4)=$false) | (((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))),inference(fac_restr,[status(thm)],{56}).
thf(86,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4)=$true) | (((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false))),inference(fac_restr,[status(thm)],{57}).
thf(87,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | (~ ((~ ((p@($WB$@SV4)=$true) | (~ ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))@SV4)),$thf(88,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false) | (~ ((~ ((p@($WB$@SV4)=$true) | (~ ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false))@SV4)),$thf(89,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false) | (~ ((~ ((p@($WB$@SV4)=$true) | (~ ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false))@SV4)),$thf(90,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((~ ((p@($WB$@SV4)=$true) | (~ ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))),inference(extcnf_or_neg,[status(thm)],{87}).
thf(91,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true),inference(extcnf_or_neg,[status(thm)],{89}).
thf(92,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))@SV4)=$true),inference(extcnf_equal_neg,[status(thm)],{85}).
thf(93,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$false) | (~ ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))),inference(extcnf_equal_neg,[status(thm)],{86}).
thf(94,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((~ ((p@($WB$@SV4)=$true) | (~ ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))@SV4)=$true),inference(extcnf_or_neg,[status(thm)],{87}).
thf(95,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true),inference(extcnf_not_neg,[status(thm)],{92}).
thf(96,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false))@SV4)=$true),inference(extcnf_not_neg,[status(thm)],{93}).
thf(100,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((~ ((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$true),inference(extcnf_or_pos,[status(thm)],{96}).
thf(101,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true),inference(extcnf_or_pos,[status(thm)],{97}).
thf(103,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | (~ ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false),inference(extcnf_not_pos,[status(thm)],{108}).
thf(105,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | ((p@($WB$@SV4)=$true) | ((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false),inference(extcnf_not_pos,[status(thm)],{108}).
thf(107,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((SV22@(![k2_S33$0@SV3])@([SX0@mu,SX1:$i] : (~ ((SV22@SX0$@SX1))))@SVB))@((!($k1_S31@(*[SX0@mu,SX1:$i] : (~ ((SV22@SX0$@SX1))))@SVB))@((!($k1_S31@(*[SX0@mu,SX1:$i] : (~ ((SV22@SX0$@SX1))))@SVB))@SV3))=$true) | ((p@($WB$@SV3)=$false) | ((p@(^SX0@mu,SX1:$i) : (~ ((SV22@SX0$@SX1))))@SV3))=$true),inference(extcnf_not_neg,[status(thm)],{78}).
thf(108,plain,!([SV11:(mu:$i$)=0],SV3:$i,SV15:(mu:$i$)=0)): (((SV15@(![$k2_S33$0@SV3]@SV11)@([SX0@mu,SX1:$i] : (~ ((SV15@SX0$@SX1))))@((!($k1_S31@(*[SX0@mu,SX1:$i] : (~ ((SV15@SX0$@SX1))))@SV11))@((!($k1_S31@(*[SX0@mu,SX1:$i] : (~ ((SV15@SX0$@SX1))))@SV11))@SV3))=$false) | ((p@(^SX0@mu,SX1:$i) : (~ ((SV15@SX0$@SX1))))@SV3))=$false) | ((p@(^SV11)@SV3)=$true)),inference(extcnf_not_pos,[status(thm)],{81}).
thf(109,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@(^SY27@mu,SY28:$i) : (~ ((SV0@SY27@SY28))))@SV4))=$false) | ((p@($WB$@SV4)=$true)),inference(sim,[status(thm)],{105}).
thf(110,plain,!([SV4:$i;SV9:(mu:$i$)=0]): (((p@($WB$@SV4)=$true) | ((p@(^SY29@mu,SY30:$i) : (~ ((SV0@SY29@SY30))))@SV4))=$true)),inference(sim,[status(thm)],{101}).
thf(111,plain,!([SV3:$i,SV8:(mu:$i$)=0]): (((p@($WB$@SV3)=$false) | ((p@(^SX0@mu,SX1:$i) : $true))@SV3))=$true),inference(sim,[status(thm)],{76}).
thf(112,plain,!([SV11:(mu:$i$)=0],SV3:$i): (((p@(^SX0@mu,SX1:$i) : $false) | ((p@($WB$@SV3)=$false) | ((p@($WB$@SV3)=$true))))),inference(sim,[status(thm)],{80}).
thf(113,plain,!([SV11:(mu:$i$)=0]): ((f0_atp_e,[status(thm)],{25,112,111,118,189,188,107,84,83,82,75,74,73,72,71,70,69,68,67,66,65,62,57,56,51,42,29})),inference(f0_atp_e,[status(thm)],{113}).
thf(114,plain,!([SV11:(mu:$i$)=0]): ((f0_atp_e,[status(thm)],{25,112,111,118,189,188,107,84,83,82,75,74,73,72,71,70,69,68,67,66,65,62,57,56,51,42,29})),inference(solved_all_splits,[solved_all_splits(join,[])]),{113}).
% $57 cutout_and_CNFRefutation.

```

```
***** End of derivation protocol ****  
***** no. of clauses in derivation: 97 ****  
***** clause counter: 113 ****
```

```

% S2S status: Unsatisfiable for ConsistencyWithoutFirstConjunctionD0.p : (rf:0,axioms:6,ps:3,u:6,ude:false,relibE0:true,rAndE0:true,use_choice:true,use_extcun_combined:true,expand_extcun:false,foapt:e,atp_timeout:25,atp_calls_frequency:10,ordering:none,proof_output:1,clause_count:113,loop_count:0,foapt_calls:2,transl_order:f0,f1,full)
ontoleo:~DemoMaterial benzmuellers []

```

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\bot

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

► $NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\bot

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

► $NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\bot

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\bot

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

- $NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$
- $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$
- $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$
- by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$
- by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$
- $\diamond \exists x[\square \perp]$
- $\diamond \square \perp$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\top(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \perp$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \perp$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller&Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \perp$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \perp$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

► $NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \perp$

\perp

Inconsistency

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

► $NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

[Benzmüller & Woltzenlogel-Paleo, IJCAI, 2016]

Def. D2*

$$\phi \text{ ess. } x \leftrightarrow \cancel{\forall \psi(\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))}$$

Lemma 1 The empty property is an essence of every entity. $\forall x(\emptyset \text{ ess. } x)$

Theorem 1 Positive Properties are possibly exemplified. $\forall \phi[P(\phi) \rightarrow \diamond \exists x \phi(x)]$

Axiom A5

► by T1, A5: $\diamond \exists x[NE(x)]$

Def. D3

$$NE(x) \leftrightarrow \forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

► $\diamond \exists x[\forall \phi[\phi \text{ ess. } x \rightarrow \square \exists y[\phi(y)]]]$

► $\diamond \exists x[\emptyset \text{ ess. } x \rightarrow \square \exists y[\emptyset(y)]]$

► by L1 $\diamond \exists x[\top \rightarrow \square \exists y[\emptyset(y)]]$

► by def. of \emptyset $\diamond \exists x[\top \rightarrow \square \perp]$

► $\diamond \exists x[\square \perp]$

► $\diamond \square \top$

Inconsistency

\perp

The last step is not hard to justify semantically: we did this in the IJCAI-16 paper!
A syntactical proof is not entirely trivial.

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$
- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$
- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$

- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$

$$\frac{\frac{\frac{\frac{\Diamond \Box \perp}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp} \quad \text{Proof Tree Diagram}$$

The diagram illustrates a proof tree in modal logic K. It starts with the assumption $\Diamond \Box \perp$ at the top. This leads to two branches: one for $\Diamond \Diamond \perp$ and one for $\Box \perp$. The branch for $\Diamond \Diamond \perp$ is closed, indicated by a double line. The branch for $\Box \perp$ also leads to a closed branch for \perp , which is enclosed in a box. Finally, the main tree is closed, indicated by a double line under the final formula $\Diamond \Diamond \perp$.

- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$

$$\frac{\frac{\frac{\frac{\Diamond \Box \perp}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp}}{\Diamond \Diamond \perp} \quad \text{Proof-Tree Diagram}$$

The diagram illustrates a proof tree in modal logic K. It starts with the formula $\Diamond \Box \perp$ at the top. A horizontal line with a slash separates it from the first inference step. This step is labeled $\Diamond E$ and consists of two parallel horizontal lines. The left line contains the formula $\Diamond \Box \perp$, and the right line contains the formula $\Box \perp$. Another horizontal line with a slash separates these from the next step. This second step is labeled $\Box E$ and also consists of two parallel horizontal lines. The left line contains the formula $\Box \perp$, and the right line contains the formula \perp . A third horizontal line with a slash separates these from the final step. This third step is labeled $\Diamond I$ and consists of a single horizontal line containing the formula $\Diamond \perp$. A fourth horizontal line with a slash separates this from the bottom result. The bottom result is also labeled $\Diamond I$ and contains the formula $\Diamond \Diamond \perp$.

- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

$\Diamond \Box \perp$ leads to Inconsistency: Syntactical Proof in Modal Logic K

- ▶ Assume: $\Diamond \Box \perp$ (holds globally)
- ▶ Show: there exist a formula φ (in current world) s.t. φ and $\neg\varphi$
- ▶ Choose $\varphi := \Diamond \Diamond \perp$
- ▶ $\Diamond \Diamond \perp$ follows from $\Diamond \Box \perp$

$$\frac{\frac{\frac{\frac{\Diamond \Box \perp}{\Diamond E}}{\frac{\Diamond \Diamond \perp}{\Diamond I}} \Diamond I}{\perp}{\Box E}}{\Box \perp} \Box E$$

- ▶ $\neg \Diamond \Diamond \perp$ holds: $\neg \Diamond \Diamond \perp \equiv \Box \Box \top$, which easily follows by necessitation
- ▶ Hence, $\Diamond \Diamond \perp$ and $\neg \Diamond \Diamond \perp$
- ▶ Hence, \perp

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

Ontologischer Beweis

Feb 10, 1970

$P(\phi)$ ϕ is positive ($\Leftrightarrow \phi \in P$)

At 1 $P(\phi), P(\psi) \vdash P(\phi \wedge \psi)$: At 2 $P(\phi) \nvdash P(\neg \phi)$

$\rho_1 G(x) \equiv (\phi)[P(\phi) \supseteq P(x)]$ (God)

$\rho_2 \phi \text{ Emx} \equiv (\psi)[\psi(x) \supset N(y)[\phi(y) \supset \psi(y)]]$ (Emptiness of x)

$P \supset Nq = N(P \supset q)$ Necessity

At 2 $\begin{cases} P(\phi) \supset N(P(\phi)) \\ \neg P(\phi) \supset N \neg P(\phi) \end{cases}$ because it follows from the nature of the property

Th. $G(x) \supset G \text{ Em. } x$

Df. $E(x) \equiv (\phi)[\phi \text{ Emx} \supset N \exists x \phi(x)]$ necessity, existence

At 3 $P(E)$

Th. $G(x) \supset N(\exists y) G(y)$

thus $(\exists x) G(x) \supset N(\exists y) G(y)$

" $M(\exists x) G(x) \supset M N(\exists y) G(y)$

" $\supset N(\exists y) G(y)$

any two instances of x are mere equivalents

exclusive or * and for any number of numerants

$M(x) G(x)$: means all pos. propo. is: compatible
This is true because of:

At 4: $P(\phi), \phi \supset \psi \vdash P(\psi)$ which impl.

$\begin{cases} x=x & \text{is positive} \\ x \neq x & \text{is negative} \end{cases}$

But if a system S of pos. propo. were incompatible, it would mean, that the num.prop. s (which is positive) would be $x \neq x$

Positive means positive in the moral aesth. sense (independently of the accidental structure of the world). Only $\neg \phi$ is the at. time pole.

Inconsistency

$$\forall \phi [P(\neg \phi) \rightarrow \neg P(\phi)]$$

$$\forall \phi \forall \psi [(P(\phi) \wedge \square \forall x[\phi(x) \rightarrow \psi(x)]) \rightarrow P(\psi)]$$

$$\phi \text{ ess. } x \leftrightarrow \forall \psi (\psi(x) \rightarrow \square \forall y(\phi(y) \rightarrow \psi(y)))$$

$$NE(x) \leftrightarrow \forall \phi [\phi \text{ ess. } x \rightarrow \square \exists y \phi(y)]$$

$$P(NE)$$

Expressivity Matters — Inconsistency in Gödel's Ontological Argument

The Heureka Step

Ontologischer Beweis

Feb 10, 1970

$\mathcal{P}(p)$ if is positive ($\psi \in \mathcal{P}$)

$M(\bar{x}) F(x)$: means all pos. prop. w.r.t. com-
the system of
the domain of

```

DemoMaterial — bash — 166x52
@SV3@$V3=false | (((p@(^[SX0:mu,SX1:$i]: $false)@$V3=$true)), inference(prim_subst,[status(thm),[66:[bind(SV11,$thf(^[SV23:mu,SY24:$i]: $false))]]))),  
thf(84,plain,!([SV2:$i,(mu:$i>$o)],SV3:$i,SV8:(mu:$i>$o)): (((($V0@((^([k2,_SY33@$V3)@(^([SX0:mu,SX1:$i]: (~ ((SV2@$X0)@$X1))))@$V8)@(((sK1,_SY1@(^[SX0:mu,SX1:$i]: (~ ((SV22@$X0)@$X1))))@$V8)@$V3)=true) | (((p@(^[SX0:mu,SX1:$i]: (~ ((SV22@$X0)@$X1))))@$V3=$true) | (((p@(^[SX0:mu,SX1:$i]: (~ ((SV22@$X0)@$X1))))@$V3=$true))), inference(prim_subst,[status(thm),[66:[bind(SV11,$thf(^[SV23:mu,SY24:$i]: $false))]]))),  
thf(85,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false) | (((p@(^[SV9]@$V4) = ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=false)), inference(fac_restr,[status(thm),[56]]),  
thf(86,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)=$true) | (((p@(^[SV9]@$V4) = ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4))=false)), inference(fac_restr,[status(thm),[57]]),  
thf(87,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)) | (~((~((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=false) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false), inference(extcnf_equal_neg,[status(thm)], [85])),  
thf(88,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((p@(^[SV9]@$V4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)) | (~((~((p@(^[SV9]@$V4) | (~((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4))=false) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)=$true)), inference(extcnf_equal_neg,[status(thm)], [86])),  
thf(92,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((~((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=false) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=$false) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false)), inference(extcnf_or_neg,[status(thm), [87]]),  
thf(93,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((~((p@(^[SV9]@$V4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4))=false) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)=$true)), inference(extcnf_or_neg,[status(thm), [89]]),  
thf(96,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((~((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=true) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false), inference(extcnf_not_neg,[status(thm), [92]]),  
thf(97,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((p@(^[SV9]@$V4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4))=true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)=$true)), inference(extcnf_not_neg,[status(thm), [93]]),  
thf(103,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=true) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false)), inference(extcnf_not_pos,[status(thm)], [100])),  
thf(105,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4)=$false) | (((p@(^[SV9]@$V4) | (~((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@$Y27)@$Y28))))@$V4))=false)), inference(extcnf_not_pos,[status(thm), [103]]),  
thf(107,plain,!([SV8:(mu:$i>$o),SV3:$i,SV22:(mu:$i>$o)): (((($V22@((^([k2,_SY33@$V3)@(^([SX0:mu,SX1:$i]: (~ ((SV2@$X0)@$X1))))@$V8)@(((sK1,_SY1@(^[SX0:mu,SX1:$i]: (~ ((SV22@$X0)@$X1))))@$V3)=true) | (((p@(^[SX0:mu,SX1:$i]: (~ ((SV22@$X0)@$X1))))@$V3=$true))), inference(extcnf_not_neg,[status(thm)], [78])),  
thf(110,plain,!([SV4:$i,SV9:(mu:$i>$o)): (((p@(^[SV9]@$V4)=$true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@$Y29)@$Y30))))@$V4)=$true)), inference(sim,[status(thm)], [101])),  
thf(111,plain,!([SV3:$i,SV8:(mu:$i>$o)): (((p@(^[SV8]@$V3)=$false) | (((p@(^[SX0:mu,SX1:$i]: $true))@$V3)=$true)), inference(sim,[status(thm)], [76])),  
thf(112,plain,!([SV11:(mu:$i>$o),SV3:$i]: (((p@(^[SX0:mu,SX1:$i]: $false))@$V3)=$false) | (((p@(^[SV11]@$V3)=$true)), inference(sim,[status(thm)], [80])),  
thf(113,plain,(!($false)=$true), inference(fa_atp_e,[status(thm)], [25,112,111,110,109,108,107,04,03,02,75,74,73,72,71,70,69,68,67,66,65,62,57,56,51,42,29])),  
thf(114,plain,($false), inference(solved_all_splits,[solved_all_splits(join,[]), [113]])),  
% S25 output end CNFRefutation

```

***** End of derivation protocol *****

***** no. of clauses in derivation: 97 *****

***** clause counter: 113 *****

Expressivity Matters — My Claim

'Heureka' steps =?= primitive substitutions

Shouldn't we focus more on exactly this challenge issue?

Universal (Meta-)Logical Reasoning

[Universal (Meta-)Logical Reasoning: Recent Successes, SCP, 2019]

[Universal (Meta-)Logical Reasoning: The Wise Men Puzzle, Data in Brief, 2019]

[Automating Free Logic in HOL, with an ..., JAR, 2019]

Expressivity Matters — Higher-Order Modal Logic

$\Box P$

P is necessary

$\Diamond P$

P is possible

\Box and \Diamond are not truth-functional

Higher-Order Logic can be extended by $\Box P$ and $\Diamond P$

Expressivity Matters — Higher-Order Modal Logic

$\Box P$

P is necessary

$\Diamond P$

P is possible

\Box and \Diamond are not truth-functional

Higher-Order Logic can be extended by $\Box P$ and $\Diamond P$

Expressivity Matters — Higher-Order Modal Logic

$\Box P$

P is necessary

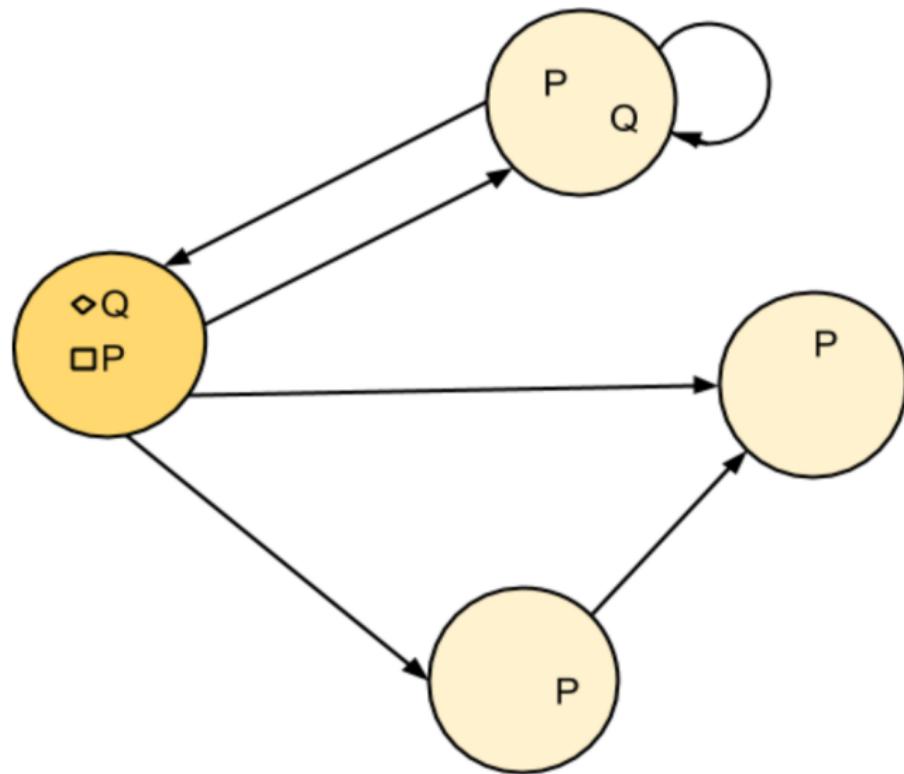
$\Diamond P$

P is possible

\Box and \Diamond are not truth-functional

Higher-Order Logic can be extended by $\Box P$ and $\Diamond P$

Expressivity Matters — Higher-Order Modal Logic in HOL



Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi w \vee \psi w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ $\text{valid} := \lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.Pxw)$ where
 $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above
 - $\square\forall x.Px \equiv \square\Pi'(\lambda x.\lambda w.Pxw)$
 - $\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw))$
 - $\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw))$
 - $\equiv \square(\lambda w.\Pi(\lambda x.Pxw))$
 - $\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw))$
 - $\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw))$
 - $\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v))$
 - $\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv)))$
 - $\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))$
- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL — Lambda's at Work

Standard translation:

- ▶ $\vee_{(i \rightarrow o) \rightarrow (i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.\lambda\psi_{i \rightarrow o}.(\lambda w_i.\varphi_w \vee \psi_w)$
- ▶ $\square_{(i \rightarrow o) \rightarrow (i \rightarrow o)} := \lambda\varphi_{i \rightarrow o}.(\lambda w_i.\forall v_i.Rwv \rightarrow \varphi_v)$ ($R_{i \rightarrow i \rightarrow o}$: accessibility relation)
- ▶ valid := $\lambda\varphi_{i \rightarrow o}.\forall w_i.\varphi_w$

Standard translation extended to quantifiers:

- ▶ remember: $\forall x.\phi x$ shorthand in HOL for $\Pi(\lambda x.\phi x)$
- ▶ $\square\forall x.Px$ is represented as $\square\Pi'(\lambda x_\alpha.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw}))$ where $\Pi' := \lambda\Phi_{\alpha \rightarrow i \rightarrow o}.\lambda w_i.\Pi(\lambda x_\alpha.\Phi_{xw})$ and \square is resolved as above

$$\begin{aligned}\square\forall x.Px &\equiv \square\Pi'(\lambda x.\lambda w.Pxw) \\ &\equiv \square((\lambda\Phi.\lambda w.\Pi(\lambda x.\Phi_{xw}))(\lambda x.\lambda w.Pxw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.(\lambda x.\lambda w.Pxw)xw)) \\ &\equiv \square(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\forall v.(Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda\varphi.\lambda w.\Pi(\lambda v.Rwv \rightarrow \varphi v))(\lambda w.\Pi(\lambda x.Pxw)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow (\lambda w.\Pi(\lambda x.Pxw))v)) \\ &\equiv (\lambda w.\Pi(\lambda v.Rwv \rightarrow \Pi(\lambda x.Pxv))) \\ &\equiv (\lambda w.\forall v.(Rwv \rightarrow \forall x.Pxv))\end{aligned}$$

- ▶ above: possibilist quantification
- ▶ actualist quantification: $\Pi' := \lambda\Phi.\lambda w.\Pi(\lambda x.\text{existsAt } x w \rightarrow \Phi_{xw})$
- ▶ also supported: local and global validity and consequence

Expressivity Matters — Higher-Order Modal Logic in HOL

```
Isabelle2018/HOL - IHOML.thy
□ IHOML.thy (~/GITHUBS/chrisgitlab/talks/2019-Dubrovnik/)

1 theory IHOML imports Main
2 begin
3   typedecl i (* possible worlds *) typedecl e (* individuals *)
4 (*Logical Operators as Truth-Sets*)
5   abbreviation mnot ("¬_"[52]53) where "¬φ ≡ λw. ¬(φ w)"
6   abbreviation negpred ("¬_"[52]53) where "¬Φ ≡ λx. ¬(Φ x)"
7   abbreviation mnegpred ("¬_"[52]53) where "¬Φ ≡ λx. λw. ¬(Φ x w)"
8   abbreviation mand (infixr "Λ"51) where "φΛψ ≡ λw. (φ w)Λ(ψ w)"
9   abbreviation mor (infixr "∨"50) where "φ∨ψ ≡ λw. (φ w)∨(ψ w)"
10  abbreviation mimp (infixr "→"49) where "φ→ψ ≡ λw. (φ w)→(ψ w)"
11  abbreviation mequ (infixr "↔"48) where "φ↔ψ ≡ λw. (φ w)↔(ψ w)"
12 (*Possibilist Quantification*)
13  abbreviation mforall ("∀") where "∀Φ ≡ λw. ∀x. (Φ x w)"
14  abbreviation mforallB (binder "∀" [8]9) where "∀x. φ(x) ≡ ∀φ"
15  abbreviation mexists ("∃") where "∃Φ ≡ λw. ∃x. (Φ x w)"
16  abbreviation mexistsB (binder "∃" [8]9) where "∃x. φ(x) ≡ ∃φ"
17 (*Actualist Quantification*)
18  consts Exists:::"(e⇒i⇒bool)" ("existsAt")
19  abbreviation mforallAct ("∀ᴱ") where "∀ᴱΦ ≡ λw. ∀x. (existsAt x w)→(Φ x w)"
20  abbreviation mexistsAct ("∃ᴱ") where "∃ᴱΦ ≡ λw. ∃x. (existsAt x w) ∧ (Φ x w)"
21  abbreviation mforallActB (binder "∀ᴱ" [8]9) where "∀ᴱx. φ(x) ≡ ∀ᴱφ"
22  abbreviation mexistsActB (binder "∃ᴱ" [8]9) where "∃ᴱx. φ(x) ≡ ∃ᴱφ"
23 (*Modal Operators*)
24  consts aRel:::"i⇒i⇒bool" (infixr "r" 70) (*accessibility relation r*)
25  abbreviation mbox ("◻_"[52]53) where "◻φ ≡ λw. ∀v. (w r v)→(φ v)"
26  abbreviation mdia ("◊_"[52]53) where "◊φ ≡ λw. ∃v. (w r v)∧(φ v)"
27 (*Meta-logical Predicates*)
28  abbreviation valid ("◻[ψ]"[8]) where "◻[ψ] ≡ ∀w. (ψ w)"
29 (*Consistency and some useful definitions on (accessibility) relations*)
30  lemma True nitpick[satisfy] oops (*Model found by model finder Nitpick*)
31 end
32
```

Expressivity Matters — Higher-Order Modal Logic in HOL

Axiom Either a property or its negation is positive, but not both: $\forall\phi[P(\neg\phi) \leftrightarrow \neg P(\phi)]$

Axiom A property necessarily implied by a positive property is positive:

$$\forall\phi\forall\psi[(P(\phi) \wedge \Box\forall x[\phi(x) \rightarrow \psi(x)]) \rightarrow P(\psi)]$$

Thm. Positive properties are possibly exemplified: $\forall\phi[P(\phi) \rightarrow \Diamond\exists x\phi(x)]$

Def. A Godlike being possesses all positive properties: $G(x) \leftrightarrow \forall\phi[P(\phi) \rightarrow \phi(x)]$

Axiom The property of being Godlike is positive: $P(G)$

Cor. Possibly, God exists: $\Diamond\exists xG(x)$

Axiom Positive properties are necessarily positive: $\forall\phi[P(\phi) \rightarrow \Box P(\phi)]$

Def. An essence of an individual is a **property possessed by it and necessarily implying any of its properties**: $\phi \text{ ess. } x \leftrightarrow \phi(x) \wedge \forall\psi(\psi(x) \rightarrow \Box\forall y(\phi(y) \rightarrow \psi(y)))$

Thm. Being Godlike is an essence of any Godlike being: $\forall x[G(x) \rightarrow G \text{ ess. } x]$

Def. Necessary existence of an individual is the necessary exemplification of all its essences: $NE(x) \leftrightarrow \forall\phi[\phi \text{ ess. } x \rightarrow \Box\exists y\phi(y)]$

Axiom Necessary existence is a positive property: $P(NE)$

Thm. Necessarily, God exists: $\Box\exists xG(x)$

Expressivity Matters — Higher-Order Modal Logic in HOL

Axiom	$\forall\phi[P(\neg\phi) \leftrightarrow \neg P(\phi)]$
Axiom	$\forall\phi\forall\psi[(P(\phi) \wedge \Box\forall x[\phi(x) \rightarrow \psi(x)]) \rightarrow P(\psi)]$
Thm.	$\forall\phi[P(\phi) \rightarrow \Diamond\exists x\phi(x)]$
Def.	$G(x) \leftrightarrow \forall\phi[P(\phi) \rightarrow \phi(x)]$
Axiom	$P(G)$
Cor.	$\Diamond\exists xG(x)$
Axiom	$\forall\phi[P(\phi) \rightarrow \Box P(\phi)]$
Def.	$\phi \text{ ess. } x \leftrightarrow \phi(x) \wedge \forall\psi(\psi(x) \rightarrow \Box\forall y(\phi(y) \rightarrow \psi(y)))$
Thm.	$\forall x[G(x) \rightarrow G \text{ ess. } x]$
Def.	$NE(x) \leftrightarrow \forall\phi[\phi \text{ ess. } x \rightarrow \Box\exists y\phi(y)]$
Axiom	$P(NE)$
Thm.	$\Box\exists xG(x)$

Expressivity Matters — Higher-Order Modal Logic in HOL

Axiom

$$\forall\phi[P(\neg\phi) \leftrightarrow \neg P(\phi)]$$

Axiom

$$\forall\phi\forall\psi[(P(\phi) \wedge \Box\forall x[\phi(x) \rightarrow \psi(x)]) \rightarrow P(\psi)]$$

Def.

$$G(x) \leftrightarrow \forall\phi[P(\phi) \rightarrow \phi(x)]$$

Axiom

$$P(G)$$

Axiom

$$\forall\phi[P(\phi) \rightarrow \Box P(\phi)]$$

Def.

$$\phi \text{ ess. } x \leftrightarrow \phi(x) \wedge \forall\psi(\psi(x) \rightarrow \Box\forall y(\phi(y) \rightarrow \psi(y)))$$

Def.

$$NE(x) \leftrightarrow \forall\phi[\phi \text{ ess. } x \rightarrow \Box\exists y\phi(y)]$$

Axiom

$$P(NE)$$

Thm.

$$\Box\exists xG(x)$$

Expressivity Matters — Higher-Order Modal Logic in HOL

The screenshot shows the Ultrafilter interface with a proof script in the main window and a video player at the bottom.

Proof Script Content:

```
theory GoedelProof imports IHOML      (* This formalization follows Fitting's textbook *)
begin
(*Positiveness/perfection: uninterpreted constant symbol*)
consts positiveProperty::"(e⇒i⇒bool)⇒i⇒bool" ("P")
(*Some auxiliary definitions needed to formalise A3*)
definition h1 ("pos")    where "pos Z ≡ ∀X. Z X → P X"
definition h2 (infix "∩" 60) where "X ∩ Z ≡ □(∀x.(X x ↔ (¬(Y x) → (Y x))))"
definition h3 (infix "⇒" 60) where "X ⇒ Y ≡ □(∀z. X z → Y z)"

(**Part I**)
(*D1*) definition G ("G") where "G ≡ (λx. ∀Y. P Y → Y x)"
(*A1*) axiomatization where Ala: "[| ∀X. P (→X) → ¬(P X)|]" and Alb:"[| ∀X. ¬(P X) → P (→X)|]"
(*A2*) axiomatization where A2: "[| ∀X Y. (P X ∧ (X ⇒ Y)) → P Y|]"
(*A3*) axiomatization where A3: "[| ∀Z X. (pos Z ∧ X ∩ Z) → P X|]"
(*T1*) theorem T1: "[| ∀X. P X → ◊∃ X|]" by (metis Ala A2 h3_def)
(*T2*) theorem T2: "[| P G |]" proof -
  {have 1: "¬∀w. ∃Z X. (P G ∨ pos Z ∧ X ∩ Z ∧ ¬P X) w" by (metis(full_types) G_def h1_def h2_def)
   have 2: "[| ∃Z X. (pos Z ∧ X ∩ Z) → P X |] → [P G]" using 1 by auto}
  thus ?thesis using A3 by blast qed
(*T3*) theorem T3: "[| ◊∃ G |]" using T1 T2 by simp
(**Part II**)
(*Logic VDK*) axiomatization where Gamma = "symmetric ⊢GOL"
```

Video Player Controls:

- Backward arrow
- Forward arrow
- Fast forward arrow
- Volume icon
- Progress bar: 00:00 - 02:26
- Zoom: 100%

Bottom Navigation:

- Output
- Query
- Sledgehammer
- Symbols



ELSEVIER

Contents lists available at ScienceDirect

Data in brief

journal homepage: www.elsevier.com/locate/dib



Data Article

Universal (meta-)logical reasoning: The Wise Men Puzzle (Isabelle/HOL dataset)



Christoph Benzmüller ^{a, b, *}

^a Freie Universität Berlin, Germany

^b University of Luxembourg, Luxembourg

ARTICLE INFO

Article history:

Received 20 November 2018

Received in revised form 22 February 2019

Accepted 4 March 2019

Available online 20 March 2019

Keywords:

Universal logical reasoning

Automated theorem proving

Higher-order logic

ABSTRACT

The authors universal (meta-)logical reasoning approach is demonstrated and assessed with a prominent riddle in epistemic reasoning: the Wise Men Puzzle. The presented solution puts a particular emphasis on the adequate modeling of common knowledge and it illustrates the elegance and the practical relevance of the shallow semantical embedding approach when utilized within modern proof assistant systems such as Isabelle/HOL. The contributed dataset provides supporting evidence for claims made in the article "Universal (meta-)logical reasoning: Recent successes" (Benzmüller, 2019).

© 2019 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



Data in brief

journal homepage: www.elsevier.com/locate/dib

Data Article

Universal (meta-)logical reasoning: The Wise Men Puzzle (Isabelle/HOL dataset)

Christoph Benzmüller ^{a,b,*}

```
definition mbox :: " $\alpha \Rightarrow \sigma \Rightarrow \sigma$ " (" $\Box_{\_ \_}$ ") where " $\Box r \varphi \equiv (\lambda w. \forall v. r w v \rightarrow \varphi v)$ "  
definition mdia :: " $\alpha \Rightarrow \sigma \Rightarrow \sigma$ " (" $\Diamond_{\_ \_}$ ") where " $\Diamond r \varphi \equiv (\lambda w. \exists v. r w v \wedge \varphi v)$ "
```

ARTICLE INFO

Article history:

Received 20 November 2018

ABSTRACT

The authors universal (meta-)logical reasoning approach is

(*In HOL the transitive closure of a relation can be defined in a single line.*)

```
definition tc :: " $\alpha \Rightarrow \alpha$ " where "tc R  $\equiv \lambda x y. \forall Q. \text{transitive } Q \rightarrow (\text{sub\_rel } R Q \rightarrow Q x y)$ "
```

Universal logical reasoning
Automated theorem proving
Higher-order logic

lized within modern proof assistant systems such as Isabelle/HOL. The contributed dataset provides supporting evidence for claims made in the article "Universal (meta-)logical reasoning: Recent successes" (Benzmüller, 2019).

© 2019 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Expressivity Matters — Higher-Order Multimodal Logic in HOL

```
theory HOMML imports Main
begin (*An Embedding of Higher-Order Multi-Modal Logic (HOMML) in HOL.*)

typedecl i (*Type of possible worlds.) typedecl μ (*Type of individuals.)
type_synonym σ ="(i⇒bool)" (*Type of world depended formulas (truth sets).*)
type_synonym α ="(i⇒i⇒bool)" (*Type of accessibility relations between worlds.*)

(*Lifted HOMML connectives: they operate on world depended formulas (truth sets).*)
definition mtop :: "σ" ("T") where "T ≡ λw. True"
definition mbot :: "σ" ("⊥") where "⊥ ≡ λw. False"
definition mneg :: "σ⇒σ" ("¬" "[52]53") where "¬φ ≡ λw. ¬φ(w)"
definition mand :: "σ⇒σ⇒σ" (infixr "∧" 51) where "φ ∧ ψ ≡ λw. φ(w) ∧ ψ(w)"
definition mor :: "σ⇒σ⇒σ" (infixr "∨" 50) where "φ ∨ ψ ≡ λw. φ(w) ∨ ψ(w)"
definition mimp :: "σ⇒σ⇒σ" (infixr "→" 49) where "φ → ψ ≡ λw. φ(w) → ψ(w)"
definition mequ :: "σ⇒σ⇒σ" (infixr "↔" 48) where "φ ↔ ψ ≡ λw. φ(w) ← ψ(w)"
definition mall :: "(‘a⇒σ)⇒σ" ("∀") where "∀Φ ≡ λw. ∀x. Φ(x)(w)"
definition mallB :: "(‘a⇒σ)⇒σ" (binder"∀"[8]9) where "∀x. φ(x) ≡ ∀φ"
definition mexi :: "(‘a⇒σ)⇒σ" ("∃") where "∃Φ ≡ λw. ∃x. Φ(x)(w)"
definition mexiB :: "(‘a⇒σ)⇒σ" (binder"∃"[8]9) where "∃x. φ(x) ≡ ∃φ"
definition mbox :: "α⇒σ⇒σ" ("□_ _") where "□r φ ≡ (λw. ∀v. r w v → φ v)"
definition mdia :: "α⇒σ⇒σ" ("◇_ _") where "◇r φ ≡ (λw. ∃v. r w v ∧ φ v)"

(*Global and local validity of lifted formulas*)
definition global_valid :: "σ ⇒ bool" ("[_]" [7]8) where "[p] ≡ ∀w. p w"
consts cw :: i (*Current world; uninterpreted constant of type i*)
definition local_valid :: "σ ⇒ bool" ("[_]cw" [9]10) where "[p]cw ≡ p cw"

(*Introducing "Defs" as the set of the above definitions; useful for convenient unfolding.*)
named_theorems Defs declare mtop_def[Defs] mbot_def[Defs] mneg_def[Defs] mand_def[Defs]
mor_def[Defs] mimp_def[Defs] mequ_def[Defs] mall_def[Defs] mallB_def[Defs] mexi_def[Defs]
mexiB_def[Defs] mbox_def[Defs] mdia_def[Defs] global_valid_def[Defs] local_valid_def[Defs]
end
```

Expressivity Matters — Higher-Order Multimodal Logic in HOL

The screenshot shows the HOL (Higher-Order Logic) IDE interface. The main window displays the file `Relations.thy` containing HOL code. The code defines various relations like reflexive, symmetric, transitive, and Euclidean, and proves properties about them. It also defines the transitive closure of a relation and adds it to a set of definitions. The interface includes tabs for Documentation, Sidekick, State, and Theories. Below the main window, a proof state is shown with a sledgehammer command being executed.

```
theory Relations imports HOMML
begin
(*Some useful relations (for constraining accessibility relations)*)
definition reflexive :: "α⇒bool" where "reflexive R ≡ ∀x. R x x"
definition symmetric :: "α⇒bool" where "symmetric R ≡ ∀x y. R x y → R y x"
definition transitive :: "α⇒bool" where "transitive R ≡ ∀x y z. R x y ∧ R y z → R x z"
definition euclidean :: "α⇒bool" where "euclidean R ≡ ∀x y z. R x y ∧ R x z → R y z"
definition intersection_rel :: "α⇒α⇒α" where "intersection_rel R Q ≡ ∀u v. R u v ∧ Q u v"
definition union_rel :: "α⇒α⇒α" where "union_rel R Q ≡ ∀u v. R u v ∨ Q u v"
definition sub_rel :: "α⇒α⇒bool" where "sub_rel R Q ≡ ∀u v. R u v → Q u v"
definition inverse_rel :: "α⇒α" where "inverse_rel R ≡ ∀u v. R v u"

(*In HOL the transitive closure of a relation can be defined in a single line.)
definition tc :: "α⇒α" where "tc R ≡ λx y.∀Q. transitive Q → (sub_rel R Q → Q x y)"

(*Adding the above definitions to the set of definitions Defs.*)
declare reflexive_def[Defs] symmetric_def[Defs] transitive_def[Defs] euclidean_def[Defs]
intersection_rel_def[Defs] union_rel_def[Defs] sub_rel_def[Defs] inverse_rel_def[Defs]

(*Some useful lemmata.*)
lemma trans_tc: "transitive (tc R)" unfolding Defs tc_def by metis
lemma trans_inv_tc: "transitive (inverse_rel (tc R))" unfolding Defs tc_def by metis
lemma sub_rel_tc: "symmetric R → (sub_rel R (inverse_rel (tc R)))"
  unfolding Defs tc_def by metis
lemma sub_rel_tc_tc: "symmetric R → (sub_rel (tc R) (inverse_rel (tc R)))"
  using sub_rel_def sub_rel_tc tc_def trans_inv_tc by fastforce
lemma symm_tc: "symmetric R → symmetric (tc R)" sledgehammer
  using inverse_rel_def sub_rel_def sub_rel_tc_tc symmetric_def by auto
end
```

Sledgehammering...
Proof found...
"e": Try this: using inverse_rel_def sub_rel_def sub_rel_tc_tc symmetric_def by auto (1.0 s)
"spass": Try this: using inverse_rel_def sub_rel_def sub_rel_tc_tc symmetric_def by presburger (79)
"cvc4": Timed out

Output Query Sledgehammer Symbols

Expressivity Matters — Higher-Order Multimodal Logic in HOL

The screenshot shows the HOL4 IDE interface with the file `WiseMenPuzzle.thy` open. The code defines a theory `WiseMenPuzzle` that imports `HOMML Relations`. It includes several axioms and lemmas about accessibility relations (`a`, `b`, `c`) and knowledge operators (`Eabc`, `Cabc`). The code uses the Sledgehammer tool to prove a theorem `T1` using the Metis prover.

```
theory WiseMenPuzzle imports HOMML Relations
begin
consts a::"o" b::"o" c::"o" (*Wise men modeled as accessibility relations.*)
(*Reflexivity, transitivity and euclideanness is postulated for these relations.*)
axiomatization where
  knowl_abc: "(x = a ∨ x = b ∨ x = c) ==> (reflexive x ∧ transitive x ∧ euclidean x)"
(*Eabc ψ stands for "Everyone in group {a,b,c} knows ψ".*)
definition Eabc where "Eabc ≡ union_rel (union_rel a b) c"
(*Cabc ψ stands for "The common knowledge of group {a,b,c}"*)
definition Cabc :: "i=i=>bool" where "Cabc ≡ tc Eabc"
(*Cabc is reflexive, transitive and euclidean; i.e. it is a knowledge operator.*)
lemma refl_Cabc: "reflexive Cabc" using Cabc_def Eabc_def knowl_abc Defs tc_def by smt
lemma symm_C_abc: "symmetric Cabc" using Cabc_def Eabc_def knowl_abc symm_tc Defs by smt
lemma eucl_Cabc: "euclidean Cabc" using Cabc_def Eabc_def knowl_abc symm_C_abc Defs tc_def by smt
(*We are now ready to model and automate the wise men puzzle.*)
consts ws :: "o=>o" (*ws a: a has a white spot.*)
consts wise :: "o=>bool" (*wise a: a is a wise man.*)
axiomatization where
  A0: "wise a ∧ wise b ∧ wise c" (*a, b and c are wise men.*)
  (*Common knowledge: at least one of a, b and c has a white spot *)
  A1: "|!Cabc (ws a ∨ ws b ∨ ws c)" and
  (*Common knowledge: if x has a white spot then y can see this (and hence know this).*)
  A2: "wise x ∧ wise y ∧ x ≠ y ==> |!Cabc (ws x → Oy (ws y))" and
  (*Common knowledge: if x not has a white spot then y can see this (hence know this) *)
  A3: "wise x ∧ wise y ∧ x ≠ y ==> |!Cabc (¬(ws x) → Oy (¬(ws y)))" and
  (*Common knowledge: a does not know whether he has a white spot.*)
  A4: "|(|!Cabc (¬(Da (ws a))))" and
  (*Common knowledge: b does not know whether he has a white spot.*)
  A5: "|(|!Cabc ¬(Db (ws b)))"
theorem (*k knows he has a white spot*)
T1: "|(|Oc (ws c))" using A0 A1 A3 A4 A5 refl_Cabc unfolding Defs sledgehammer[] by metis
lemma True nitpick [satisfy,user_axioms,expect=genuine,show_all] oops (*Consistency confirmed*)
end
```

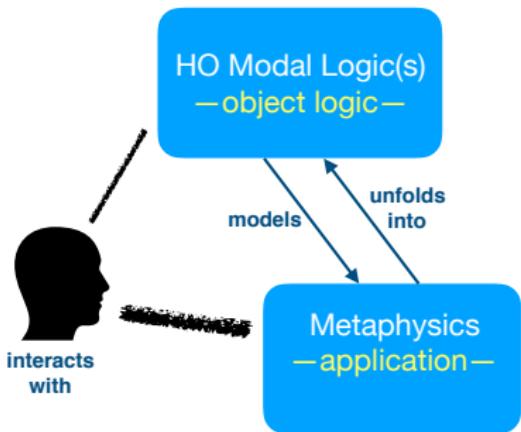
Sledgehammering...
Proof found...
"e": Try this: by metis (684 ms)
"spass": Try this: by metis (666 ms)
"cvc4": Try this: bv metis (488 ms)

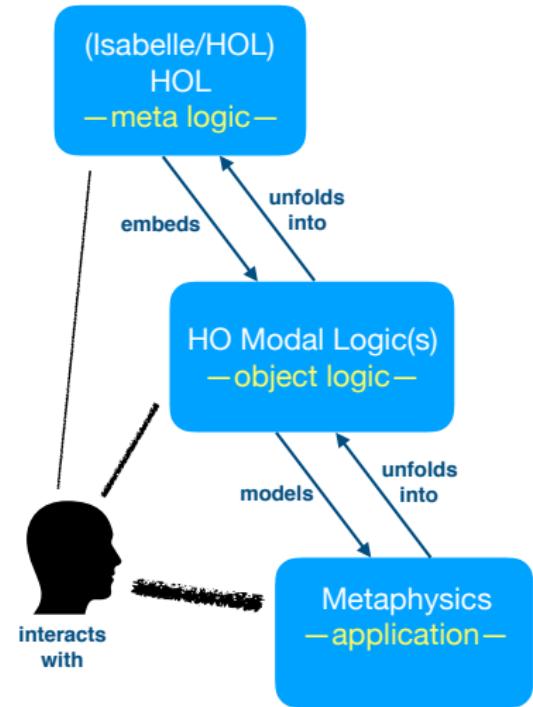
☐ Proof state Auto update Update Search: 100%

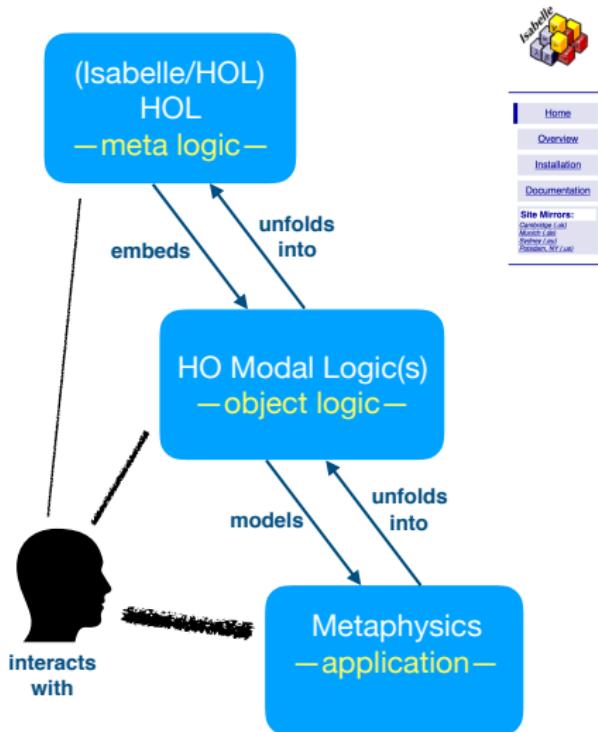
☐ Output Query Sledgehammer Symbols



Metaphysics
—application—







Isabelle

UNIVERSITY OF CAMBRIDGE
Computer Laboratory
TUM
Technische Universität München

What is Isabelle?

Isabelle is a generic proof assistant. It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus. Isabelle was originally developed at the University of Cambridge and Technische Universität München, but now includes numerous contributions from institutions and individuals worldwide. See the [Isabelle overview](#) for a brief introduction.

Now available: Isabelle2017 (October 2017)



[Download for Linux](#) · [Download for Windows \(32bit\)](#) · [Download for Windows \(64bit\)](#) · [Download for Mac OS X](#)

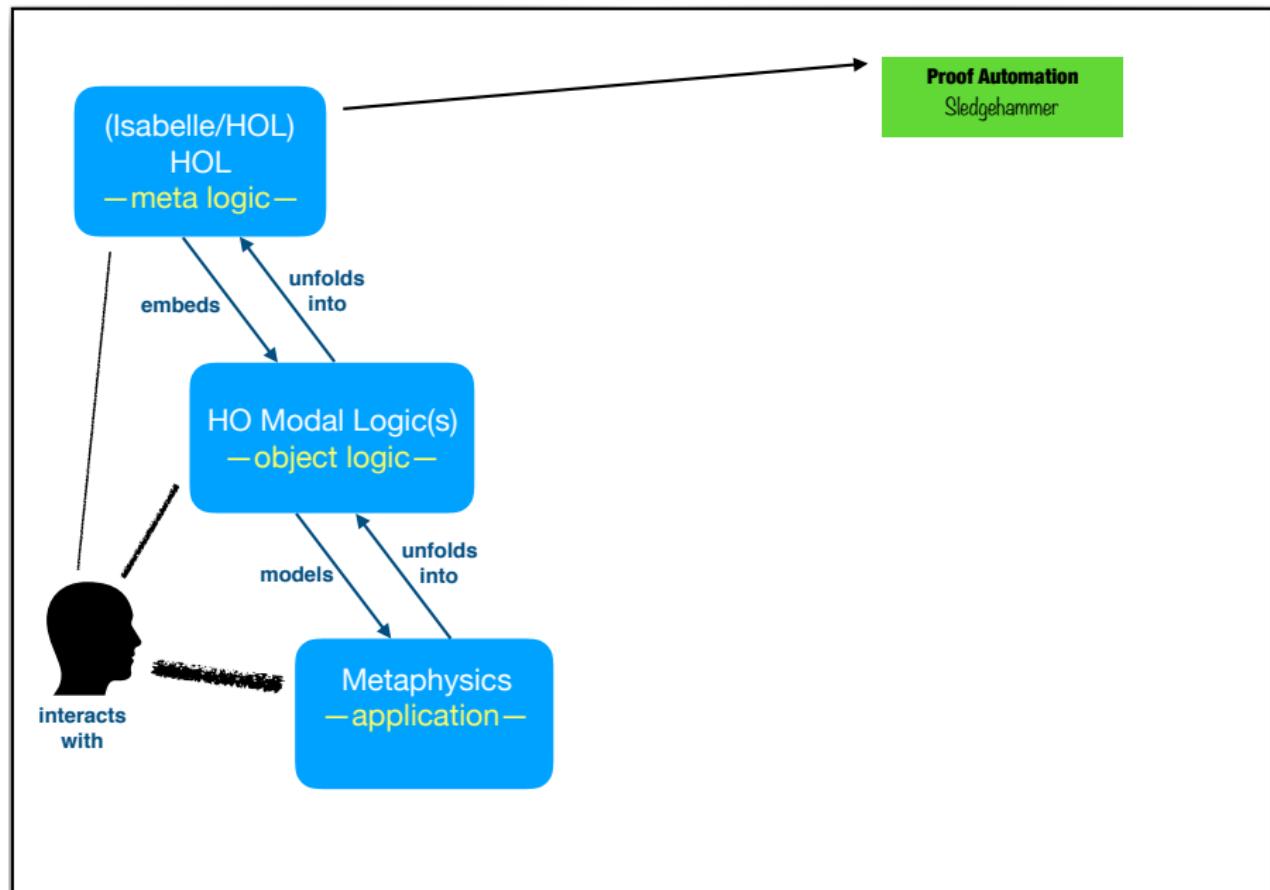
Some notable changes:

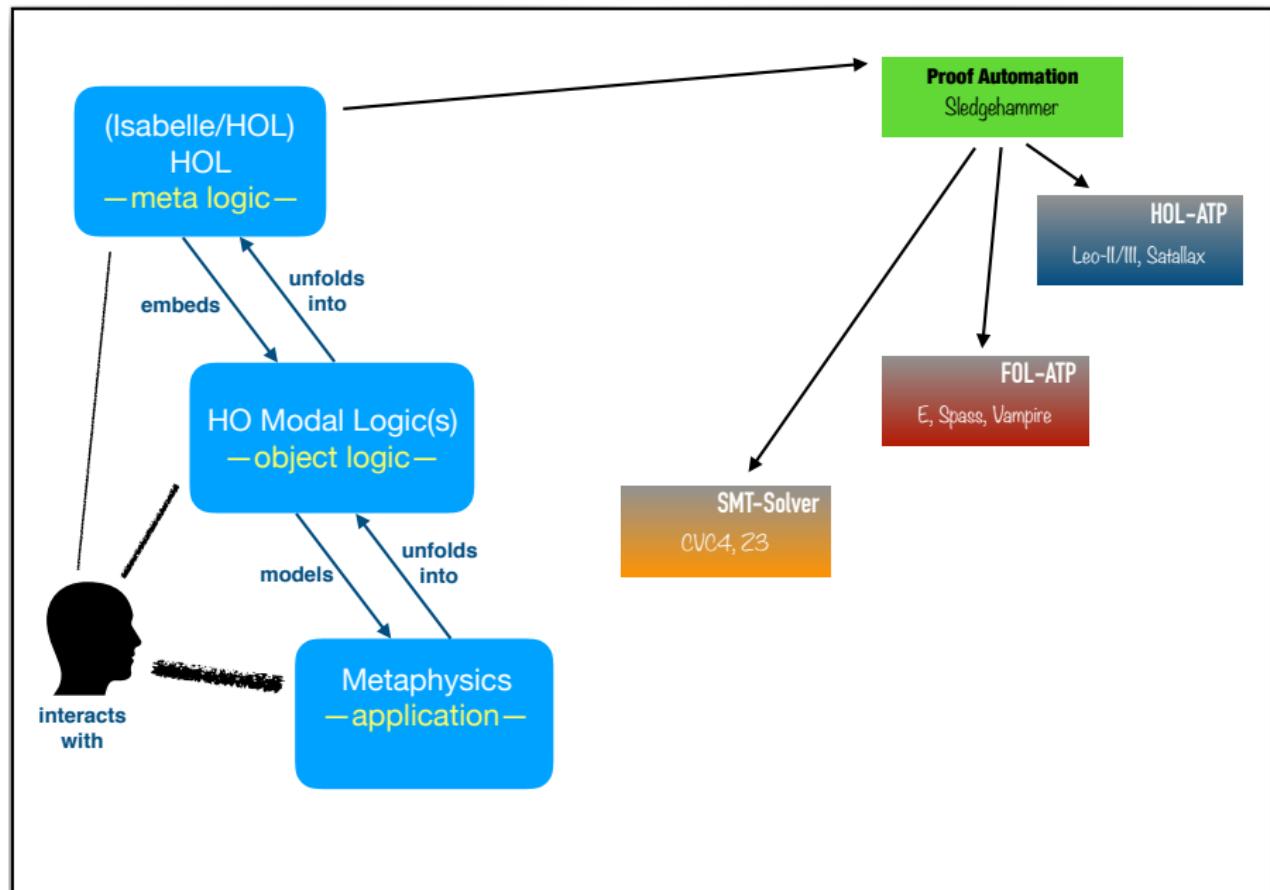
- Experimental support for Visual Studio Code as alternative PIDE front-end.
- Improved Isabelle/Edit Prover IDE: management of session sources independently of editor buffers, removal of unused theories, explicit indication of theory status, more careful auto-indentation.
- Separated theory imports.
- Code generator improvements: support for statically embedded computations.
- Numerous HOL library improvements.
- More material in HOL-Algebra, HOL-Computational_Algebra and HOL-Analysis (ported from HOL-Light).
- Improved Nunchaku model finder, now in main HOL.
- SQL database support in Isabelle/SQLs.

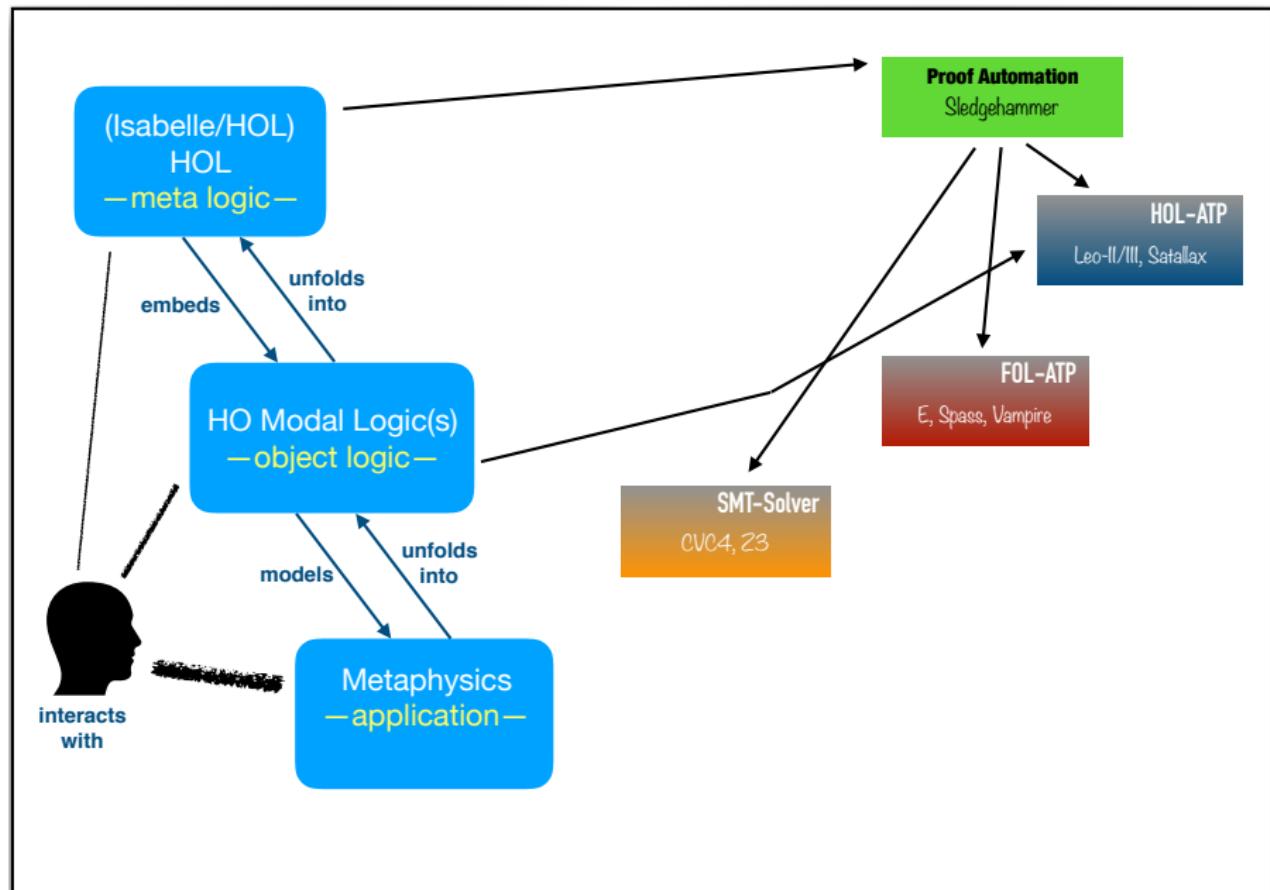
See also the cumulative [NEWS](#).

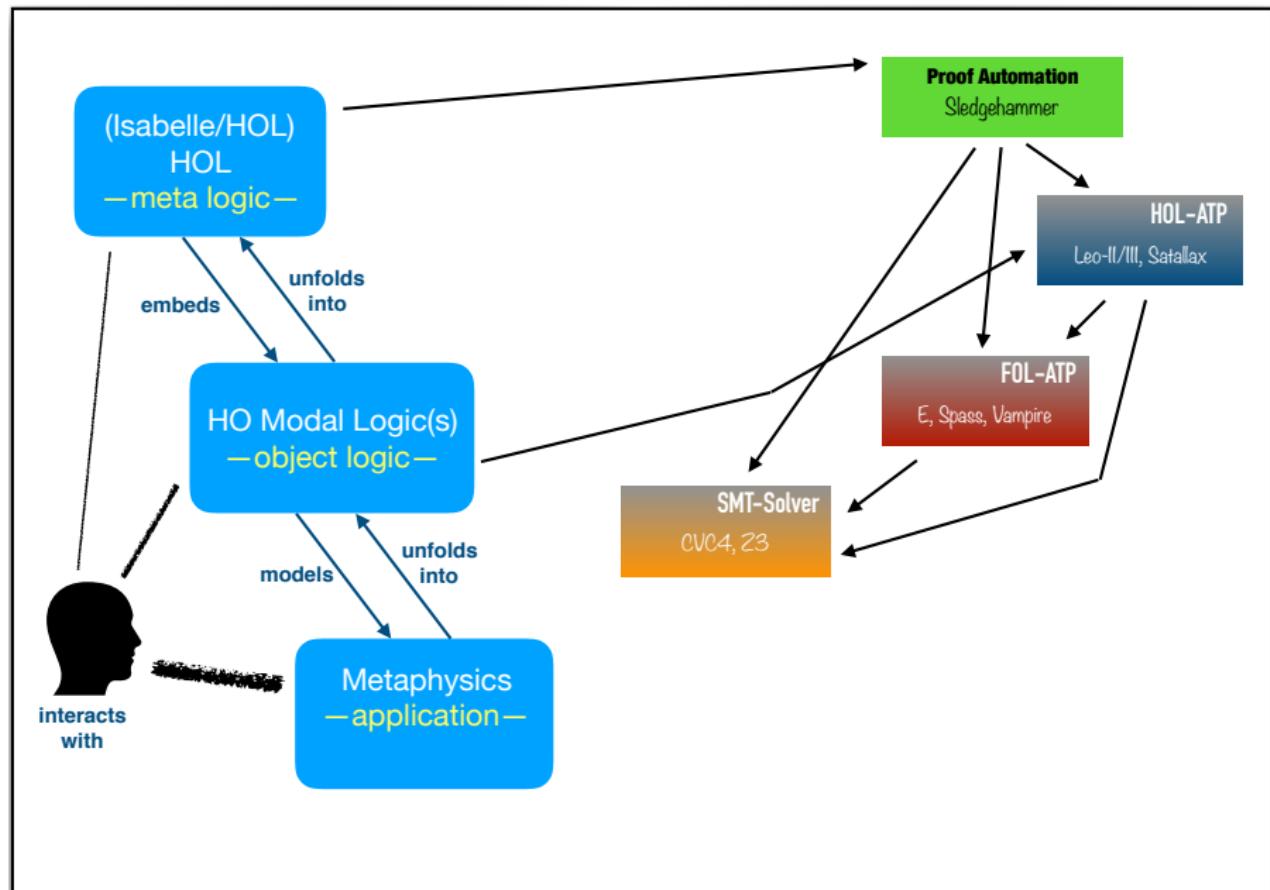
Distribution & Support

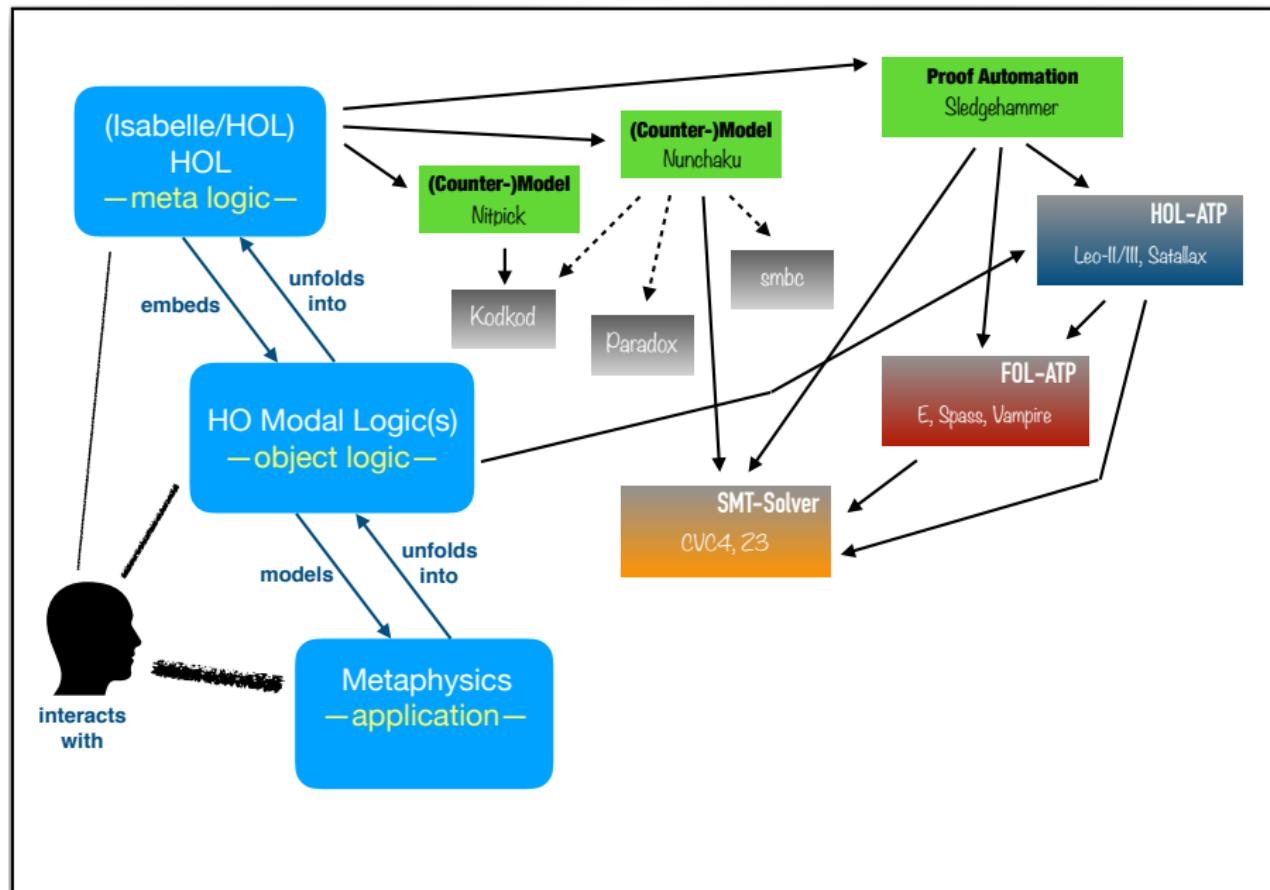
Isabelle is distributed for free under a conglomerate of open-source licenses, but the main code-base is subject to BSD-style regulations. The application bundles include source and binary packages and documentation, see the detailed [Installation Instructions](#). A vast collection of Isabelle examples and applications is available from the [Archive of Formal Proofs](#).

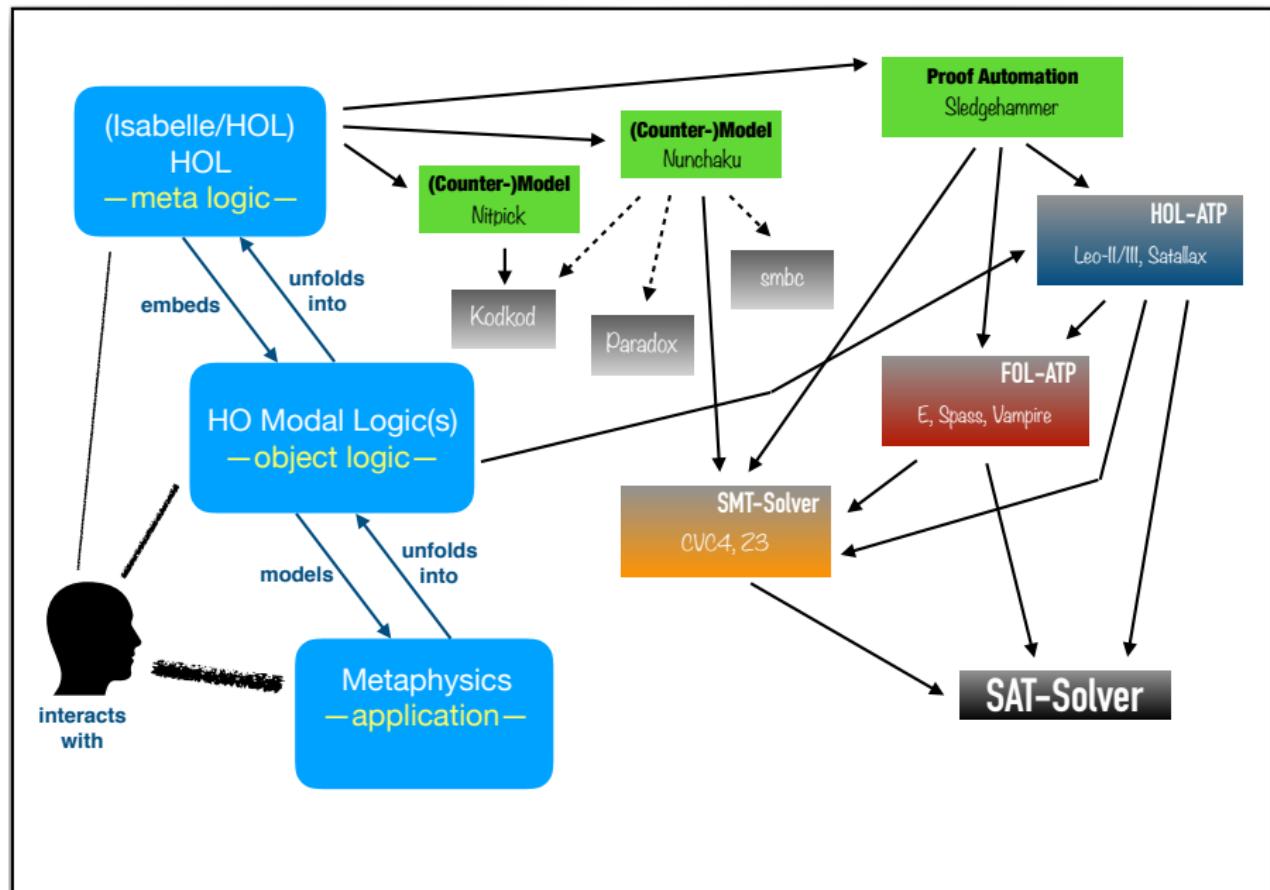


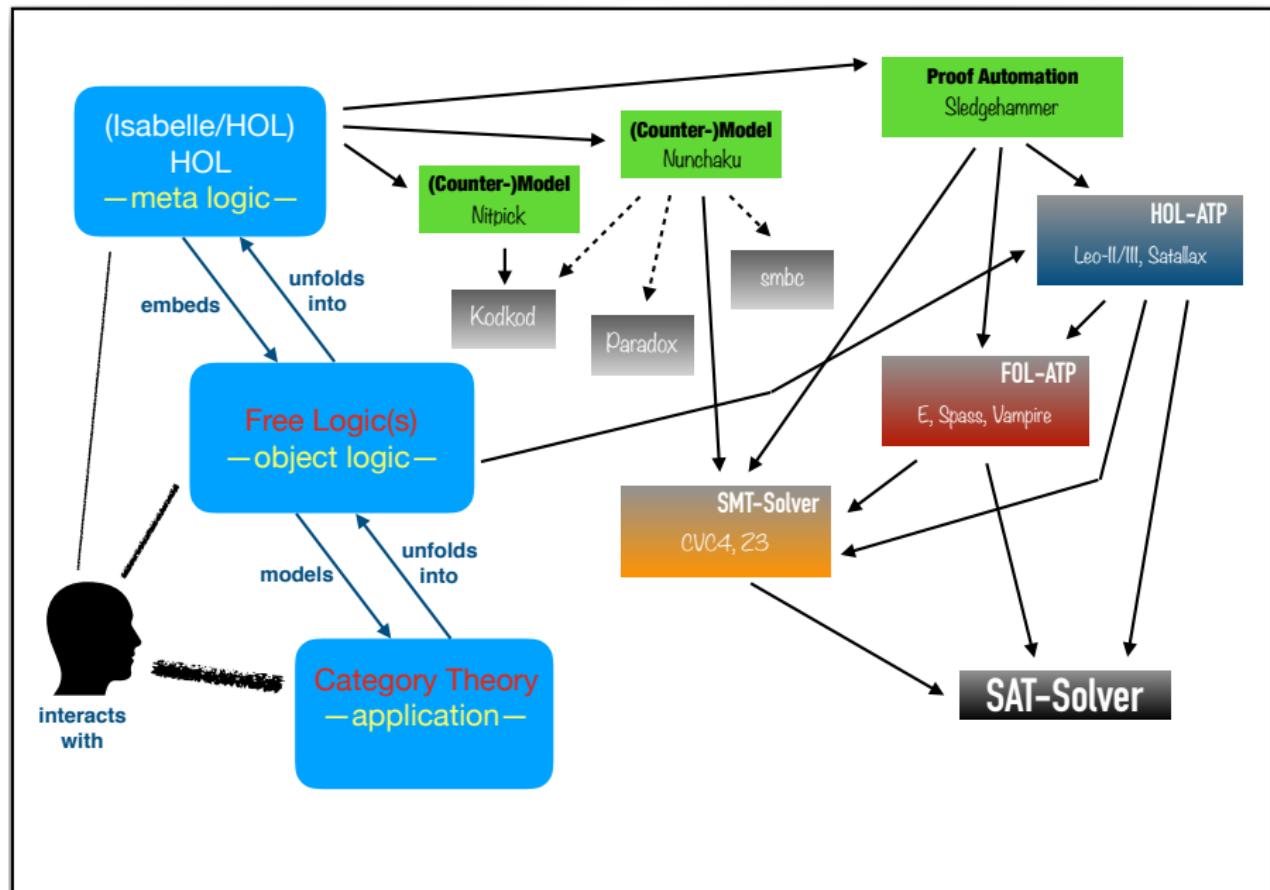


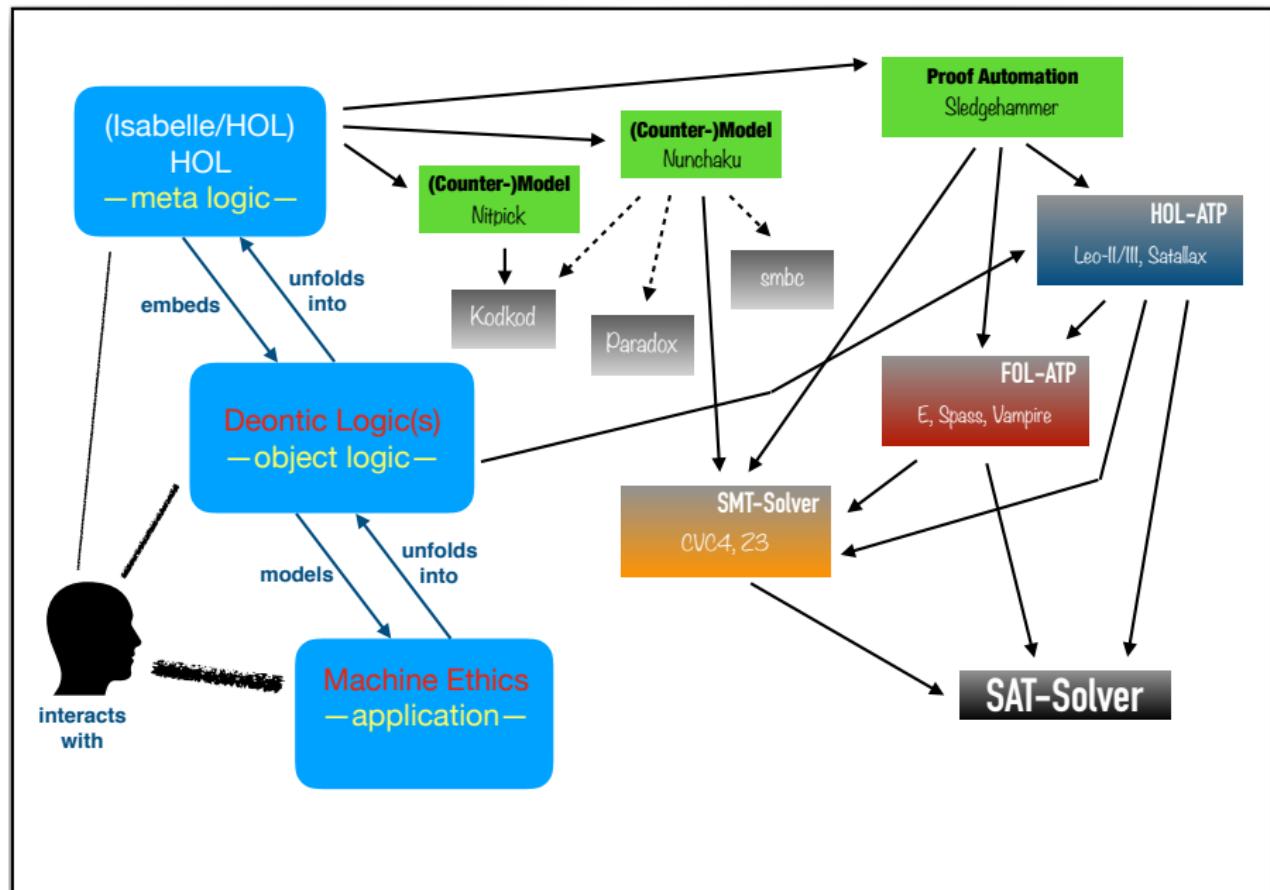


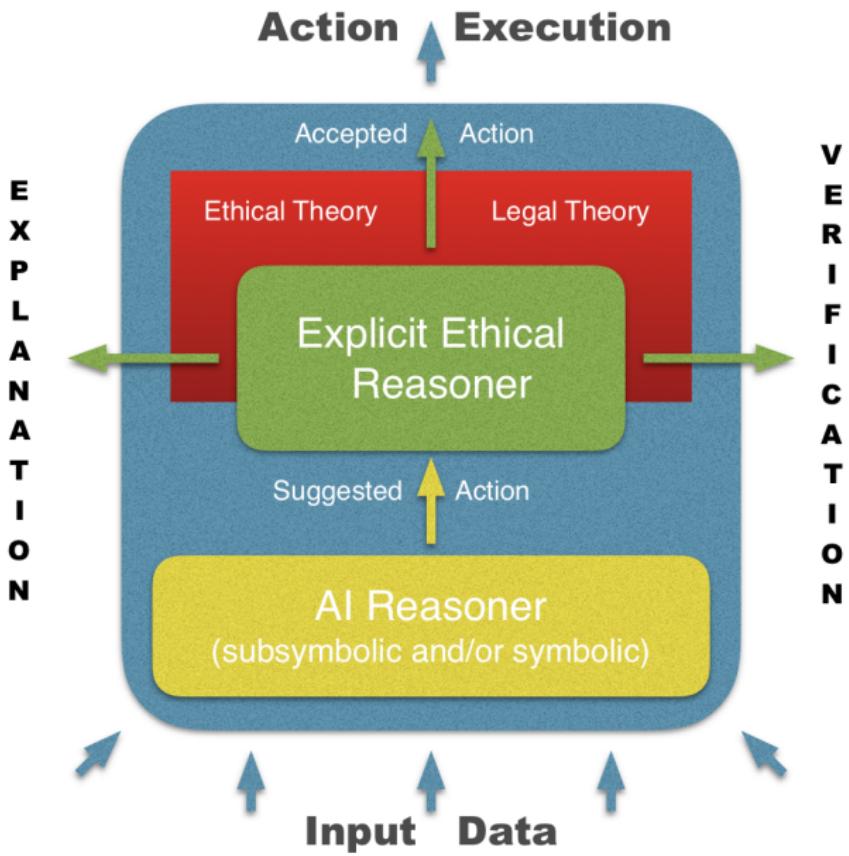


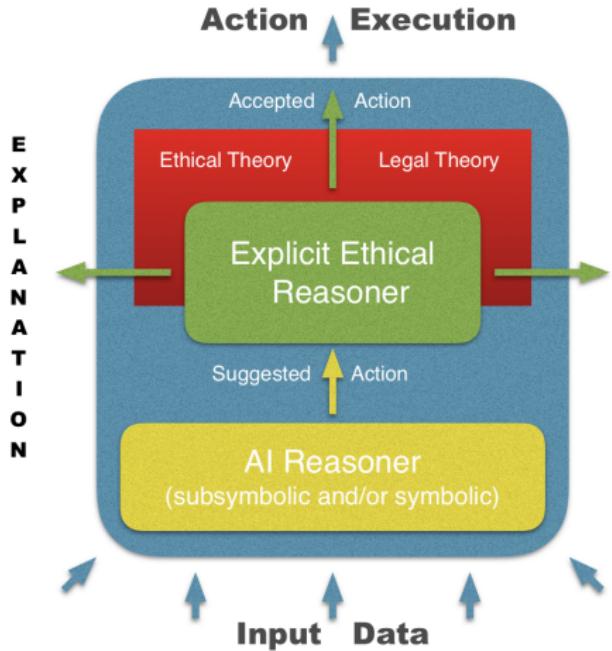








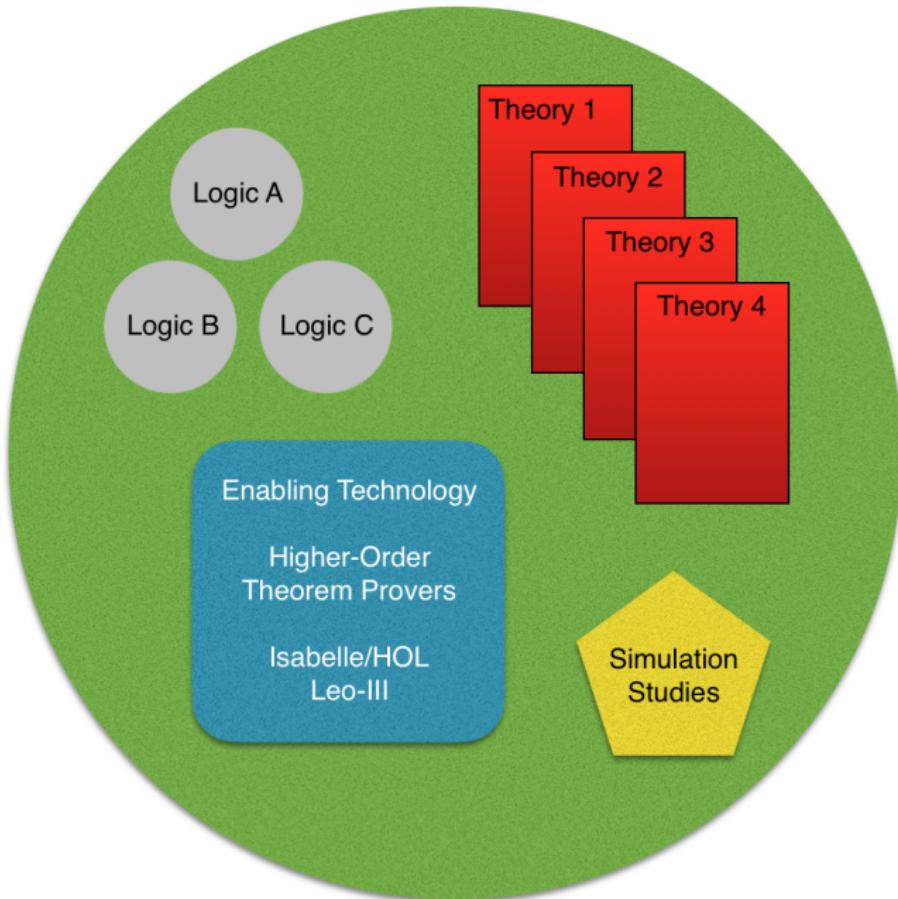




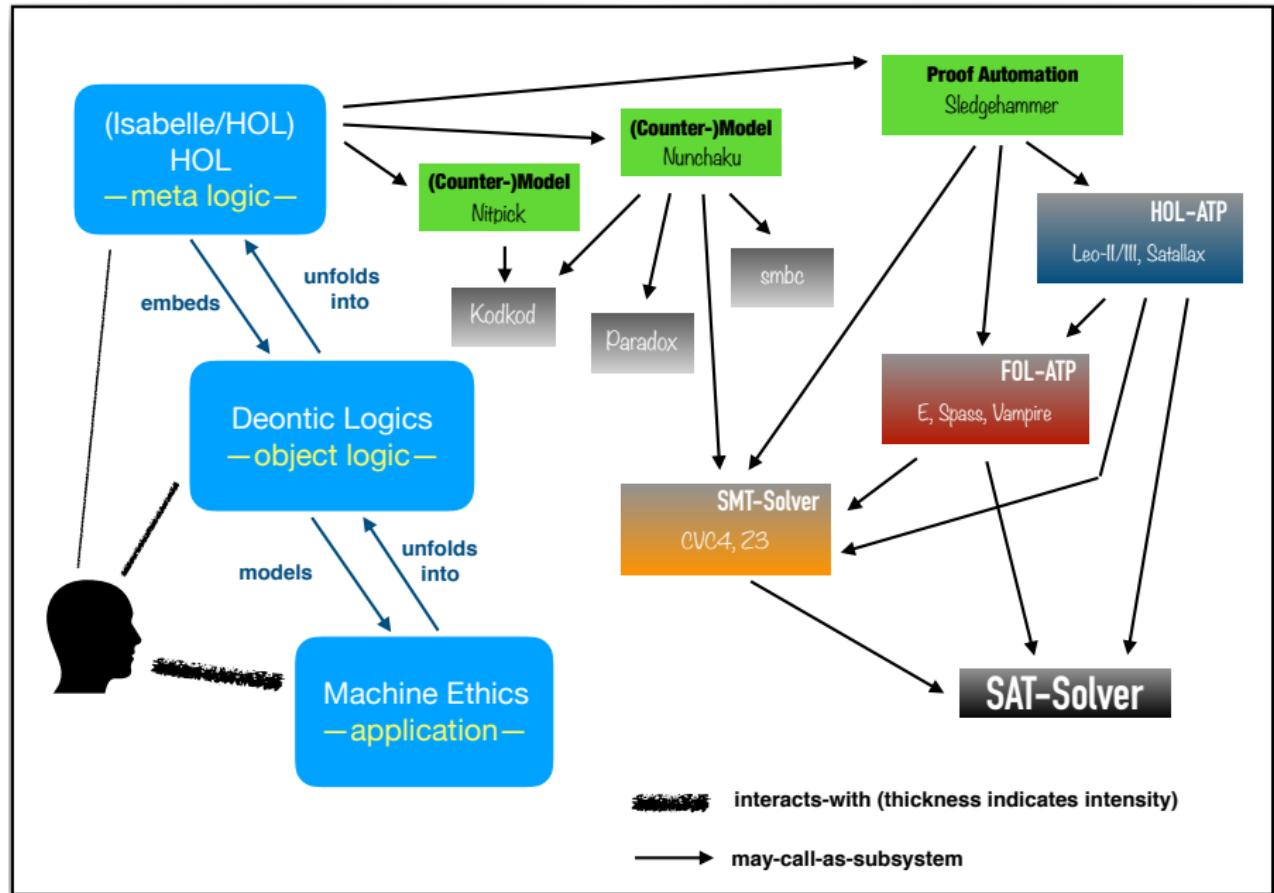
Related Work

- ▶ 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]
- ▶ ...

Normative Reasoning Experimentation Platform (Deontic Logics are required!)



Universal (Meta-)Logical Reasoning Framework (supports multiple Deontic Logics)



Standard Deontic Logic (SDL) in Isabelle/HOL

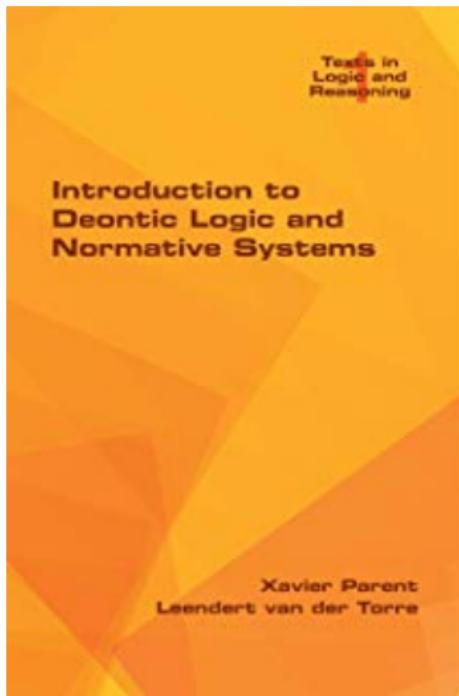
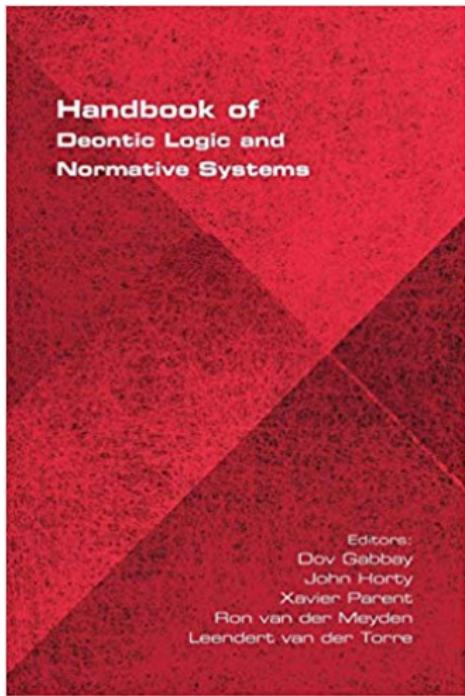
```
SDL.thy
1 theory SDL imports Main (* Christoph Benzmueller & Xavier Parent, 2018 *)
2
3 begin (* SDL: Standard Deontic Logic (Modal Logic D) *)
4 typecl型 i (*type for possible worlds*) type_synonym σ = "(i⇒bool)"
5 consts r::"i⇒i⇒bool" (infixr "r" 70) (*Accessibility relation.*) cw::i (*Current world.*)
6
7 abbreviation mtop ("T") where "T ≡ λw. True"
8 abbreviation mbot ("⊥") where "⊥ ≡ λw. False"
9 abbreviation mnnot ("¬"[52]53) where "¬φ ≡ λw. ¬φ(w)"
10 abbreviation mand (infixr " ∧ " 51) where "φ ∧ ψ ≡ λw. φ(w) ∧ ψ(w)"
11 abbreviation mor (infixr " ∨ " 50) where "φ ∨ ψ ≡ λw. φ(w) ∨ ψ(w)"
12 abbreviation mimp (infixr " → " 49) where "φ → ψ ≡ λw. φ(w) → ψ(w)"
13 abbreviation mequ (infixr "↔" 48) where "φ ↔ ψ ≡ λw. φ(w) ↔ ψ(w)"
14 abbreviation mobligatory ("OB") where "OB φ ≡ λw. ∀v. w r v → φ(v)" (*obligatory*)
15 abbreviation mpermissible ("PE") where "PE φ ≡ ¬(OB(¬φ))" (*permissible*)
16 abbreviation impermissible ("IM") where "IM φ ≡ OB(¬φ)" (*impermissible*)
17 abbreviation omissible ("OM") where "OM φ ≡ ¬(OB φ)" (*omissible*)
18 abbreviation moptional ("OP") where "OP φ ≡ (¬(OB φ) ∧ ¬(OB(¬φ)))" (*optional*)
19
20 abbreviation ddlderived::"σ ⇒ bool" ("[_]"[7]105) (*Global Validity*)
21   where "[A] ≡ ∀w. A w"
22 abbreviation ddlderivedcw::"σ ⇒ bool" ("[_]cw"[7]105) (*Local Validity (in cw)*)
23   where "[A]cw ≡ A cw"
24
25 (* The D axiom is postulated *)
26 axiomatization where D: "[¬ ((OB φ) ∧ (OB (¬ φ)))]"
27
28 (* Meta-level study: D corresponds to seriality *)
29 lemma "[¬ ((OB φ) ∧ (OB (¬ φ)))] ←→ (λw. ∃v. w r v)" by auto
30
31 (* Standardised syntax: unary operator for obligation in SDL *)
32 abbreviation obligatorSDL::"σ⇒σ" ("0(_)") where "0(A) ≡ OB A"
33
34 (* Consistency *)
35 lemma True nitpick [satisfy] oops
```

Dyadic Deontic Logic (DDL) in Isabelle/HOL

The screenshot shows the Isabelle/HOL IDE interface with the DDL.thy theory file open. The theory defines Dyadic Deontic Logic (DDL) based on Main. It includes type declarations for possible worlds, accessibility relations, and axioms for where, ddlneg, ddland, ddlor, ddlimp, ddlequiv, ddbox, ddboxa, ddboxb, ddldia, ddldlia, ddldiag, ddlo, ddloa, ddlop, ddltop, ddbot, ddvalid, ddlidcw, and obligatoryDDL. The code also includes a consistency lemma and nitpick oops. The right sidebar shows navigation links for Documentation, Sidekick, State, and Theories.

```
theory DDL imports Main
begin (* DDL: Dyadic Deontic Logic by Carmo and Jones *)
type_synonym σ = "i⇒bool"
consts av:::"i⇒σ" pv:::"σ⇒(σ⇒bool)" (*accessibility relations*) cw::i (*current world*)
axiomatization where
  ax_3a: "∃x. av(w)(x)" and ax_4a: "∀x. av(w)(x) → pv(w)(x)" and ax_4b: "pv(w)(w)" and
  ax_5a: "¬ob(X)(x). False)" and
  ax_5b: "(∀w. ((Y(w) ∧ X(w)) → (Z(w) ∧ X(w)))) → (ob(X)(Y) → ob(X)(Z))" and
  ax_5c: "((Y. β(Z) → ob(X)(Z)) ∧ (∃Z. β(Z))) →
    (((Y. ((Aw. Y.Z) → (Z w)) ∧ X(y)) → ob(X)(λw. ∀Z. (β Z) → (Z w))))" and
  ax_5d: "((Y. ((Aw. Y.w) → X(w)) ∧ ob(X)(Y) ∧ (∀w. X(w) → Z(w))) →
    ob(Z)(λw. (Z(w) ∧ ¬X(w)) ∨ Y(w)))" and
  ax_5e: "((Y. Y(w) → X(w)) ∧ ob(X)(Z) ∧ (∃w. Y(w) ∧ Z(w))) → ob(Y)(Z)"
abbreviation ddlneg ("¬"[52]53) where "¬A ≡ λw. ¬A(w)"
abbreviation ddland (infixr "∧" 51) where "A ∧ B ≡ λw. A(w) ∧ B(w)"
abbreviation ddlor (infixr "∨" 50) where "A ∨ B ≡ λw. A(w) ∨ B(w)"
abbreviation ddlimp (infixr "→" 49) where "A → B ≡ λw. A(w) → B(w)"
abbreviation ddlequiv (infixr "↔" 48) where "A ↔ B ≡ λw. A(w) ↔ B(w)"
abbreviation ddbox ("□") where "□A ≡ λw. ∀v. A(v)" (*A = (λw. True)*)
abbreviation ddboxa ("□a") where "□a ≡ λw. (Y. av(w)(x) → A(x))" (*in all actual worlds*)
abbreviation ddboxb ("□o") where "□o ≡ λw. (Y. pv(w)(x) → A(x))" (*in all potential worlds*)
abbreviation ddldia ("◊") where "◊A ≡ ▦(¬A)"
abbreviation ddldlia ("◊a") where "◊a ≡ ▦(¬a)"
abbreviation ddldiag ("◊o") where "◊o ≡ ▦(¬A)"
abbreviation ddlo ("O[_)") [52]53) where "O[B|A] ≡ λw. ob(A)(B)" (*it ought to be w, given φ*)
abbreviation ddloa ("Oa") where "Oa ≡ λw. ▦(ob(av(w))(A) ∧ (∃x. av(w)(x) ∧ ¬A(x)))" (*actual obligation*)
abbreviation ddlop ("Oo") where "Oo ≡ λw. ▦(ob(pv(w))(A) ∧ (∃x. pv(w)(x) ∧ ¬A(x)))" (*primary obligation*)
abbreviation ddltop ("T") where "T ≡ λw. True"
abbreviation ddbot ("⊥") where "⊥ ≡ λw. False"
abbreviation ddvalid:::"σ ⇒ bool" ("[_]" [7]105) where "[A] ≡ λw. A w" (*Global validity*)
abbreviation ddlidcw:::"σ ⇒ bool" ("[_]cw" [7]105) where "[A]cw ≡ A cw" (*Local validity (in cw)*)
(* A is obligatory *)
abbreviation obligatoryDDL:::"σ ⇒ σ" ("O[_)") where "O(A) ≡ O(A|T)"
(* Consistency *)
lemma True nitpick [satisfy] oops
```

One SDL \Rightarrow Many Post-SDL's \Rightarrow One HOL (Meta-Logic)



- ▶ Translation to First-Order Logic (FOL): supports propositional non-classical logics
- ▶ Embedding in Higher-Order Logic (HOL): additionally supports **quantified** non-classical logics
(furthermore: **competitive automation** and **intuitive user-interaction**)

Experimenting with Deontic Logics in Isabelle/HOL

The screenshot shows the Isabelle/HOL interface with the file `GDPR.thy` open. The code defines a theory `GDPR imports SDL` with various axioms and lemmas related to data processing obligations. The interface includes a toolbar, a vertical navigation bar on the right, and a status bar at the bottom.

```
theory GDPR imports SDL (* Christoph Benzmueller & Xavier Parent, 2018 *)
begin (** GDPR Example **)
consts process_data_lawfully::σ erase_data::σ kill_boss::σ
axiomatization where
  (* It is an obligation to process data lawfully. *)
  A1: "[0(process_data_lawfully)]" and
  (* Implicit: It is an obligation to keep the data if it was processed lawfully. *)
  Implicit: "[0(process_data_lawfully → ¬erase_data)]" and
  (* If data was not processed lawfully, then it is an obligation to erase the data. *)
  A2: "[¬process_data_lawfully → 0(erase_data)]"
  (* Given a situation where data is processed unlawfully. *) and
  A3: "[¬process_data_lawfully]_cw"
(** Some Experiments **)
lemma True nitpick [satisfy] oops (* Consistency-check: Is there a model? *)
lemma False sledgehammer oops (* Inconsistency-check: Can Falsum be derived? *)
lemma "[0(erase_data)]" sledgehammer nitpick oops (* Should the data be erased? *)
lemma "[0(¬erase_data)]" sledgehammer nitpick oops (* Should the data be kept? *)
lemma "[0(kill_boss)]" sledgehammer nitpick oops (* Should the boss be killed? *)
end
```

Sledgehammering...
Proof found...
"spass": The prover derived "False" from "A1", "A2", "A3", "D", and "Implicit", which could
"e": The prover derived "False" from "A1", "A2", "A3", "D", and "Implicit", which could be d
"cvc4": Try this: by (metis A1 A2 A3 D Implicit) (68 ms)
"z3": Try this: by (metis A1 A2 A3 D Implicit) (59 ms)

Output Query Sledgehammer Symbols

Our framework scales for first-order and higher-order deontic logics!!!

Experimenting with Deontic Logics in Isabelle/HOL

The screenshot shows the Isabelle2018/HOL interface with a theory file named `GDPR.thy`. The code implements a deontic logic for GDPR, defining obligations and their consequences. The interface includes a toolbar, a navigation bar, and a sidebar with tabs for Documentation, Sidekick, State, and Theories.

```
theory GDPR imports DDL (* Christoph Benzmueller & Xavier Parent, 2018 *)
begin (** GDPR Example **)
consts process_data_lawfully::σ erase_data::σ kill_boss::σ

axiomatization where
(* It is an obligation to process data lawfully. *)
A1: "[0(process_data_lawfully)]" and
(* Implicit: It is an obligation to keep the data if it was processed lawfully. *)
Implicit: "[0(process_data_lawfully → ¬erase_data)]" and
(* If data was not processed lawfully, then it is an obligation to erase the data. *)
A2: "[¬process_data_lawfully → 0(erase_data)]"
(* Given a situation where data is processed unlawfully. *) and
A3: "[¬process_data_lawfully]_cw"

(** Some Experiments **)
lemma True nitpick [satisfy] oops (* Consistency-check: Is there a model? *)
lemma False sledgehammer oops (* Inconsistency-check: Can Falsum be derived? *)

lemma "[0(erase_data)]" sledgehammer nitpick oops (* Should the data be erased? *)
lemma "[0(¬erase_data)]" sledgehammer nitpick oops (* Should the data be kept? *)
lemma "[0(kill_boss)]" sledgehammer nitpick oops (* Should the boss be killed? *)

end
```

The bottom pane displays the results of a nitpicking session:

- Nitpicking formula...
- Nitpick found a counterexample for card i = 2:
- Constants:
av = ($\lambda x. _.$)((i₁, i₁) := True, (i₁, i₂) := False, (i₂, i₁) := False, (i₂, i₂) := True)
cw = i₂

Navigation buttons at the bottom include Output, Query, Sledgehammer, and Symbols. The status bar at the bottom shows the time as 22:48 (1127/1197) and the memory usage as 1.25/1170MB 8:28 AM.

Input/Output (I/O) Logic

[Makinson, JPL, 2000], [GabbayHortyParentEtAl-Handbook, 2013]

- ▶ I/O-operators, such as `out1` (simple-minded output), accept set G of conditional norms as argument
- ▶ Conditional norms: pairs (a,x) with input “ a ” (condition) and output “ x ” (obligation)
- ▶ Pairs (a,x) are not given a truth-functional semantics in I/O logic

Semantics of `out1`

- ▶ $\text{out1}(G, A) := \text{Cn}(G(\text{Cn}(A)))$
- ▶ where $\text{Cn}(X) := \{s \mid X \models s\}$ and $G(X) := \{s \mid \exists a \in X. (a, s) \in G\}$.

```
(*I0 logic in HOL*)
typedecl i -- "type for possible worlds"      type_synonym e = "(i⇒bool)"
abbreviation ktop   :: "e"      ("⊤")  where "⊤ ≡ λw. True"
abbreviation kbot   :: "e"      ("⊥")  where "⊥ ≡ λw. False"
abbreviation knot   :: "e⇒e"    ("¬_"[52]53) where "¬φ ≡ λw. ¬φ(w)"
abbreviation kor    :: "e⇒e⇒e" (infixr "∨"50) where "φ∨ψ ≡ λw. φ(w)∨ψ(w)"
abbreviation kand   :: "e⇒e⇒e" (infixr "∧"51) where "φ∧ψ ≡ λw. φ(w)∧ψ(w)"
abbreviation kimp   :: "e⇒e⇒e" (infixr "⇒"49) where "φ⇒ψ ≡ λw. φ(w)→ψ(w)"
abbreviation kvalid :: "e⇒bool" ("_|"[8]109) where "|p| ≡ ∀w. p w"

abbreviation "outpre ≡ λG.λa.λy::e. ∃f. [a ⊢ f] ∧ G (f,y)"
abbreviation "out1 ≡ λG.λa.λx. [x] ∨
  (∃i j k. outpre G a i ∧ outpre G a j ∧ outpre G a k ∧ [(i ∧ j ∧ k) ⊢ x])"
```

I/O-Logic in Isabelle/HOL

The screenshot shows the Isabelle/HOL IDE interface. The main window displays the theory file `IO_Logic.thy`. The code contains several lemmas annotated with the `nitpick oops` command, which is highlighted in yellow. The sidebar on the right shows navigation links: Documentation, Sidekick, State, and Theories.

```
(* Some Tests *)
consts a::e b::e c::e
abbreviation "G1 ≡ (λX. X=(a,e) ∨ X=(b,e))" (* G = {(a,e),(b,e)} *)
lemma "outl G1 a e" by blast (*proof*)
lemma "outpre G1 a e" by blast (*proof*)
lemma "outpre G1 (a ∨ b) e" nitpick oops (*countermodel*)
lemma "outl G1 (a ∨ b) e" nitpick oops (*countermodel*)
lemma "[x] ==> outpre G1 (a ∨ b) x" nitpick oops (*countermodel*)
lemma "[x] ==> outl G1 (a ∨ b) x" by blast (*proof*)

(* GDPR Example from before *)
consts pr_d_lawf::e erase_d::e kill_boss::e
abbreviation (* G = {(T,pr_d_lawf),(pr_d_lawf,¬erase_d),(¬pr_d_lawf,erase_d)} *)
"G ≡ (λX. X=(T,pr_d_lawf) ∨ X=(pr_d_lawf,¬erase_d) ∨ X=(¬pr_d_lawf,erase_d)) "
lemma "outl G (¬pr_d_lawf) erase_d" by smt (*proof*)
lemma "outl G (¬pr_d_lawf) (¬erase_d)" nitpick oops (*countermodel*)
lemma "outl G (¬pr_d_lawf) kill_boss" nitpick oops (*countermodel*)
lemma "outl G (¬pr_d_lawf) ⊥" nitpick oops (*countermodel*)
```

Below the theory editor, there is a toolbar with checkboxes for "Proof state" and "Auto update", an "Update" button, a "Search" input field, and a zoom level of "100%". The output pane shows the results of the nitpicking process:

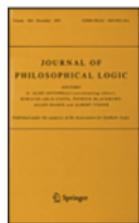
```
Nitpicking formula...
Nitpick found a counterexample for card i = 2:

Skolem constant:
w = i1
Constants:
erase_d = (λx::i. _)(i1 := True, i2 := True)
kill_boss = (λx::i. _)(i1 := False, i2 := False)
pr_d_lawf = (λx::i. _)(i1 := False, i2 := True)
```

The bottom navigation bar includes tabs for Output, Query, Sledgehammer, and Symbols.

Preference-based DDL in Isabelle/HOL

Journal of Philosophical Logic / Vol. 43, No. 6, December 2014 / Maximality vs. Optim...



X. Parent

The screenshot shows the Isabelle/HOL IDE interface. The top part displays a proof script named `PrefDDL.thy` with several lemmas and their proofs. The lemmas include `classical`, `OM`, `OD`, `DfP`, `COK`, `abs`, `nec`, `exti`, `lema_id`, `Sh`, `MP`, `N`, `D`, and `CM`. The `CM` lemma is highlighted with a yellow background and a red error message "nitpick oops". The bottom part of the interface shows the nitpick results, indicating a counterexample was found for the CM lemma with card `w = 3`. The nitpick formula is shown, along with free variables and their assignments.

```
246 (* axioms of proof theory for E, check for soundness *)
247 lemma classical: " $A \rightarrow (\lambda w. A)$ " by simp -- "all classical tautologies"
248
249 lemma "OM": " $\Box A \rightarrow A$ " by simp -- "part of SS schema for  $\Box$ "
250 lemma "OD": " $\Diamond A \rightarrow \Box(\Diamond A)$ " by simp -- "part of SS schema for  $\Diamond$ "
251
252 lemma DfP: " $[P(B|A) \leftrightarrow \neg(O(\neg B)|A)]$ " by (simp add: prefDDLBase.truthSet_def)
253 lemma COK: " $[O(B \rightarrow C)|A] \rightarrow (O(B|A) \rightarrow O(C|A))$ " by (simp add: prefDDLBase.truthSet_def)
254 lemma abs: " $[O(B|A) \rightarrow \Box(O(B|A))]$ " by simp
255 lemma nec: " $[O(A \rightarrow B) \rightarrow (O(C|A) \rightarrow O(C|B))]$ " by (simp add: prefDDLBase.truthSet_def)
256 lemma exti: " $[O(A \leftrightarrow B) \rightarrow (O(C|A) \leftrightarrow O(C|B))]$ " by (simp add: prefDDLBase.truthSet_def)
257 lemma lema_id: " $[O(A|A)]$ " by (simp add: optChoice)
258 lemma Sh: " $[O(C|(A \wedge B)) \rightarrow O(B \rightarrow C)|A]$ " by (smt optBest.optChoice optBest_axioms prefDDLBase.
259
260 (* soundness of inference rules *)
261 lemma MP: " $[A] \rightarrow [A \rightarrow B] \rightarrow [B]$ " by simp
262 lemma N: " $[A] \rightarrow [\Box A]$ " by simp
263
264 (* D+ should hold in F, this can be verified: *)
265 lemma "D+": " $[\Diamond A \rightarrow (O(B|A) \rightarrow P(B|A))]$ " by (metis FOpt.opt_limitatedness FOpt_axioms truthSet_def)
266
267
268 (* (CM) should not be provable in system F but only as of system F+CM, verified by nitpick *)
269 lemma CM: " $[(O(B|A) \wedge O(C|A)) \rightarrow O(C|(A \wedge B))]$ " nitpick oops
270
```

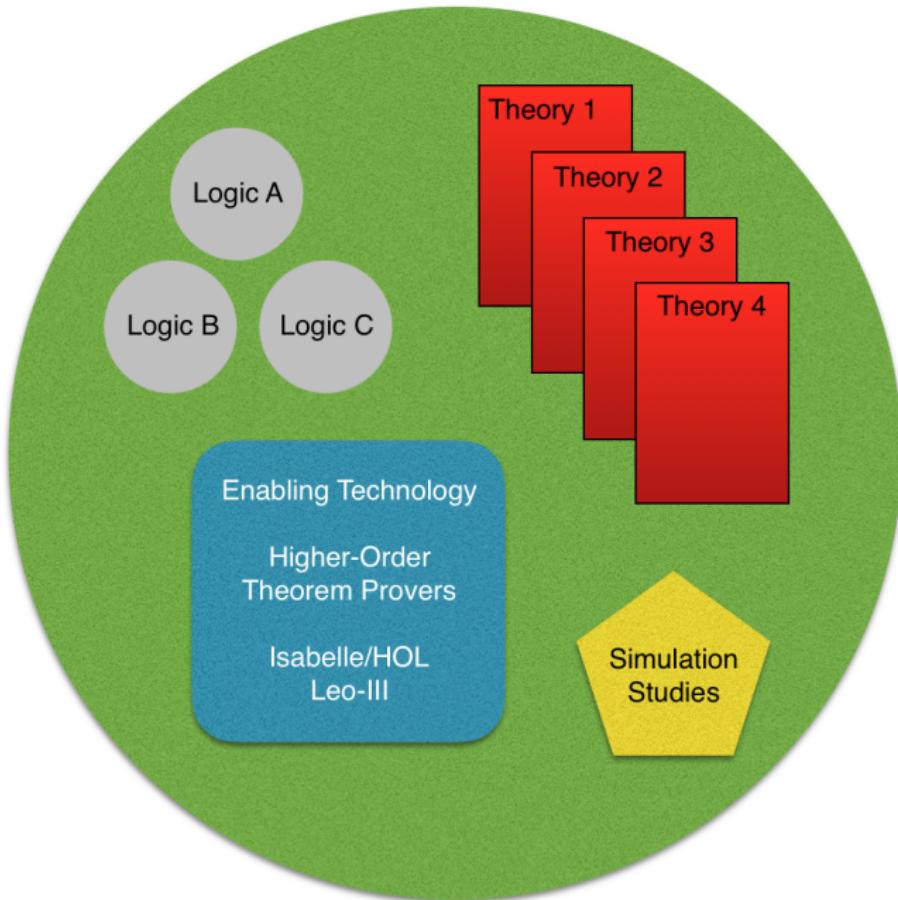
Nitpicking formula...
Nitpick found a counterexample for card `w = 3`:

Free variables:

```
A = ( $\lambda x. \_)(w_1 := \text{True}, w_2 := \text{True}, w_3 := \text{True})$ 
B = ( $\lambda x. \_)(w_1 := \text{True}, w_2 := \text{True}, w_3 := \text{False})$ 
C = ( $\lambda x. \_)(w_1 := \text{False}, w_2 := \text{True}, w_3 := \text{False})$ 
op ⪰ =
  ( $\lambda x. \_$ )
  ( $w_1 := (\lambda x. \_)(w_1 := \text{True}, w_2 := \text{True}, w_3 := \text{False}),$ 
    $w_2 := (\lambda x. \_)(w_1 := \text{True}, w_2 := \text{True}, w_3 := \text{True}),$ 
    $w_3 := (\lambda x. \_)(w_1 := \text{True}, w_2 := \text{False}, w_3 := \text{True}))$ 
opt = ( $\lambda x. \_$ )
```

Output Query Sledgehammer Symbols

Normative Reasoning Experimentation Platform (Deontic Logics are required!)



What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

What you should do in the next 60 years!

Let EasyChair make you rich!

But spent more time with Vampire

and also go in bed with Lambda's

Conclusion: HOL and Universal Reasoning deserves more attention!

- ▶ **Heureka steps:** can be expected in HOL, but less so in FOL or PL
- ▶ **Lean and elegant** approach to integrate and combine heterogeneous logics
- ▶ **Intuitive user interaction** at abstract level (in modern proof assistants)
- ▶ **Reuse** of existing systems and tools
- ▶ Important **link** between different **research communities**
- ▶ Approach very well suited for (interdisciplinary) **teaching of logics**

Leo-III — Universal Higher-Order Theorem Prover

- ▶ The Higher-Order Prover Leo-III, IJCAR 2018, Springer LNCS, 2018.
- ▶ Theorem Provers for Every Normal Modal Logic, LPAR 2017, Springer LNCS, 2018.

Universal (Meta-)Logical Reasoning

- ▶ Universal (Meta-)Logical Reasoning: Recent Successes, SCP, 2019.
- ▶ Universal (Meta-)Logical Reasoning: The Wise Men Puzzle, Data in Brief, 2019.
- ▶ Automating Free Logic in HOL, with an Experimental Application in Category Theory, JAR, 2019.

Deontic Logic Reasoning Infrastructure

- ▶ Designing Normative Theories of Ethical Reasoning, submitted, Url (preprint): [here](#)
- ▶ A Dyadic Deontic Logic in HOL, DEON 2018, 2018. (Best Paper Award)
- ▶ A Deontic Logic Reasoning Infrastructure, CiE 2018, Springer LNCS, 2018.
- ▶ Aqvist's Dyadic Deontic Logic E in HOL, Journal of Applied Logics, in print.
- ▶ I/O Logic in HOL, Journal of Applied Logics, in print.

Conclusion: HOL and Universal Reasoning deserves more attention!

- ▶ **Heureka steps:** can be expected in HOL, but less so in FOL or PL
- ▶ **Lean and elegant** approach to integrate and combine heterogeneous logics
- ▶ **Intuitive user interaction** at abstract level (in modern proof assistants)
- ▶ **Reuse** of existing systems and tools
- ▶ Important **link** between different **research communities**
- ▶ Approach very well suited for (interdisciplinary) **teaching of logics**

Leo-III — Universal Higher-Order Theorem Prover

- ▶ The Higher-Order Prover Leo-III, IJCAR 2018, Springer LNCS, 2018.
- ▶ Theorem Provers for Every Normal Modal Logic, LPAR 2017, Springer LNCS, 2018.

Universal (Meta-)Logical Reasoning

- ▶ Universal (Meta-)Logical Reasoning: Recent Successes, SCP, 2019.
- ▶ Universal (Meta-)Logical Reasoning: The Wise Men Puzzle, Data in Brief, 2019.
- ▶ Automating Free Logic in HOL, with an Experimental Application in Category Theory, JAR, 2019.

Deontic Logic Reasoning Infrastructure

- ▶ Designing Normative Theories of Ethical Reasoning, submitted, Url (preprint): [here](#)
- ▶ A Dyadic Deontic Logic in HOL, DEON 2018, 2018. (Best Paper Award)
- ▶ A Deontic Logic Reasoning Infrastructure, CiE 2018, Springer LNCS, 2018.
- ▶ Aqvist's Dyadic Deontic Logic E in HOL, Journal of Applied Logics, in print.
- ▶ I/O Logic in HOL, Journal of Applied Logics, in print.

Machine Ethics — Gewirth's Principle of Generic Consistency (PGC) in HOL

Emendation of the Golden Rule:

"Act in accord with the generic rights of your recipients as well as of yourself. I shall call this the Principle of Generic Consistency (PGC), since it combines the formal consideration of consistency with the material consideration of rights to the generic features or goods of action."

(Alan Gewirth, Reason and Morality, 1978)

- ▶ **Gewirth's PGC has**
 - ▶ stirred much controversy in moral philosophy
 - ▶ been discussed as means to bound the impact of artificial general intelligence (AGI)
- ▶ **Idea (in a nutshell):**
 - ▶ devise a safety mechanism of a mathematical (deductive) nature
 - ▶ to ensure that an AGI respects human's freedom and well-being
 - ▶ mechanism is based on assumption that it is able to recognize itself, as well as us humans, as agents (prospective purposive agents, PPA) which
 - ▶ act voluntarily on self-chosen purposes, and
 - ▶ reason rationally
- ▶ **References:**
 - ▶ A. Gewirth. Reason and morality. U of Chicago Press, 1978.
 - ▶ D. Beyleveld. The dialectical necessity of morality: An analysis and defense of Alan Gewirth's argument to the principle of generic consistency. U of Chicago Press, 1991.
 - ▶ A. Kornai. Bounding the impact of AGI. J. Experimental & Theoretical AI, 2014.
- ▶ **See our Automation of the PGC in Isabelle/HOL:**
<https://www.isa-afp.org/entries/GewirthPGCProof.html>

But what if we don't want Full Comprehension?

This has been the case for Ed Zalta's
Principia-Logico Metaphysica

Starting point:

- ▶ Higher-order modal logic S5
- ▶ Relational type-theory as foundation
- ▶ Functional type theory (HOL) not directly suited: comes with full comprehension which leads to paradoxes

Example of paradoxical term to exclude:

$$\lambda x \exists F(xF \wedge \neg Fx)$$

See: [Oppenheimer & Zalta, Relations versus functions at the foundations of logic:
Type-theoretic considerations, Journal of Logic and Computation, 2011]

Challenge:

- ▶ Encoding of relational higher-order modal logic S5 in HOL
- ▶ Restricted comprehension to avoid paradoxes

Our solution in HOL: explicitly model denoting and non-denoting terms

[Computer Science and Metaphysics: A Cross-Fertilization, submitted]

[Mechanizing Principia Logico-Metaphysica in Functional Type Theory, submitted]

Russell discovered the well-known paradox in Cantor's set theory.

Similar construction possible in a naive representation of AOT in HOL

Assume that term $[\lambda x \exists F(xF \wedge \neg Fx)]$ denotes a valid property K . (property of being x , such that there exists a property that x encodes, but does not exemplify)

Comprehension axiom then ensures that there is an abstract object which encodes and only encodes K . The question whether this abstract object exemplifies K leads to paradox.

The relational formulation of AOT tries to avoid this issue by restricting the matrix of λ -expressions to so called *propositional formulas* only, that is, to formulas without encoding subformulas. This way the term $[\lambda x \exists F(xF \wedge \neg Fx)]$ is no longer well-formed and the construction of the paradox fails.

Solution in HOL: explicitly model denoting and non-denoting terms

[Kirchner, Benzmüller & Zalta, Computer Science and Metaphysics: A Cross-Fertilization, submitted, preprint: see my website]

[Kirchner, Benzmüller & Zalta, Mechanizing Principia Logico-Metaphysica in Functional Type Theory, submitted, preprint: see my website]