Faithful Logic Embeddings in HOL: Deep and Shallow, Propositional and Quantified

Christoph Benzmüller and Luca Pasetto

U Bamberg | FU Berlin | U Luxemburg

ESSLLI 2026

Many related papers on deep and shallow embeddings in HOL since the early 1990s — see references in CADE 30 paper (and references therein)

Our interest here:

- Interlinked logic enbeddings in LogiKEy: deep & shallow, prop. & quantified
- Reasoning at meta- and object level
- Faithfulness proofs — can they be automated?
- Evaluation — which type of embedding performs best?

Propositional Modal Logic (PML) and Quantified Boolean Formulas (QBF) are used used in this work only for illustration and reuse in education (courses on universal logical reasoning) — the QBF part is new !!!

PML

2 Propositional modal logic (PML)

Syntax

The *syntax* of PML is defined by the grammar $\varphi, \psi := a \mid \neg\varphi \mid \varphi \supset \psi \mid \Box\varphi$, where $a \in \mathcal{S}$ are propositional symbols declared in a (nonempty) signature $\mathcal{S}$. Further logical connectives can be defined as usual, e.g.: $\varphi \vee \psi := \neg\varphi \supset \psi$, $\varphi \wedge \psi := \neg(\varphi \supset \neg\psi)$, $\Diamond\varphi := \neg\Box\neg\varphi$, $\top := p \supset p$ (for some $p \in \mathcal{S}$), and $\bot := \neg\top$.

A *semantics* for PML, called relational semantics or possible-world semantics, determines the truth of a formula φ relative to a so-called possible world w and a relational model. A relational model is a tuple $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$, where $\mathcal{W}$ is a set of possible worlds (called the universe), $\mathcal{R}$ is a binary relation on W (called the accessibility relation), $\mathcal{V}$ is a valuation function that assigns a truth value to each symbol $a \in \mathcal{S}$ in each world $w \in \mathcal{W}$, where $\mathcal{S}$ is the signature of propositional symbols. $\langle \mathcal{W}, \mathcal{R} \rangle$ is called a *frame*. Formally, the truth of a formula φ with respect to a given model $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$ and a possible world $w \in \mathcal{W}$ is defined as follows:

Semantics

$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models a$ iff $\mathcal{V}(a)(w)$ is true

$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \neg\varphi$ iff not $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \varphi$

$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \varphi \supset \psi$ iff $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \varphi$ implies $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \psi$

$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \Box\varphi$ iff for all v with $\mathcal{R}wv$ we have $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, v \models \varphi$

Note that in this recursive definition of truth the only parameter that may be recursively modified is the possible world w in the $\Box$-clause, which states that a formula φ is "necessary" in a world w iff φ is true at all reachable worlds v from w. Based on the above definitions a notion of validity can be defined: A formula φ is valid (in so called base modal logic **K**) iff $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models \varphi$ for all $\mathcal{W}$, $\mathcal{R}$, $\mathcal{V}$, and $w \in \mathcal{W}$.

Deep vs. Shallow (deep)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

$\varphi, \psi :=$ at(a) | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\varphi, \psi :=$ a | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\langle W, R, V\rangle, w \vDash$ at(a) iff $V(a)(w)$ is true

$\langle W, R, V\rangle, w \vDash \neg\varphi$ iff not $\langle W, R, V\rangle, w \vDash \varphi$

$\langle W, R, V\rangle, w \vDash \varphi \supset \psi$ iff $\langle W, R, V\rangle, w \vDash \varphi$ implies $\langle W, R, V\rangle, w \vDash \psi$

$\langle W, R, V\rangle, w \vDash \Box\varphi$ iff for all $v \in W$ with Rwv : $\langle W, R, V\rangle, v \vDash \varphi$

$\vDash \varphi$ **iff** $\langle W, R, V\rangle, v \vDash \varphi$ **for all** W, R, V, v

$\vDash \quad \neg\Box a \supset \neg(\Box\neg(a \supset b))$

Deep vs. Shallow (deep)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

```
3 —‹Deep embedding (of propositional modal logic in HOL)›
4  datatype PML = AtmD 𝒮 ("_ᵈ") | NotD PML ("¬ᵈ") | ImpD PML PML (infixr "⊃ᵈ" 93) | BoxD PML ("□ᵈ")
```

```
11 —‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
12  primrec RelativeTruthD :: "𝒲⇒ℛ⇒𝒱 ⇒w⇒PML⇒bool" ("⟨_,_,_⟩,_ ⊨ᵈ _") where
13      "⟨W,R,V⟩, w ⊨ᵈ aᵈ        = (V a w)"
14    | "⟨W,R,V⟩, w ⊨ᵈ ¬ᵈφ       = (¬ ⟨W,R,V⟩, w ⊨ᵈ φ)"
15    | "⟨W,R,V⟩, w ⊨ᵈ φ ⊃ᵈ ψ    = (⟨W,R,V⟩, w ⊨ᵈ φ ⟶ ⟨W,R,V⟩, w ⊨ᵈ ψ)"
16    | "⟨W,R,V⟩, w ⊨ᵈ □ᵈφ       = (∀v:W. R w v ⟶ ⟨W,R,V⟩, v ⊨ᵈ φ)"
17 —‹Definition of validity›
18  definition ValD ("⊨ᵈ _") where "(⊨ᵈ φ) ≡ (∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ)"
```

```
⊨ᵈ ¬ᵈ (□ᵈ aᵈ) ⊃ᵈ ¬ᵈ (□ᵈ (¬ᵈ (aᵈ ⊃ᵈ bᵈ)))
```

Superscript d for "deep"

Deep vs. Shallow (shallow — maximal)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

A shallow embedding eschews the algebraic datatype, and hence the explicit representation of the abstract syntax of the language; instead, the language is defined directly in terms of its semantics."

$\varphi, \psi := \mathsf{at}(a) \mid \neg\varphi \mid \varphi \supset \psi \mid \Box\varphi \qquad (a \in S)$

$\varphi, \psi := \quad a \quad \mid \neg\varphi \mid \varphi \supset \psi \mid \Box\varphi \qquad (a \in S)$

$\langle W, R, V\rangle, w \vDash \mathsf{at}(a)$ iff $V(a)(w)$ is true

$\langle W, R, V\rangle, w \vDash \neg\varphi$ iff not $\langle W, R, V\rangle, w \vDash \varphi$

$\langle W, R, V\rangle, w \vDash \varphi \supset \psi$ iff $\langle W, R, V\rangle, w \vDash \varphi$ implies $\langle W, R, V\rangle, w \vDash \psi$

$\langle W, R, V\rangle, w \vDash \Box\varphi$ iff for all $v \in W$ with $Rwv : \langle W, R, V\rangle, v \vDash \varphi$

$\vDash \varphi$ **iff** $\langle W, R, V\rangle, v \vDash \varphi$ **for all** W, R, V, v

at	$\lambda a\,.\,\lambda W R V w\,.\,V a w$
$\neg$	$\lambda\varphi\,.\,\lambda W R V w\,.\,\neg(\varphi\, W R V w)$
$\supset$	$\lambda\varphi\,.\,\lambda\psi\,.\,\lambda W R V w\,.\,(\varphi\, W R V w) \rightarrow (\psi\, W R V w)$
$\Box$	$\lambda\varphi\,.\,\lambda\, W R V w\,.\,\forall v{:}W\,.\,R w v \rightarrow (\varphi\, W R V v)$
$\vDash$	$\lambda\varphi\,.\,\forall\, W R V w\,.\,(\varphi\, W R V w)$

$\vDash \quad \neg\Box a \supset \neg(\Box\neg(a \supset b))$

Deep vs. Shallow (shallow — maximal)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

A shallow embedding eschews the algebraic datatype, and hence the explicit representation of the abstract syntax of the language; instead, the language is defined directly in terms of its semantics."

Superscript
^{s}
for "shallow"

```
3  —‹Shallow embedding (of propositional modal logic in HOL)›
4  type_synonym σ = "𝒲⇒ℛ⇒𝒱⇒w⇒bool"
5  definition AtmS::"𝒮⇒σ"      ("_ˢ")          where   "aˢ ≡ λW R V w. V a w"
6  definition NegS::"σ⇒σ"      ("¬ˢ")          where   "¬ˢ φ ≡ λW R V w. ¬(φ W R V w)"
7  definition ImpS::"σ⇒σ⇒σ" (infixr "⊃ˢ" 93) where   "φ ⊃ˢ ψ ≡ λW R V w. (φ W R V w) ⟶ (ψ W R V w)"
8  definition BoxS::"σ⇒σ"      ("□ˢ")          where   "□ˢ φ ≡ λW R V w. ∀v:W. R w v ⟶ (φ W R V v)"

15 —‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
16 definition RelativeTruthS :: "𝒲⇒ℛ⇒𝒱⇒w⇒σ⇒bool" ("⟨_,_,_⟩,_ ⊨ˢ _") where "⟨W,R,V⟩,w ⊨ˢ φ ≡ φ W R V w"
17 —‹Definition of validity›
18 definition  ValS ("⊨ˢ _") where "⊨ˢ φ  ≡  ∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ˢ φ"
```

$\models^{s} \neg^{s} (\Box^{s} a^{s}) \supset^{s} \neg^{s} (\Box^{s} (\neg^{s} (a^{s} \supset^{s} b^{s})))$

Deep vs. Shallow (shallow — maximal)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

A shallow embedding eschews the algebraic datatype, and hence the explicit representation of the abstract syntax of the language; instead, the language is defined directly in terms of its semantics."

$\varphi, \psi :=$ at(a) | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\varphi, \psi :=$ a | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\langle W, R, V\rangle, w \vDash$ at(a) iff $V(a)(w)$ is true

$\langle W, R, V\rangle, w \vDash \neg\varphi$ iff not $\langle W, R, V\rangle, w \vDash \varphi$

$\langle W, R, V\rangle, w \vDash \varphi \supset \psi$ iff $\langle W, R, V\rangle, w \vDash \varphi$ implies $\langle W, R, V\rangle, w \vDash \psi$

$\langle W, R, V\rangle, w \vDash \Box\varphi$ iff for all $v \in W$ with Rwv : $\langle W, R, V\rangle, v \vDash \varphi$

$\vDash \varphi$ **iff** $\langle W, R, V\rangle, v \vDash \varphi$ **for all** W, R, V, v

at	$\lambda a\,.\,\lambda W R V w\,.\,V a w$
$\neg$	$\lambda\varphi\,.\,\lambda W R V w\,.\,\neg(\varphi\, W R V w)$
$\supset$	$\lambda\varphi\,.\,\lambda\psi\,.\,\lambda W R V w\,.\,(\varphi\, W R V w) \rightarrow (\psi\, W R V w)$
$\Box$	$\lambda\varphi\,.\,\lambda W R V w\,.\,\forall v{:}W\,.\,R w v \rightarrow (\varphi\, W R V v)$
$\vDash$	$\lambda\varphi\,.\,\forall W R V w\,.\,(\varphi\, W R V w)$

$\vDash \quad \neg\Box a \supset \neg(\Box\neg(a \supset b))$

Deep vs. Shallow (shallow — minimal)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

A shallow embedding eschews the algebraic datatype, and hence the explicit representation of the abstract syntax of the language; instead, the language is defined directly in terms of its semantics."

$\varphi, \psi :=$ at(a) | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\varphi, \psi :=$ a | $\neg\varphi$ | $\varphi \supset \psi$ | $\Box\varphi$ $\quad (a \in S)$

$\langle W, R, V\rangle, w \vDash$ at(a) iff $V(a)(w)$ is true

$\langle W, R, V\rangle, w \vDash \neg\varphi$ iff not $\langle W, R, V\rangle, w \vDash \varphi$

$\langle W, R, V\rangle, w \vDash \varphi \supset \psi$ iff $\langle W, R, V\rangle, w \vDash \varphi$ implies $\langle W, R, V\rangle, w \vDash \psi$

$\langle W, R, V\rangle, w \vDash \Box\varphi$ iff for all $v \in W$ with Rwv : $\langle W, R, V\rangle, v \vDash \varphi$

$\vDash \varphi$ **iff** $\langle W, R, V\rangle, v \vDash \varphi$ **for all** W, R, V, v

consts V, R

at	$\lambda a\,.\,\lambda w\,.\,V\,a\,w$
$\neg$	$\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\,w)$
$\supset$	$\lambda\varphi\,.\,\lambda\psi\,.\,\lambda w\,.\,(\varphi\,w) \rightarrow (\psi\,w)$
$\Box$	$\lambda\varphi\,.\,\lambda w\,.\,\forall v\,.\,R\,w\,v \rightarrow (\varphi\,v)$
$\vDash$	$\lambda\varphi\,.\,\forall w\,.\,\varphi\,w$

$\vDash \quad \neg\Box a \supset \neg(\Box\neg(a \supset b))$

Deep vs. Shallow (shallow — minimal)

Gibbons & Wu (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). Doi: 10.1145/2628136.2628138

"[...] a deep embedding consists of a representation of the abstract syntax as an algebraic datatype, together with some functions that assign semantics to that syntax by traversing the algebraic datatype.

A shallow embedding eschews the algebraic datatype, and hence the explicit representation of the abstract syntax of the language; instead, the language is defined directly in terms of its semantics."

Superscript m for "min. shallow"

```
3  —‹The acessibility relation R and the valuation function V are introduced as constants at the meta-level HOL›
4  consts R::𝓡 V::𝓥
5  —‹Shallow embedding (of propositional modal logic in HOL)›
6  type_synonym σ = "w⇒bool"
7  definition AtmM::"𝒮⇒σ"      ("_ᵐ")          where   "aᵐ ≡ λw. V a w"
8  definition NegM::"σ⇒σ"      ("¬ᵐ")          where   "¬ᵐ φ ≡ λw. ¬φ w"
9  definition ImpM::"σ⇒σ⇒σ" (infixr "⊃ᵐ" 93)  where   "φ ⊃ᵐ ψ ≡ λw. φ w ⟶ ψ w"
10 definition BoxM::"σ⇒σ"      ("□ᵐ")          where   "□ᵐ φ ≡ λw. ∀v. R w v ⟶ φ v"
```

```
17 —‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
18 definition RelativeTruthM :: "w⇒σ⇒bool" ("_ ⊨ᵐ _") where "w ⊨ᵐ φ ≡ φ w"
19 —‹Definition of validity›
20 definition  ValM ("⊨ᵐ _") where "⊨ᵐ φ  ≡ ∀w::w. w ⊨ᵐ φ"
```

```
⊨ᵐ ¬ᵐ (□ᵐ aᵐ) ⊃ᵐ ¬ᵐ (□ᵐ (¬ᵐ (aᵐ ⊃ᵐ bᵐ)))
```

Example (minimal shallow):

$\models \neg\Box a \supset \neg(\Box \neg(a \supset b))$

$\models^{m} \neg^{m} (\Box^{m} a^{m}) \supset^{m} \neg^{m} (\Box^{m} (\neg^{m} (a^{m} \supset^{m} b^{m})))$

$\models \neg\Box a \supset \neg(\Box \neg(a \supset b))$

$\downarrow \neg \quad \lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w)$

$\models (\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w))\,\Box a \supset (\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w))(\Box\,(\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w))(a \supset b))$

$\downarrow \Box \quad \lambda\varphi\,.\,\lambda w\,.\,\forall v\,.\,R\,w\,v \rightarrow \varphi\, v$

$\models (\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w))((\lambda\varphi\,.\,\lambda w\,.\,\forall v\,.\,R\,w\,v \rightarrow \varphi\, v)\,a) \supset (\lambda w\,.\,\neg((\lambda\varphi\,.\,\lambda w\,.\,\forall v\,.\,R\,w\,v \rightarrow \varphi\, v)(\lambda\varphi\,.\,\lambda w\,.\,\neg(\varphi\, w))(a \supset b))\, w))$

$\downarrow \supset \quad \lambda\varphi\,.\,\lambda\psi\,.\,\lambda w\,.\,(\varphi\, w) \rightarrow (\psi\, w)$

$\models \quad \ldots \qquad \ldots$

$\downarrow \mathrm{at} \quad \lambda a\,.\,\lambda w\,.\,(V\,a\,w)$

$\models \quad \ldots \qquad \ldots$

$\downarrow \models \quad \lambda\varphi\,.\,\forall w\,.\,(\varphi\, w)$

$\ldots \qquad \ldots \qquad \ldots$

$\downarrow \beta\text{-reduction}$

$\forall w\,.\,\neg(\forall v\,.\,R\,w\,v \rightarrow V\,a\,v) \rightarrow \neg(\forall v\,.\,R\,w\,v \rightarrow \neg(V\,a\,v \rightarrow V\,b\,v))$

$\downarrow$ simplification

$\forall w\,.\,(\exists v\,.\,R\,w\,v \wedge \neg V\,a\,v) \rightarrow (\exists v\,.\,R\,w\,v \wedge (V\,a\,v \rightarrow V\,b\,v))$

Standard translation of PML to FOL via unfolding and β-reduction in HOL

Example (all embeddings): $\models \neg\Box a \supset \neg(\Box \neg(a \supset b))$

shallow minimal ($.^m$)

$\models^m \neg^m (\Box^m a^m) \supset^m \neg^m (\Box^m (\neg^m (a^m \supset^m b^m)))$

$\forall w.\ (\exists v.\ R\ w\ v \wedge \neg\ V\ a\ v) \longrightarrow (\exists v.\ R\ w\ v \wedge (V\ a\ v \longrightarrow V\ b\ v))$

shallow maximal ($.^s$)

$\models^s \neg^s (\Box^s a^s) \supset^s \neg^s (\Box^s (\neg^s (a^s \supset^s b^s)))$

$\forall W\ R\ V\ x.$
$W\ x \longrightarrow$
$(\exists xa.\ W\ xa \wedge R\ x\ xa \wedge \neg\ V\ a\ xa) \longrightarrow (\exists xa.\ W\ xa \wedge R\ x\ xa \wedge (V\ a\ xa \longrightarrow V\ b\ xa))$

deep ($.^d$)

$\models^d \neg^d (\Box^d a^d) \supset^d \neg^d (\Box^d (\neg^d (a^d \supset^d b^d)))$

$\forall W\ R\ V\ x.$
$W\ x \longrightarrow$
$(\exists xa.\ W\ xa \wedge R\ x\ xa \wedge \neg\ V\ a\ xa) \longrightarrow (\exists xa.\ W\ xa \wedge R\ x\ xa \wedge (V\ a\ xa \longrightarrow V\ b\ xa))$

Example (all embeddings):

~~$\vdash \neg \Box a \supset \neg(\Box \neg(a \supset b))$~~

$\nvdash \neg \Box \varphi \supset \neg(\Box \neg(\varphi \supset \psi))$

shallow minimal ($.^m$)

$\models^m \neg^m (\Box^m \varphi) \supset^m \neg^m (\Box^m (\neg^m (\varphi \supset^m \psi)))$

$\forall w.\ (\exists v.\ R\ w\ v \wedge \neg\ \varphi\ v) \longrightarrow (\exists v.\ R\ w\ v \wedge (\varphi\ v \longrightarrow \psi\ v))$

shallow maximal ($.^s$)

$\models^s \neg^s (\Box^s \varphi) \supset^s \neg^s (\Box^s (\neg^s (\varphi \supset^s \psi)))$

$\forall W\ R\ V\ x.$
$\quad W\ x \longrightarrow$
$\quad (\exists xa.\ W\ xa \wedge R\ x\ xa \wedge \neg\ \varphi\ W\ R\ V\ xa) \longrightarrow$
$\quad (\exists xa.\ W\ xa \wedge R\ x\ xa \wedge (\varphi\ W\ R\ V\ xa \longrightarrow \psi\ W\ R\ V\ xa))$

deep ($.^d$)

$\models^d \neg^d (\Box^d \varphi) \supset^d \neg^d (\Box^d (\neg^d (\varphi \supset^d \psi)))$

```
∀W R V x.
  W x ⟶
  (∃xa.
     W xa ∧
     R x xa ∧
     ¬ rec_PML (λa W R V. V a) (λφ φ' W R V w. ¬ φ' W R V w)
       (λφ ψ φ' ψ' W R V w. φ' W R V w ⟶ ψ' W R V w)
       (λφ φ' W R V w. ∀x. W x ⟶ R w x ⟶ φ' W R V x) φ W R V xa) ⟶
  (∃xa.
     W xa ∧
     R x xa ∧
     (rec_PML (λa W R V. V a) (λφ φ' W R V w. ¬ φ' W R V w)
       (λφ ψ φ' ψ' W R V w. φ' W R V w ⟶ ψ' W R V w)
       (λφ φ' W R V w. ∀x. W x ⟶ R w x ⟶ φ' W R V x) φ W R V xa ⟶
      rec_PML (λa W R V. V a) (λφ φ' W R V w. ¬ φ' W R V w)
       (λφ ψ φ' ψ' W R V w. φ' W R V w ⟶ ψ' W R V w)
       (λφ φ' W R V w. ∀x. W x ⟶ R w x ⟶ φ' W R V x) ψ W R V xa))
```

Schematic formulas φ
lead to more complex
proof problems

Interlinked Embeddings

shared preliminaries

```
—‹Type declarations common for both the deep and shallow embedding›
typedecl w —‹Type for possible worlds›
typedecl 𝒮 —‹Type for propositional constant symbols›
consts p::𝒮 q::𝒮 —‹Some propositional constant symbols›
type_synonym 𝒲 = "w⇒bool" —‹Type for sets of possible worlds›
type_synonym ℛ = "w⇒w⇒bool" —‹Type for accessibility relations›
type_synonym 𝒱 = "𝒮⇒w⇒bool" —‹Type for valuation functions›
```

shallow maximal ($.^s$)

```
theory PMLinHOL_shallow imports PMLinHOL_preliminaries (* Christoph Benzmüller, 2025 *)
begin
—‹Shallow embedding (of propositional modal logic in HOL)›
type_synonym σ = "𝒲⇒ℛ⇒𝒱⇒w⇒bool"
definition AtmS::"𝒮⇒σ"     ("_ˢ")          where "aˢ ≡ λW R V w. V a w"
definition NegS::"σ⇒σ"     ("¬ˢ")          where "¬ˢ φ ≡ λW R V w. ¬(φ W R V w)"
definition ImpS::"σ⇒σ⇒σ"   (infixr "⊃ˢ" 93) where "φ ⊃ˢ ψ ≡ λW R V w. (φ W R V w) ⟶ (ψ W R V w)"
definition BoxS::"σ⇒σ"     ("□ˢ")          where "□ˢ φ ≡ λW R V w. ∀v:W. R w v ⟶ (φ W R V v)"
—‹Further logical connectives as definitions›
definition OrS    (infixr "∨ˢ" 92) where "φ ∨ˢ ψ ≡ ¬ˢφ ⊃ˢ ψ"
definition AndS   (infixr "∧ˢ" 95) where "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
definition DiaS   ("◇ˢ")           where "◇ˢφ ≡ ¬ˢ (□ˢ (¬ˢφ)) "
definition TopS   ("⊤ˢ")           where "⊤ˢ ≡ pˢ ⊃ˢ pˢ"
definition BotS   ("⊥ˢ")           where "⊥ˢ ≡ ¬ˢ⊤ˢ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
definition RelativeTruthS :: "𝒲⇒ℛ⇒𝒱⇒w⇒σ⇒bool" ("⟨_,_,_⟩,_ ⊨ˢ _") where "⟨W,R,V⟩,w ⊨ˢ φ ≡ φ W R V w"
—‹Definition of validity›
definition ValS ("⊨ˢ _") where "⊨ˢ φ ≡ ∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
named_theorems DefS declare AtmS_def[DefS,simp] NegS_def[DefS,simp] ImpS_def[DefS,simp]
  BoxS_def[DefS,simp] OrS_def[DefS,simp] AndS_def[DefS,simp] DiaS_def[DefS,simp] TopS_def[DefS,simp]
  BotS_def[DefS,simp] RelativeTruthS_def[DefS,simp] ValS_def[DefS,simp]
end
```

deep ($.^d$)

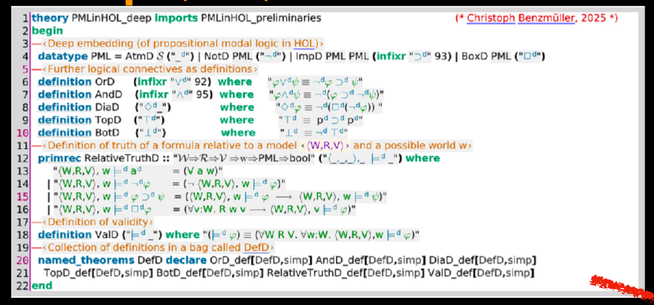
```
theory PMLinHOL_deep imports PMLinHOL_preliminaries (* Christoph Benzmüller, 2025 *)
begin
—‹Deep embedding (of propositional modal logic in HOL)›
datatype PML = AtmD 𝒮 ("_ᵈ") | NotD PML ("¬ᵈ") | ImpD PML PML (infixr "⊃ᵈ" 93) | BoxD PML ("□ᵈ")
—‹Further logical connectives as definitions›
definition OrD    (infixr "∨ᵈ" 92) where "φ∨ᵈψ ≡ ¬ᵈφ ⊃ᵈ ψ"
definition AndD   (infixr "∧ᵈ" 95) where "φ∧ᵈψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
definition DiaD   ("◇ᵈ_")          where "◇ᵈφ ≡ ¬ᵈ(□ᵈ(¬ᵈφ)) "
definition TopD   ("⊤ᵈ")           where "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
definition BotD   ("⊥ᵈ")           where "⊥ᵈ ≡ ¬ᵈ⊤ᵈ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
primrec RelativeTruthD :: "𝒲⇒ℛ⇒𝒱⇒w⇒PML⇒bool" ("⟨_,_,_⟩,_ ⊨ᵈ _") where
  "⟨W,R,V⟩, w ⊨ᵈ aᵈ     = (V a w)"
| "⟨W,R,V⟩, w ⊨ᵈ ¬ᵈφ    = (¬ ⟨W,R,V⟩, w ⊨ᵈ φ)"
| "⟨W,R,V⟩, w ⊨ᵈ φ ⊃ᵈ ψ  = (⟨W,R,V⟩, w ⊨ᵈ φ ⟶ ⟨W,R,V⟩, w ⊨ᵈ ψ)"
| "⟨W,R,V⟩, w ⊨ᵈ □ᵈφ    = (∀v:W. R w v ⟶ ⟨W,R,V⟩, v ⊨ᵈ φ)"
—‹Definition of validity›
definition ValD ("⊨ᵈ _") where "⊨ᵈ φ ≡ (∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ)"
—‹Collection of definitions in a bag called DefD›
named_theorems DefD declare OrD_def[DefD,simp] AndD_def[DefD,simp] DiaD_def[DefD,simp]
  TopD_def[DefD,simp] BotD_def[DefD,simp] RelativeTruthD_def[DefD,simp] ValD_def[DefD,simp]
end
```

shallow minimal ($.^m$)

```
theory PMLinHOL_shallow_minimal imports PMLinHOL_preliminaries (* Christoph Benzmüller, 2025 *)
begin
—‹The acessibility relation R and the valuation function V are introduced as constants at the meta-level HOL›
consts R::ℛ V::𝒱
—‹Shallow embedding (of propositional modal logic in HOL)›
type_synonym σ = "w⇒bool"
definition AtmM::"𝒮⇒σ"     ("_ᵐ")          where "aᵐ ≡ λw. V a w"
definition NegM::"σ⇒σ"     ("¬ᵐ")          where "¬ᵐ φ ≡ λw. ¬φ w"
definition ImpM::"σ⇒σ⇒σ"   (infixr "⊃ᵐ" 93) where "φ ⊃ᵐ ψ ≡ λw. φ w ⟶ ψ w"
definition BoxM::"σ⇒σ"     ("□ᵐ")          where "□ᵐ φ ≡ λw. ∀v. R w v ⟶ φ v"
—‹Further logical connectives as definitions›
definition OrM    (infixr "∨ᵐ" 92) where "φ ∨ᵐ ψ ≡ ¬ᵐφ ⊃ᵐ ψ"
definition AndM   (infixr "∧ᵐ" 95) where "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
definition DiaM   ("◇ᵐ_")          where "◇ᵐφ ≡ ¬ᵐ (□ᵐ (¬ᵐφ)) "
definition TopM   ("⊤ᵐ")           where "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
definition BotM   ("⊥ᵐ")           where "⊥ᵐ ≡ ¬ᵐ⊤ᵐ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
definition RelativeTruthM :: "w⇒σ⇒bool" ("_ ⊨ᵐ _") where "w ⊨ᵐ φ ≡ φ w"
—‹Definition of validity›
definition ValM ("⊨ᵐ _") where "⊨ᵐ φ ≡ ∀w::w. w ⊨ᵐ φ"
—‹Collection of definitions in a bag called DefM›
named_theorems DefM declare AtmM_def[DefM,simp] NegM_def[DefM,simp] ImpM_def[DefM,simp]
  BoxM_def[DefM,simp] OrM_def[DefM,simp] AndM_def[DefM,simp] DiaM_def[DefM,simp] TopM_def[DefM,simp]
  BotM_def[DefM,simp] RelativeTruthM_def[DefM,simp] ValM_def[DefM,simp]
end
```

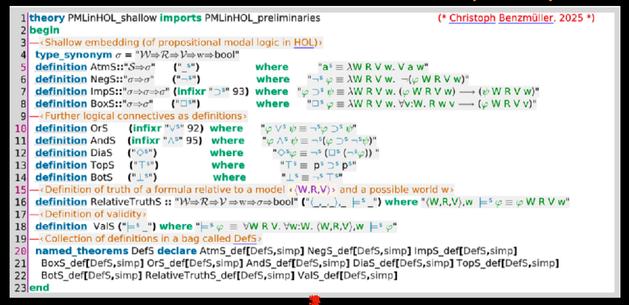
Faithfulness automated

deep ($.^d$)



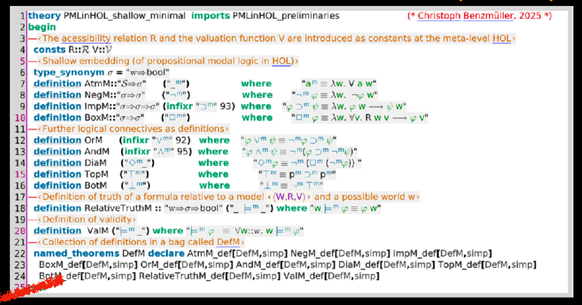
```
theory PMLinHOL_deep imports PMLinHOL_preliminaries                    (* Christoph Benzmüller, 2025 *)
begin
—‹Deep embedding (of propositional modal logic in HOL)›
datatype PML = AtmD 𝒮 ("_ᵈ") | NotD PML ("¬ᵈ") | ImpD PML PML (infixr "⊃ᵈ" 93) | BoxD PML ("□ᵈ")
—‹Further logical connectives as definitions›
definition OrD     (infixr "∨ᵈ" 92) where   "φ∨ᵈψ ≡ ¬ᵈφ⊃ᵈψ"
definition AndD    (infixr "∧ᵈ" 95) where   "φ∧ᵈψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
definition DiaD    ("◇ᵈ_")           where   "◇ᵈφ ≡ ¬ᵈ(□ᵈ(¬ᵈφ))"
definition TopD    ("⊤ᵈ")            where   "⊤ᵈ ≡ pᵈ⊃ᵈpᵈ"
definition BotD    ("⊥ᵈ")            where   "⊥ᵈ ≡ ¬ᵈ⊤ᵈ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
primrec RelativeTruthD :: "W⇒R⇒V ⇒w⇒PML⇒bool" ("⟨_,_,_⟩,_ ⊨ᵈ _") where
  "⟨W,R,V⟩, w ⊨ᵈ aᵈ      = (V a w)"
| "⟨W,R,V⟩, w ⊨ᵈ ¬ᵈφ     = (¬ ⟨W,R,V⟩, w ⊨ᵈ φ)"
| "⟨W,R,V⟩, w ⊨ᵈ φ ⊃ᵈ ψ  = ((⟨W,R,V⟩, w ⊨ᵈ φ ⟶ ⟨W,R,V⟩, w ⊨ᵈ ψ)"
| "⟨W,R,V⟩, w ⊨ᵈ □ᵈφ     = (∀v:W. R w v ⟶ ⟨W,R,V⟩, v ⊨ᵈ φ)"
—‹Definition of validity›
definition ValD ("⊨ᵈ _") where "(⊨ᵈ φ) ≡ (∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ)"
—‹Collection of definitions in a bag called DefD›
named_theorems DefD declare OrD_def[DefD,simp] AndD_def[DefD,simp] DiaD_def[DefD,simp]
  TopD_def[DefD,simp] BotD_def[DefD,simp] RelativeTruthD_def[DefD,simp] ValD_def[DefD,simp]
end
```

shallow maximal ($.^s$)



```
theory PMLinHOL_shallow imports PMLinHOL_preliminaries                 (* Christoph Benzmüller, 2025 *)
begin
—‹Shallow embedding (of propositional modal logic in HOL)›
type_synonym σ = "W⇒R⇒V⇒w⇒bool"
definition AtmS::"𝒮⇒σ"     ("_ˢ")      where   "aˢ ≡ λW R V w. V a w"
definition NegS::"σ⇒σ"      ("¬ˢ")      where   "¬ˢφ ≡ λW R V w. ¬(φ W R V w)"
definition ImpS::"σ⇒σ⇒σ"    (infixr "⊃ˢ" 93) where "φ ⊃ˢ ψ ≡ λW R V w. (φ W R V w) ⟶ (ψ W R V w)"
definition BoxS::"σ⇒σ"      ("□ˢ")      where   "□ˢφ ≡ λW R V w. ∀v:W. R w v ⟶ (φ W R V v)"
—‹Further logical connectives as definitions›
definition OrS     (infixr "∨ˢ" 92) where   "φ ∨ˢ ψ ≡ ¬ˢφ ⊃ˢ ψ"
definition AndS    (infixr "∧ˢ" 95) where   "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
definition DiaS    ("◇ˢ_")          where   "◇ˢφ ≡ ¬ˢ(□ˢ(¬ˢφ))"
definition TopS    ("⊤ˢ")           where   "⊤ˢ ≡ pˢ⊃ˢpˢ"
definition BotS    ("⊥ˢ")           where   "⊥ˢ ≡ ¬ˢ⊤ˢ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
definition RelativeTruthS :: "W⇒R⇒V ⇒w⇒σ⇒bool" ("⟨_,_,_⟩,_ ⊨ˢ _") where "⟨W,R,V⟩,w ⊨ˢ φ ≡ φ W R V w"
—‹Definition of validity›
definition ValS ("⊨ˢ _") where "⊨ˢ φ ≡ ∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
named_theorems DefS declare AtmS_def[DefS,simp] NegS_def[DefS,simp] ImpS_def[DefS,simp]
  BoxS_def[DefS,simp] OrS_def[DefS,simp] AndS_def[DefS,simp] DiaS_def[DefS,simp] TopS_def[DefS,simp]
  BotS_def[DefS,simp] RelativeTruthS_def[DefS,simp] ValS_def[DefS,simp]
end
```

shallow minimal ($.^m$)



```
theory PMLinHOL_shallow_minimal imports PMLinHOL_preliminaries         (* Christoph Benzmüller, 2025 *)
begin
—‹The accessibility relation R and the valuation function V are introduced as constants at the meta-level HOL›
consts R::𝒲 and V::𝒱
—‹Shallow embedding (of propositional modal logic in HOL)›
type_synonym σ = "w⇒bool"
definition AtmM::"𝒮⇒σ"     ("_ᵐ")      where   "aᵐ ≡ λw. V a w"
definition NegM::"σ⇒σ"      ("¬ᵐ")      where   "¬ᵐφ ≡ λw. ¬w φ"
definition ImpM::"σ⇒σ⇒σ"    (infixr "⊃ᵐ" 93) where "φ ⊃ᵐ ψ ≡ λw. φ w ⟶ ψ w"
definition BoxM::"σ⇒σ"      ("□ᵐ")      where   "□ᵐφ ≡ λw. ∀v. R w v ⟶ φ v"
—‹Further logical connectives as definitions›
definition OrM     (infixr "∨ᵐ" 92) where   "φ ∨ᵐ ψ ≡ ¬ᵐφ ⊃ᵐ ψ"
definition AndM    (infixr "∧ᵐ" 95) where   "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
definition DiaM    ("◇ᵐ_")          where   "◇ᵐφ ≡ ¬ᵐ(□ᵐ(¬ᵐφ))"
definition TopM    ("⊤ᵐ")           where   "⊤ᵐ ≡ pᵐ⊃ᵐpᵐ"
definition BotM    ("⊥ᵐ")           where   "⊥ᵐ ≡ ¬ᵐ⊤ᵐ"
—‹Definition of truth of a formula relative to a model ‹⟨W,R,V⟩› and a possible world w›
definition RelativeTruthM :: "w⇒σ⇒bool" ("_ ⊨ᵐ _") where "w ⊨ᵐ φ ≡ φ w"
—‹Definition of validity›
definition ValM ("⊨ᵐ _") where "⊨ᵐ φ ≡ ∀w::w. w ⊨ᵐ φ"
—‹Collection of definitions in a bag called DefM›
named_theorems DefM declare AtmM_def[DefM,simp] NegM_def[DefM,simp] ImpM_def[DefM,simp]
  BoxM_def[DefM,simp] OrM_def[DefM,simp] AndM_def[DefM,simp] DiaM_def[DefM,simp] TopM_def[DefM,simp]
  BotM_def[DefM,simp] RelativeTruthM_def[DefM,simp] ValM_def[DefM,simp]
end
```

```
theory PMLinHOL_faithfulness  imports PMLinHOL_deep PMLinHOL_shallow PMLinHOL_shallow_minimal  (* C.B., 2025 *)
begin
—‹Mappings: deep to maximal shallow and deep to minimal shallow›
  primrec DpToShMax ("(_)") where "(aᵈ)  = aˢ" | "(¬ᵈ φ) = ¬ˢ (φ)" | "(φ ⊃ᵈ ψ) = (φ) ⊃ˢ (ψ)" | "(□ᵈ φ) = □ˢ (φ)"
  primrec DpToShMin  ("[_]") where "[aᵈ]  = aᵐ" | "[¬ᵈ φ] = ¬ᵐ [φ]" | "[φ ⊃ᵈ ψ] = [φ] ⊃ᵐ [ψ]" | "[□ᵈ φ] = □ᵐ [φ]"
—‹Proving faithfulness between deep and maximal shallow›
  theorem Faithful1a:   "∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ ⟷ ⟨W,R,V⟩,w ⊨ˢ (φ)"     apply induct by auto
  theorem Faithful1b:   "⊨ᵈ φ ⟷ ⊨ˢ (φ)"                                    using Faithful1a by auto
—‹Proving faithfulness between deep and minimal shallow›
  theorem Faithful2:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ᵈ φ ⟷ w ⊨ᵐ [φ]"          apply induct by auto
—‹Proving faithfulness maximal shallow and minimal shallow›
  theorem Faithful3:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ˢ (φ) ⟷ w ⊨ᵐ [φ]"       apply induct by auto
—‹Additional check for soundness for the minimal shallow embedding›
  lemma Sound1: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵈ φ)" by (smt Faithful2 DefM DefD RelativeTruthD.simps ext[of ψ "[x⊃ᵈx]"])
  lemma Sound2: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵐ [φ])" using Sound1 by blast
end
```

Faithfulness automated

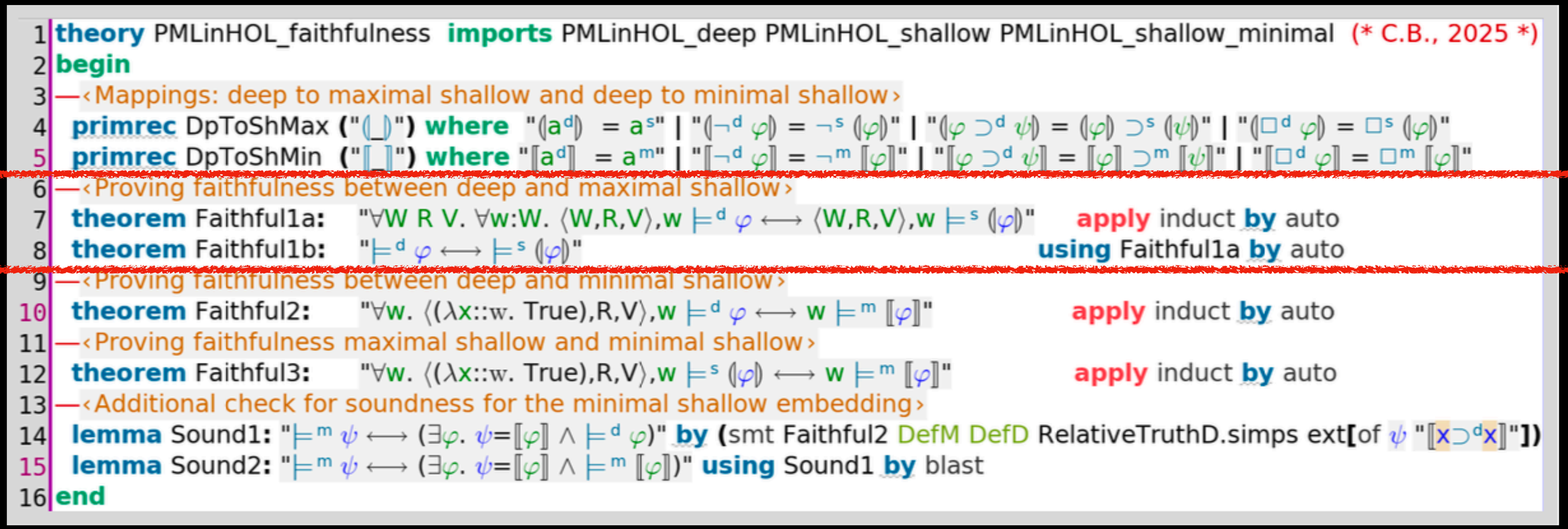
⦇.⦈ - Mapping of deep formulas to maximal shallow formulas

⟦.⟧ - Mapping of deep formulas to minimal shallow formulas

```
theory PMLinHOL_faithfulness  imports PMLinHOL_deep PMLinHOL_shallow PMLinHOL_shallow_minimal  (* C.B., 2025 *)
begin
—‹Mappings: deep to maximal shallow and deep to minimal shallow›
  primrec DpToShMax ("⦇_⦈") where  "⦇aᵈ⦈  = aˢ" | "⦇¬ᵈ φ⦈ = ¬ˢ ⦇φ⦈" | "⦇φ ⊃ᵈ ψ⦈ = ⦇φ⦈ ⊃ˢ ⦇ψ⦈" | "⦇□ᵈ φ⦈ = □ˢ ⦇φ⦈"
  primrec DpToShMin  ("⟦_⟧") where "⟦aᵈ⟧  = aᵐ" | "⟦¬ᵈ φ⟧ = ¬ᵐ ⟦φ⟧" | "⟦φ ⊃ᵈ ψ⟧ = ⟦φ⟧ ⊃ᵐ ⟦ψ⟧" | "⟦□ᵈ φ⟧ = □ᵐ ⟦φ⟧"
—‹Proving faithfulness between deep and maximal shallow›
  theorem Faithful1a:   "∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ ⟷ ⟨W,R,V⟩,w ⊨ˢ ⦇φ⦈"     apply induct by auto
  theorem Faithful1b:   "⊨ᵈ φ ⟷ ⊨ˢ ⦇φ⦈"                                  using Faithful1a by auto
—‹Proving faithfulness between deep and minimal shallow›
  theorem Faithful2:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ᵈ φ ⟷ w ⊨ᵐ ⟦φ⟧"          apply induct by auto
—‹Proving faithfulness maximal shallow and minimal shallow›
  theorem Faithful3:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ˢ ⦇φ⦈ ⟷ w ⊨ᵐ ⟦φ⟧"          apply induct by auto
—‹Additional check for soundness for the minimal shallow embedding›
  lemma Sound1: "⊨ᵐ ψ ⟷ (∃φ. ψ=⟦φ⟧ ∧ ⊨ᵈ φ)" by (smt Faithful2 DefM DefD RelativeTruthD.simps ext[of ψ "⟦x⊃ᵈx⟧"])
  lemma Sound2: "⊨ᵐ ψ ⟷ (∃φ. ψ=⟦φ⟧ ∧ ⊨ᵐ ⟦φ⟧)" using Sound1 by blast
end
```

Faithfulness automated

A deep formula is valid if and only if its maximal shallow translation is valid.



```
theory PMLinHOL_faithfulness  imports PMLinHOL_deep PMLinHOL_shallow PMLinHOL_shallow_minimal  (* C.B., 2025 *)
begin
—‹Mappings: deep to maximal shallow and deep to minimal shallow›
  primrec DpToShMax ("(_)") where  "(aᵈ)  = aˢ" | "(¬ᵈ φ) = ¬ˢ (φ)" | "(φ ⊃ᵈ ψ) = (φ) ⊃ˢ (ψ)" | "(□ᵈ φ) = □ˢ (φ)"
  primrec DpToShMin  ("[_]") where "[aᵈ]  = aᵐ" | "[¬ᵈ φ] = ¬ᵐ [φ]" | "[φ ⊃ᵈ ψ] = [φ] ⊃ᵐ [ψ]" | "[□ᵈ φ] = □ᵐ [φ]"
—‹Proving faithfulness between deep and maximal shallow›
  theorem Faithful1a:   "∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ ⟷ ⟨W,R,V⟩,w ⊨ˢ (φ)"     apply induct by auto
  theorem Faithful1b:   "⊨ᵈ φ ⟷ ⊨ˢ (φ)"                                    using Faithful1a by auto
—‹Proving faithfulness between deep and minimal shallow›
  theorem Faithful2:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ᵈ φ ⟷ w ⊨ᵐ [φ]"          apply induct by auto
—‹Proving faithfulness maximal shallow and minimal shallow›
  theorem Faithful3:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ˢ (φ) ⟷ w ⊨ᵐ [φ]"          apply induct by auto
—‹Additional check for soundness for the minimal shallow embedding›
  lemma Sound1: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵈ φ)" by (smt Faithful2 DefM DefD RelativeTruthD.simps ext[of ψ "[x⊃ᵈx]"])
  lemma Sound2: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵐ [φ])" using Sound1 by blast
end
```

Faithfulness automated

Similar relative faithfulness theorems for the
- deep and minimal shallow embedding, and for the
- maximal shallow and minimal shallow embedding

```
theory PMLinHOL_faithfulness  imports PMLinHOL_deep PMLinHOL_shallow PMLinHOL_shallow_minimal  (* C.B., 2025 *)
begin
—‹Mappings: deep to maximal shallow and deep to minimal shallow›
  primrec DpToShMax ("(_)") where  "(aᵈ)  = aˢ" | "(¬ᵈ φ) = ¬ˢ (φ)" | "(φ ⊃ᵈ ψ) = (φ) ⊃ˢ (ψ)" | "(□ᵈ φ) = □ˢ (φ)"
  primrec DpToShMin  ("[_]") where "[aᵈ]  = aᵐ" | "[¬ᵈ φ] = ¬ᵐ [φ]" | "[φ ⊃ᵈ ψ] = [φ] ⊃ᵐ [ψ]" | "[□ᵈ φ] = □ᵐ [φ]"
—‹Proving faithfulness between deep and maximal shallow›
  theorem Faithful1a:   "∀W R V. ∀w:W. ⟨W,R,V⟩,w ⊨ᵈ φ ⟷ ⟨W,R,V⟩,w ⊨ˢ (φ)"     apply induct by auto
  theorem Faithful1b:   "⊨ᵈ φ ⟷ ⊨ˢ (φ)"                                    using Faithful1a by auto
—‹Proving faithfulness between deep and minimal shallow›
  theorem Faithful2:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ᵈ φ ⟷ w ⊨ᵐ [φ]"          apply induct by auto
—‹Proving faithfulness maximal shallow and minimal shallow›
  theorem Faithful3:    "∀w. ⟨(λx::w. True),R,V⟩,w ⊨ˢ (φ) ⟷ w ⊨ᵐ [φ]"          apply induct by auto
—‹Additional check for soundness for the minimal shallow embedding›
  lemma Sound1: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵈ φ)" by (smt Faithful2 DefM DefD RelativeTruthD.simps ext[of ψ "[x⊃ᵈx]"])
  lemma Sound2: "⊨ᵐ ψ ⟷ (∃φ. ψ=[φ] ∧ ⊨ᵐ [φ])" using Sound1 by blast
end
```

Experiments

Conducted with all embeddings — but presented here for the minimal shallow embedding

Experiments

```
1 theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
2 begin
3 —‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
4   lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
5   lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
6   lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
7   lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
```

Hilbert axioms and modus ponens rule are implied

H1: $\models \varphi \supset (\psi \supset \varphi)$

H2: $\models (\varphi \supset (\psi \supset \gamma)) \supset ((\varphi \supset \psi) \supset (\varphi \supset \gamma))$

H3: $\models (\neg\varphi \supset \neg\psi) \supset (\psi \supset \varphi)$

MP: $\models \varphi \implies \models \varphi \supset \psi \implies \models \psi$

Experiments

```
theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
begin
—‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
  lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
  lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
  lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
  lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
—‹Reasoning with the Hilbert calculus: interactive and fully automated›
  lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
```

Simple automated proof of $\varphi \supset \varphi$ at level of this Hilbert calculus

Experiments

```
1  theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
2  begin
3  —‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
4    lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
5    lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
6    lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
7    lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
8  —‹Reasoning with the Hilbert calculus: interactive and fully automated›
9    lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
10     proof -  have 1: "⊨ᵐ φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)" using H1 by auto
11              have 2: "⊨ᵐ (φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)) ⊃ᵐ ((φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ))" using H2 by auto
12              have 3: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ)" using 1 2 MP by meson
13              have 4: "⊨ᵐ φ ⊃ᵐ (ψ  ⊃ᵐ φ)" using H1 by auto
14              thus ?thesis using 3 4 MP by meson qed
```

Interactive proof of $\varphi \supset \varphi$ in this Hilbert calculus

Experiments

```
theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
begin
—‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
 lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
 lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
—‹Reasoning with the Hilbert calculus: interactive and fully automated›
 lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
  proof -  have 1: "⊨ᵐ φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)" using H1 by auto
           have 2: "⊨ᵐ (φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)) ⊃ᵐ ((φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ))" using H2 by auto
           have 3: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ)" using 1 2 MP by meson
           have 4: "⊨ᵐ φ ⊃ᵐ (ψ  ⊃ᵐ φ)" using H1 by auto
           thus ?thesis using 3 4 MP by meson qed
 lemma HCderived2: "⊨ᵐ φ ⊃ᵐ (¬ᵐφ  ⊃ᵐ ψ) " by (metis H1 H2 H3 MP)
 lemma HCderived3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ φ) ⊃ᵐ φ" by (metis H1 H2 H3 MP)
 lemma HCderived4: "⊨ᵐ (φ ⊃ᵐ ψ) ⊃ᵐ (¬ᵐψ ⊃ᵐ ¬ᵐφ) " by auto
```

Further automated proofs of simple example problems

Experiments

```
theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
begin
—‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
 lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
 lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
—‹Reasoning with the Hilbert calculus: interactive and fully automated›
 lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
  proof -  have 1: "⊨ᵐ φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)" using H1 by auto
           have 2: "⊨ᵐ (φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)) ⊃ᵐ ((φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ))" using H2 by auto
           have 3: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ)" using 1 2 MP by meson
           have 4: "⊨ᵐ φ ⊃ᵐ (ψ  ⊃ᵐ φ)" using H1 by auto
           thus ?thesis using 3 4 MP by meson qed
 lemma HCderived2: "⊨ᵐ φ ⊃ᵐ (¬ᵐφ  ⊃ᵐ ψ) " by (metis H1 H2 H3 MP)
 lemma HCderived3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ φ) ⊃ᵐ φ" by (metis H1 H2 H3 MP)
 lemma HCderived4: "⊨ᵐ (φ ⊃ᵐ ψ) ⊃ᵐ (¬ᵐψ ⊃ᵐ ¬ᵐφ) " by auto
—‹Modal logic: the schematic necessitation rule and distribution axiom are implied›
 lemma Nec: "⊨ᵐ φ ⟹ ⊨ᵐ □ᵐφ" by (smt DefM)
 lemma Dist:  "⊨ᵐ □ᵐ(φ ⊃ᵐ ψ) ⊃ᵐ (□ᵐφ ⊃ᵐ □ᵐψ) " by auto
```

Necessitation rule and distribution axiom are implied

Nec: $\models \varphi \implies \models \Box\varphi$

Dist: $\models \Box(\varphi \supset \psi) \supset (\Box\varphi \supset \Box\psi)$

Experiments

```
theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
begin
—‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
 lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
 lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
—‹Reasoning with the Hilbert calculus: interactive and fully automated›
 lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
  proof -  have 1: "⊨ᵐ φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)" using H1 by auto
           have 2: "⊨ᵐ (φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)) ⊃ᵐ ((φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ))" using H2 by auto
           have 3: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ)" using 1 2 MP by meson
           have 4: "⊨ᵐ φ ⊃ᵐ (ψ  ⊃ᵐ φ)" using H1 by auto
           thus ?thesis using 3 4 MP by meson qed
 lemma HCderived2: "⊨ᵐ φ ⊃ᵐ (¬ᵐφ  ⊃ᵐ ψ) " by (metis H1 H2 H3 MP)
 lemma HCderived3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ φ) ⊃ᵐ φ" by (metis H1 H2 H3 MP)
 lemma HCderived4: "⊨ᵐ (φ ⊃ᵐ ψ) ⊃ᵐ (¬ᵐψ ⊃ᵐ ¬ᵐφ) " by auto
—‹Modal logic: the schematic necessitation rule and distribution axiom are implied›
 lemma Nec: "⊨ᵐ φ ⟹ ⊨ᵐ □ᵐφ" by (smt DefM)
 lemma Dist:  "⊨ᵐ □ᵐ(φ ⊃ᵐ ψ) ⊃ᵐ (□ᵐφ ⊃ᵐ □ᵐψ) " by auto
—‹Correspondence theory: correct statements›
 lemma cM:  "reflexive R ⟷ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ φ)" by auto
 lemma cBa: "symmetric R ⟶ (∀φ. ⊨ᵐ φ ⊃ᵐ □ᵐ◇ᵐφ)" by auto
 lemma cBb: "symmetric R ⟵ (∀φ. ⊨ᵐ φ ⊃ᵐ □ᵐ◇ᵐφ)" by (metis DefM)
 lemma c4a: "transitive R ⟶ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" by (smt DefM)
 lemma c4b: "transitive R ⟵ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" by auto
```

Sahlqvist correspondences automated

Experiments

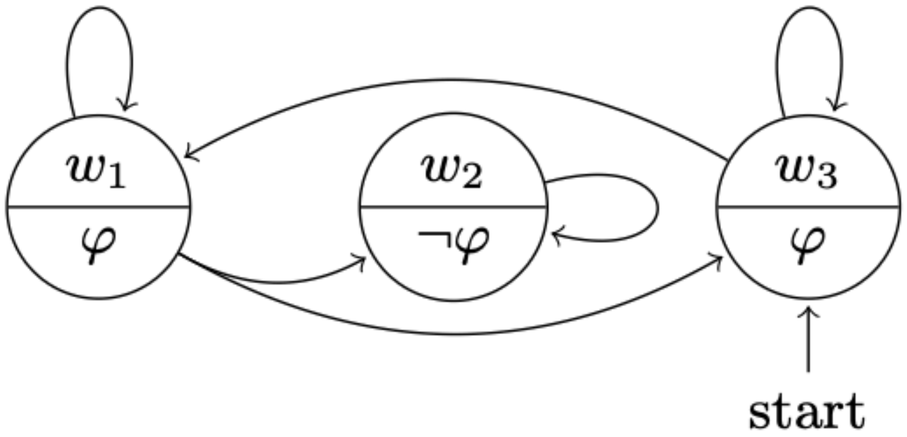
```
theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal        (* Christoph Benzmüller, 2025 *)
begin
—‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
 lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
 lemma H3: "⊨ᵐ (¬ᵐφ ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
 lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
—‹Reasoning with the Hilbert calculus: interactive and fully automated›
 lemma HCderived1: "⊨ᵐ (φ ⊃ᵐ φ)" —‹sledgehammer(H1 H2 H3 MP) returns: by (metis H1 H2 MP)›
  proof - have 1: "⊨ᵐ φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)" using H1 by auto
         have 2: "⊨ᵐ (φ ⊃ᵐ ((ψ ⊃ᵐ φ) ⊃ᵐ φ)) ⊃ᵐ ((φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ))" using H2 by auto
         have 3: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ φ)) ⊃ᵐ (φ ⊃ᵐ φ)" using 1 2 MP by meson
         have 4: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" using H1 by auto
         thus ?thesis using 3 4 MP by meson qed
 lemma HCderived2: "⊨ᵐ φ ⊃ᵐ (¬ᵐφ ⊃ᵐ ψ) " by (metis H1 H2 H3 MP)
 lemma HCderived3: "⊨ᵐ (¬ᵐφ ⊃ᵐ φ) ⊃ᵐ φ" by (metis H1 H2 H3 MP)
 lemma HCderived4: "⊨ᵐ (φ ⊃ᵐ ψ) ⊃ᵐ (¬ᵐψ ⊃ᵐ ¬ᵐφ) " by auto
—‹Modal logic: the schematic necessitation rule and distribution axiom are implied›
 lemma Nec: "⊨ᵐ φ ⟹ ⊨ᵐ □ᵐφ" by (smt DefM)
 lemma Dist: "⊨ᵐ □ᵐ(φ ⊃ᵐ ψ) ⊃ᵐ (□ᵐφ ⊃ᵐ □ᵐψ) " by auto
—‹Correspondence theory: correct statements›
 lemma cM: "reflexive R ⟷ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ φ)" by auto
 lemma cBa: "symmetric R ⟶ (∀φ. ⊨ᵐ φ ⊃ᵐ □ᵐ◇ᵐφ)" by auto
 lemma cBb: "symmetric R ⟵ (∀φ. ⊨ᵐ φ ⊃ᵐ □ᵐ◇ᵐφ)" by (metis DefM)
 lemma c4a: "transitive R ⟶ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" by (smt DefM)
 lemma c4b: "transitive R ⟵ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" by auto
—‹Correspondence theory: incorrect statements›
 lemma "reflexive R ⟶ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" nitpick[card w=3,show_all] oops —‹nitpick: Counterexample›
```

Incorrect correspondence claim

$$\text{reflexive}\, R \longrightarrow (\forall \varphi\,.\, \vDash \Box\varphi \supset \Box\Box\varphi)$$

Experiments

```
1 theory PMLinHOL_shallow_minimal_tests imports PMLinHOL_shallow_minimal   (* Christoph Benzmüller, 2025 *)
2 begin
3 —‹Hilbert calculus: proving that the schematic axioms and rules are implied by the embedding›
4   lemma H1: "⊨ᵐ φ ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
5   lemma H2: "⊨ᵐ (φ ⊃ᵐ (ψ ⊃ᵐ γ)) ⊃ᵐ ((φ ⊃ᵐ ψ) ⊃ᵐ (φ ⊃ᵐ γ)) " by auto
6   lemma H3: "⊨ᵐ (¬ᵐφ  ⊃ᵐ ¬ᵐψ) ⊃ᵐ (ψ ⊃ᵐ φ)" by auto
7   lemma MP: "⊨ᵐ φ ⟹ ⊨ᵐ (φ  ⊃ᵐ ψ) ⟹ ⊨ᵐ ψ" by auto
```




```
Nitpick found a counterexample for card w = 3:
  Skolem constants:
    ??.BoxM.v = w₁
    ??.BoxM.v = w₂
    ??.ValM.w = w₃
    φ = (λx. _)(w₁ := True, w₂ := False, w₃ := True)
  Constant:
    R =
      (λx. _)
      (w₁ := (λx. _)(w₁ := True, w₂ := True, w₃ := True),
        w₂ := (λx. _)(w₁ := False, w₂ := True, w₃ := False),
        w₃ := (λx. _)(w₁ := True, w₂ := False, w₃ := True))
```

Fig. 5. Counterexample reported by `nitpick` in line 28 of Fig. 4

[See paper at LPLR 2025]

```
27 —‹Correspondence theory: incorrect statements›
28 lemma "reflexive R ⟶ (∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ(□ᵐφ))" nitpick[card w=3,show_all] oops —‹nitpick: Counterexample›
```

Incorrect correspondence claim — counterexample by Nitpick

reflexive $R \longrightarrow (\forall \varphi \,.\, \vDash \Box\varphi \supset \Box\Box\varphi)$

Experiments

modal reasoning

```
theory PMLinHOL_shallow_minimal_further_tests imports PMLinHOL_shallow_minimal_tests        (* C.B., 2025 *)
begin
—‹Implied modal principle›
 lemma K_Dia: "⊨ᵐ (□ᵐ(φ ⊃ᵐ ψ)) ⊃ᵐ ((◇ᵐφ) ⊃ᵐ ◇ᵐψ)"    by auto
—‹Example 6.10 of Sider (2009) Logic for Philosophy›
 lemma T1a:  "⊨ᵐ □ᵐpᵐ ⊃ᵐ ((◇ᵐqᵐ) ⊃ᵐ ◇ᵐ (pᵐ ∧ᵐ qᵐ))" by auto  —‹fast automation in meta-logic HOL›
 lemma T1b: "⊨ᵐ □ᵐpᵐ ⊃ᵐ ((◇ᵐqᵐ) ⊃ᵐ ◇ᵐ (pᵐ ∧ᵐ qᵐ))"
  proof -                                          —‹alternative interactive proof in modal object logic K›
    have 1: "⊨ᵐ pᵐ ⊃ᵐ (qᵐ ⊃ᵐ (pᵐ ∧ᵐ qᵐ))"          unfolding AndM_def using H1 H2 H3 MP by metis
    have 2: "⊨ᵐ □ᵐ(pᵐ ⊃ᵐ (qᵐ ⊃ᵐ (pᵐ ∧ᵐ qᵐ)))"                          using 1 Nec          by metis
    have 3: "⊨ᵐ □ᵐpᵐ ⊃ᵐ □ᵐ(qᵐ ⊃ᵐ (pᵐ ∧ᵐ qᵐ))"                           using 2 Dist MP      by metis
    have 4: "⊨ᵐ (□ᵐ(qᵐ ⊃ᵐ (pᵐ ∧ᵐ qᵐ))) ⊃ᵐ ((◇ᵐqᵐ) ⊃ᵐ ◇ᵐ(pᵐ ∧ᵐ qᵐ))" using K_Dia          by metis
    have 5: "⊨ᵐ □ᵐpᵐ ⊃ᵐ ((◇ᵐqᵐ) ⊃ᵐ ◇ᵐ (pᵐ ∧ᵐ qᵐ))"                    using 3 4 H1 H2 MP by metis
    thus ?thesis . qed
end
```

inspecting, and reasoning with, the Löb formula

```
theory PMLinHOL_shallow_minimal_Loeb_tests  imports PMLinHOL_shallow_minimal
begin
 —‹Löb axiom: with the minimal shallow embedding automated reasoning tools are still partly responsive›
 lemma Loeb1M:  "∀φ. ⊨ᵐ □ᵐ( □ᵐφ ⊃ᵐφ) ⊃ᵐ □ᵐφ" nitpick[card w=1,card 𝒮=1] oops —‹nitpick: Counterexample.›
 lemma Loeb2M:  "(conversewf R ∧ transitive R) ⟶ (∀φ. ⊨ᵐ □ᵐ(□ᵐφ ⊃ᵐφ) ⊃ᵐ □ᵐφ)"
  by (smt (verit, del_insts) BoxM_def ImpM_def RelativeTruthM_def ValM_def)
 lemma Loeb3aM: "conversewf R ⟵ (∀φ. ⊨ᵐ □ᵐ(□ᵐφ ⊃ᵐφ) ⊃ᵐ □ᵐφ)" unfolding DefM by blast
 lemma Loeb3bM: "transitive R ⟵ (∀φ. ⊨ᵐ □ᵐ(□ᵐφ ⊃ᵐφ) ⊃ᵐ □ᵐφ)"  —‹sledgehammer: no proof›
  proof assume 1: "∀φ. ⊨ᵐ □ᵐ (□ᵐ φ ⊃ᵐ φ) ⊃ᵐ □ᵐ φ"
       have 2: "∀φ. ⊨ᵐ □ᵐφ ⊃ᵐ □ᵐ((□ᵐ(□ᵐφ ∧ᵐ φ)) ⊃ᵐ ((□ᵐφ) ∧ᵐ φ))" by auto
       show "transitive R"  by (smt (z3) 1 2 DefM(10,11,2,3,4,6)) qed
 lemma Loeb3M: "(conversewf R ∧ transitive R) ⟵ (∀φ. ⊨ᵐ □ᵐ(□ᵐφ ⊃ᵐφ) ⊃ᵐ □ᵐφ)"
  using Loeb3aM Loeb3bM by force
end
```

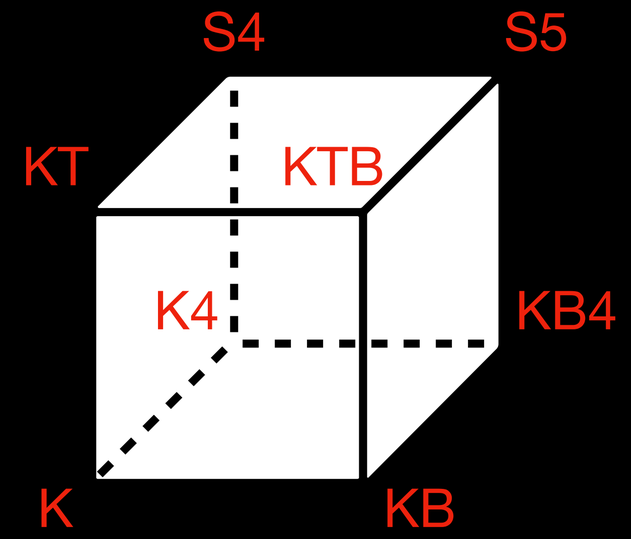
PML Evaluation

(new — extending results from CADE 2025)

Evaluation

F_1: $\Diamond\Diamond\varphi \supset \Diamond\varphi$
F_2: $\Diamond\Box\varphi \supset \Box\Diamond\varphi$
F_3: $\Diamond\Box\varphi \supset \Box\varphi$
F_4: $\Box\Diamond\Box\Diamond\varphi \supset \Box\Diamond\varphi$
F_5: $\Diamond(\varphi \wedge \Diamond\psi)) \supset (\Diamond\varphi \wedge \Diamond\psi)$
F_6: $((\Box(\varphi \supset \psi)) \wedge \Diamond\Box\neg\psi) \supset \neg\Diamond\psi$
F_7: $\Diamond\varphi \supset \Box(\varphi \vee \Diamond\varphi)$
F_8: $\Diamond(\Box\varphi) \supset (\varphi \vee \Diamond\varphi)$
F_9: $(\Box\Diamond\varphi \wedge \Box\Diamond\neg\varphi) \supset \Diamond\Diamond\varphi$
F_{10}: $(\Box(\varphi \supset \Box\psi) \wedge \Box\Diamond\neg\psi) \supset \neg\Box\psi$

(schematic)
problem formulas



AxT: $\forall\varphi\,.\,\models \Box\varphi \supset \varphi$
AxB: $\forall\varphi\,.\,\models \varphi \supset \Box\Diamond\varphi$
$Ax4$: $\forall\varphi\,.\,\models \Box\varphi \supset \Box\Box\varphi$

$SemT$: reflexive(R)
$SemB$: symmetric(R)
$Sem4$: transitive(R)

S5:	$AxT \wedge AxB \wedge Ax4$	$\models F_n$
KTB:	$AxT \wedge AxB$	$\models F_n$
S4:	$AxT \wedge Ax4$	$\models F_n$
KB4:	$AxB \wedge Ax4$	$\models F_n$
KT:	AxT	$\models F_n$
KB:	AxB	$\models F_n$
K4:	$Ax4$	$\models F_n$
K:		$\models F_n$

80 problems
with/without “apply simp”
160 calls to sledgehammer and nitpick

$M \models F$

S5:	$SemT \wedge SemB \wedge Sem4$	$\models F_n$
KTB:	$SemT \wedge SemB$	$\models F_n$
S4:	$SemT \wedge Sem4$	$\models F_n$
KB4:	$SemB \wedge Sem4$	$\models F_n$
KT:	$SemT$	$\models F_n$
KB:	$SemB$	$\models F_n$
K4:	$Sem4$	$\models F_n$
K:		$\models F_n$

80 problems
with/without “apply simp”
160 calls to sledgehammer and nitpick

$N \models F$

	nitpick			sledgehammer	
	A	B	C	D	E
Problems	counterex.	no counterex.	unknown	proof found	no proof
$M \models^m F$ $N \models^m F$	minimal shallow			minimal shallow	
$M \models^s F$ $N \models^s F$	maximal shallow			maximal shallow	
$M \models^d F$ $N \models^d F$	deep			deep	

Sledgehammer and Nitpick were run in default settings in Isabelle 2025

Evaluation

A MacBook Pro 16,1 (6-Core Intel Core i7 processor, 2,6 GHz, 6 Cores, 256 KB L2 Cache per Core, 12 MB L3 Cache, 16 GB Memory) and Isabelle 2025-1-RC1 in standard configuration was used.

		sledgehammer[14] (38 THM)									nitpick (42 CSA)				
		1s	2s	5s	10s	20s	40s	80s	∑	%	2s	10s	40s	∑	%
$M \models^m F$		9	9	12	19	24	34	37	144	45%	42	42	42	126	100%
$M \models^m F$	simp	18	36	37	**38**	**38**	**38**	**38**	243	91%	42	42	42	126	100%
$N \models^m F$		**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$N \models^m F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^s F$		0	0	0	23	23	24	29	99	37%	16	16	16	48	38%
$M \models^s F$	simp	7	26	26	28	28	28	28	181	68%	16	16	16	48	38%
$N \models^s F$		7	7	8	37	37	37	37	170	64%	42	42	42	126	100%
$N \models^s F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^d F$		7	12	21	24	26	30	30	150	56%	0	6	6	12	10%
$M \models^d F$	simp	0	7	13	26	28	28	28	130	49%	2	8	8	18	14%
$N \models^d F$		25	**38**	**38**	**38**	**38**	**38**	**38**	253	95%	6	24	24	54	43%
$N \models^d F$	simp	30	32	**38**	**38**	**38**	**38**	**38**	252	95%	2	8	8	18	14%

Fig. 10. Performance results of the automated PML formula classification task conducted with sledgehammer and nitpick; for the problem set $M \models^m F$ a perfect classification is achieved (with or without the use of simp) already for a very short timeout of 2s (and also 1s), meaning that for each formula F in this mode either a proof is reported, if F is a theorem (THM), or a counterexample is presented, if F is a countersatisfiable (CSA). There were no conflicting results reported by sledgehammer and nitpick.

minimal shallow with modal axioms

minimal shallow with semantic axioms

maximal shallow with modal axioms

maximal shallow with semantic axioms

deep with modal axioms

deep with semantic axioms

Results

- Winner(s) — perfect classification — “no counterex.”/“proof” for all valid statement and “counterex.” else

Evaluation

A MacBook Pro 16,1 (6-Core Intel Core i7 processor, 2,6 GHz, 6 Cores, 256 KB L2 Cache per Core, 12 MB L3 Cache, 16 GB Memory) and Isabelle 2025-1-RC1 in standard configuration was used.

		sledgehammer[14] (38 THM)									nitpick (42 CSA)				
		1s	2s	5s	10s	20s	40s	80s	∑	%	2s	10s	40s	∑	%
$M \models^m F$		9	9	12	19	24	34	37	144	45%	42	42	42	126	100%
$M \models^m F$	simp	18	36	37	**38**	**38**	**38**	**38**	243	91%	42	42	42	126	100%
$N \models^m F$		**38**	**38**	**38**	**38**	**38**	**38**	**38**	266	100%	42	42	42	126	100%
$N \models^m F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	266	100%	42	42	42	126	100%
$M \models^s F$		0	0	0	23	23	24	29	99	37%	16	16	16	48	38%
$M \models^s F$	simp	7	26	26	28	28	28	28	181	68%	16	16	16	48	38%
$N \models^s F$		7	7	8	37	37	37	37	170	64%	42	42	42	126	100%
$N \models^s F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	266	100%	42	42	42	126	100%
$M \models^d F$		7	12	21	24	26	30	30	150	56%	0	6	6	12	10%
$M \models^d F$	simp	0	7	13	26	28	28	28	130	49%	2	8	8	18	14%
$N \models^d F$		25	**38**	**38**	**38**	**38**	**38**	**38**	253	95%	6	24	24	54	43%
$N \models^d F$	simp	30	32	**38**	**38**	**38**	**38**	**38**	252	95%	2	8	8	18	14%

Fig. 10. Performance results of the automated PML formula classification task conducted with sledgehammer and nitpick; for the problem set $M \models^m F$ a perfect classification is achieved (with or without the use of simp) already for a very short timeout of 2s (and also 1s), meaning that for each formula F in this mode either a proof is reported, if F is a theorem (THM), or a counterexample is presented, if F is a countersatisfiable (CSA). There were no conflicting results reported by sledgehammer and nitpick.

minimal shallow with modal axioms

minimal shallow with semantic axioms

maximal shallow with modal axioms

maximal shallow with semantic axioms

deep with modal axioms

deep with semantic axioms

Results

- Winner — perfect classification — “no counterex.”/“proof” for all valid statement and “counterex.” else
- Second best

Evaluation

A MacBook Pro 16,1 (6-Core Intel Core i7 processor, 2,6 GHz, 6 Cores, 256 KB L2 Cache per Core, 12 MB L3 Cache, 16 GB Memory) and Isabelle 2025-1-RC1 in standard configuration was used.

		sledgehammer[14] (38 THM)									nitpick (42 CSA)				
		1s	2s	5s	10s	20s	40s	80s	∑	%	2s	10s	40s	∑	%
$M \models^m F$		9	9	12	19	24	34	37	144	45%	42	42	42	126	100%
$M \models^m F$	simp	18	36	37	**38**	**38**	**38**	**38**	243	91%	42	42	42	126	100%
$N \models^m F$		**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$N \models^m F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^s F$		0	0	0	23	23	24	29	99	37%	16	16	16	48	38%
$M \models^s F$	simp	7	26	26	28	28	28	28	181	68%	16	16	16	48	38%
$N \models^s F$		7	7	8	37	37	37	37	170	64%	42	42	42	126	100%
$N \models^s F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^d F$		7	12	21	24	26	30	30	150	56%	0	6	6	12	10%
$M \models^d F$	simp	0	7	13	26	28	28	28	130	49%	2	8	8	18	14%
$N \models^d F$		25	**38**	**38**	**38**	**38**	**38**	**38**	253	95%	6	24	24	54	43%
$N \models^d F$	simp	30	32	**38**	**38**	**38**	**38**	**38**	252	95%	2	8	8	18	14%

Fig. 10. Performance results of the automated PML formula classification task conducted with sledgehammer and nitpick; for the problem set $M \models^m F$ a perfect classification is achieved (with or without the use of simp) already for a very short timeout of 2s (and also 1s), meaning that for each formula F in this mode either a proof is reported, if F is a theorem (THM), or a counterexample is presented, if F is a countersatisfiable (CSA). There were no conflicting results reported by sledgehammer and nitpick.

minimal shallow with modal axioms

minimal shallow with semantic axioms

maximal shallow with modal axioms

maximal shallow with semantic axioms

deep with modal axioms

deep with semantic axioms

Results

- Winner — perfect classification — “no counterex.”/“proof” for all valid statement and “counterex.” else
- Second best
- Third best

Evaluation

A MacBook Pro 16,1 (6-Core Intel Core i7 processor, 2,6 GHz, 6 Cores, 256 KB L2 Cache per Core, 12 MB L3 Cache, 16 GB Memory) and Isabelle 2025-1-RC1 in standard configuration was used.

		sledgehammer[14] (38 THM)									nitpick (42 CSA)				
		1s	2s	5s	10s	20s	40s	80s	$\sum$	%	2s	10s	40s	$\sum$	%
$M \models^m F$		9	9	12	19	24	34	37	144	45%	42	42	42	126	100%
$M \models^m F$	simp	18	36	37	**38**	**38**	**38**	**38**	243	91%	42	42	42	126	100%
$N \models^m F$		**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$N \models^m F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^s F$		0	0	0	23	23	24	29	99	37%	16	16	16	48	38%
$M \models^s F$	simp	7	26	26	28	28	28	28	181	68%	16	16	16	48	38%
$N \models^s F$		7	7	8	37	37	37	37	170	64%	42	42	42	126	100%
$N \models^s F$	simp	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**266**	**100%**	42	42	42	126	100%
$M \models^d F$		7	12	21	24	26	30	30	150	56%	0	6	6	12	10%
$M \models^d F$	simp	0	7	13	26	28	28	28	130	49%	2	8	8	18	14%
$N \models^d F$		25	**38**	**38**	**38**	**38**	**38**	**38**	253	95%	6	24	24	54	43%
$N \models^d F$	simp	30	32	**38**	**38**	**38**	**38**	**38**	252	95%	2	8	8	18	14%

Fig. 10. Performance results of the automated PML formula classification task conducted with sledgehammer and nitpick; for the problem set $M \models^m F$ a perfect classification is achieved (with or without the use of simp) already for a very short timeout of 2s (and also 1s), meaning that for each formula F in this mode either a proof is reported, if F is a theorem (THM), or a counterexample is presented, if F is a countersatisfiable (CSA). There were no conflicting results reported by sledgehammer and nitpick.

minimal shallow with modal axioms

minimal shallow with semantic axioms

maximal shallow with modal axioms

maximal shallow with semantic axioms

deep with modal axioms

deep with semantic axioms

Results

- Winner — perfect classification — “no counterex.”/“proof” for all valid statement and “counterex.” else
- Second best
- Third best

Comments

- Remember that reasoning for **schematic formulas** was tested here
- Approach supports the **parallelization** of deep, minimal shallow and maximal shallow

Does this approach scale for Quantifiers? (Example: QBF)

(new — extending results from CADE 2025)

QBF — Syntax and Semantics

The *syntax* of QBF with free variables is defined by the following grammar, where $a \in \mathcal{S}$ are propositional variable symbols declared in a (nonempty) signature $\mathcal{S}$:

$$\varphi, \psi := a \mid \neg\varphi \mid \varphi \vee \psi \mid \forall a.\varphi$$

Further logical connectives can be defined as usual, e.g.: $\varphi \supset \psi := \neg\varphi \vee \psi$, $\varphi \wedge \psi := \neg(\varphi \supset \neg\psi)$, $\exists a.\varphi := \neg\forall a.\neg\varphi$, $\top := a \supset a$ (for some $a \in \mathcal{S}$), and $\bot := \neg\top$.

A *semantics* for QBF determines the truth of a formula φ relative to an assignment function I, which assigns a truth value to each symbol $a \in \mathcal{S}$. Formally, we have:

$$I \models a \text{ iff } I(a)$$

$$I \models \neg\varphi \text{ iff not } I \models \varphi$$

$$I \models \varphi \vee \psi \text{ iff } I \models \varphi \text{ or } I \models \psi$$

$$I \models \forall a.\varphi \text{ iff for all } v \in \{T, F\} \text{ we have } I\langle a \leftarrow v\rangle \models \varphi$$

The modified assignment function $I\langle a \leftarrow v\rangle$ in the last clause is defined such that it is identical to I except on a, for which it now maps to v.

Based on the above definitions, a notion of validity can be defined: A formula φ is valid iff $I \models \varphi$ for all assignment functions I.

QBF: Linked Embeddings

$$"\forall^m x.\ \Phi\ x \ \equiv\ \forall x.\ \Phi\ x"$$

$$"\forall^s a.\ \varphi \ \equiv\ \lambda I.\ \forall v.\ \varphi\ (I\langle a \leftarrow v\rangle)"$$

$$\mid "I \models^d \forall^d x.\ \varphi \ = (\forall v.\ \ I\langle x \leftarrow v\rangle \models^d \varphi)"$$

shallow minimal ($.^m$)

```
theory QBFinHOL_shallow_minimal imports QBFinHOL_preliminaries
begin
 type_synonym σ=bool
—‹Surjective assignment functions are assumed at the meta-level›
 consts  IF::𝓘  axiomatization where IF_surj: "surj IF"
—‹Minimal shallow embedding (of QBF in HOL)›
 definition VarM :: "𝒮⇒σ"      ("_ᵐ")          where   "vᵐ ≡ IF v"
 definition NotM :: "σ⇒σ"      ("¬ᵐ _")        where   "¬ᵐ φ ≡ ¬ φ"
 definition OrM  :: "σ⇒σ⇒σ"    (infixr "∨ᵐ" 930) where "φ ∨ᵐ ψ ≡ φ ∨ ψ"
 definition AllM :: "(𝒮⇒σ)⇒σ" (binder "∀ᵐ" 910) where "∀ᵐx. Φ x ≡ ∀x. Φ x"
—‹Further logical connectives as derived definitions›
 definition ImpM :: "σ⇒σ⇒σ"    (infixr "⊃ᵐ" 92)  where "φ ⊃ᵐ ψ ≡ (¬ᵐφ) ∨ᵐ ψ"
 definition AndM :: "σ⇒σ⇒σ"    (infixr "∧ᵐ" 95)  where "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
 definition ExM  :: "(𝒮⇒σ)⇒σ" (binder "∃ᵐ" 910) where "∃ᵐx. Φ x ≡ ¬ᵐ∀ᵐx. ¬ᵐ(Φ x)"
 consts p::𝒮
 definition TopM :: "σ"        ("⊤ᵐ")          where   "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
 definition BotM :: "σ"        ("⊥ᵐ")          where   "⊥ᵐ ≡ ¬ᵐ ⊤ᵐ"
—‹Validity›
 definition ValM :: "σ⇒bool"   ("⊨ᵐ _" 9)      where   "⊨ᵐ φ ≡ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefM declare VarM_def[DefM] NotM_def[DefM] OrM_def[DefM] AllM_def[DefM]
   ImpM_def[DefM] AndM_def[DefM] ExM_def[DefM] TopM_def[DefM] BotM_def[DefM] ValM_def[DefM]
end
```

deep ($.^d$)

```
theory QBFinHOL_deep imports QBFinHOL_preliminaries
begin
—‹Deep embedding (of QBF in HOL)›
 datatype ℱ = VarD 𝒮 ("_ᵈ") | NotD ℱ ("¬ᵈ _") | OrD ℱ ℱ (infixr "∨ᵈ" 925) | AllD 𝒮 ℱ ("∀ᵈ_. _")
—‹Further logical connectives as definitions›
 definition ImpD  (infixr "⊃ᵈ" 92)  where "φ ⊃ᵈ ψ ≡ (¬ᵈφ) ∨ᵈ ψ"
 definition AndD  (infixr "∧ᵈ" 95)  where "φ ∧ᵈ ψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
 definition ExD   ("∃ᵈ_. _")        where "∃ᵈa. φ ≡ ¬ᵈ(∀ᵈa. ¬ᵈφ)"
 consts p :: 𝒮
 definition TopD  ("⊤ᵈ")            where "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
 definition BotD  ("⊥ᵈ")            where "⊥ᵈ ≡ ¬ᵈ ⊤ᵈ"
—‹Relative truth›
 primrec RelativeTruthD ("_ ⊨ᵈ _" [100,100] 100) where
   "I ⊨ᵈ (aᵈ)      = I a"
 | "I ⊨ᵈ (¬ᵈφ)     = (¬ I ⊨ᵈ φ)"
 | "I ⊨ᵈ φ ∨ᵈ ψ    = (I ⊨ᵈ φ ∨ I ⊨ᵈ ψ)"
 | "I ⊨ᵈ ∀ᵈx. φ    = (∀v. I⟨x ← v⟩ ⊨ᵈ φ)"
—‹Validity›
 definition ValD ("⊨ᵈ _" 9) where "⊨ᵈ φ ≡ ∀I. I ⊨ᵈ φ"
—‹Collection of definitions in a bag called DefD›
 named_theorems DefD
 declare  ImpD_def[DefD] AndD_def[DefD] ExD_def[DefD] TopD_def[DefD] BotD_def[DefD] ValD_def[DefD]
end
```

shallow maximal ($.^s$)

```
theory QBFinHOL_shallow imports QBFinHOL_preliminaries
begin
 type_synonym σ = "𝓘⇒bool"
—‹Maximal shallow embedding (of QBF in HOL)›
 definition VarS :: "𝒮⇒σ"      ("_ˢ")          where   "aˢ ≡ λI. I a"
 definition NotS :: "σ⇒σ"      ("¬ˢ _")        where   "¬ˢ φ ≡ λI. ¬ (φ I)"
 definition OrS  :: "σ⇒σ⇒σ"    (infixr "∨ˢ" 930) where "φ ∨ˢ ψ ≡ λI. φ I ∨ ψ I"
 definition AllS :: "𝒮⇒σ⇒σ"    ("∀ˢ_. _")      where   "∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"
—‹Further logical connectives as derived definitions›
 definition ImpS :: "σ⇒σ⇒σ"    (infixr "⊃ˢ" 92)  where "φ ⊃ˢ ψ ≡ (¬ˢφ) ∨ˢ ψ"
 definition AndS :: "σ⇒σ⇒σ"    (infixr "∧ˢ" 95)  where "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
 definition ExS  :: "𝒮⇒σ⇒σ"    ("∃ˢ_. _")      where   "∃ˢa. φ ≡ ¬ˢ(∀ˢa. ¬ˢφ) "
 consts q :: 𝒮
 definition TopS :: "σ"        ("⊤ˢ")          where   "⊤ˢ ≡ qˢ ⊃ˢ qˢ"
 definition BotS :: "σ"        ("⊥ˢ")          where   "⊥ˢ ≡ ¬ˢ ⊤ˢ"
—‹Relative truth›
 definition RelTruthS :: "𝓘⇒σ⇒bool" (infix "⊨ˢ" 10) where "I ⊨ˢ φ ≡ φ I"
—‹Validity›
 definition ValS :: "σ⇒bool"   ("⊨ˢ _" 9)      where   "⊨ˢ φ ≡ ∀I. I ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefS declare VarS_def[DefS] NotS_def[DefS] OrS_def[DefS] AllS_def[DefS]
   ImpS_def[DefS] AndS_def[DefS] ExS_def[DefS] TopS_def[DefS] BotS_def[DefS] RelTruthS_def[DefS]  ValS_def[DefS]
end
```

QBF: Linked Embeddings

shared preliminaries

```
theory QBFinHOL_preliminaries imports Main
begin
—‹Type declarations common for both the deep and shallow embedding›
 type_synonym 𝒮 = nat —‹Signature of variable symbols: we simply use the natural numbers for this›
 type_synonym ℐ = "𝒮⇒bool" —‹Assignment functions›
—‹Modification of assignment functions›
 definition IntMod :: "ℐ⇒𝒮⇒bool⇒ℐ" ("_⟨_ ← _⟩") where      "I⟨x←y⟩ ≡ λz. if z = x then y else I z"
—‹Some lemmata›
 lemma L1 [simp]:      "(I⟨y ← s⟩) x = I x" if "x ≠ y"                    using that IntMod_def by auto
 lemma L2:             "a ≠ c ⟹ (I⟨a ← b⟩⟨c ← d⟩) = (I⟨c ← d⟩⟨a ← b⟩)"    using IntMod_def by auto
 lemma L3 [simp]:      "(I⟨a ← b⟩) a = b"                                 using IntMod_def by auto
 lemma L4 [simp]:      "I⟨a ← b⟩⟨a ← c⟩ = (I⟨a ← c⟩)"                     using IntMod_def by auto
—‹Further settings› declare[[syntax_ambiguity_warning=false]] nitpick_params[user_axioms,expect=genuine]
end
```

```
| "I ⊨ᵈ ∀ᵈx. φ   = (∀v.  I⟨x ← v⟩ ⊨ᵈ φ)"
```

shallow minimal ($.^m$)

```
theory QBFinHOL_shallow_minimal imports QBFinHOL_preliminaries
begin
 type_synonym σ=bool
—‹Surjective assignment functions are assumed at the meta-level›
 consts  IF::ℐ  axiomatization where IF_surj: "surj IF"
—‹Minimal shallow embedding (of QBF in HOL)›
 definition VarM :: "𝒮⇒σ"       ("_ᵐ")          where      "vᵐ ≡ IF v"
 definition NotM :: "σ⇒σ"       ("¬ᵐ _")        where      "¬ᵐ φ ≡ ¬ φ"
 definition OrM  :: "σ⇒σ⇒σ"     (infixr "∨ᵐ" 930) where    "φ ∨ᵐ ψ ≡ φ ∨ ψ"
 definition AllM :: "(𝒮⇒σ)⇒σ" (binder "∀ᵐ" 910) where    "∀ᵐx. Φ x ≡ ∀x. Φ x"
—‹Further logical connectives as derived definitions›
 definition ImpM :: "σ⇒σ⇒σ"     (infixr "⊃ᵐ" 92)  where    "φ ⊃ᵐ ψ ≡ (¬ᵐφ) ∨ᵐ ψ"
 definition AndM :: "σ⇒σ⇒σ"     (infixr "∧ᵐ" 95)  where    "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
 definition ExM  :: "(𝒮⇒σ)⇒σ" (binder "∃ᵐ" 910) where    "∃ᵐx. Φ x ≡ ¬ᵐ∀ᵐx. ¬ᵐ(Φ x)"
 consts p::𝒮
 definition TopM :: "σ"         ("⊤ᵐ")          where      "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
 definition BotM :: "σ"         ("⊥ᵐ")          where      "⊥ᵐ ≡ ¬ᵐ ⊤ᵐ"
—‹Validity›
 definition ValM :: "σ⇒bool"    ("⊨ᵐ _" 9)      where      "⊨ᵐ φ ≡ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefM declare VarM_def[DefM] NotM_def[DefM] OrM_def[DefM] AllM_def[DefM]
   ImpM_def[DefM] AndM_def[DefM] ExM_def[DefM] TopM_def[DefM] BotM_def[DefM] ValM_def[DefM]
end
```

```
theory QBFinHOL_deep imports QBFinHOL_preliminaries
begin
—‹Deep embedding (of QBF in HOL)›
 datatype ℱ = VarD 𝒮 ("_ᵈ") | NotD ℱ ("¬ᵈ _") | OrD ℱ ℱ (infixr "∨ᵈ" 925) | AllD 𝒮 ℱ ("∀ᵈ_. _")
—‹Further logical connectives as definitions›
 definition ImpD  (infixr "⊃ᵈ" 92)  where      "φ ⊃ᵈ ψ ≡ (¬ᵈφ) ∨ᵈ ψ"
 definition AndD  (infixr "∧ᵈ" 95)  where      "φ ∧ᵈ ψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
 definition ExD   ("∃ᵈ_. _")        where      "∃ᵈa. φ  ≡ ¬ᵈ(∀ᵈa. ¬ᵈφ) "
 consts p :: 𝒮
 definition TopD  ("⊤ᵈ") where                 "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
 definition BotD  ("⊥ᵈ") where                 "⊥ᵈ ≡ ¬ᵈ ⊤ᵈ"
—‹Relative truth›
 primrec RelativeTruthD ("_ ⊨ᵈ _" [100,100] 100) where   "I ⊨ᵈ(aᵈ)      = I a"
                                                        | "I ⊨ᵈ (¬ᵈφ)    = (¬ I ⊨ᵈ φ)"
                                                        | "I ⊨ᵈ φ ∨ᵈ ψ   = (I ⊨ᵈ φ ∨ I ⊨ᵈ ψ)"
                                                        | "I ⊨ᵈ ∀ᵈx. φ   = (∀v.  I⟨x ← v⟩ ⊨ᵈ φ)"
—‹Validity›
 definition ValD ("⊨ᵈ _" 9) where              "⊨ᵈ φ ≡ ∀I. I ⊨ᵈ φ"
—‹Collection of definitions in a bag called DefD›
 named_theorems DefD
 declare  ImpD_def[DefD] AndD_def[DefD] ExD_def[DefD] TopD_def[DefD] BotD_def[DefD] ValD_def[DefD]
end
```

deep ($.^d$)

```
theory QBFinHOL_shallow imports QBFinHOL_preliminaries
begin
 type_synonym σ = "ℐ⇒bool"
—‹Maximal shallow embedding (of QBF in HOL)›
 definition VarS :: "𝒮⇒σ"       ("_ˢ")          where      "aˢ ≡ λI. I a"
 definition NotS :: "σ⇒σ"       ("¬ˢ _")        where      "¬ˢ φ ≡ λI. ¬ (φ I)"
 definition OrS  :: "σ⇒σ⇒σ"     (infixr "∨ˢ" 930) where    "φ ∨ˢ ψ ≡ λI. φ I ∨ ψ I"
 definition AllS :: "𝒮⇒σ⇒σ"     ("∀ˢ_. _")      where      "∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"
—‹Further logical connectives as derived definitions›
 definition ImpS :: "σ⇒σ⇒σ"     (infixr "⊃ˢ" 92)  where    "φ ⊃ˢ ψ ≡ (¬ˢφ) ∨ˢ ψ"
 definition AndS :: "σ⇒σ⇒σ"     (infixr "∧ˢ" 95)  where    "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
 definition ExS  :: "𝒮⇒σ⇒σ"     ("∃ˢ_. _")      where      "∃ˢa. φ ≡ ¬ˢ(∀ˢa. ¬ˢφ) "
 consts q :: 𝒮
 definition TopS :: "σ"         ("⊤ˢ")          where      "⊤ˢ ≡ qˢ ⊃ˢ qˢ"
 definition BotS :: "σ"         ("⊥ˢ")          where      "⊥ˢ ≡ ¬ˢ ⊤ˢ"
—‹Relative truth›
 definition RelTruthS :: "ℐ⇒σ⇒bool" (infix "⊨ˢ" 10) where   "I ⊨ˢ φ ≡ φ I"
—‹Validity›
 definition ValS :: "σ⇒bool"    ("⊨ˢ _" 9)      where      "⊨ˢ φ ≡ ∀I. I ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefS declare VarS_def[DefS] NotS_def[DefS] OrS_def[DefS] AllS_def[DefS]
   ImpS_def[DefS] AndS_def[DefS] ExS_def[DefS] TopS_def[DefS] BotS_def[DefS] RelTruthS_def[DefS]  ValS_def[DefS]
end
```

shallow maximal ($.^s$)

QBF: Linked Embeddings

shared preliminaries

```
theory QBFinHOL_preliminaries imports Main
begin
—‹Type declarations common for both the deep and shallow embedding›
 type_synonym 𝒮 = nat —‹Signature of variable symbols: we simply use the natural numbers for this›
 type_synonym ℐ = "𝒮⇒bool" —‹Assignment functions›
—‹Modification of assignment functions›
 definition IntMod :: "ℐ⇒𝒮⇒bool⇒ℐ" ("_⟨_ ← _⟩") where      "I⟨x←y⟩ ≡ λz. if z = x then y else I z"
—‹Some lemmata›
 lemma L1 [simp]:      "(I⟨y ← s⟩) x = I x" if "x ≠ y"                    using that IntMod_def by auto
 lemma L2:             "a ≠ c ⟹ (I⟨a ← b⟩⟨c ← d⟩) = (I⟨c ← d⟩⟨a ← b⟩)"    using IntMod_def by auto
 lemma L3 [simp]:      "(I⟨a ← b⟩) a = b"                                 using IntMod_def by auto
 lemma L4 [simp]:      "I⟨a ← b⟩⟨a ← c⟩ = (I⟨a ← c⟩)"                     using IntMod_def by auto
—‹Further settings› declare[[syntax_ambiguity_warning=false]] nitpick_params[user_axioms,expect=genuine]
end
```

"∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"

shallow minimal ($.^m$)

```
theory QBFinHOL_shallow_minimal imports QBFinHOL_preliminaries
begin
 type_synonym σ=bool
—‹Surjective assignment functions are assumed at the meta-level›
 consts  IF::ℐ  axiomatization where IF_surj: "surj IF"
—‹Minimal shallow embedding (of QBF in HOL)›
 definition VarM :: "𝒮⇒σ"       ("_ᵐ")          where      "vᵐ ≡ IF v"
 definition NotM :: "σ⇒σ"       ("¬ᵐ _")        where      "¬ᵐ φ ≡ ¬ φ"
 definition OrM  :: "σ⇒σ⇒σ"     (infixr "∨ᵐ" 930) where    "φ ∨ᵐ ψ ≡ φ ∨ ψ"
 definition AllM :: "(𝒮⇒σ)⇒σ" (binder "∀ᵐ" 910) where    "∀ᵐx. Φ x ≡ ∀x. Φ x"
—‹Further logical connectives as derived definitions›
 definition ImpM :: "σ⇒σ⇒σ"     (infixr "⊃ᵐ" 92)  where    "φ ⊃ᵐ ψ ≡ (¬ᵐφ) ∨ᵐ ψ"
 definition AndM :: "σ⇒σ⇒σ"     (infixr "∧ᵐ" 95)  where    "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
 definition ExM  :: "(𝒮⇒σ)⇒σ" (binder "∃ᵐ" 910) where    "∃ᵐx. Φ x ≡ ¬ᵐ∀ᵐx. ¬ᵐ(Φ x)"
 consts p::𝒮
 definition TopM :: "σ"         ("⊤ᵐ")          where      "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
 definition BotM :: "σ"         ("⊥ᵐ")          where      "⊥ᵐ ≡ ¬ᵐ ⊤ᵐ"
—‹Validity›
 definition ValM :: "σ⇒bool"    ("⊨ᵐ _" 9)      where      "⊨ᵐ φ ≡ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefM declare VarM_def[DefM] NotM_def[DefM] OrM_def[DefM] AllM_def[DefM]
   ImpM_def[DefM] AndM_def[DefM] ExM_def[DefM] TopM_def[DefM] BotM_def[DefM] ValM_def[DefM]
end
```

deep ($.^d$)

```
theory QBFinHOL_deep imports QBFinHOL_preliminaries
begin
—‹Deep embedding (of QBF in HOL)›
 datatype ℱ = VarD 𝒮 ("_ᵈ") | NotD ℱ ("¬ᵈ _") | OrD ℱ ℱ (infixr "∨ᵈ" 925) | AllD 𝒮 ℱ ("∀ᵈ_. _")
—‹Further logical connectives as definitions›
 definition ImpD  (infixr "⊃ᵈ" 92)  where      "φ ⊃ᵈ ψ ≡ (¬ᵈφ) ∨ᵈ ψ"
 definition AndD  (infixr "∧ᵈ" 95)  where      "φ ∧ᵈ ψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
 definition ExD   ("∃ᵈ_. _")        where      "∃ᵈa. φ ≡ ¬ᵈ(∀ᵈa. ¬ᵈφ) "
 consts p :: 𝒮
 definition TopD  ("⊤ᵈ") where                 "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
 definition BotD  ("⊥ᵈ") where                 "⊥ᵈ ≡ ¬ᵈ ⊤ᵈ"
—‹Relative truth›
 primrec RelativeTruthD ("_ ⊨ᵈ _" [100,100] 100) where   "I ⊨ᵈ (aᵈ)      = I a"
                                                        | "I ⊨ᵈ (¬ᵈφ)     = (¬ I ⊨ᵈ φ)"
                                                        | "I ⊨ᵈ φ ∨ᵈ ψ     = (I ⊨ᵈ φ ∨ I ⊨ᵈ ψ)"
                                                        | "I ⊨ᵈ ∀ᵈx. φ     = (∀v. I⟨x ← v⟩ ⊨ᵈ φ)"
—‹Validity›
 definition ValD ("⊨ᵈ _" 9) where              "⊨ᵈ φ ≡ ∀I. I ⊨ᵈ φ"
—‹Collection of definitions in a bag called DefD›
 named_theorems DefD
 declare  ImpD_def[DefD] AndD_def[DefD] ExD_def[DefD] TopD_def[DefD] BotD_def[DefD] ValD_def[DefD]
end
```

shallow maximal ($.^s$)

```
theory QBFinHOL_shallow imports QBFinHOL_preliminaries
begin
 type_synonym σ = "ℐ⇒bool"
—‹Maximal shallow embedding (of QBF in HOL)›
 definition VarS :: "𝒮⇒σ"       ("_ˢ")          where      "aˢ ≡ λI. I a"
 definition NotS :: "σ⇒σ"       ("¬ˢ _")        where      "¬ˢ φ ≡ λI. ¬ (φ I)"
 definition OrS  :: "σ⇒σ⇒σ"     (infixr "∨ˢ" 930) where    "φ ∨ˢ ψ ≡ λI. φ I ∨ ψ I"
 definition AllS :: "𝒮⇒σ⇒σ"     ("∀ˢ_. _")      where      "∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"
—‹Further logical connectives as derived definitions›
 definition ImpS :: "σ⇒σ⇒σ"     (infixr "⊃ˢ" 92)  where    "φ ⊃ˢ ψ ≡ (¬ˢφ) ∨ˢ ψ"
 definition AndS :: "σ⇒σ⇒σ"     (infixr "∧ˢ" 95)  where    "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
 definition ExS  :: "𝒮⇒σ⇒σ"     ("∃ˢ_. _")      where      "∃ˢa. φ ≡ ¬ˢ(∀ˢa. ¬ˢφ) "
 consts q :: 𝒮
 definition TopS :: "σ"         ("⊤ˢ")          where      "⊤ˢ ≡ qˢ ⊃ˢ qˢ"
 definition BotS :: "σ"         ("⊥ˢ")          where      "⊥ˢ ≡ ¬ˢ ⊤ˢ"
—‹Relative truth›
 definition RelTruthS :: "ℐ⇒σ⇒bool" (infix "⊨ˢ" 10) where   "I ⊨ˢ φ ≡ φ I"
—‹Validity›
 definition ValS :: "σ⇒bool"    ("⊨ˢ _" 9)      where      "⊨ˢ φ ≡ ∀I. I ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefS declare VarS_def[DefS] NotS_def[DefS] OrS_def[DefS] AllS_def[DefS]
   ImpS_def[DefS] AndS_def[DefS] ExS_def[DefS] TopS_def[DefS] BotS_def[DefS] RelTruthS_def[DefS]  ValS_def[DefS]
end
```

QBF: Linked Embeddings

shared preliminaries

```
theory QBFinHOL_preliminaries imports Main
begin
—‹Type declarations common for both the deep and shallow embedding›
 type_synonym 𝒮 = nat —‹Signature of variable symbols: we simply use the natural numbers for this›
 type_synonym ℐ = "𝒮⇒bool" —‹Assignment functions›
—‹Modification of assignment functions›
 definition IntMod :: "ℐ⇒𝒮⇒bool⇒ℐ" ("_⟨_←_⟩") where      "I⟨x←y⟩ ≡ λz. if z = x then y else I z"
—‹Some lemmata›
 lemma L1 [simp]:      "(I⟨y ← s⟩) x = I x" if "x ≠ y"                          using that IntMod_def by auto
 lemma L2:             "a ≠ c ⟹ (I⟨a ← b⟩⟨c ← d⟩) = (I⟨c ← d⟩⟨a ← b⟩)"          using IntMod_def by auto
 lemma L3 [simp]:      "(I⟨a ← b⟩) a = b"                                       using IntMod_def by auto
 lemma L4 [simp]:      "I⟨a ← b⟩⟨a ← c⟩ = (I⟨a ← c⟩)"                           using IntMod_def by auto
—‹Further settings› declare[[syntax_ambiguity_warning=false]] nitpick_params[user_axioms,expect=genuine]
end
```

"∀ᵐx. Φ x ≡ ∀x. Φ x"

shallow minimal ($.^m$)

```
theory QBFinHOL_shallow_minimal imports QBFinHOL_preliminaries
begin
 type_synonym σ=bool
—‹Surjective assignment functions are assumed at the meta-level›
 consts  IF::ℐ  axiomatization where IF_surj: "surj IF"
—‹Minimal shallow embedding (of QBF in HOL)›
 definition VarM :: "𝒮⇒σ"       ("_ᵐ")          where    "vᵐ ≡ IF v"
 definition NotM :: "σ⇒σ"       ("¬ᵐ _")        where    "¬ᵐ φ ≡ ¬ φ"
 definition OrM  :: "σ⇒σ⇒σ"     (infixr "∨ᵐ" 930) where  "φ ∨ᵐ ψ ≡ φ ∨ ψ"
 definition AllM :: "(𝒮⇒σ)⇒σ" (binder "∀ᵐ" 910) where  "∀ᵐx. Φ x ≡ ∀x. Φ x"
—‹Further logical connectives as derived definitions›
 definition ImpM :: "σ⇒σ⇒σ"     (infixr "⊃ᵐ" 92)  where  "φ ⊃ᵐ ψ ≡ (¬ᵐφ) ∨ᵐ ψ"
 definition AndM :: "σ⇒σ⇒σ"     (infixr "∧ᵐ" 95)  where  "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
 definition ExM  :: "(𝒮⇒σ)⇒σ" (binder "∃ᵐ" 910) where  "∃ᵐx. Φ x ≡ ¬ᵐ∀ᵐx. ¬ᵐ(Φ x)"
 consts p::𝒮
 definition TopM :: "σ"         ("⊤ᵐ")          where    "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
 definition BotM :: "σ"         ("⊥ᵐ")          where    "⊥ᵐ ≡ ¬ᵐ ⊤ᵐ"
—‹Validity›
 definition ValM :: "σ⇒bool"    ("⊨ᵐ _" 9)      where    "⊨ᵐ φ ≡ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefM declare VarM_def[DefM] NotM_def[DefM] OrM_def[DefM] AllM_def[DefM]
   ImpM_def[DefM] AndM_def[DefM] ExM_def[DefM] TopM_def[DefM] BotM_def[DefM] ValM_def[DefM]
end
```

```
theory QBFinHOL_deep imports QBFinHOL_preliminaries
begin
—‹Deep embedding (of QBF in HOL)›
 datatype ℱ = VarD 𝒮 ("_ᵈ") | NotD ℱ ("¬ᵈ _") | OrD ℱ ℱ (infixr "∨ᵈ" 925) | AllD 𝒮 ℱ ("∀ᵈ_. _")
—‹Further logical connectives as definitions›
 definition ImpD  (infixr "⊃ᵈ" 92)  where  "φ ⊃ᵈ ψ ≡ (¬ᵈφ) ∨ᵈ ψ"
 definition AndD  (infixr "∧ᵈ" 95)  where  "φ ∧ᵈ ψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
 definition ExD   ("∃ᵈ_. _")        where  "∃ᵈa. φ ≡ ¬ᵈ(∀ᵈa. ¬ᵈφ) "
 consts p :: 𝒮
 definition TopD  ("⊤ᵈ")  where  "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
 definition BotD  ("⊥ᵈ")  where  "⊥ᵈ ≡ ¬ᵈ ⊤ᵈ"
—‹Relative truth›
 primrec RelativeTruthD ("_ ⊨ᵈ _" [100,100] 100) where  "I ⊨ᵈ (aᵈ)      = I a"
                                                       | "I ⊨ᵈ (¬ᵈφ)     = (¬ I ⊨ᵈ φ)"
                                                       | "I ⊨ᵈ φ ∨ᵈ ψ     = (I ⊨ᵈ φ ∨ I ⊨ᵈ ψ)"
                                                       | "I ⊨ᵈ ∀ᵈx. φ     = (∀v. I⟨x ← v⟩ ⊨ᵈ φ)"
—‹Validity›
 definition ValD ("⊨ᵈ _" 9) where  "⊨ᵈ φ ≡ ∀I. I ⊨ᵈ φ"
—‹Collection of definitions in a bag called DefD›
 named_theorems DefD
 declare  ImpD_def[DefD] AndD_def[DefD] ExD_def[DefD] TopD_def[DefD] BotD_def[DefD] ValD_def[DefD]
end
```

deep ($.^d$)

```
theory QBFinHOL_shallow imports QBFinHOL_preliminaries
begin
 type_synonym σ = "ℐ⇒bool"
—‹Maximal shallow embedding (of QBF in HOL)›
 definition VarS :: "𝒮⇒σ"       ("_ˢ")          where  "aˢ ≡ λI. I a"
 definition NotS :: "σ⇒σ"       ("¬ˢ _")        where  "¬ˢ φ ≡ λI. ¬ (φ I)"
 definition OrS  :: "σ⇒σ⇒σ"     (infixr "∨ˢ" 930) where  "φ ∨ˢ ψ ≡ λI. φ I ∨ ψ I"
 definition AllS :: "𝒮⇒σ⇒σ"     ("∀ˢ_. _")      where  "∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"
—‹Further logical connectives as derived definitions›
 definition ImpS :: "σ⇒σ⇒σ"     (infixr "⊃ˢ" 92)  where  "φ ⊃ˢ ψ ≡ (¬ˢφ) ∨ˢ ψ"
 definition AndS :: "σ⇒σ⇒σ"     (infixr "∧ˢ" 95)  where  "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
 definition ExS  :: "𝒮⇒σ⇒σ"     ("∃ˢ_. _")       where  "∃ˢa. φ ≡ ¬ˢ(∀ˢa. ¬ˢφ) "
 consts q :: 𝒮
 definition TopS :: "σ"         ("⊤ˢ")          where  "⊤ˢ ≡ qˢ ⊃ˢ qˢ"
 definition BotS :: "σ"         ("⊥ˢ")          where  "⊥ˢ ≡ ¬ˢ ⊤ˢ"
—‹Relative truth›
 definition RelTruthS :: "ℐ⇒σ⇒bool" (infix "⊨ˢ" 10) where  "I ⊨ˢ φ ≡ φ I"
—‹Validity›
 definition ValS :: "σ⇒bool"    ("⊨ˢ _" 9)      where  "⊨ˢ φ ≡ ∀I. I ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefS declare VarS_def[DefS] NotS_def[DefS] OrS_def[DefS] AllS_def[DefS]
   ImpS_def[DefS] AndS_def[DefS] ExS_def[DefS] TopS_def[DefS] BotS_def[DefS] RelTruthS_def[DefS]  ValS_def[DefS]
end
```

shallow maximal ($.^s$)

Proving Faithfulness:
QBF deep —
QBF maximal shallow —
QBF minimal shallow

QBF: Linked Embeddings

shared preliminaries

```
theory QBFinHOL_preliminaries imports Main
begin
—‹Type declarations common for both the deep and shallow embedding›
 type_synonym 𝒮 = nat —‹Signature of variable symbols: we simply use the natural numbers for this›
 type_synonym ℐ = "𝒮⇒bool" —‹Assignment functions›
—‹Modification of assignment functions›
 definition IntMod :: "ℐ⇒𝒮⇒bool⇒ℐ" ("_⟨_ ← _⟩") where      "I⟨x←y⟩ ≡ λz. if z = x then y else I z"
—‹Some lemmata›
 lemma L1 [simp]:   "(I⟨y ← s⟩) x = I x" if "x ≠ y"                    using that IntMod_def by auto
 lemma L2:          "a ≠ c ⟹ (I⟨a ← b⟩⟨c ← d⟩) = (I⟨c ← d⟩⟨a ← b⟩)"    using IntMod_def by auto
 lemma L3 [simp]:   "(I⟨a ← b⟩) a = b"                                 using IntMod_def by auto
 lemma L4 [simp]:   "I⟨a ← b⟩⟨a ← c⟩ = (I⟨a ← c⟩)"                     using IntMod_def by auto
—‹Further settings› declare[[syntax_ambiguity_warning=false]] nitpick_params[user_axioms,expect=genuine]
end
```

shallow minimal ($.^m$)

```
theory QBFinHOL_shallow_minimal imports QBFinHOL_preliminaries
begin
 type_synonym σ=bool
—‹Surjective assignment functions are assumed at the meta-level›
 consts  IF::ℐ  axiomatization where IF_surj: "surj IF"
—‹Minimal shallow embedding (of QBF in HOL)›
 definition VarM :: "𝒮⇒σ"       ("_ᵐ")          where    "vᵐ ≡ IF v"
 definition NotM :: "σ⇒σ"       ("¬ᵐ _")        where    "¬ᵐ φ ≡ ¬ φ"
 definition OrM  :: "σ⇒σ⇒σ"     (infixr "∨ᵐ" 930) where  "φ ∨ᵐ ψ ≡ φ ∨ ψ"
 definition AllM :: "(𝒮⇒σ)⇒σ" (binder "∀ᵐ" 910) where  "∀ᵐx. Φ x ≡ ∀x. Φ x"
—‹Further logical connectives as derived definitions›
 definition ImpM :: "σ⇒σ⇒σ"     (infixr "⊃ᵐ" 92)  where  "φ ⊃ᵐ ψ ≡ (¬ᵐφ) ∨ᵐ ψ"
 definition AndM :: "σ⇒σ⇒σ"     (infixr "∧ᵐ" 95)  where  "φ ∧ᵐ ψ ≡ ¬ᵐ(φ ⊃ᵐ ¬ᵐψ)"
 definition ExM  :: "(𝒮⇒σ)⇒σ" (binder "∃ᵐ" 910) where  "∃ᵐx. Φ x ≡ ¬ᵐ∀ᵐx. ¬ᵐ(Φ x)"
 consts p::𝒮
 definition TopM :: "σ"         ("⊤ᵐ")          where    "⊤ᵐ ≡ pᵐ ⊃ᵐ pᵐ"
 definition BotM :: "σ"         ("⊥ᵐ")          where    "⊥ᵐ ≡ ¬ᵐ ⊤ᵐ"
—‹Validity›
 definition ValM :: "σ⇒bool"    ("⊨ᵐ _" 9)      where    "⊨ᵐ φ ≡ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefM declare VarM_def[DefM] NotM_def[DefM] OrM_def[DefM] AllM_def[DefM]
   ImpM_def[DefM] AndM_def[DefM] ExM_def[DefM] TopM_def[DefM] BotM_def[DefM] ValM_def[DefM]
end
```

deep ($.^d$)

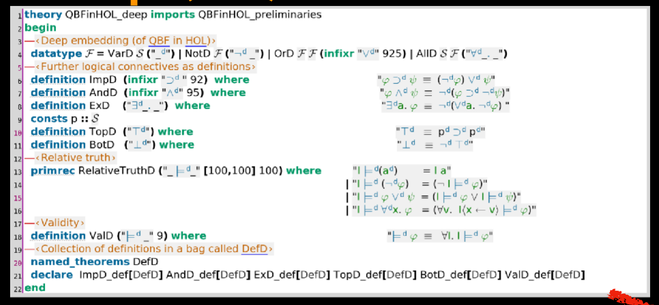
```
theory QBFinHOL_deep imports QBFinHOL_preliminaries
begin
—‹Deep embedding (of QBF in HOL)›
 datatype ℱ = VarD 𝒮 ("_ᵈ") | NotD ℱ ("¬ᵈ _") | OrD ℱ ℱ (infixr "∨ᵈ" 925) | AllD 𝒮 ℱ ("∀ᵈ_. _")
—‹Further logical connectives as definitions›
 definition ImpD  (infixr "⊃ᵈ" 92)  where   "φ ⊃ᵈ ψ ≡ (¬ᵈφ) ∨ᵈ ψ"
 definition AndD  (infixr "∧ᵈ" 95)  where   "φ ∧ᵈ ψ ≡ ¬ᵈ(φ ⊃ᵈ ¬ᵈψ)"
 definition ExD   ("∃ᵈ_. _")        where   "∃ᵈa. φ  ≡ ¬ᵈ(∀ᵈa. ¬ᵈφ) "
 consts p :: 𝒮
 definition TopD  ("⊤ᵈ") where              "⊤ᵈ ≡ pᵈ ⊃ᵈ pᵈ"
 definition BotD  ("⊥ᵈ") where              "⊥ᵈ ≡ ¬ᵈ ⊤ᵈ"
—‹Relative truth›
 primrec RelativeTruthD ("_ ⊨ᵈ _" [100,100] 100) where   "I ⊨ᵈ (aᵈ)      = I a"
                                                        | "I ⊨ᵈ (¬ᵈφ)     = (¬ I ⊨ᵈ φ)"
                                                        | "I ⊨ᵈ φ ∨ᵈ ψ     = (I ⊨ᵈ φ ∨ I ⊨ᵈ ψ)"
                                                        | "I ⊨ᵈ ∀ᵈx. φ     = (∀v. I⟨x ← v⟩ ⊨ᵈ φ)"
—‹Validity›
 definition ValD ("⊨ᵈ _" 9) where                      "⊨ᵈ φ ≡ ∀I. I ⊨ᵈ φ"
—‹Collection of definitions in a bag called DefD›
 named_theorems DefD
 declare  ImpD_def[DefD] AndD_def[DefD] ExD_def[DefD] TopD_def[DefD] BotD_def[DefD] ValD_def[DefD]
end
```

shallow maximal ($.^s$)

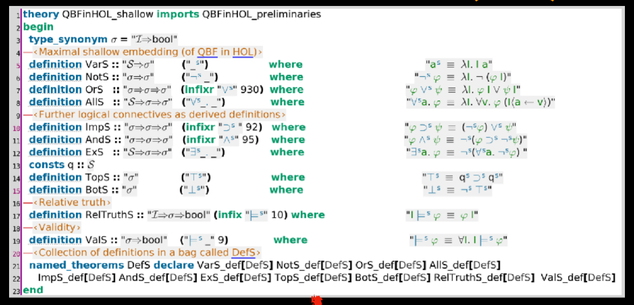
```
theory QBFinHOL_shallow imports QBFinHOL_preliminaries
begin
 type_synonym σ = "ℐ⇒bool"
—‹Maximal shallow embedding (of QBF in HOL)›
 definition VarS :: "𝒮⇒σ"       ("_ˢ")          where    "aˢ ≡ λI. I a"
 definition NotS :: "σ⇒σ"       ("¬ˢ _")        where    "¬ˢ φ ≡ λI. ¬ (φ I)"
 definition OrS  :: "σ⇒σ⇒σ"     (infixr "∨ˢ" 930) where  "φ ∨ˢ ψ ≡ λI. φ I ∨ ψ I"
 definition AllS :: "𝒮⇒σ⇒σ"     ("∀ˢ_. _")      where    "∀ˢa. φ ≡ λI. ∀v. φ (I⟨a ← v⟩)"
—‹Further logical connectives as derived definitions›
 definition ImpS :: "σ⇒σ⇒σ"     (infixr "⊃ˢ" 92)  where  "φ ⊃ˢ ψ ≡ (¬ˢφ) ∨ˢ ψ"
 definition AndS :: "σ⇒σ⇒σ"     (infixr "∧ˢ" 95)  where  "φ ∧ˢ ψ ≡ ¬ˢ(φ ⊃ˢ ¬ˢψ)"
 definition ExS  :: "𝒮⇒σ⇒σ"     ("∃ˢ_. _")      where    "∃ˢa. φ ≡ ¬ˢ(∀ˢa. ¬ˢφ) "
 consts q :: 𝒮
 definition TopS :: "σ"         ("⊤ˢ")          where    "⊤ˢ ≡ qˢ ⊃ˢ qˢ"
 definition BotS :: "σ"         ("⊥ˢ")          where    "⊥ˢ ≡ ¬ˢ ⊤ˢ"
—‹Relative truth›
 definition RelTruthS :: "ℐ⇒σ⇒bool" (infix "⊨ˢ" 10) where   "I ⊨ˢ φ ≡ φ I"
—‹Validity›
 definition ValS :: "σ⇒bool"    ("⊨ˢ _" 9)      where    "⊨ˢ φ ≡ ∀I. I ⊨ˢ φ"
—‹Collection of definitions in a bag called DefS›
 named_theorems DefS declare VarS_def[DefS] NotS_def[DefS] OrS_def[DefS] AllS_def[DefS]
   ImpS_def[DefS] AndS_def[DefS] ExS_def[DefS] TopS_def[DefS] BotS_def[DefS] RelTruthS_def[DefS]  ValS_def[DefS]
end
```

QBF — Faithfulness Automated

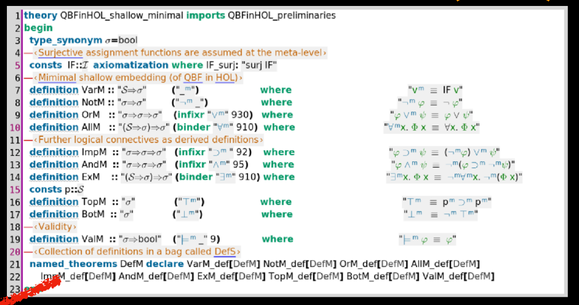
deep ($.^d$)



shallow maximal ($.^s$)



shallow minimal ($.^m$)



```
theory QBFinHOL_faithfulness
  imports QBFinHOL_deep QBFinHOL_shallow_minimal QBFinHOL_shallow QBFinHOL_deep_subst_lemma QBFinHOL_surj
begin
—‹Mappings: deep to maximal shallow and deep to minimal shallow›
  fun DpToShS ("⟦_⟧") where      "⟦xᵈ⟧ = xˢ" | "⟦¬ᵈφ⟧ = ¬ˢ⟦φ⟧"  | "⟦φ∨ᵈψ⟧ = ⟦φ⟧∨ˢ⟦ψ⟧" | "⟦∀ᵈv. φ⟧ = ∀ˢv. ⟦φ⟧"
  fun DpToShM ("(|_|)") where    "(|xᵈ|) = xᵐ" | "(|¬ᵈφ|) = ¬ᵐ(|φ|)" | "(|φ∨ᵈψ|) = (|φ|)∨ᵐ(|ψ|)" | "(|∀ᵈv. φ|) = (∀ᵐx. (|[v←ᵣxᵈ](φ)|))"
—‹Proving faithfulness between deep and maximal shallow›
  theorem FaithfulSDlem: "(I ⊨ˢ ⟦φ⟧) ⟷ (I ⊨ᵈ φ)"          by (induct φ arbitrary: I) (auto simp: DefS DefD)
  theorem FaithfulSD:    "(⊨ˢ ⟦φ⟧) ⟷  (⊨ᵈ φ)"            by (simp add: FaithfulSDlem ValD_def ValS_def)
—‹Proving faithfulness between deep and minimal shallow›
  theorem FaithfulMD:    "(⊨ᵐ (|φ|)) ⟷ (IF ⊨ᵈ φ)" by (induct φ rule: QInduct; simp add: DefD DefM; metis IF_surj surjE)
—‹Proving faithfulness between maximal and minimal shallow›
  theorem FaithfulMS:    "(⊨ᵐ (|φ|)) ⟷ (IF ⊨ˢ ⟦φ⟧)"          using FaithfulSDlem FaithfulMD by auto
end
```

QBF: Lemmas for deep, e.g. substitution lemma

```
theory QBFinHOL_deep_subst_lemma imports QBFinHOL_deep
begin
 primrec is_free (infix "free'_in" 900)  where
    "x free_in (y ᵈ) = (y = x)" | "x free_in (¬ ᵈφ) = (x free_in φ)"
  | "x free_in (φ ∨ᵈ ψ) = (x free_in φ ∨ x free_in ψ)" | "x free_in (∀ ᵈy. φ) = (x free_in φ ∧ x ≠ y)"
 abbreviation is_not_free (infix "not'_free'_in" 900) where "x not_free_in φ ≡ ¬(x free_in φ)"
 fun is_bound (infix "bound'_in" 900)  where
    "x bound_in (y ᵈ) = False" | "x bound_in (¬ ᵈφ) = (x bound_in φ)"
  | "x bound_in (φ ∨ᵈ ψ) = (x bound_in φ ∨ x bound_in ψ)" | "x bound_in (∀ ᵈ y. φ) = (x = y ∨ (x bound_in φ))"
 abbreviation is_not_bound (infix "not'_bound'_in" 900) where "x not_bound_in φ ≡ ¬(x bound_in φ)"
 abbreviation occurs (infix "occurs'_in" 900) where "x occurs_in φ ≡ x free_in φ ∨ x bound_in φ"
 abbreviation not_in (infix "not'_in" 900) where "x not_in φ ≡ x not_free_in φ ∧ x not_bound_in φ"
 primrec fresh ("fresh'(_')") where
  "fresh x ᵈ = x+1" | "fresh ¬ ᵈφ = fresh φ" | "fresh (φ∨ᵈψ) = max (fresh φ) (fresh ψ)" | "fresh ∀ᵈx. φ = max (x+1) (fresh φ)"
 lemma L5: "x bound_in φ ⟹ x < (fresh φ)" by (induct φ) auto
 lemma L6: "x free_in φ ⟹ x < (fresh φ)" by (induct φ) auto
 lemma L7: "(fresh φ) not_in φ" using L5 L6 by blast
 lemma L8: "max (fresh φ) (fresh ψ) not_free_in φ" by (metis L6 L7 max.absorb3 max_def)
 lemma L9: "max (fresh φ) (fresh ψ) not_bound_in φ" by (metis L5 L7 max.absorb3 max_def)
 lemma L10: "max (fresh φ) (fresh ψ) not_free_in ψ" by (metis L8 max.commute)
 lemma L11: "max (fresh φ) (fresh ψ) not_bound_in ψ" by (metis L9 max.commute)
 lemma L12: "y not_free_in φ ⟹ (I(y:=s)) ⊨ᵈ φ ≡ I ⊨ᵈ φ" by (induct φ arbitrary: I; simp) (metis L4 L2)
 primrec Subst ("[_←_]'(_')") where ‹Substitution ignoring potential variable capture›
    "[x←τ](y ᶜ) = (if x = y then τ else (y ᵈ))"     | "[x←τ](¬ ᵈφ) = ¬ ᵈ[x←τ](φ)"
  | "[x←τ](φ ∨ᵈ ψ) = ([x←τ](φ) ∨ ᵈ [x←τ](ψ))" | "[x←τ](∀ ᵈy. φ) = (if x = y then (∀ ᵈy. φ) else (∀ ᵈy. [x←τ](φ)))"
 lemma L13[simp]: "size [x←y ᵈ](φ) = size φ" by (induct φ; auto)
 lemma L14[simp]: "[x←x ᶜ](φ) = φ" by (induct φ; auto)
 lemma L15: assumes "x ≠ a" shows "[a←τ]([a←x ᶜ](φ)) = [a←x ᵈ](φ)" using assms by (induct φ) auto
 lemma L16[simp]: assumes "a ≠ x" shows "a not_free_in ([a←x ᵈ](φ))" using assms by (induct φ) auto
 lemma SInduct:  ‹Induction over the size of a formula›
   assumes "⋀ x. P (x ᵈ)"
   assumes "⋀ φ. (⋀ ψ. size ψ ≤ size φ ⟹ P ψ) ⟹ P (¬ ᵈφ)"
   assumes "⋀ φ ψ. (⋀ χ. size χ ≤ size φ + size ψ ⟹ P χ) ⟹ P (φ ∨ᵈ ψ)"
   assumes "⋀ y φ. (⋀ ψ. size ψ ≤ size φ ⟹ P ψ) ⟹ P (∀ ᵈy. φ)"
   shows "P φ"
  using assms proof (induct "size φ" arbitrary: φ rule: less_induct) case less thus ?case by (induct φ) auto qed
```

```
 lemma QInduct: ‹Stronger induction›
   assumes "⋀ x. P (x ᵈ)"
   assumes "⋀ φ. P φ ⟹ P (¬ ᵈφ)"
   assumes "⋀ φ ψ. P φ ⟹ P ψ ⟹ P (φ ∨ᵈ ψ)"
   assumes "⋀ y φ. (⋀ ψ. size ψ ≤ size φ ⟹ P ψ) ⟹ P (∀ ᵈy. φ)"
   shows "P φ" using assms by (induct φ rule: SInduct) auto
 primrec SubstitutableForIn ‹Substitutable for a variable›
    ("_ is'_subst'_for _ in _" [999,1,999] 999) where "τ is_subst_for x in (y ᵈ) = True"
  | "τ is_subst_for x in (¬ ᵈφ) = τ is_subst_for x in (φ)"
  | "τ is_subst_for x in (φ ∨ᵈ ψ) = (τ is_subst_for x in (φ) ∧ τ is_subst_for x in (ψ))"
  | "τ is_subst_for x in (∀ᵈ y. φ) = (y = x ∨ (x not_free_in φ ∨ y not_free_in τ) ∧ τ is_subst_for x in φ)"
 lemma SubstitutionLemma [simp]: ‹Substitution lemma›
   assumes "τ is_subst_for x in φ" shows "(I ⊨ ᵈ ([x←τ](φ))) = I(x←(I ⊨ ᵈ τ)) ⊨ ᵈ φ"
   using assms by (induction φ arbitrary: I; auto simp: L12 L2)
 fun ren_for_subst where ‹Alphabetic variants for unrestricted substitution›
    "ren_for_subst x τ (y ᵈ) = y ᵈ" | "ren_for_subst x τ (¬ ᵈφ) = (¬ ᵈ(ren_for_subst x τ φ))"
  | "ren_for_subst x τ (φ ∨ᵈ ψ) = (ren_for_subst x τ φ ∨ ᵈ ren_for_subst x τ ψ)"
  | "ren_for_subst x τ (∀ ᵈ y. φ) = (if y free_in τ ∧ x free_in φ  then
       let f = fresh(φ ∨ ᵈ τ); φ' = [y←f ᵈ](φ) in (∀ ᵈf. ren_for_subst x τ φ') else ∀ ᵈy. ren_for_subst x τ φ)"
 lemma L17 [simp]: "size (ren_for_subst x τ φ) = size φ" by (induct φ arbitrary: τ x rule: QInduct; simp add: Let_def)
 lemma L18: "α not_in φ ⟹ (α ᵈ) is_subst_for β in φ" by (induct φ) auto
 lemma L19: "x free_in ψ ⟹ y ≠ x ⟹ x free_in [y←τ](ψ)" by (induct ψ) auto
 lemma L20 [simp]: "x free_in ren_for_subst x τ φ = (x free_in φ)"
  by (induct φ rule: QInduct; simp add: Let_def) (metis L16 L6 L7 L19 max.absorb3 max_def_raw)
 lemma L21: "(I ⊨ ᶜ φ) = (I ⊨ ᵈ (ren_for_subst x τ φ))" by (induct φ arbitrary: x τ I rule: QInduct;
   simp add: Let_def; subst SubstitutionLemma; safe intro!: L18; insert L8 L9; simp; metis L8 L2 L12)
 lemma L22: "τ is_subst_for x in (ren_for_subst x τ φ)" by (induct φ rule: QInduct; auto simp add: Let_def) (metis L10)
 lemma L23: "(α ᵈ) is_subst_for α in φ" by (induct φ) auto
 lemma L24: assumes "α not_free_in φ" shows "[α←τ](φ) = φ" using assms by (induct φ) auto
 lemma L25: "β not_free_in φ ⟹ (α ᵈ) is_subst_for β in φ" by (induct φ) auto
 lemma L26 [simp]: "z bound_in [x←y ᵈ](φ) = z bound_in φ" by (induct φ) auto
 definition ren_subst ("[_ ←ᵣ _]'(_')") where "[x ←ᵣ τ](φ) = [x←τ](ren_for_subst x τ φ)" ‹Renaming substitution›
 lemma L27 [simp]: "(I ⊨ ᵈ [x ←ᵣ τ](φ)) = (I(x←I ⊨ ᵈ τ) ⊨ ᵈ φ)" using L22 L21 unfolding ren_subst_def by force
 lemma L28 [simp]: "size ([x ←ᵣ y ᵈ](φ)) = size φ" by (induct φ rule: QInduct) (auto simp: ren_subst_def Let_def)
 lemma L29: "I ⊨ ᵈ (∀ ᵈx. φ) = (∀τ. I ⊨ ᵈ ([x ←ᵣ τ](φ)))" by (smt L27 RelativeTruthD.simps(4))
end
```

QBF: Lemmas (for deep), e.g. Substitution lemma

```
3  primrec is_free (infix "free'_in" 900)  where
4     "x free_in (y ᵈ) = (y = x)" | "x free_in (¬ ᵈφ) = (x free_in φ)"
5   | "x free_in (φ ∨ᵈ ψ) = (x free_in φ ∨ x free_in ψ)" | "x free_in (∀ ᵈy. φ) = (x free_in φ ∧ x ≠ y)"
6  abbreviation is_not_free (infix "not'_free'_in" 900) where "x not_free_in φ ≡ ¬(x free_in φ)"
```

```
23 primrec Subst ("[_←_]'(_')") where —‹Substitution ignoring potential variable capture›
24    "[x←τ](y ᵈ) = (if x = y then τ else (y ᵈ))"     | "[x←τ](¬ ᵈφ) = ¬ ᵈ[x←τ](φ)"
25  | "[x←τ](φ ∨ᵈ ψ) = ([x←τ](φ) ∨ ᵈ [x←τ](ψ))" | "[x←τ](∀ ᵈy. φ) = (if x = y then (∀ ᵈy. φ) else (∀ ᵈy. [x←τ](φ)))"
```

```
43 primrec SubstitutableForIn —‹Substitutable for a variable›
44    ("_ is'_subst'_for _ in _" [999,1,999] 999) where "τ is_subst_for x in (y ᵈ) = True"
45  | "τ is_subst_for x in (¬ ᵈφ) = τ is_subst_for x in (φ)"
46  | "τ is_subst_for x in (φ ∨ᵈ ψ) = (τ is_subst_for x in (φ) ∧ τ is_subst_for x in (ψ))"
47  | "τ is_subst_for x in (∀ᵈ y. φ) = (y = x ∨ (x not_free_in φ ∨ y not_free_in τ) ∧ τ is_subst_for x in φ)"
48 lemma SubstitutionLemma [simp]: —‹Substitution lemma›
49  assumes "τ is_subst_for x in φ" shows "(I ⊨ᵈ ([x←τ](φ))) = I⟨x←(I ⊨ᵈ τ)⟩ ⊨ᵈ φ"
50  using assms by (induction φ arbitrary: I; auto simp: L12 L2)
```

QBF: Discharging the surjectivity assumption (deep)

```
1  theory QBFinHOL_surj imports QBFinHOL_deep_subst_lemma
2  begin
3   theorem SurjectiveDeep:      "(∀I. I ⊨ᵈ φ) ⟷ (∀I. surj I ⟶ I ⊨ᵈ φ)"
4     proof(safe) fix I
5       obtain a b where          "a ≠ b ∧ a not_free_in φ ∧ b not_free_in φ"      by (metis L10 L8 is_free.simps(1))
6       moreover define J where   "J ≡ λx. if x free_in φ then I x else x = a"
7       ultimately have           "surj J"                                        by (smt surj_def)
8       moreover assume           "∀ I. surj I ⟶ I ⊨ᵈ φ"
9       ultimately have           "J ⊨ᵈ φ"                                        by blast
10      moreover have             "∀ x. x free_in φ ⟶ J x = I x"                  unfolding J_def by force
11      ultimately show           "I ⊨ᵈ φ"                  by (induct φ arbitrary: I J; simp add: DefD; smt IntMod_def)
12    qed(auto)
13 end
```

Playing with QBF in Isabelle/HOL

(new — extending results from CADE 2025)

```
theory QBFinHOL_experiments imports QBFinHOL_deep  QBFinHOL_shallow  QBFinHOL_shallow_minimal
begin
 abbreviation "(x::𝒮) ≡ 1"  abbreviation "(y::𝒮) ≡ 2"  abbreviation "(z::𝒮) ≡ 3"
 lemma "⊨ᵈ ∀ᵈx.∃ᵈy. (xᵈ ∨ᵈ yᵈ)"      unfolding DefD apply simp by auto
 lemma "⊨ˢ ∀ˢx.∃ˢy. (xˢ ∨ˢ yˢ)"      unfolding DefS apply simp by auto
 lemma "⊨ᵐ ∀ᵐx.∃ᵐy. (xᵐ ∨ᵐ yᵐ)"   unfolding DefM apply simp by (metis IF_surj surjE)

 lemma "⊨ᵈ ∀ᵈx.∃ᵈy. (xᵈ ∧ᵈ yᵈ)"      unfolding DefD apply simp nitpick oops —‹Counterexample›
 lemma "⊨ˢ ∀ˢx.∃ˢy. (xˢ ∧ˢ yˢ)"      unfolding DefS apply simp nitpick oops —‹Counterexample›
 lemma "⊨ᵐ ∀ᵐx.∃ᵐy. (xᵐ ∧ᵐ yᵐ)"  unfolding DefM apply simp nitpick oops —‹Counterexample›

 lemma "⊨ᵈ ∀ᵈx.∀ᵈy.∃ᵈz. (((zᵈ ⊃ᵈ ((xᵈ ⊃ᵈ ¬ᵈ yᵈ) ∧ᵈ (yᵈ ⊃ᵈ ¬ᵈ xᵈ))) ∧ᵈ (((xᵈ ⊃ᵈ ¬ᵈ yᵈ) ∧ᵈ (yᵈ ⊃ᵈ ¬ᵈ xᵈ)) ⊃ᵈ zᵈ)))"
  unfolding DefD apply simp by auto
 lemma "⊨ˢ ∀ˢx.∀ˢy.∃ˢz. (((zˢ ⊃ˢ ((xˢ ⊃ˢ ¬ˢ yˢ) ∧ˢ (yˢ ⊃ˢ ¬ˢ xˢ))) ∧ˢ (((xˢ ⊃ˢ ¬ˢ yˢ) ∧ˢ (yˢ ⊃ˢ ¬ˢ xˢ)) ⊃ˢ zˢ)))"
  unfolding DefS apply simp by auto
 lemma "⊨ᵐ ∀ᵐx.∀ᵐy.∃ᵐz. (((zᵐ⊃ᵐ((xᵐ⊃ᵐ ¬ᵐyᵐ) ∧ᵐ (yᵐ⊃ᵐ ¬ᵐxᵐ))) ∧ᵐ (((xᵐ⊃ᵐ ¬ᵐyᵐ) ∧ᵐ (yᵐ⊃ᵐ ¬ᵐxᵐ))⊃ᵐzᵐ)))"
  unfolding DefM apply simp by (metis IF_surj surjE)
end
```

```
theory QBFinHOL_experiments_medium imports QBFinHOL_deep  QBFinHOL_shallow  QBFinHOL_shallow_minimal
begin
  abbreviation "EquivD φ ψ ≡  (AndD (ImpD φ ψ) (ImpD ψ φ))"
  lemma QBFmediumD: "
⊨ᵈ (AllD 30 (ExD 29 (AllD 28 (ExD 27 (AllD 26 (ExD 25 (AllD 24 (ExD 23 (AllD 22 (ExD 21 (AllD 20 (ExD 19 (AllD 18 (ExD 17
(AllD 16 (ExD 15 (AllD 14 (ExD 13 (AllD 12 (ExD 11 (AllD 10 (ExD 9 (AllD 8 (ExD 7 (AllD 6 (ExD 5 (AllD 4 (ExD 3 (AllD 2 (ExD 1
(AndD (EquivD (AndD (AndD (EquivD (OrD (EquivD (VarD 24) (VarD 21)) (EquivD (VarD 1) (VarD 3))) (OrD (OrD (VarD 21)
(NotD (VarD 16))) (OrD (VarD 16) (VarD 20)))) (ImpD (EquivD (EquivD (VarD 17) (NotD (VarD 15))) (EquivD (VarD 24)
(NotD (VarD 17)))) (ImpD (AndD (VarD 20) (VarD 19)) (EquivD (VarD 14) (VarD 1))))) (OrD (ImpD (AndD (ImpD (NotD (VarD 15))
(VarD 2)) (EquivD (VarD 24) (VarD 22))) (OrD (AndD (VarD 8) (VarD 7)) (EquivD (VarD 14) (NotD (VarD 14))))) (ImpD (EquivD
(ImpD (NotD (VarD 28)) (VarD 25)) (EquivD (VarD 19) (VarD 10))) (AndD (AndD (VarD 16) (NotD (VarD 25))) (EquivD (VarD 21)
(NotD (VarD 13)))))) (AndD (AndD (EquivD (EquivD (OrD (VarD 10) (VarD 23)) (OrD (NotD (VarD 3)) (VarD 22))) (EquivD (OrD
(VarD 12) (NotD (VarD 21))) (AndD (VarD 4) (VarD 2)))) (OrD (OrD (ImpD (VarD 23) (VarD 27)) (EquivD (VarD 26) (VarD 9)))
(AndD (ImpD (VarD 18) (NotD (VarD 8))) (AndD (VarD 19) (VarD 23))))) (EquivD (OrD (ImpD (ImpD (VarD 28) (VarD 2)) (EquivD
(NotD (VarD 6)) (VarD 6))) (ImpD (EquivD (VarD 26) (VarD 22)) (ImpD (NotD (VarD 25)) (VarD 30)))) (EquivD (EquivD (EquivD
(VarD 28) (VarD 12)) (OrD (VarD 17) (NotD (VarD 29)))) (EquivD (AndD (NotD (VarD 28)) (NotD (VarD 8))) (OrD (VarD 17)
(VarD 10)))))) (EquivD (OrD (ImpD (EquivD (OrD (EquivD (VarD 5) (VarD 6)) (ImpD (VarD 21) (VarD 16))) (EquivD (OrD
(VarD 25) (VarD 12)) (ImpD (NotD (VarD 13)) (VarD 15)))) (ImpD (EquivD (AndD (VarD 25) (NotD (VarD 4))) (EquivD (NotD
(VarD 26)) (VarD 26))) (ImpD (AndD (VarD 13) (NotD (VarD 11))) (ImpD (NotD (VarD 13)) (NotD (VarD 25)))))) (EquivD (AndD
(EquivD (AndD (VarD 12) (VarD 18)) (ImpD (VarD 4) (VarD 14))) (ImpD (OrD (VarD 12) (NotD (VarD 14))) (ImpD (VarD 3) (VarD
20)))) (AndD (ImpD (EquivD (VarD 27) (VarD 7)) (EquivD (NotD (VarD 2)) (VarD 1))) (AndD (OrD (NotD (VarD 10)) (VarD 14))
(EquivD (NotD (VarD 11)) (NotD (VarD 9)))))) (ImpD (EquivD (AndD (EquivD (OrD (VarD 5) (VarD 7)) (EquivD (VarD 14)
(VarD 10))) (OrD (EquivD (NotD (VarD 15)) (VarD 9)) (EquivD (VarD 11) (VarD 3)))) (EquivD (AndD (AndD (VarD 27) (NotD
(VarD 30))) (ImpD (NotD (VarD 2)) (NotD (VarD 30)))) (ImpD (EquivD (NotD (VarD 4)) (VarD 13)) (OrD (VarD 24) (VarD 7)))))
(EquivD (AndD (EquivD (OrD (VarD 2) (VarD 19)) (EquivD (VarD 24) (VarD 3))) (ImpD (OrD (VarD 3) (VarD 8)) (ImpD (VarD 6)
(VarD 28)))) (AndD (AndD (OrD (NotD (VarD 30)) (VarD 8)) (EquivD (VarD 27) (VarD 27))) (ImpD (EquivD (VarD 14) (VarD 18))
(OrD (VarD 22) (NotD (VarD 26))))))))))))))))))))))))))))))))))))))))))))))))"
    unfolding DefD apply simp sledgehammer[z3,timeout=4]() oops —‹Proof by z3 in 4s›
end
```

Fig. 22. A valid, moderate size QBF formula, termed QBFmediumD, encoded in the deep embedding using ASCII prefix notation only.

Fig. 23. The formula QBFmediumD from Figure 22 as depicted by Isabelle/HOL after unfolding of definitions and simplification.

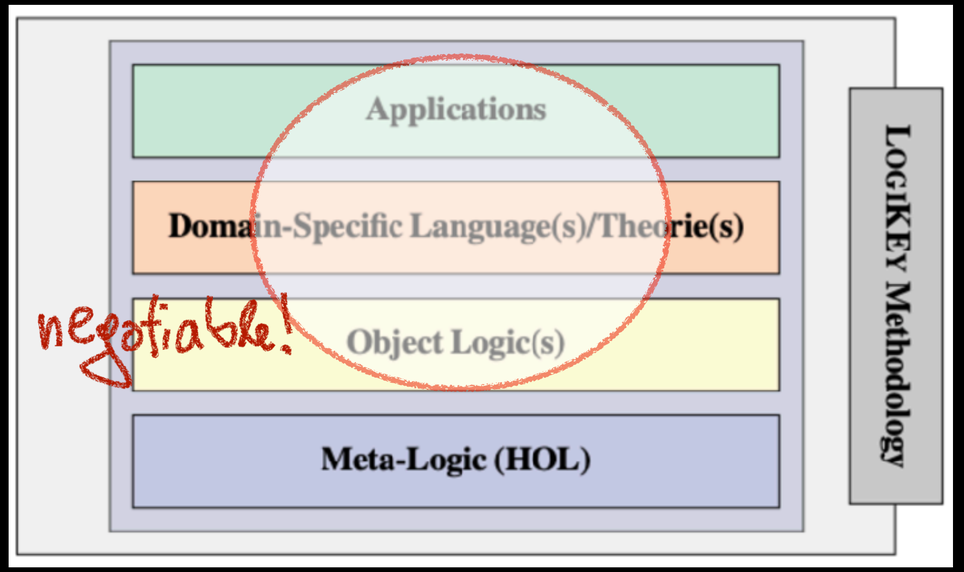
```
theory QBFinHOL_experiments_medium imports QBFinHOL_deep  QBFinHOL_shallow  QBFinHOL_shallow_minimal
begin
  abbreviation "EquivD φ ψ ≡  (AndD (ImpD φ ψ) (ImpD ψ φ))"
  lemma QBFmediumD: "
⊨d (AllD 30 (ExD 29 (AllD 28 (ExD 27 (AllD 26 (ExD 25 (AllD 24 (ExD 23 (AllD 22 (ExD 21 (AllD 20 (ExD 19 (AllD 18 (ExD 17
(AllD 16 (ExD 15 (AllD 14 (ExD 13 (AllD 12 (ExD 11 (AllD 10 (ExD 9 (AllD 8 (ExD 7 (AllD 6 (ExD 5 (AllD 4 (ExD 3 (AllD 2 (ExD 1
(AndD (EquivD (AndD (AndD (EquivD (OrD (EquivD (VarD 24) (VarD 21)) (EquivD (VarD 1) (VarD 3))) (OrD (OrD (VarD 21)
(NotD (VarD 16))) (OrD (VarD 16) (VarD 20)))) (ImpD (EquivD (EquivD (VarD 17) (NotD (VarD 15))) (EquivD (VarD 24)
(NotD (VarD 17)))) (ImpD (AndD (VarD 20) (VarD 19)) (EquivD (VarD 14) (VarD 1))))) (OrD (ImpD (AndD (ImpD (NotD (VarD 15))
(VarD 2)) (EquivD (VarD 24) (VarD 22))) (OrD (AndD (VarD 8) (VarD 7)) (EquivD (VarD 14) (NotD (VarD 14))))) (ImpD (EquivD
(ImpD (NotD (VarD 28)) (VarD 25)) (EquivD (VarD 19) (VarD 10))) (AndD (AndD (VarD 16) (NotD (VarD 25))) (EquivD (VarD 21)
(NotD (VarD 13))))))) (AndD (AndD (EquivD (EquivD (OrD (VarD 10) (VarD 23)) (OrD (NotD (VarD 3)) (VarD 22))) (EquivD (OrD
(VarD 12) (NotD (VarD 21))) (AndD (VarD 4) (VarD 2)))) (OrD (OrD (ImpD (VarD 23) (VarD 27)) (EquivD (VarD 26) (VarD 9)))
(AndD (ImpD (VarD 18) (NotD (VarD 8))) (AndD (VarD 19) (VarD 23))))) (EquivD (OrD (ImpD (ImpD (VarD 28) (VarD 2)) (EquivD
(NotD (VarD 6)) (VarD 6))) (ImpD (EquivD (VarD 26) (VarD 22)) (ImpD (NotD (VarD 25)) (VarD 30)))) (EquivD (EquivD (EquivD
(VarD 28) (VarD 12)) (OrD (VarD 17) (NotD (VarD 29)))) (EquivD (AndD (NotD (VarD 28)) (NotD (VarD 8))) (OrD (VarD 17)
(VarD 10))))))) (EquivD (OrD (ImpD (EquivD (OrD (EquivD (VarD 5) (VarD 6)) (ImpD (VarD 21) (VarD 16))) (EquivD (OrD
(VarD 25) (VarD 12)) (ImpD (NotD (VarD 13)) (VarD 15)))) (ImpD (EquivD (AndD (VarD 25) (NotD (VarD 4))) (EquivD (NotD
(VarD 26)) (VarD 26))) (ImpD (AndD (VarD 13) (NotD (VarD 11))) (ImpD (NotD (VarD 13)) (NotD (VarD 25)))))) (EquivD (AndD
(EquivD (AndD (VarD 12) (VarD 18)) (ImpD (VarD 4) (VarD 14))) (ImpD (OrD (VarD 12) (NotD (VarD 14))) (ImpD (VarD 3) (VarD
20)))) (AndD (ImpD (EquivD (VarD 27) (VarD 7)) (EquivD (NotD (VarD 2)) (VarD 1))) (AndD (OrD (NotD (VarD 10)) (VarD 14))
(EquivD (NotD (VarD 11)) (NotD (VarD 9))))))) (ImpD (EquivD (AndD (EquivD (OrD (VarD 5) (VarD 7)) (EquivD (VarD 14)
(VarD 10))) (OrD (EquivD (NotD (VarD 15)) (VarD 9)) (EquivD (VarD 11) (VarD 3)))) (EquivD (AndD (AndD (VarD 27) (NotD
(VarD 30))) (ImpD (NotD (VarD 2)) (NotD (VarD 30)))) (ImpD (EquivD (NotD (VarD 4)) (VarD 13)) (OrD (VarD 24) (VarD 7)))))
(EquivD (AndD (EquivD (OrD (VarD 2) (VarD 19)) (EquivD (VarD 24) (VarD 3))) (ImpD (OrD (VarD 3) (VarD 8)) (ImpD (VarD 6)
(VarD 28)))) (AndD (AndD (OrD (NotD (VarD 30)) (VarD 8)) (EquivD (VarD 27) (VarD 27))) (ImpD (EquivD (VarD 14) (VarD 18))
(OrD (VarD 22) (NotD (VarD 26))))))))))))))))))))))))))))))))))))))"
    unfolding DefD apply simp sledgehammer[z3,timeout=4]() oops —‹Proof by z3 in 4s›
end
```

Fig. 22. A valid, moderate size QBF formula, termed QBFmediumD, encoded in the deep embedding using ASCII prefix notation only.

```
; smt.random_seed=1 smt.refine_inj_axioms=false -T:30 -smt2
(set-option :produce-proofs true)
(set-logic AUFLIA)
(assert (! (not (forall ((?v0 Bool)) (exists ((?v1 Bool)) (forall ((?v2 Bool)) (exists ((?v3 Bool)) (forall ((?v4 Bool)) (exists ((?v5 Bool)) (forall ((?v6 Bool))
(exists ((?v7 Bool)) (forall ((?v8 Bool)) (exists ((?v9 Bool)) (forall ((?v10 Bool)) (exists ((?v11 Bool)) (forall ((?v12 Bool)) (exists ((?v13 Bool)) (forall ((?
v14 Bool)) (exists ((?v15 Bool)) (forall ((?v16 Bool)) (exists ((?v17 Bool)) (forall ((?v18 Bool)) (exists ((?v19 Bool)) (forall ((?v20 Bool)) (exists ((?v21
Bool)) (forall ((?v22 Bool)) (exists ((?v23 Bool)) (forall ((?v24 Bool)) (exists ((?v25 Bool)) (forall ((?v26 Bool)) (exists ((?v27 Bool)) (forall ((?v28 Bool))
(exists ((?v29 Bool)) (and (or (and (or (and ?v6 (not ?v9)) (and ?v9 (not ?v6))) (or (and ?v29 (not ?v27)) (and ?v27 (not ?v29)))) (or (and ?v11 (and ?
v10 (and (or (and ?v13 ?v15) (or (and (not ?v15) (not ?v13)) (and (=> ?v6 (not ?v13)) (or ?v13 ?v6)))) (and (or (and ?v6 ?v13) (or (and (not ?v13)
(not ?v6)) (and (=> ?v13 (not ?v15)) (or ?v15 ?v13)))) (or (and ?v16 (not ?v29)) (and ?v29 (not ?v16))))))) (or (and (or ?v15 ?v28) (and (=> ?v6 ?v8)
(and (=> ?v8 ?v6) (and (=> ?v22 (not ?v23)) (and (or (and (not ?v2) (not ?v5)) (and (=> ?v11 ?v20) (=> ?v20 ?v11))) (and (or (and ?v11 (not ?v20)) (or
(and ?v20 (not ?v11)) (or ?v2 ?v5))) (=> ?v14 (or ?v5 (or (and ?v9 ?v17) (and (not ?v17) (not ?v9)))))))))) (and (or (and ?v27 (and (or ?v20 ?v7) (not ?
v8))) (or (and (=> ?v27 ?v8) (and (not ?v20) (not ?v7))) (and (or (and (not ?v18) ?v9) (and ?v26 ?v28)) (=> (and ?v28 ?v26) (or ?v18 (not ?v9)))))) (and
(or (and (or ?v18 (not ?v9)) (=> ?v26 (not ?v28))) (or (and ?v28 (and ?v26 (and (not ?v18) ?v9))) (and (=> ?v27 (or (and (not ?v20) (not ?v7)) ?v8)) (or
(and ?v27 (not ?v8)) (or ?v20 ?v7))))) (and (=> ?v7 (or ?v3 (or (and (=> ?v4 ?v21) (=> ?v21 ?v4)) (and ?v11 (=> ?v12 (not ?v22)))))) (and (or (and
(=> ?v2 ?v28) (and (=> ?v4 ?v8) (and (=> ?v8 ?v4) (and (not ?v5) (not ?v0))))) (and (or (and (=> ?v2 ?v18) (and (=> ?v18 ?v2) (and (not ?v13) ?v1)))
(or (and (or ?v13 (not ?v1)) (or (and ?v2 (not ?v18)) (and ?v18 (not ?v2)))) (and (or ?v2 (or ?v22 (or ?v13 ?v20))) (or (and (not ?v13) (not ?v20)) (and
(not ?v2) (not ?v22)))))) (or (and (not ?v2) (and (not ?v22) (and (not ?v13) (not ?v20)))) (or (and (or ?v13 ?v20) (or ?v2 ?v22)) (and (or (and ?v2 (not ?
v18)) (or (and ?v18 (not ?v2)) (or ?v13 (not ?v1)))) (or (and (not ?v13) ?v1) (and (=> ?v2 ?v18) (=> ?v18 ?v2))))))) (or (and (or (and ?v2 (not ?v18)) (or
(and ?v18 (not ?v2)) (or ?v13 (not ?v1)))) (and (or (and (not ?v13) ?v1) (and (=> ?v2 ?v18) (=> ?v18 ?v2))) (or (and (not ?v2) (and (not ?v22) (and
(not ?v13) (not ?v20)))) (and (or ?v13 ?v20) (or ?v2 ?v22))))) (or (and (or ?v2 (or ?v22 (or ?v13 ?v20))) (and (or (and (not ?v13) (not ?v20)) (and (not ?
v2) (not ?v22))) (or (and (=> ?v2 ?v18) (and (=> ?v18 ?v2) (and (not ?v13) ?v1))) (and (or ?v13 (not ?v1)) (or (and ?v2 (not ?v18)) (and ?v18 (not ?
v2))))))) (or (and ?v2 (not ?v28)) (or (and ?v4 (not ?v8)) (or (and ?v8 (not ?v4)) (or ?v5 ?v0))))))))))))) (and (or (and (=> ?v27 (or (and (not ?v20) (not ?
v7)) ?v8)) (and (or (and ?v27 (not ?v8)) (or ?v20 ?v7)) (or (and (or ?v18 (not ?v9)) (=> ?v26 (not ?v28))) (and ?v28 (and ?v26 (and (not ?v18) ?v9))))))
(or (and (or (and (not ?v18) ?v9) (and ?v26 ?v28)) (and (=> (and ?v26 ?v28) (or ?v18 (not ?v9))) (or (and ?v27 (and (or ?v20 ?v7) (not ?v8))) (and
(=> ?v27 ?v8) (and (not ?v20) (not ?v7)))))) (or (and ?v7 (and (not ?v3) (and (or (and ?v4 (not ?v21)) (and ?v21 (not ?v4))) (=> ?v11 (and ?v12 ?
v22))))) (or (and (or (and ?v2 (not ?v28)) (or (and ?v4 (not ?v8)) (or (and ?v8 (not ?v4)) (or ?v5 ?v0)))) (or (and (or (and ?v2 (not ?v18)) (or (and ?v18
(not ?v2)) (or ?v13 (not ?v1)))) (and (or (and (not ?v13) ?v1) (and (=> ?v2 ?v18) (=> ?v18 ?v2))) (or (and (not ?v2) (and (not ?v22) (and (not ?v13)
(not ?v20)))) (and (or ?v13 ?v20) (or ?v2 ?v22))))) (and (or ?v2 (or ?v22 (or ?v13 ?v20))) (and (or (and (not ?v13) (not ?v20)) (and (not ?v2) (not ?
v22))) (or (and (=> ?v2 ?v18) (and (=> ?v18 ?v2) (and (not ?v13) ?v1))) (and (or ?v13 (not ?v1)) (or (and ?v2 (not ?v18)) (and ?v18 (not ?v2)))))))) (or
(and (or (and (=> ?v2 ?v18) (and (=> ?v18 ?v2) (and (not ?v13) ?v1))) (or (and (or ?v13 (not ?v1)) (or (and ?v2 (not ?v18)) (and ?v18 (not ?v2)))) (and
(or ?v2 (or ?v22 (or ?v13 ?v20))) (or (and (not ?v13) (not ?v20)) (and (not ?v2) (not ?v22)))))) (and (or (and (not ?v2) (and (not ?v22) (and (not ?v13)
(not ?v20)))) (or (and (or ?v13 ?v20) (or ?v2 ?v22)) (and (or (and ?v2 (not ?v18)) (or (and ?v18 (not ?v2)) (or ?v13 (not ?v1)))) (or (and (not ?v13) ?v1)
(and (=> ?v2 ?v18) (=> ?v18 ?v2)))))) (and (=> ?v2 ?v28) (and (=> ?v4 ?v8) (and (=> ?v8 ?v4) (and (not ?v5) (not ?v0))))))) (and (or (and (=> ?v6 ?v9)
(=> ?v9 ?v6)) (and (=> ?v29 ?v27) (=> ?v27 ?v29))) (and (=> (and ?v11 ?v10) (or (and (=> ?v13 (not ?v15)) (and (or ?v15 ?v13) (or (and ?v6 ?v13)
(and (not ?v13) (not ?v6))))) (or (and (=> ?v6 (not ?v13)) (and (or ?v13 ?v6) (or (and ?v13 ?v15) (and (not ?v15) (not ?v13))))) (and (=> ?v16 ?v29)
(=> ?v29 ?v16))))) (or (and (not ?v15) (not ?v28)) (or (and ?v6 (not ?v8)) (or (and ?v8 (not ?v6)) (or (and ?v22 ?v23) (or (and (or ?v2 ?v5) (or (and ?
v11 (not ?v20)) (and ?v20 (not ?v11)))) (or (and (=> ?v11 ?v20) (and (=> ?v20 ?v11) (and (not ?v2) (not ?v5)))) (and ?v14 (and (not ?v5) (and (=> ?v9
(not ?v17)) (or ?v17 ?v9))))))))))))))))) (or (and (or (and (not ?v25) (not ?v23)) (and (=> ?v16 ?v20) (=> ?v20 ?v16))) (and (or (and ?v16 (not ?v20)) (or
(and ?v20 (not ?v16)) (or ?v25 ?v23))) (and (or (and (or ?v15 ?v21) (=> ?v21 (not ?v15))) (and (=> ?v19 ?v27) (=> ?v27 ?v19))) (or (and ?v3 (and
(not ?v0) (and (or ?v28 (not ?v0)) (and (or ?v26 ?v17) (and (=> ?v17 (not ?v26)) (and (not ?v6) (not ?v23))))))) (and (or (and (not ?v26) (not ?v17)) (or
(and ?v17 ?v26) (or ?v6 ?v23))) (=> ?v3 (or ?v0 (and (not ?v28) ?v0)))))))) (or (and (=> ?v3 (or ?v0 (or (and (not ?v28) ?v0) (or (and (not ?v26) (not ?
v17)) (or (and ?v17 ?v26) (or ?v6 ?v23)))))) (and (or (and (or ?v26 ?v17) (and (=> ?v17 (not ?v26)) (and (not ?v6) (not ?v23)))) (and ?v3 (and (not ?v0)
(or ?v28 (not ?v0)))))) (or (and (or ?v25 ?v23) (or (and ?v16 (not ?v20)) (and ?v20 (not ?v16)))) (or (and (=> ?v16 ?v20) (and (=> ?v20 ?v16) (and
(not ?v25) (not ?v23)))) (and (or (and (not ?v15) (not ?v21)) (and ?v21 ?v15)) (or (and ?v19 (not ?v27)) (and ?v27 (not ?v19)))))))) (and (or (and (or ?
v28 ?v11) (or (and ?v6 (not ?v27)) (and ?v27 (not ?v6)))) (or (and (=> ?v6 ?v27) (and (=> ?v27 ?v6) (and (not ?v28) (not ?v11)))) (or (and ?v24 (and
(or ?v27 ?v22) (not ?v2))) (and (=> ?v0 ?v22) (or (and ?v16 (not ?v12)) (or (and ?v12 (not ?v16)) (or ?v8 (not ?v4)))))))) (or (and ?v0 (not ?v22)) (or
(and (=> ?v16 ?v12) (and (=> ?v12 ?v16) (and (not ?v8) ?v4))) (and (or (and (not ?v28) (not ?v11)) (and (=> ?v6 ?v27) (=> ?v27 ?v6))) (and (or (and ?
v6 (not ?v27)) (or (and ?v27 (not ?v6)) (or ?v28 ?v11))) (=> ?v24 (or (and (not ?v27) (not ?v22)) ?v2)))))))))))))))))))))))))))))))))))))))))))) :named
conjecture0))
(check-sat)
(get-proof)
```

Fig. 24. Proof input file for z3 as generated by sledgehemmer (in overlord mode) for the simplified formula QBFmediumD from Figure 22.

Discussion: LogiKEy — Logical Pluralism in Proof Assistant Systems —



- supports the exploration of (expressive) **logics & logic combinations**
- supports the exploration & verification of **meta-logical properties**
- enables the **simultaneous use** of deep & shallow embeddings
- enables **integration of generic and specialist tools** (opportunities for further work)
- has **innovative interdisciplinary applications** (philosophy, law & ethics, maths, …)
- creates **uniform proofs** at the meta-level (after unfolding)
- …
- supports **uniform logic education** (since 2016 at FU Berlin & U Bamberg)

Reading

(preprints: http://christoph-benzmueller.de/publications.html)

HOL (see also the papers of Church, Henkin, Andrews and Muskens in JSL as mentioned before)

- Benzmüller, Brown, Kohlhase (2004), Higher-Order Semantics and Extensionality, J. Symb. Log., http://doi.org/10.2178/jsl/1102022211
- Benzmüller & Miller (2014), Automation of Higher-Order Logic, Handbook of the History of Logic, http://doi.org/10.1016/B978-0-444-51624-4.50005-8
- Benzmüller & Andrews (2019), Church's Type Theory, SEP, https://plato.stanford.edu/entries/type-theory-church/

Leo Provers

- Steen & Benzmüller (2021), Extensional Higher-Order Paramodulation in Leo-III, J. Autom. Reason., http://doi.org/10.1007/s10817-021-09588-x
- Benzmüller, Sultana, Paulson (2015), Thhe Higher-Order Prover Leo-II, J. Autom. Reason., http://doi.org/10.1007/s10817-015-9348-y
- Benzmüller & Kohlhase (1998), LEO – A Higher-Order Theorem Prover, CADE, http://doi.org/10.1007/BFb0054256

LogiKEy & Universal (Meta-)Logical Reasoning

- Benzmüller (2019), **Universal (meta-)logical reasoning: Recent successes.** Sci. Comp. Progr. (http://doi.org/10.1016/j.scico.2018.10.008)
- __, Parent & van der Torre (2020), **Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support.** Artificial Intelligence. (https://doi.org/10.1016/j.artint.2020.103348)
- Benzmüller et.al. (2020), **LogiKEy Workbench: Deontic Logics, Logic Combinations and Expressive Ethical and Legal Reasoning (Isabelle/HOL Dataset)**. Data in Brief (https://doi.org/10.1016/j.dib.2020.106409)
- Bibel (2022), **Computer Kreiert Wissenschaft**. Informatik Spektrum. (https://doi.org/10.1007/s00287-022-01456-1)

Reading (cont'd)

(preprints: http://christoph-benzmueller.de/publications.html)

Foundational Studies in Metaphysics

- Kirchner, Benzmüller & Zalta (2019), **Computer Science and Metaphysics: A Cross-Fertilization**. Open Philosophy (http://doi.org/10.1515/opphil-2019-0015)
- __ (2020), **Mechanizing Principia Logico-Metaphysica in Functional Type Theory**. Review of Symbolic Logic (https://doi.org/10.1017/S1755020319000297)
- Kirchner (2021). **Embedding of Abstract Object Theory in Isabelle/HOL.** Full sources. (https://github.com/ekpyron/AOT/tree/dissertation)
- __ (2022), **Computer-Verified Foundations of Metaphysics and an Ontology of Natural Numbers in Isabelle/HOL**. PhD thesis, Dep. of Maths and CS, FU Berlin.

Foundational Studies in Ethics & Law

- Fuenmayor & Benzmüller (2018), **Formalisation and Evaluation of Alan Gewirth's Proof for the Principle of Generic Consistency in Isabelle/HOL**. (https://www.isa-afp.org/entries/GewirthPGCProof.html)
- __ (2019), **Harnessing Higher-Order (Meta-)Logic to Represent and Reason with Complex Ethical Theories**. PRICAI 2019 (https://doi.org/10.1007/978-3-030-29908-8_34)
- Benzmüller & Fuenmayor (2021), **Value-oriented Legal Argumentation in Isabelle/HOL**. ITP 2021 (https://doi.org/10.4230/LIPIcs.ITP.2021.7)
- __, Fuenmayor & Lomfeld (2022), **Modelling Value-oriented Legal Reasoning in LogiKEy**. (https://arxiv.org/abs/2006.12789)

Studies on the Ontological Argument

- Benzmüller & Woltzenlogel Paleo (2014), **Automating Gödel's Ontological Proof of God's Existence with Higher-order Automated Theorem Provers.** ECAI 2014 (http://doi.org/10.3233/978-1-61499-419-0-93)
- __ (2016), **The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics.** IJCAI 2016 (http://www.ijcai.org/Proceedings/16/Papers/137.pdf)

Reading (cont'd)

(preprints: http://christoph-benzmueller.de/publications.html)

Studies on the Ontological Argument (cont'd)

- Benzmüller & Woltzenlogel Paleo (2016), **An Object-Logic Explanation for the Inconsistency in Gödel's Ontological Theory** (Extended Abstract, Sister Conferences). KI 2016 (http://doi.org/10.1007/978-3-319-46073-4)
- __ (2017). **Computer-Assisted Analysis of the Anderson-Hájek Controversy.** Logica Universalis (http://doi.org/10.1007/s11787-017-0160-9)
- Fuenmayor & Benzmüller (2017), **Automating Emendations of the Ontological Argument in Intensional Higher-Order Modal Logic.** KI 2017 (http://doi.org/10.1007/978-3-319-67190-1_9)
- __ (2018). **A Case Study On Computational Hermeneutics: E. J. Lowe's Modal Ontological Argument.** IfCoLoG Journal of Logics and their Applications (https://www.researchgate.net/publication/333804824)
- Benzmüller & Fuenmayor (2020), **Computer-supported Analysis of Positive Properties, Ultrafilters and Modal Collapse in Variants of Gödel's Ontological Argument.** Bulletin of the Section of Logic (http://doi.org/10.18778/0138-0680.2020.08)
- Benzmüller (2020), **A (Simplified) Supreme Being Necessarily Exists, says the Computer: Computationally Explored Variants of Gödel's Ontological Argument.** KR 2020 (https://doi.org/10.24963/kr.2020/80)
- __ (2022), **A Simplified Variant of Gödel's Ontological Argument.** To appear (https://www.researchgate.net/publication/358607847)

Isabelle/HOL:

- Website: https://isabelle.in.tum.de
- Documentation: https://isabelle.in.tum.de/documentation.html
- Nipkow, Paulson & Wenzel (2002), **Isabelle/HOL: A Proof Assistant for Higher-Order Logic.** Springer (https://doi.org/10.1007/3-540-45949-9)
- Blanchette, Bulwahn & Tobias Nipkow (2011), **Automatic Proof and Disproof in Isabelle/HOL**. FroCoS 2011 (https://doi.org/10.1007/978-3-642-24364-6_2)

Reading

(preprints: http://christoph-benzmueller.de/publications.html)

Free Logic & Axiomatic Category Theory in Free Logic

- Scott (1967), **Existence and description in formal logic.** Reprinted in: Lambert (1991), Philosophical Application of Free Logic, OUP. (ISBN 0195061314)
- Scott (1977), **Identity and existence in intuitionistic logic.** Applications of Sheaves, Springer (https://doi.org/10.1007/BFb0061839)
- Benzmüller & Scott (2016), **Automating Free Logic in Isabelle/HOL.** ICMS 2016 (https://doi.org/10.1007/978-3-319-42432-3_6)
- __ (2018), **Axiom systems for category theory in free logic**. Archive of Formal Proofs (https://www.isa-afp.org/entries/AxiomaticCategoryTheory.html)
- __ (2020), **Automating Free Logic in HOL, with an Experimental Application in Category Theory.** JAR (http://doi.org/10.1007/s10817-018-09507-7)
- Tiemens, Scott, Benzmüller & Benda (2020), **Computer-supported Exploration of a Categorical Axiomatization of Modeloids.** RAMiCS 2020 (https://doi.org/10.1007/978-3-030-43520-2_19)
- Makarenko & Benzmüller (2020), **Positive Free Higher-Order Logic and its Automation via a Semantical Embedding**. KI 2020 (http://doi.org/10.1007/978-3-030-58285-2_9)
- Bayer (2021), **Exploring categories, formally**. BSc Thesis, Dep. of Maths and CS, FU Berlin.
- Gonus (2021), **Categorical semantics of Intuitionistic Multiplicative Linear Logic and its formalization in Isabelle/HOL**. MSc Thesis, Dep. of Maths and CS, FU Berlin.