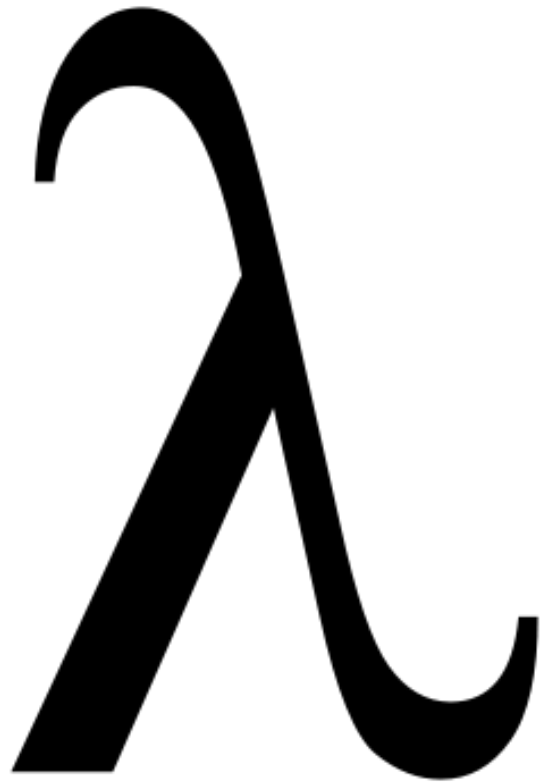


Higher-order Logic (HOL)

Classical Higher-Order Logic (HOL)

Expressivity	FOL	HOL	Example
Quantification over			
- Individuals	✓	✓	$\forall x\, p(f(x))$
- Functions	-	✓	$\forall f\, p(f(a))$
- Predicates/Sets/Rels	-	✓	$\forall p\, p(f(a))$
Unnamed			
- Functions	-	✓	$(\lambda x\, x)$
- Predicates/Sets/Rels	-	✓	$(\lambda x\, x \neq a)$
Statements about			
- Functions	-	✓	$continuous(\lambda x\, x)$
- Predicates/Sets/Rels	-	✓	$reflexive(=)$
Powerful abbreviations	-	✓	$reflexive = \lambda r\, \forall x\, r(x, x)$

Classical Higher-Order Logic (HOL)

Expressivity	FOL	HOL	Example
Quantification over			
- Individuals	✓	✓	$\forall x_\iota\, p_{\iota\to o}(f_{\iota\to\iota}(x_\iota))$
- Functions	-	✓	$\forall f_{\iota\to\iota}\, p_{\iota\to o}(f_{\iota\to o}(a_\iota))$
- Predicates/Sets/Rels	-	✓	$\forall p_{\iota\to o}\, p_{\iota\to o}(f_{\iota\to\iota}(a_\iota))$
Unnamed			
- Functions	-	✓	$(\lambda x_\iota\, x_\iota)$
- Predicates/Sets/Rels	-	✓	$(\lambda x_{\iota\to\iota}\, x_{\iota\to\iota} \neq_{\iota\to\iota\to p} a)_\iota)$
Statements about			
- Functions	-	✓	$continuous_{(\iota\to\iota)\to o}(\lambda x_\iota\, x_\iota)$
- Predicates/Sets/Rels	-	✓	$reflexive_{(\iota\to\iota\to o)\to o}(=_{\iota\to\iota\to o})$
Powerful abbreviations	-	✓	$reflexive_{(\iota\to\iota\to o)\to o} = \lambda r_{(\iota\to\iota\to o)}\, \forall x_\iota\, r(x,x)$

Simple Types: Prevent Some Paradoxes and Inconsistencies

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x\neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x\neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad \boxed{P_\alpha} \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$

$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x\neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid \boxed{(\lambda x_\alpha\, s_\beta)_{\alpha\to\beta}} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$

$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x\neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid \boxed{(s_{\alpha\to\beta}\, t_\alpha)_\beta} \mid$$

$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$

$$\boxed{(\neg_{o\to o}\, s_o)} \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$

$$(\neg_{o\to o}\, s_o) \mid \boxed{((\vee_{o\to o\to o}\, s_o)\, t_o)} \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction (we may use infix notation, omit types and brackets: $s \vee t$)
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$

$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \boxed{\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)}$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification (we may use syntactical sugar: $\forall x_\alpha s$)

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists x s$ stands for $\neg\forall x \neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

Simple Types: $\alpha, \beta \quad ::= \quad \iota \mid o \mid (\alpha \to \beta)$

(we may add further base types, e.g. μ)

HOL Language:

$$s, t \quad ::= \quad P_\alpha \mid x_\alpha \mid (\lambda x_\alpha\, s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o) \mid ((\vee_{o\to o\to o}\, s_o)\, t_o) \mid \forall_{(\alpha\to o)\to o}(\lambda x_\alpha\, s_o)$$

constant symbols
variable symbols
lambda abstraction
application
negation
disjunction
universal quantification

Terms of type o: formulas

Other logical connectives can be defined, e.g. $\exists xs$ stands for $\neg\forall x\neg s$

Equality may also be defined: $s \doteq t$ stands for $\forall P(Ps \Rightarrow Pt)$
(but it is stongly recommended to add primitive equality!)

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Syntax

α-conversion

is considered implicitly, e.g. $(\lambda x\, Px)$ is identified with $(\lambda y\, Py)$.

Substitution ($[s/x]t$)

of a term s_α for x_α in t_β is denoted by $[s/x]t$. We assume the bound variables of t avoid variable capture: $[y/x](\lambda y\, .Ryx)$ returns $(\lambda z\, .Rzy)$ and not $(\lambda y\, .Ryy)$

β-reduction and η-reduction

β-redex has the form $(\lambda x\, s)t$ and β-reduces to $[t/x]s$.

$$(\lambda x\, s)t \longrightarrow_\beta [t/x]s$$

η-redex has the form $(\lambda x\, sx)$ where x is not free in s; it η-reduces to s.

$$(\lambda x\, sx) \longrightarrow_\eta s \qquad (x \text{ is not free in } s)$$

$s \equiv_\beta t$ means s can be converted to t by β-reductions and expansions.
$s \equiv_{\beta\eta} t$ means s can be converted to t using both β and η.

Unique normalforms exist

For each $s_\alpha \in \mathcal{HOL}$ there is a unique β-normal form and a unique $\beta\eta$-normal form.

HOL: Semantics

Semantics of HOL is (meanwhile) well understood

- Origin [Church, J.Symb.Log., 1940]
- Henkin-Semantics [Henkin, J.Symb.Log., 1950]
  [Andrews, J.Symb.Log., 1971, 1972]
- Extensionality/Intensionality [BenzmüllerEtAl., J.Symb.Log., 2004]
  [Muskens, J.Symb.Log., 2007]

HOL with Henkin-Semantics:
semi-decidable & compact & existence of countable models (like FOL)

Recommended Reading: [BenzmüllerMiller, HandbookHistoryOfLogic, Vol.9, 2014]

HOL: Semantics

A Frame D
is a collection $\{D_\alpha\}_{\alpha \in \mathrm{T}}$ of nonempty sets D_α, such that

- D_ι can be chosen freely
- $D_o = \{T, F\}$ (for truth and falsehood), and
- $D_{\alpha\to\beta}$ are collections of functions mapping D_α into D_β

A Model M
for HOL is a tuple $M = \langle D, I\rangle$, where

- D is a frame;
- I is a family of typed interpretation functions mapping constant symbols p_α to appropriate elements of D_α, called the denotation of p_α;
- the logical connectives $\neg$, $\vee$, and $\forall$ are always given the standard denotations;
- moreover, we assume that the domains $D_{\alpha\to\alpha\to o}$ contain the respective identity relations.

A Frame D

is a collection $\{D_\alpha\}_{\alpha \in \mathrm{T}}$ of nonempty sets D_α, such that

- D_ι can be chosen freely
- $D_o = \{T, F\}$ (for truth and falsehood), and
- $D_{\alpha\to\beta}$ are collections of functions mapping D_α into D_β

A Model M

for HOL is a tuple $\boldsymbol{M = \langle D, I \rangle}$, where

- $\boldsymbol{D}$ is a frame;
- $\boldsymbol{I}$ is a family of typed interpretation functions mapping constant symbols $\boldsymbol{p_\alpha}$ to appropriate elements of $\boldsymbol{D_\alpha}$, called the denotation of $\boldsymbol{p_\alpha}$;
- the logical connectives $\neg$, $\vee$, and $\forall$ are always given the standard denotations;
- moreover, we assume that the domains $\boldsymbol{D_{\alpha\to\alpha\to o}}$ contain the respective identity relations.

Standard and Henkin Models

In a standard model $\boldsymbol{M = \langle D, I \rangle}$ we have

- $\boldsymbol{D_{\alpha\to\beta}} = \{f \mid f : D_\alpha \longrightarrow D_\beta\}$ (for all types α, β)

In a Henkin model $\boldsymbol{M = \langle D, I \rangle}$ we only require

- $\boldsymbol{D_{\alpha\to\beta}} \subseteq \{f \mid f : \boldsymbol{D_\alpha} \longrightarrow \boldsymbol{D_\beta}\}$ (for all types α, β)
- the valuation function $\|\cdot\|^{M,g}$ from above is total (every term denotes)

Any standard model is obviously also a Henkin model.
We consider Henkin models in the remainder (for Henkin semantics we have semi-decidability, compactness and existence of countable models).

A Remark on Equality

Primitive equality is not needed in HOL, since equality can be defined:

$\doteq \quad := \lambda x_\alpha \lambda y_\alpha \forall p_{\alpha\to o}. px \to py$ (Leibniz)

$\doteqdot \quad := \lambda x_\alpha \lambda y_\alpha \forall q_{\alpha\to\alpha\to o}. \forall z_\alpha (qzz) \to qxy$ (Andrews)

For theoretical and pragmatic reasons the use of primitive equality is nevertheless highly recommended.

Standard and Henkin Models

In a standard model $\boldsymbol{M = \langle D, I \rangle}$ we have

- $\boldsymbol{D_{\alpha\to\beta}} = \{f \mid f : D_\alpha \longrightarrow D_\beta\}$ (for all types α, β)

In a Henkin model $\boldsymbol{M = \langle D, I \rangle}$ we only require

- $\boldsymbol{D_{\alpha\to\beta}} \subseteq \{f \mid f : \boldsymbol{D_\alpha} \longrightarrow \boldsymbol{D_\beta}\}$ (for all types α, β)
- the valuation function $\|\cdot\|^{M,g}$ from above is total (every term denotes)

Any standard model is obviously also a Henkin model.
We consider Henkin models in the remainder (for Henkin semantics we have semi-decidability, compactness and existence of countable models).

A Remark on Equality

Primitive equality is not needed in HOL, since equality can be defined:

$\doteq \quad := \lambda x_\alpha \lambda y_\alpha \forall p_{\alpha\to o}. px \to py$ (Leibniz)

$\doteqdot \quad := \lambda x_\alpha \lambda y_\alpha \forall q_{\alpha\to\alpha\to o}. \forall z_\alpha (qzz) \to qxy$ (Andrews)

For theoretical and pragmatic reasons the use of primitive equality is nevertheless highly recommended.

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I\rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g} =$ the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $||(\neg_{o\to o}\ s_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = F$
6. $||((\vee_{o\to o\to o}\ s_o)\ t_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = T$ or $||t_o||^{M,g} = T$
7. $||(\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\ s_o))_o||^{M,g} = T$ iff for all $d \in D_\alpha$ we have $||s_o||^{M,g[d/x_\alpha]} = T$

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I \rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g} =$ the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $||(\neg_{o\to o}\ s_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = F$
6. $||((\vee_{o\to o\to o}\ s_o)\ t_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = T$ or $||t_o||^{M,g} = T$
7. $||(\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\ s_o))_o||^{M,g} = T$ iff for all $d \in D_\alpha$ we have $||s_o||^{M,g[d/x_\alpha]} = T$

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I \rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g}$ = the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $||(\neg_{o\to o}\ s_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = F$
6. $||((\vee_{o\to o\to o}\ s_o)\ t_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = T$ or $||t_o||^{M,g} = T$
7. $||(\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\ s_o))_o||^{M,g} = T$ iff for all $d \in D_\alpha$ we have $||s_o||^{M,g[d/x_\alpha]} = T$

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I \rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g}$ = the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $I(\neg_{o\to o})$ = not $\in D_{o\to o}$ such that: not$(F) = T$ and not$(T) = F$
6. $||((\vee_{o\to o\to o}\ s_o)\ t_o)_o||^{M,g} = T$ iff $||s_o||^{M,g} = T$ or $||t_o||^{M,g} = T$
7. $||(\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\ s_o))_o||^{M,g} = T$ iff for all $d \in D_\alpha$ we have $||s_o||^{M,g[d/x_\alpha]} = T$

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I \rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g}$ = the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $I(\neg_{o\to o})$ = not $\in D_{o\to o}$ such that: not$(F) = T$ and not$(T) = F$
6. $I(\vee_{o\to o\to o})$ = or $\in D_{o\to o\to o}$ such that: or$(a, b) = T$ iff $a = T$ or $b = T$
7. $||(\forall_{(\alpha\to o)\to o}(\lambda x_\alpha\ s_o))_o||^{M,g} = T$ iff for all $d \in D_\alpha$ we have $||s_o||^{M,g[d/x_\alpha]} = T$

Variable Assignment

A variable assignment g maps variables x_α to elements in D_α.
$g[d/W]$ denotes the assignment that is identical to g, except for variable W, which is now mapped to d.

Interpretation/Value of a HOL term

The value $||s_\alpha||^{M,g}$ of a HOL term s_α on a model $M = \langle D, I \rangle$ under assignment g is an element $d \in D_\alpha$ defined in the following way:

1. $||P_\alpha||^{M,g} = I(P_\alpha)$
2. $||x_\alpha||^{M,g} = g(x_\alpha)$
3. $||(s_{\alpha\to\beta}\ t_\alpha)_\beta||^{M,g} = ||s_{\alpha\to\beta}||^{M,g}(||t_\alpha||^{M,g})$
4. $||(\lambda x_\alpha\ s_\beta)_{\alpha\to\beta}||^{M,g}$ = the function f from D_α to D_β such that $f(d) = ||s_\beta||^{M,g[d/x_\alpha]}$ for all $d \in D_\alpha$
5. $I(\neg_{o\to o})$ = not $\in D_{o\to o}$ such that: not$(F) = T$ and not$(T) = F$
6. $I(\vee_{o\to o\to o})$ = or $\in D_{o\to o\to o}$ such that: or$(a, b) = T$ iff $a = T$ or $b = T$
7. $I(\forall_{(\alpha\to o)\to o})$ = All $\in D_{(\alpha\to o)\to o}$ such that: for $p \in D_{\alpha\to o}$ we have All$(p) = T$ iff $p(d) = T$ for all $d \in D_\alpha$

Truth in model / Validity

A formula s_o is true in model M under assignment g if and only if $\|s_o\|^{M,g} = T$; this is also denoted as $M, g \models s_o$.

A formula s_o is called valid in M if and only if $M, g \models s_o$ for all assignments g; this is also denoted as $M \models s_o$.

A formula s_o is called valid, which we denote by $\models s_o$, if and only if $M \models s_o$ for all M.

We define $\models \varphi$, where φ is a set of HOL formulas, if and only if $\models s$ for all $s \in \varphi$.

Logical Consequence

Let φ and ψ be set of HOL formulas. We define

$$\varphi \models \psi \qquad \text{(logical consequence)}$$

if and only if for each model M we have:

$$M \models \varphi \quad \text{implies} \quad M \models \psi$$

HOL: Semantics

Truth in model / Validity

A formula s_o is true in model M under assignment g if and only if $\|s_o\|^{M,g} = T$; this is also denoted as $M, g \models s_o$.

A formula s_o is called valid in M if and only if $M, g \models s_o$ for all assignments g; this is also denoted as $M \models s_o$.

A formula s_o is called valid, which we denote by $\models s_o$, if and only if $M \models s_o$ for all M.

We define $\models \varphi$, where φ is a set of HOL formulas, if and only if $\models s$ for all $s \in \varphi$.

Logical Consequence

Let φ and ψ be set of HOL formulas. We define

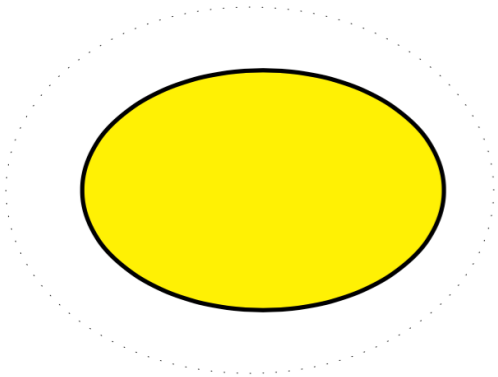
$$\varphi \models \psi \qquad \text{(logical consequence)}$$

if and only if for each model M we have:

$$M \models \varphi \quad \text{implies} \quad M \models \psi$$

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Standard Models $\mathfrak{ST}(\Sigma)$

- Idea of Standard Semantics:

$$\iota \longrightarrow \mathcal{D}_\iota \qquad \text{(choose)}$$

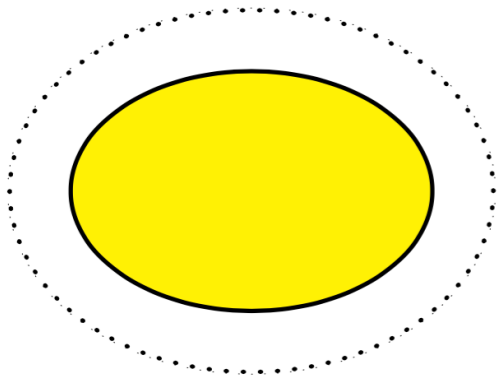
$$o \longrightarrow \mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\} \qquad \text{(fixed)}$$

$$(\alpha \to \beta) \longrightarrow$$

$$\mathcal{D}_{\alpha\to\beta} = \mathcal{F}(\mathcal{D}_\alpha, \mathcal{D}_\beta) \qquad \text{(fixed)}$$

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Standard Models $\mathfrak{ST}(\Sigma)$

- Idea of Standard Semantics:

$\iota \longrightarrow \mathcal{D}_\iota$ (choose)

$o \longrightarrow \mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$ (fixed)

$(\alpha \to \beta) \longrightarrow$

$\mathcal{D}_{\alpha\to\beta} = \mathcal{F}(\mathcal{D}_\alpha, \mathcal{D}_\beta)$ (fixed)

- Henkin's Generalization:

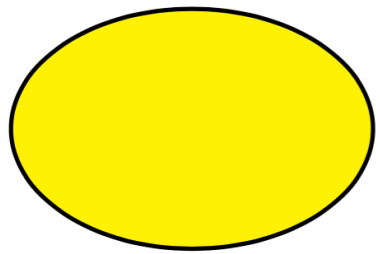
$\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha, \mathcal{D}_\beta)$ (choose)

but elements are still functions!

[Henkin-50]

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



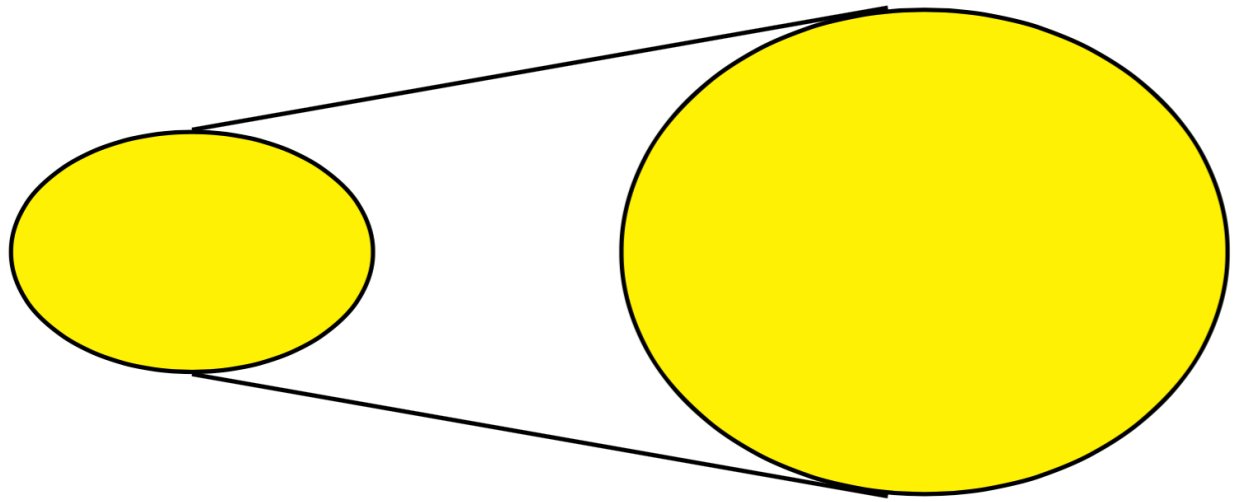
Standard Models $\mathfrak{ST}(\Sigma)$

choose: $\mathcal{D}_\iota$

fixed: $\mathcal{D}_o, \mathcal{D}_{\alpha\to\beta}$, functions

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Standard Models $\mathfrak{ST}(\Sigma)$

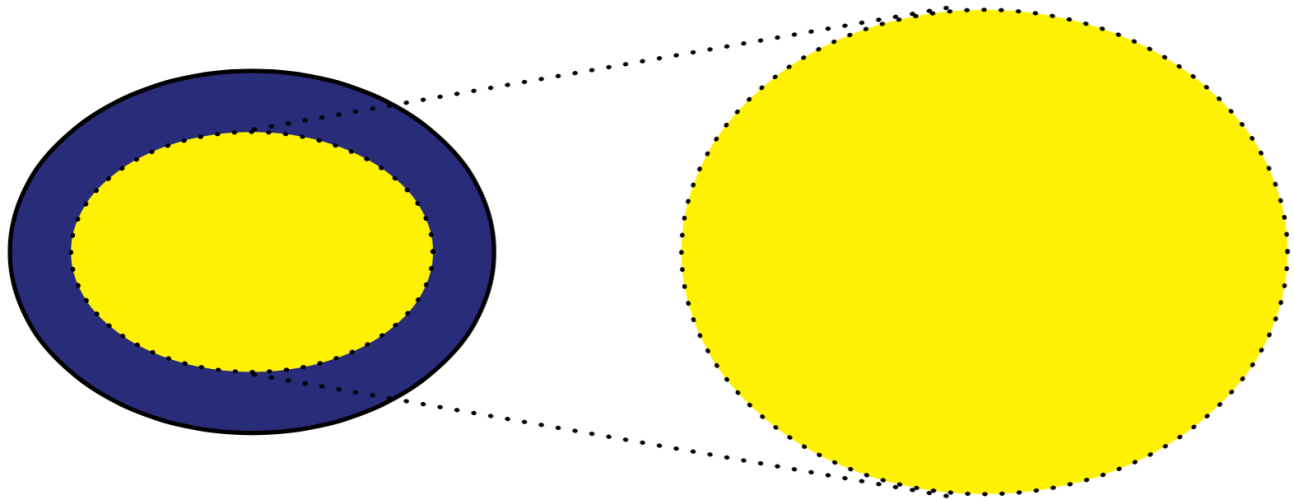
Formulas valid in $\mathfrak{ST}(\Sigma)$

choose: $\mathcal{D}_\iota$

fixed: $\mathcal{D}_o, \mathcal{D}_{\alpha\to\beta}$, functions

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Henkin Models $\mathfrak{H}(\Sigma) = \mathfrak{M}_{\beta\mathfrak{fb}}(\Sigma)$

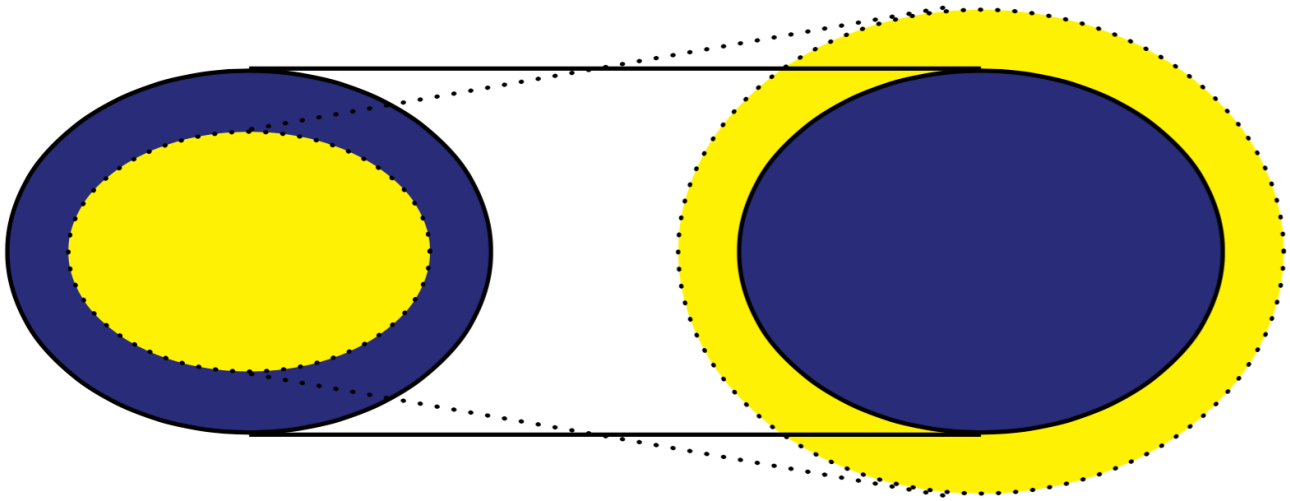
Formulas valid in $\mathfrak{M}_{\beta\mathfrak{fb}}(\Sigma)$?

choose: $\mathcal{D}_\iota, \mathcal{D}_{\alpha\to\beta}$

fixed: $\mathcal{D}_o$, functions

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Henkin Models $\mathfrak{H}(\Sigma) = \mathfrak{M}_{\beta\mathfrak{fb}}(\Sigma)$

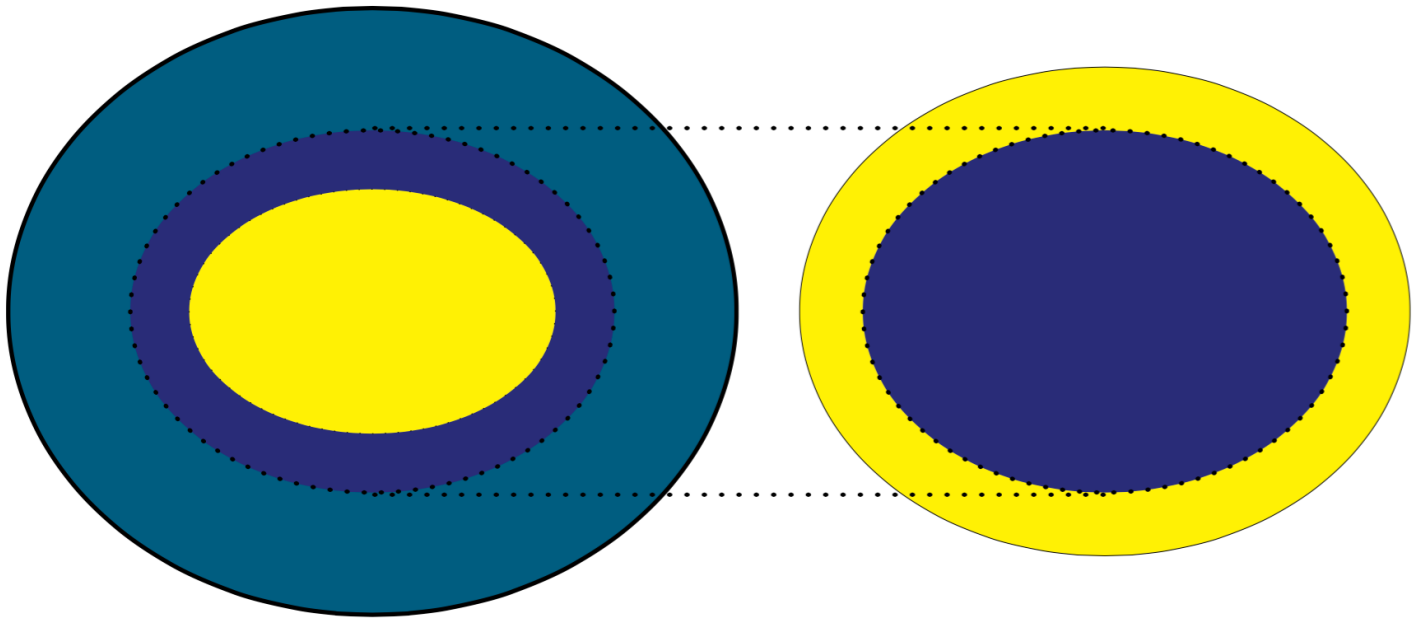
Formulas valid in $\mathfrak{M}_{\beta\mathfrak{fb}}(\Sigma)$

choose: $\mathcal{D}_\iota, \mathcal{D}_{\alpha\to\beta}$

fixed: $\mathcal{D}_o$, functions

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Non-Extensional Models $\mathfrak{M}_\beta(\Sigma)$

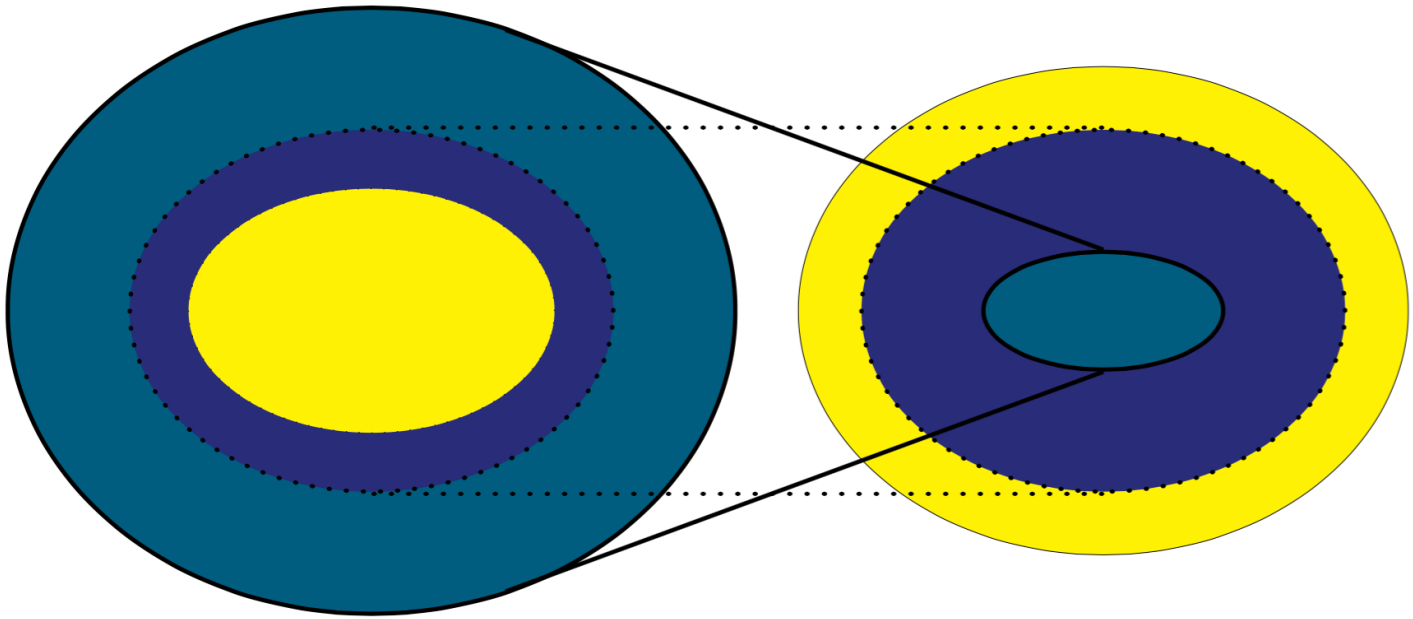
Formulas valid in $\mathfrak{M}_\beta(\Sigma)$?

choose: $\mathcal{D}_\iota, \mathcal{D}_{\alpha\to\beta}$, also non−functions, $\mathcal{D}_o$

fixed:

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



Non-Extensional Models $\mathfrak{M}_\beta(\Sigma)$

choose: $\mathcal{D}_\iota, \mathcal{D}_{\alpha\to\beta}$, also non−functions, $\mathcal{D}_o$

fixed:

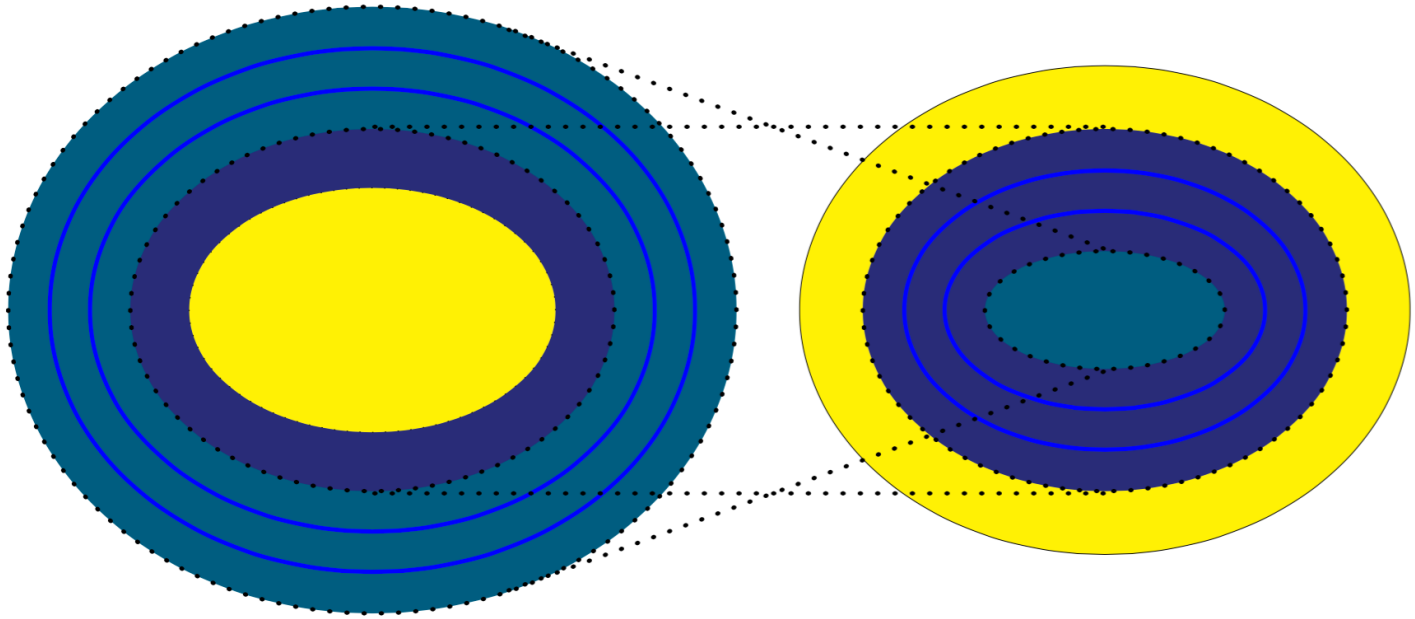
Formulas valid in $\mathfrak{M}_\beta(\Sigma)$?

Ex.: $\forall X.\forall Y.X \vee Y \Leftrightarrow Y \vee X$

vs. $\vee \doteq \lambda X.\lambda Y.Y \vee X$

HOL: Semantics

(see [BenzmüllerBrownKohlhase, J.Symb.Log., 2004] and [BenzmüllerBrownKohlhase, LMCS, 2009]



We additionally studied different model classes with 'varying degrees of extensionality'

$\forall X.\forall Y.X \vee Y \Leftrightarrow Y \vee X$ $\quad$ $\forall X.\forall Y.X \vee Y \doteq Y \vee X$

$\lambda X.\lambda Y.X \vee Y \doteq \lambda X.\lambda Y.Y \vee X$ $\quad$ $\vee \doteq \lambda X.\lambda Y.Y \vee X$