Interacting with Modal Logics in the Coq Proof Assistant

Christoph Benzmüller and **Bruno Woltzenlogel Paleo**
Freie Universität Berlin
Vienna University of Technology
Australian National University

CSR, Listvyanka, 16th of July 2015

$$\Box P$$

P is necessary, P is obligatory, P is known,
P is believed, always P . . .

$$\Diamond P$$

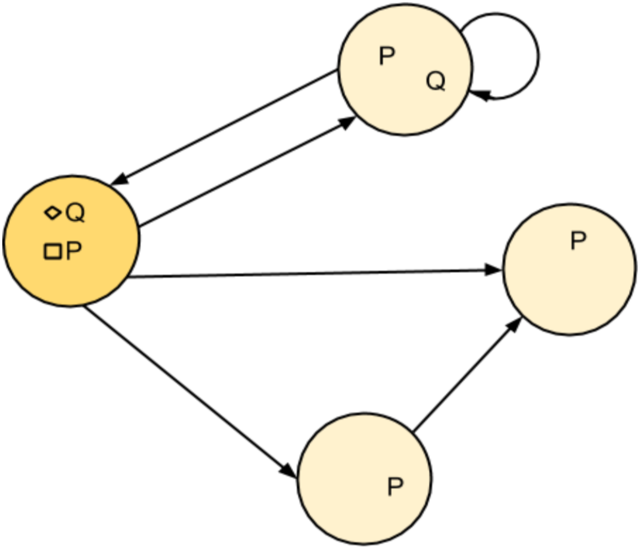
P is possible, P is permissible, P is epistemically possible,
P is doxastically possible, eventually P . . .

$$\Box P$$

P is necessary, P is obligatory, P is known,
P is believed, always P . . .

$$\Diamond P$$

P is possible, P is permissible, P is epistemically possible,
P is doxastically possible, eventually P . . .



P
Q
◇Q
□P
P
P

$$\forall x.(G(x) \equiv \forall \varphi.P(\varphi) \rightarrow \varphi(x))$$

$$(\lambda \varphi_{\iota \rightarrow o}.(P\ \varphi))\ G \qquad \rightsquigarrow_{\beta} \qquad (P\ G)$$

Motivation : Applications (Ontologies, Paraconsistency, *Philosophy*, . . .)

Motivation : Applications (Ontologies, Paraconsistency, *Philosophy*, . . .)

Challenge : No provers for *Higher-order Quantified Modal Logics* (QML)

Motivation : Applications (Ontologies, Paraconsistency, *Philosophy*, . . .)

Challenge : No provers for *Higher-order Quantified Modal Logics* (QML)

Our goals :

efficient automated provers for QML

user-friendly interactive provers for QML

Motivation : Applications (Ontologies, Paraconsistency, *Philosophy*, . . .)

Challenge : No provers for *Higher-order Quantified Modal Logics* (QML)

Our goals :

- efficient automated provers for QML
- user-friendly interactive provers for QML

Approach : Embed QML in *Higher-Order Classical Logic* (HOL)

Then use existing HOL theorem provers for reasoning in QML

interactive:	Isabelle, HOL4, Hol Light, Coq, PVS, . . .
automated:	TPS, LEO-II, Satallax, Nitpick, . . .

QML $\varphi, \psi \quad ::= \quad \ldots \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \Box\varphi \mid \Diamond\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi$

HOL $s, t \quad ::= \quad C \mid x \mid \lambda x s \mid s\, t \mid \neg s \mid s \vee t \mid \forall x\, t$

QML in HOL: QML formulas φ are mapped to HOL predicates $\varphi_{\iota\rightarrow o}$

$$
\begin{array}{lcl}
\neg & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \neg\varphi w \\
\wedge & = & \lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda w_\iota(\varphi w \wedge \psi w) \\
\rightarrow & = & \lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda w_\iota(\neg\varphi w \vee \psi w) \\
\forall & = & \lambda h_{\gamma\rightarrow(\iota\rightarrow o)}\lambda w_\iota \forall d_\gamma\, h d w \\
\exists & = & \lambda h_{\gamma\rightarrow(\iota\rightarrow o)}\lambda w_\iota \exists d_\gamma\, h d w \\
 & & \\
\Box & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \forall u_\iota\, (\neg r w u \vee \varphi u) \\
\Diamond & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \exists u_\iota\, (r w u \wedge \varphi u) \\
 & & \\
\text{valid} & = & \lambda\varphi_{\iota\rightarrow o}\forall w_\iota \varphi w
\end{array}
$$

Ax

The equations in Ax are given as axioms to the HOL provers.

QML $\varphi, \psi \quad ::= \quad \ldots \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \Box\varphi \mid \Diamond\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi$

HOL $s, t \quad ::= \quad C \mid x \mid \lambda x s \mid s\, t \mid \neg s \mid s \vee t \mid \forall x\, t$

QML in HOL: QML formulas φ are mapped to HOL predicates $\varphi_{\iota\rightarrow o}$

$$
\begin{array}{lcl}
\neg & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \neg\varphi w \\
\wedge & = & \lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda w_\iota(\varphi w \wedge \psi w) \\
\rightarrow & = & \lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda w_\iota(\neg\varphi w \vee \psi w) \\
\forall & = & \lambda h_{\gamma\rightarrow(\iota\rightarrow o)}\lambda w_\iota \forall d_\gamma\, h d w \\
\exists & = & \lambda h_{\gamma\rightarrow(\iota\rightarrow o)}\lambda w_\iota \exists d_\gamma\, h d w \\
 & & \\
\Box & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \forall u_\iota\, (\neg r w u \vee \varphi u) \\
\Diamond & = & \lambda\varphi_{\iota\rightarrow o}\lambda w_\iota \exists u_\iota\, (r w u \wedge \varphi u) \\
 & & \\
\text{valid} & = & \lambda\varphi_{\iota\rightarrow o}\forall w_\iota \varphi w
\end{array}
\qquad \text{Ax}
$$

The equations in Ax are given as axioms to the HOL provers.

QML $\varphi, \psi \quad ::= \quad \ldots \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \Box\varphi \mid \Diamond\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi$

HOL $s, t \quad ::= \quad C \mid x \mid \lambda x s \mid s\, t \mid \neg s \mid s \vee t \mid \forall x\, t$

QML in HOL: QML formulas φ are mapped to HOL predicates $\varphi_{\iota\to o}$

$$
\begin{array}{lcl}
\neg & = & \lambda\varphi_{\iota\to o}\lambda w_\iota \neg\varphi w \\
\wedge & = & \lambda\varphi_{\iota\to o}\lambda\psi_{\iota\to o}\lambda w_\iota(\varphi w \wedge \psi w) \\
\rightarrow & = & \lambda\varphi_{\iota\to o}\lambda\psi_{\iota\to o}\lambda w_\iota(\neg\varphi w \vee \psi w) \\
\forall & = & \lambda h_{\gamma\to(\iota\to o)}\lambda w_\iota \forall d_\gamma\, hdw \\
\exists & = & \lambda h_{\gamma\to(\iota\to o)}\lambda w_\iota \exists d_\gamma\, hdw \\
 & & \\
\Box & = & \lambda\varphi_{\iota\to o}\lambda w_\iota \forall u_\iota\, (\neg rwu \vee \varphi u) \\
\Diamond & = & \lambda\varphi_{\iota\to o}\lambda w_\iota \exists u_\iota\, (rwu \wedge \varphi u) \\
 & & \\
\text{valid} & = & \lambda\varphi_{\iota\to o}\forall w_\iota \varphi w
\end{array}
\qquad \text{Ax}
$$

The equations in Ax are given as axioms to the HOL provers.

Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

Example

QML formula	$\Diamond\exists x G(x)$
QML formula in HOL	valid $(\Diamond\exists x G(x))_{\iota\to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond\exists x G(x))_{\iota\to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota\to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

What do we do?

In order to prove that φ is valid in QML,
we instead prove that valid $\varphi_{\iota \to o}$ can be derived from Ax in HOL.

This can be done with interactive or automated HOL theorem provers.

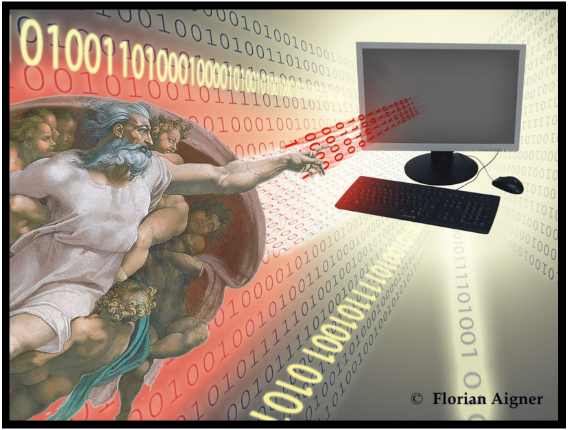
Example

QML formula	$\Diamond \exists x G(x)$
QML formula in HOL	valid $(\Diamond \exists x G(x))_{\iota \to o}$
expansion, $\beta\eta$-conversion	$\forall w_\iota (\Diamond \exists x G(x))_{\iota \to o}\, w$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge (\exists x G(x))_{\iota \to o} u)$
expansion, $\beta\eta$-conversion	$\forall w_\iota \exists u_\iota (rwu \wedge \exists x Gxu)$

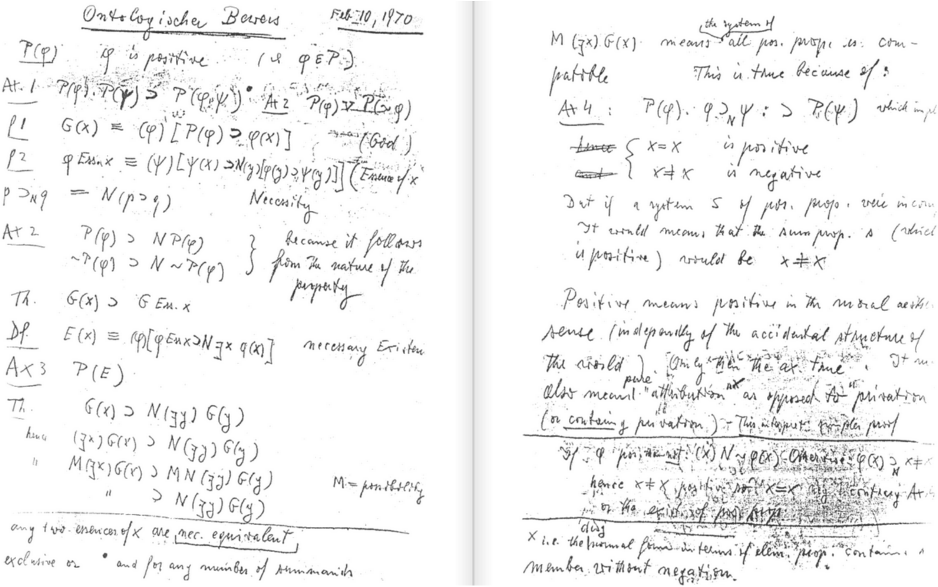
Does this actually work in practice?

Is it efficient ??

Is it user-friendly ??



An Application to Philosophy:
Formalization and Verification of Ontological Arguments


Ontologischer Beweis
Feb 10, 1970
$P(\varphi)$ φ is positive (or $\varphi \in P$)
At. 1 $P(\varphi).P(\psi) \supset P(\varphi.\psi)$
At 2 $P(\varphi) \vee P(\sim\varphi)$
P1 $G(x) \equiv (\varphi)[P(\varphi) \supset \varphi(x)]$ (God)
P2 $\varphi \text{ Ess } x \equiv (\psi)[\psi(x) \supset N(y)[\varphi(y) \supset \psi(y)]]$ (Essence of x)
$p \supset_N q = N(p \supset q)$ Necessity
At 2 $P(\varphi) \supset NP(\varphi)$
$\sim P(\varphi) \supset N \sim P(\varphi)$
because it follows from the nature of the property
Th. $G(x) \supset G \text{ Ess. } x$
Df. $E(x) \equiv (\varphi)[\varphi \text{ Ess } x \supset N \exists x\, \varphi(x)]$ necessary Existence
Ax 3 $P(E)$
Th. $G(x) \supset N(\exists y)\, G(y)$
hence $(\exists x)\, G(x) \supset N(\exists y)\, G(y)$
" $M(\exists x)\, G(x) \supset MN(\exists y)\, G(y)$
" $\supset N(\exists y)\, G(y)$
M = possibility
any two essences of x are nec. equivalent
exclusive or • and for any number of summands
$M(\exists x)\, G(x)$ means the system of all pos. props. is compatible
This is true because of:
At 4: $P(\varphi).\varphi \supset_N \psi : \supset P(\psi)$ which impl
$x = x$ is positive
$x \neq x$ is negative
But if a system S of pos. props. were incon
It would mean that the sum 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 then the ax. true. It m. also mean pure "attribution"* as opposed to privation (or containing privation). This interpret. simpler proof
If φ pos. not: $(x) N \sim\varphi(x)$. Otherwise: $\varphi(x) \supset_N x \neq x$
hence $x \neq x$ positive not $x = x$
* i.e. the disj. normal form in terms of elem. prop. contains a member without negation.

A Long History

pros and cons

. . . Anselm v. C. Gaunilo . . . Th. Aquinas . . . . . . Descartes Spinoza Leibniz . . . Hume Kant . . . Hegel . . . Frege . . . Hartshorne Malcolm Lewis Plantinga Gödel . . .

Anselm's notion of God:

> *"God is that, than which nothing greater can be conceived."*

Gödel's notion of God:

> *"A God-like being possesses all 'positive' properties."*

… Anselm v. C. Gaunilo … Th. Aquinas … … Descartes Spinoza Leibniz … Hume Kant … Hegel … Frege … Hartshorne Malcolm Lewis Plantinga Gödel …

Anselm's notion of God:

"God is that, than which nothing greater can be conceived."

Gödel's notion of God:

"A God-like being possesses all 'positive' properties."

The Ontological Proof Today

Proof Overview

$$\textbf{D1: } G(x) \equiv \forall \varphi.[P(\varphi) \to \varphi(x)]$$

$$\textbf{D2: } \varphi \; ess \; x \equiv \varphi(x) \wedge \forall \psi.(\psi(x) \to \Box \forall x.(\varphi(x) \to \psi(x)))$$

$$\textbf{D3: } NE(x) \equiv \forall \varphi.[\varphi \; ess \; x \to \Box \exists y.\varphi(y)]$$

$$\dfrac{\dfrac{\textbf{A3}}{P(G)} \qquad \dfrac{\dfrac{\textbf{A2}}{\forall \varphi.\forall \psi.[(P(\varphi) \wedge \Box \forall x.[\varphi(x) \to \psi(x)]) \to P(\psi)]} \qquad \dfrac{\textbf{A1a}}{\forall \varphi.[P(\neg \varphi) \to \neg P(\varphi)]}}{\textbf{T1: } \forall \varphi.[P(\varphi) \to \Diamond \exists x.\varphi(x)]}}{\textbf{C1: } \Diamond \exists z.G(z)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\textbf{A1b}}{\forall \varphi.[\neg P(\varphi) \to P(\neg \varphi)]} \qquad \dfrac{\textbf{A4}}{\forall \varphi.[P(\varphi) \to \Box \; P(\varphi)]}}{\textbf{T2: } \forall y.[G(y) \to G \; ess \; y]} \qquad \dfrac{\textbf{A5}}{P(NE)}}{\textbf{L1: } \exists z.G(z) \to \Box \exists x.G(x)}}{\Diamond \exists z.G(z) \to \Diamond \Box \exists x.G(x)} \qquad \dfrac{\textbf{S5}}{\forall \xi.[\Diamond \Box \xi \to \Box \xi]}}{\textbf{L2: } \Diamond \exists z.G(z) \to \Box \exists x.G(x)}$$

$$\dfrac{\textbf{C1: } \Diamond \exists z.G(z) \qquad \textbf{L2: } \Diamond \exists z.G(z) \to \Box \exists x.G(x)}{\textbf{T3: } \Box \exists x.G(x)}$$

	HOL encoding	dependencies	logic	status	LEO-II const/vary	Satallax const/vary	Nitpick const/vary
A1	$[\dot\forall \phi_{\mu\to\sigma}. p_{(\mu\to\sigma)\to\sigma}(\lambda X_\mu. \dot\neg(\phi X)) \dot\equiv \dot\neg(p\phi)]$						
A2	$[\dot\forall \phi_{\mu\to\sigma}. \dot\forall\psi_{\mu\to\sigma}. (p_{(\mu\to\sigma)\to\sigma}\phi \dot\wedge \dot\Box\dot\forall X_\mu.(\phi X \dot\supset \psi X)) \dot\supset p\psi]$						
T1	$[\dot\forall \phi_{\mu\to\sigma}. p_{(\mu\to\sigma)\to\sigma}\phi \dot\supset \dot\Diamond\dot\exists X_\mu. \phi X]$	A1(⊃), A2	K	THM	0.1/0.1	0.0/0.0	—/—
		A1, A2	K	THM	0.1/0.1	0.0/5.2	—/—
D1	$g_{\mu\to\sigma} = \lambda X_\mu. \dot\forall\phi_{\mu\to\sigma}. p_{(\mu\to\sigma)\to\sigma}\phi \dot\supset \phi X$						
A3	$[p_{(\mu\to\sigma)\to\sigma} g_{\mu\to\sigma}]$						
C	$[\dot\Diamond\dot\exists X_\mu. g_{\mu\to\sigma}X]$	T1, D1, A3	K	THM	0.0/0.0	0.0/0.0	—/—
		A1, A2, D1, A3	K	THM	0.0/0.0	5.2/31.3	—/—
A4	$[\dot\forall\phi_{\mu\to\sigma}. p_{(\mu\to\sigma)\to\sigma}\phi \dot\supset \dot\Box p\phi]$						
D2	$\text{ess}_{(\mu\to\sigma)\to\mu\to\sigma} = \lambda\phi_{\mu\to\sigma}.\lambda X_\mu. \phi X \dot\wedge \dot\forall\psi_{\mu\to\sigma}.(\psi X \dot\supset \dot\Box\dot\forall Y_\mu.(\phi Y \dot\supset \psi Y))$						
T2	$[\dot\forall X_\mu. g_{\mu\to\sigma}X \dot\supset (\text{ess}_{(\mu\to\sigma)\to\mu\to\sigma}\, g X)]$	A1, D1, A4, D2	K	THM	19.1/18.3	0.0/0.0	—/—
		A1, A2, D1, A3, A4, D2	K	THM	12.9/14.0	0.0/0.0	—/—
D3	$\text{NE}_{\mu\to\sigma} = \lambda X_\mu. \dot\forall\phi_{\mu\to\sigma}.(\text{ess}\,\phi X \dot\supset \dot\Box\dot\exists Y_\mu.\phi Y)$						
A5	$[p_{(\mu\to\sigma)\to\sigma}\text{NE}_{\mu\to\sigma}]$						
T3	$[\dot\Box\dot\exists X_\mu. g_{\mu\to\sigma}X]$	D1, C, T2, D3, A5	K	CSA	—/—	—/—	3.8/6.2
		A1, A2, D1, A3, A4, D2, D3, A5	K	CSA	—/—	—/—	8.2/7.5
		D1, C, T2, D3, A5	KB	THM	0.0/0.1	0.1/5.3	—/—
		A1, A2, D1, A3, A4, D2, D3, A5	KB	THM	—/—	—/—	—/—
MC	$[s_\sigma \dot\supset \dot\Box s_\sigma]$	D2, T2, T3	KB	THM	17.9/—	3.3/3.2	—/—
		A1, A2, D1, A3, A4, D2, D3, A5	KB	THM	—/—	—/—	—/—
FG	$[\dot\forall\phi_{\mu\to\sigma}.\dot\forall X_\mu.(g_{\mu\to\sigma}X \dot\supset (\dot\neg(p_{(\mu\to\sigma)\to\sigma}\phi) \dot\supset \dot\neg(\phi X)))]$	A1, D1	KB	THM	16.5/—	0.0/0.0	—/—
		A1, A2, D1, A3, A4, D2, D3, A5	KB	THM	12.8/15.1	0.0/5.4	—/—
MT	$[\dot\forall X_\mu.\dot\forall Y_\mu.(g_{\mu\to\sigma}X \dot\supset (g_{\mu\to\sigma}Y \dot\supset X \dot= Y))]$	D1, FG	KB	THM	—/—	0.0/3.3	—/—
		A1, A2, D1, A3, A4, D2, D3, A5	KB	THM	—/—	—/—	—/—
CO	$\emptyset$ (no goal, check for consistency)	A1, A2, D1, A3, A4, D2, D3, A5	KB	SAT	—/—	—/—	7.3/7.4
D2’	$\text{ess}_{(\mu\to\sigma)\to\mu\to\sigma} = \lambda\phi_{\mu\to\sigma}.\lambda X_\mu. \dot\forall\psi_{\mu\to\sigma}.(\psi X \dot\supset \dot\Box\dot\forall Y_\mu.(\phi Y \dot\supset \psi Y))$						
CO’	$\emptyset$ (no goal, check for consistency)	A1(⊃), A2, D2’, D3, A5	KB	UNS	7.5/7.8	—/—	—/—
		A1, A2, D1, A3, A4, D2’, D3, A5	KB	UNS	—/—	—/—	—/—

Proof	**D1'**	**A:A1'**	**A2'**	**A3'**	**A4**	**A4'**	**H:A4**	**A5**	**A5'**	**H:A5**	**T3**	**T3'**	**MC**
Scott (const)	-	-	-	-	N/I	-	-	N/I	-	-	P	-	P
Scott (var)	-	-	-	-	N/I	-	-	N/I	-	-	P	-	P
Anderson (const)	-	-	-	-	R (K4B)	-	-	-	R	-	-	P	CS
Anderson (var)	-	-	-	-	R (K4B)	-	-	-	R	-	-	P	CS
Anderson (mix)	-	-	-	-	R (K4B)	-	-	-	I	-	-	CS	CS
Hájek AOE' (var)	-	-	CS		S/I	-	-	-	S/I	-	-	P (KB)	CS
Hájek AOE'_0 (var)	-	-	-	CS	R	-	-	-	S/U	-	-	P (KB)	CS
Hájek AOE'' (var)	-	-			-	-	S/I	-	-	S/I	-	P (KB)	CS
Anderson (simp) (var)	-	R	R			R (K4B)	-	-		-	-		
Bjørdal (const)	R (K4)	-	R	R	-	R (KT)	-	-	N/I	-	-	P (KB)	CS
Bjørdal (var)	CS	-	R	R	-	R (KT)	-	-	N/I	-	-	P (KB)	CS

Proof	D1'	A:A1'	A2'	A3'	A4	A4'	H:A4	A5	A5'	H:A5	T3	T3'	MC
Scott (const)	-	-	-	-	N/I	-	-	N/I	-	-	P	-	P
Scott (var)	-	-	-	-	N/I	-	-	N/I	-	-	P	-	P
Anderson (const)	-	-	-	-	R (K4B)	-	-	-	R	-	-	P	CS
Anderson (var)	-	-	-	-	R (K4B)	-	-	-	R	-	-	P	CS
Anderson (mix)	-	-	-	-	R (K4B)	-	-	-	I	-	-	CS	CS
Hájek AOE' (var)	-	-	CS		S/I	-	-	-	S/I	-	-	P (KB)	CS
Hájek AOE'_0 (var)	-	-	-	CS	R	-	-	-	S/U	-	-	P (KB)	CS
Hájek AOE'' (var)	-	-			-	-	S/I	-	-	S/I	-	P (KB)	CS
Anderson (simp) (var)	-	R	R			R (K4B)	-	-		-	-		
Bjørdal (const)	R (K4)	-	R	R	-	R (KT)	-	-	N/I	-	-	P (KB)	CS
Bjørdal (var)	CS	-	R	R	-	R (KT)	-	-	N/I	-	-	P (KB)	CS

Leibniz (1646–1716)

characteristica universalis *and* ***calculus ratiocinator***

If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down to their slates, and to say to each other . . . : Let us calculate.

Issues with Fully Automated Reasoning

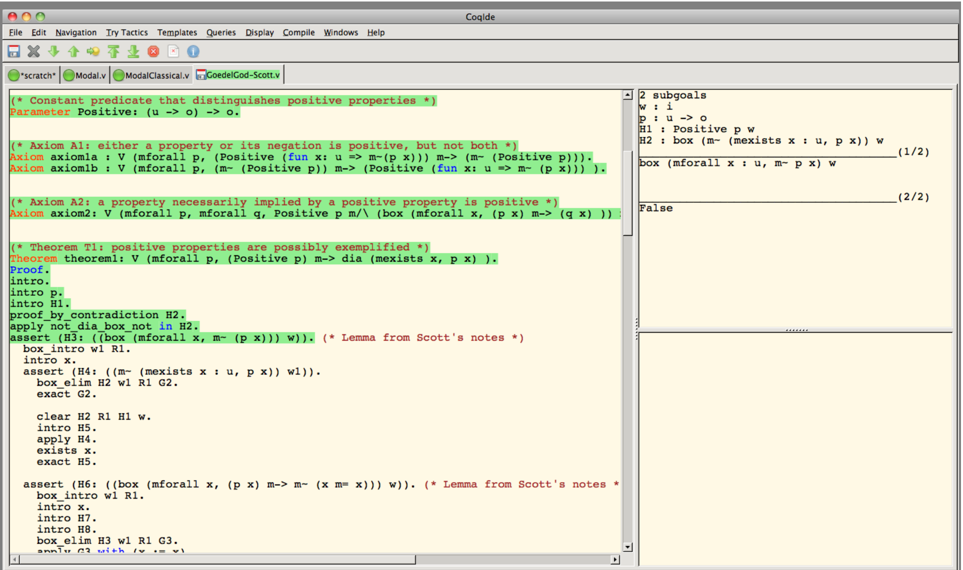
Proofs are hard to read and do not necessarily correspond to the informal proofs being verified

DemoMaterial — bash — 166×52

```
@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>$o)),SV3:$i,SV8:(mu>($i>$o))]: ((((SV8@(((sK2_SY33@SV3)@(^[SX0:mu,SX1:$i]: (~ ((SV22@SX0)@SX1))))@SV8))@(((sK1_SY31@(^[SX0:mu,SX1:$i]: (
~ ((SV22@SX0)@SX1))))@SV8)@SV3))=$true) | (((p@SV8)@SV3)=$false) | (((p@(^[SX0:mu,SX1:$i]: (~ ((SV22@SX0)@SX1))))@SV3)=$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>$o))]: ((((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)=$false) | ((((p@SV9)@SV4) = ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)
@SY28))))@SV4))=$false))),inference(fac_restr,[status(thm)],[56])).
 thf(86,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)=$true) | ((((p@SV9)@SV4) = ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@
SY30))))@SV4))=$false))),inference(fac_restr,[status(thm)],[57])).
 thf(87,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((~ (((p@SV9)@SV4) | ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4))) | (~ ((~ ((p@SV9)@SV4)) | (~ ((p@(^[SY27:mu,
SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)))))=$false) | (((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_equal_neg,[status(thm)],[85])).
 thf(89,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((~ (((p@SV9)@SV4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4))) | (~ ((~ ((p@SV9)@SV4)) | (~ ((p@(^[SY29:mu,
SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)))))=$false) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)=$true))),inference(extcnf_equal_neg,[status(thm)],[86])).
 thf(92,plain,(![SV4:$i,SV9:(mu>($i>$o))]: (((~ ((~ ((p@SV9)@SV4)) | (~ ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4))))=$false) | (((p@(^[SY27:mu,SY28:$i]: (
~ ((SV9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_or_neg,[status(thm)],[87])).
 thf(93,plain,(![SV4:$i,SV9:(mu>($i>$o))]: (((~ (((p@SV9)@SV4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)))=$false) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@
SY29)@SY30))))@SV4)=$true))),inference(extcnf_or_neg,[status(thm)],[89])).
 thf(96,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((~ ((p@SV9)@SV4)) | (~ ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)))=$true) | (((p@(^[SY27:mu,SY28:$i]: (~ ((S
V9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_not_neg,[status(thm)],[92])).
 thf(97,plain,(![SV4:$i,SV9:(mu>($i>$o))]: (((((p@SV9)@SV4) | ((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4))=$true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)
@SY30))))@SV4)=$true))),inference(extcnf_not_neg,[status(thm)],[93])).
 thf(100,plain,(![SV4:$i,SV9:(mu>($i>$o))]: (((~ ((p@SV9)@SV4))=$true) | ((~ ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4))=$true) | (((p@(^[SY27:mu,SY28:$i]:
 (~ ((SV9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_or_pos,[status(thm)],[96])).
 thf(101,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@SV9)@SV4)=$true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)=$true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV
9@SY29)@SY30))))@SV4)=$true))),inference(extcnf_or_pos,[status(thm)],[97])).
 thf(103,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@SV9)@SV4)=$false) | ((~ ((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4))=$true) | (((p@(^[SY27:mu,SY28:$i]: (~
 ((SV9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_not_pos,[status(thm)],[100])).
 thf(105,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)=$false) | (((p@SV9)@SV4)=$false) | (((p@(^[SY27:mu,SY28:$i]: (~ ((
SV9@SY27)@SY28))))@SV4)=$false))),inference(extcnf_not_pos,[status(thm)],[103])).
 thf(107,plain,(![SV8:(mu>($i>$o)),SV3:$i,SV22:(mu>($i>$o))]: ((((SV22@(((sK2_SY33@SV3)@(^[SX0:mu,SX1:$i]: (~ ((SV22@SX0)@SX1))))@SV8))@(((sK1_SY31@(^[SX0:mu,SX1:$i]:
 (~ ((SV22@SX0)@SX1))))@SV8)@SV3))=$true) | (((p@SV8)@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>$o)),SV3:$i,SV15:(mu>($i>$o))]: ((((SV15@(((sK2_SY33@SV3)@SV11)@(^[SX0:mu,SX1:$i]: (~ ((SV15@SX0)@SX1)))))@(((sK1_SY31@SV11)@(^[SX0:mu,
SX1:$i]: (~ ((SV15@SX0)@SX1))))@SV3))=$false) | (((p@(^[SX0:mu,SX1:$i]: (~ ((SV15@SX0)@SX1))))@SV3)=$false) | (((p@SV11)@SV3)=$true))),inference(extcnf_not_pos,[statu
s(thm)],[81])).
 thf(109,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@(^[SY27:mu,SY28:$i]: (~ ((SV9@SY27)@SY28))))@SV4)=$false) | (((p@SV9)@SV4)=$false))),inference(sim,[status(thm)],[10
5])).
 thf(110,plain,(![SV4:$i,SV9:(mu>($i>$o))]: ((((p@SV9)@SV4)=$true) | (((p@(^[SY29:mu,SY30:$i]: (~ ((SV9@SY29)@SY30))))@SV4)=$true))),inference(sim,[status(thm)],[101]
)).
 thf(111,plain,(![SV3:$i,SV8:(mu>($i>$o))]: ((((p@SV8)@SV3)=$false) | (((p@(^[SX0:mu,SX1:$i]: $true))@SV3)=$true))),inference(sim,[status(thm)],[76])).
 thf(112,plain,(![SV11:(mu>($i>$o)),SV3:$i]: ((((p@(^[SX0:mu,SX1:$i]: $false))@SV3)=$false) | (((p@SV11)@SV3)=$true))),inference(sim,[status(thm)],[80])).
 thf(113,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[25,112,111,110,109,108,107,84,83,82,75,74,73,72,71,70,69,68,67,66,65,62,57,56,51,42,29])).
 thf(114,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[113])).
% SZS output end CNFRefutation

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

% SZS status Unsatisfiable for ConsistencyWithoutFirstConjunctinD2.p : (rf:0,axioms:6,ps:3,u:6,ude:false,rLeibEQ:true,rAndEQ:true,use_choice:true,use_extuni:true,use_
extcnf_combined:true,expand_extuni:false,foatp:e,atp_timeout:25,atp_calls_frequency:10,ordering:none,proof_output:1,clause_count:113,loop_count:0,foatp_calls:2,transl
ation:fof_full)
ontoleo:DemoMaterial cbenzmueller$
```



CoqIde
File Edit Navigation Try Tactics Templates Queries Display Compile Windows Help
scratch
Modal.v
ModalClassical.v
GoedelGod-Scott.v
(* Constant predicate that distinguishes positive properties *)
Parameter Positive: (u -> o) -> o.
(* Axiom A1: either a property or its negation is positive, but not both *)
Axiom axiom1a : V (mforall p, (Positive (fun x: u => m~(p x))) m-> (m~ (Positive p))).
Axiom axiom1b : V (mforall p, (m~ (Positive p)) m-> (Positive (fun x: u => m~ (p x)))).
(* Axiom A2: a property necessarily implied by a positive property is positive *)
Axiom axiom2: V (mforall p, mforall q, Positive p m/\ (box (mforall x, (p x) m-> (q x)))
(* Theorem T1: positive properties are possibly exemplified *)
Theorem theorem1: V (mforall p, (Positive p) m-> dia (mexists x, p x)).
Proof.
intro.
intro p.
intro H1.
proof_by_contradiction H2.
apply not_dia_box_not in H2.
assert (H3: ((box (mforall x, m~ (p x))) w)). (* Lemma from Scott's notes *)
box_intro w1 R1.
intro x.
assert (H4: ((m~ (mexists x : u, p x)) w1)).
box_elim H2 w1 R1 G2.
exact G2.
clear H2 R1 H1 w.
intro H5.
apply H4.
exists x.
exact H5.
assert (H6: ((box (mforall x, (p x) m-> m~ (x m= x))) w)). (* Lemma from Scott's notes *
box_intro w1 R1.
intro x.
intro H7.
intro H8.
box_elim H3 w1 R1 G3.
2 subgoals
w : i
p : u -> o
H1 : Positive p w
H2 : box (m~ (mexists x : u, p x)) w
(1/2)
box (mforall x : u, m~ p x) w
(2/2)
False

- Calculus of Inductive Constructions (CIC)
- Related to CC and λC (cf. previous talk).
- A minimalistic higher-order natural deduction calculus.
- Typical natural deduction rules are admissible.

Typical Natural Deduction Rules

Interactive proving using basic tactics in Coq feels roughly like constructing a natural deduction proof

$$\dfrac{A \vee B \quad \begin{array}{c}\overline{A}\\ \vdots \\ C\end{array} \quad \begin{array}{c}\overline{B}\\ \vdots \\ C\end{array}}{C}\ \vee_E \qquad \dfrac{A \quad B}{A \wedge B}\ \wedge_I \qquad \dfrac{\begin{array}{c}\overline{A}^{\,h}\\ \vdots \\ B\end{array}}{A \rightarrow B}\ \rightarrow_I^h$$

$$\dfrac{A}{A \vee B}\ \vee_{I_1} \qquad \dfrac{A \wedge B}{A}\ \wedge_{E_1} \qquad \dfrac{B}{A \rightarrow B}\ \rightarrow_I$$

$$\dfrac{B}{A \vee B}\ \vee_{I_2} \qquad \dfrac{A \wedge B}{B}\ \wedge_{E_2} \qquad \dfrac{A \quad A \rightarrow B}{B}\ \rightarrow_E$$

$$\dfrac{A[\alpha]}{\forall x.A[x]}\ \forall_I \qquad \dfrac{\forall x.A[x]}{A[t]}\ \forall_E \qquad \dfrac{A[t]}{\exists x.A[x]}\ \exists_I \qquad \dfrac{\exists x.A[x]}{A[\beta]}\ \exists_E$$

$$\neg A \equiv A \rightarrow \bot \qquad \color{red}{\dfrac{\neg\neg A}{A}\ \neg\neg_E}$$

- Challenges:
  - Can we hide the semantic embedding from the user?
  - Can we provide an interaction experience to the user that differs as little as possible from what he is already used to?
  - Can we reconstruct, step-by-step in Coq, precisely Scott's formulation of Gödel's argument?

- Challenges:
  - Can we hide the semantic embedding from the user?
  - Can we provide an interaction experience to the user that differs as little as possible from what he is already used to?
  - Can we reconstruct, step-by-step in Coq, precisely Scott's formulation of Gödel's argument?

- Challenges:
  - Can we hide the semantic embedding from the user?
  - Can we provide an interaction experience to the user that differs as little as possible from what he is already used to?
  - Can we reconstruct, step-by-step in Coq, precisely Scott's formulation of Gödel's argument?

Modal Logics in the Coq Proof Assistant

```
Parameter i: Type. (* Type for worlds *)
Parameter u: Type. (* Type for individuals *)
Definition o := i -> Prop. (* Type of modal propositions *)

Parameter r: i -> i -> Prop. (* Accessibility relation for worlds *)

Definition mnot (p: o)(w: i) := ~ (p w).
Notation "m~  p" := (mnot p) (at level 74, right associativity).

Definition mand (p q:o)(w: i) := (p w) /\ (q w).
Notation "p m/\ q" := (mand p q) (at level 79, right associativity).

Definition mor (p q:o)(w: i) := (p w) \/ (q w).
Notation "p m\/ q" := (mor p q) (at level 79, right associativity).

Definition mimplies (p q:o)(w:i) := (p w) -> (q w).
Notation "p m-> q" := (mimplies p q) (at level 99, right associativity).

Definition mequiv (p q:o)(w:i) := (p w) <-> (q w).
Notation "p m<-> q" := (mequiv p q) (at level 99, right associativity).
```

```
Definition A {t: Type}(p: t -> o)(w: i) := forall x, p x w.
Notation "'mforall'  x , p" := (A (fun x => p))
  (at level 200, x ident, right associativity) : type_scope.
Notation "'mforall' x : t , p" := (A (fun x:t => p))
  (at level 200, x ident, right associativity,
    format "'[' 'mforall' '/ '  x  :  t , '/ '  p ']'")
  : type_scope.

Definition E {t: Type}(p: t -> o)(w: i) := exists x, p x w.
Notation "'mexists' x , p" := (E (fun x => p))
  (at level 200, x ident, right associativity) : type_scope.
Notation "'mexists' x : t , p" := (E (fun x:t => p))
  (at level 200, x ident, right associativity,
    format "'[' 'mexists' '/ '  x  :  t , '/ '  p ']'")
  : type_scope.
```

Modal Logics in the Coq Proof Assistant

```
Definition box (p: o) := fun w => forall w1, (r w w1) -> (p w1).
Definition dia (p: o) := fun w => exists w1, (r w w1) /\ (p w1).
```

```
Lemma mp_dia:
  [mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)].
Proof. mv.
intros p q H1 H2. unfold dia. unfold dia in H1. unfold box in H2.
destruct H1 as [w0 [R1 H1]]. exists w0. split.
  exact R1.
  apply H2.
    exact R1.
    exact H1.
Qed.
```

```
Ltac box_i := let w := fresh "w" in let R := fresh "R"
              in (intro w at top; intro R at top).

Ltac box_elim H w1 H1 := match type of H with
     ((box ?p) ?w) => cut (p w1);
                      [intros H1 | (apply (H w1); try assumption)] end.

Ltac box_e H H1:= match goal with | [ |- (_ ?w) ] => box_elim H w H1 end.

Ltac dia_e H := let w := fresh "w" in let R := fresh "R" in
                (destruct H as [w [R H]]; move w at top; move R at top).
Ltac dia_i w := (exists w; split; [assumption | idtac]).
```

```
Lemma mp_dia:
  [mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)].
Proof. mv.
intros p q H1 H2. unfold dia. unfold dia in H1. unfold box in H2.
destruct H1 as [w0 [R1 H1]]. exists w0. split.
  exact R1.
  apply H2.
    exact R1.
    exact H1.
Qed.
```

```
Lemma mp_dia:
  [mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)].
Proof. mv.
intros p q H1 H2. dia_e H1. dia_i w0. box_e H2 H3. apply H3. exact H1.
Qed.
```

$$\dfrac{\alpha : \boxed{\begin{array}{c}\vdots\\ A\end{array}}}{\Box A}\ \Box_I \qquad\qquad \dfrac{\Box A}{t : \boxed{\begin{array}{c}A\\ \vdots\end{array}}}\ \Box_E$$

$$\dfrac{t : \boxed{\begin{array}{c}\vdots\\ A\end{array}}}{\Diamond A}\ \Diamond_I \qquad\qquad \dfrac{\Diamond A}{\beta : \boxed{\begin{array}{c}A\\ \vdots\end{array}}}\ \Diamond_E$$

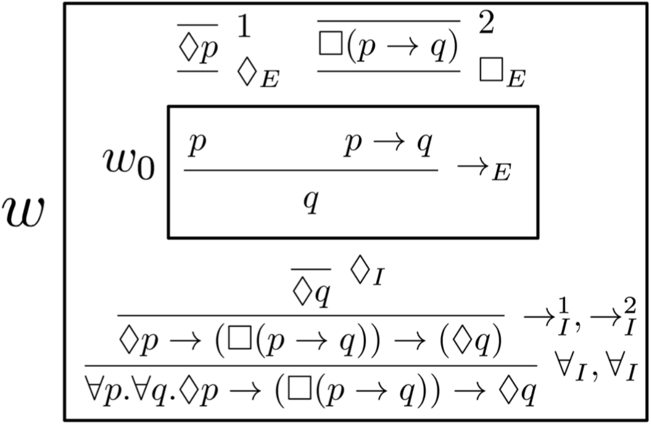
eigen-box condition:

$\Box_I$ and $\Diamond_E$ are *strong* modal rules:
α and β must be fresh names for the boxes they access
(in analogy to the eigen-variable condition for strong quantifier rules).
Every box must be accessed by *exactly one* strong modal inference.

boxed assumption condition:

assumptions should be discharged within
the box where they are created.

```
Lemma mp_dia:
  [mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)].
Proof. mv.
intros p q H1 H2. dia_e H1. dia_i w0. box_e H2 H3. apply H3. exact H1.
Qed.
```



```
(* Theorem T1: positive properties are possibly exemplified *)
Theorem theorem1: [ mforall p, (Positive p) m-> dia (mexists x, p x) ].
Proof. mv.
intro p. intro H1. proof_by_contradiction H2. apply not_dia_box_not in H2.
assert (H3: ((box (mforall x, m~ (p x))) w)). (* Scott *)
  box_i. intro x. assert (H4: ((m~ (mexists x : u, p x)) w0)).
    box_e H2 G2. exact G2.
    clear H2 R H1 w. intro H5. apply H4. exists x. exact H5.
  assert (H6: ((box (mforall x, (p x) m-> m~ (x m= x))) w)). (* Scott *)
    box_i. intro x. intros H7 H8. box_elim H3 w0 G3. eapply G3. exact H7.
    assert (H9: ((Positive (fun x => m~ (x m= x))) w)). (* Scott *)
      apply (axiom2 w p (fun x => m~ (x m= x))). split.
        exact H1.
        exact H6.
      assert (H10: ((box (mforall x, (p x) m-> (x m= x))) w)). (* Scott *)
        box_i. intros x H11. reflexivity.
        assert (H11 : ((Positive (fun x => (x m= x))) w)). (* Scott *)
          apply (axiom2 w p (fun x => x m= x )). split.
            exact H1.
            exact H10.
          apply axiom1a in H9. contradiction.
Qed.
```

Conclusions

- Contributions:
  - Efficient automated reasoning for QML
  - User-friendly interactive reasoning with QML in Coq
  - A new natural deduction calculus for higher-order modal logics
  - Non-trivial new benchmark problems for HOL provers
  - Verification of existing results about Gödel's proof (e.g. Modal Collapse)
  - New results about Gödel's proof (e.g. Inconsistency)
  - Resolution of philosophical controversies (Leibniz's dream)
  - Computer-Assisted Theoretical Philosophy
    - Ed Zalta's *Computational Metaphysics* project at Stanford University
    - John Rushby's formalization of Anselm's proof using PVS