

# Encoding Hybrid Logic into Higher-Order Logic

Chad E. Brown

`cebrown@ags.uni-sb.de`

Universität des Saarlandes, Saarbrücken, Germany

# Simple Types

Simple Types  $\mathcal{T}$ :

- $o$  (truth values)
- $\iota$  (individuals)
- $(\alpha\beta)$  (functions from  $\beta$  to  $\alpha$ )

# Simple Types

Simple Types  $\mathcal{T}$ :

$o$	(truth values)
$\iota$	(individuals)
$(\alpha\beta)$	(functions from $\beta$ to $\alpha$ )

$(\alpha\beta\gamma)$  abbreviates  $((\alpha\beta)\gamma)$

# Simple Types

Simple Types $\mathcal{T}$ :	$o$	(truth values)
	$\iota$	(individuals)
	$(\alpha\beta)$	(functions from $\beta$ to $\alpha$ )
	$o\iota$	“sets of individuals” (characteristic functions of sets)
$o(o\iota)$	“sets of sets”	

# Simply Typed $\lambda$ -Terms

Terms:

$x_\alpha$	Variables ( $\mathcal{V}$ )
$A_\alpha$	Parameters ( $\mathcal{P}$ )
$c_\alpha$	Logical Constants ( $\mathcal{S}$ )
$(\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta)_\alpha$	Application
$(\lambda y_\beta \mathbf{A}_\alpha)_{\alpha\beta}$	$\lambda$ -abstraction

# Simply Typed $\lambda$ -Terms

Terms:	$x_\alpha$	Variables ( $\mathcal{V}$ )
	$A_\alpha$	Parameters ( $\mathcal{P}$ )
	$c_\alpha$	Logical Constants ( $\mathcal{S}$ )
	$(\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta)_\alpha$	Application
	$(\lambda y_\beta \mathbf{A}_\alpha)_{\alpha\beta}$	$\lambda$ -abstraction

Equality of terms:  $\alpha\beta\eta$

# Simply Typed $\lambda$ -Terms

Terms:	$x_\alpha$	Variables ( $\mathcal{V}$ )
	$A_\alpha$	Parameters ( $\mathcal{P}$ )
	$c_\alpha$	Logical Constants ( $\mathcal{S}$ )
	$(\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta)_\alpha$	Application
	$(\lambda y_\beta \mathbf{A}_\alpha)_{\alpha\beta}$	$\lambda$ -abstraction

Equality of terms:  $\alpha\beta\eta$

$\alpha$ -conversion    Changing Bound Variables

$\beta$ -reduction     $((\lambda y_\beta \mathbf{A}_\alpha) \mathbf{B}) \xrightarrow{\beta} [\mathbf{B}/y] \mathbf{A}$

$\eta$ -reduction     $(\lambda y_\beta (\mathbf{F}_{\alpha\beta} y)) \xrightarrow{\eta} \mathbf{F}$      $(y_\beta \notin \mathbf{Free}(\mathbf{F}))$

# Simply Typed $\lambda$ -Terms

Terms:

$x_\alpha$	Variables ( $\mathcal{V}$ )
$A_\alpha$	Parameters ( $\mathcal{P}$ )
$c_\alpha$	Logical Constants ( $\mathcal{S}$ )
$(\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta)_\alpha$	Application
$(\lambda y_\beta \mathbf{A}_\alpha)_{\alpha\beta}$	$\lambda$ -abstraction

Equality of terms:  $\alpha\beta\eta$

Every term has a unique  $\beta\eta$ -normal form,  
up to  $\alpha$ -conversion.

# Logical Constants

$\neg_{oo}$  – negation

$\vee_{ooo}$  – disjunction

$\prod_{o(o\alpha)}^\alpha$  – universal quantification over type  $\alpha$

$=_{o\alpha\alpha}^\alpha$  – equality

# Abbreviations for Logical Operators

$(\mathbf{A}_o \vee \mathbf{B}_o)$  means  $(\vee_{ooo} \mathbf{A}_o \mathbf{B}_o)$

$(\mathbf{A}_o \supset \mathbf{B}_o)$  means  $(\neg \mathbf{A}_o \vee \mathbf{B}_o)$

$(\forall x_\alpha \mathbf{A}_o)$  means  $(\Pi_{o(o\alpha)}^\alpha \cdot \lambda x_\alpha \mathbf{A}_o)$ .

$(\exists x_\alpha \mathbf{A}_o)$  means  $(\neg \forall x_\alpha \neg \mathbf{A}_o)$ .

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice
- Axiom of Infinity

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice
- Axiom of Infinity

Elementary Type Theory ( $\text{TPS}$  – for automated theorem proving)

# Church's Type Theory

## Church's Type Theory:

- Simply typed  $\lambda$ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice
- Axiom of Infinity

Extensional Type Theory (TPS and LEO – automated theorem proving)

# Multi-Modal Logic

“Propositional” Symbols:

$$PROP = \{P, Q, \dots\}$$

# Multi-Modal Logic

“Propositional” Symbols:

$$PROP = \{P, Q, \dots\}$$

“Modalities”:

$$MOD = \{R, S, \dots\}$$

# Multi-Modal Logic

“Propositional” Symbols:

$$PROP = \{P, Q, \dots\}$$

“Modalities”:

$$MOD = \{R, S, \dots\}$$

Multi-Modal WFF's ( $\varphi, \psi, \dots$ ):

$$P | \neg\varphi | \varphi \vee \psi | \langle R \rangle \varphi | [R] \varphi$$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$
- $ST_x(\neg\varphi) = \neg ST_x(\varphi)$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$
- $ST_x(\neg\varphi) = \neg ST_x(\varphi)$
- $ST_x(\varphi \vee \psi) = ST_x(\varphi) \vee ST_x(\psi)$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$
- $ST_x(\neg\varphi) = \neg ST_x(\varphi)$
- $ST_x(\varphi \vee \psi) = ST_x(\varphi) \vee ST_x(\psi)$
- $ST_x(\langle R \rangle \varphi) = \exists y(\bar{R}(x, y) \wedge ST_y(\varphi))$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$
- $ST_x(\neg\varphi) = \neg ST_x(\varphi)$
- $ST_x(\varphi \vee \psi) = ST_x(\varphi) \vee ST_x(\psi)$
- $ST_x(\langle R \rangle \varphi) = \exists y(\bar{R}(x, y) \wedge ST_y(\varphi))$
- $ST_x([R]\varphi) = \forall y(\bar{R}(x, y) \supset ST_y(\varphi))$

# Multi-Modal Logic

Standard Translation to First-Order (relative to  $x$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}$ .
- $ST_x(P) = \bar{P}(x)$
- $ST_x(\neg\varphi) = \neg ST_x(\varphi)$
- $ST_x(\varphi \vee \psi) = ST_x(\varphi) \vee ST_x(\psi)$
- $ST_x(\langle R \rangle \varphi) = \exists y(\bar{R}(x, y) \wedge ST_y(\varphi))$
- $ST_x([R]\varphi) = \forall y(\bar{R}(x, y) \supset ST_y(\varphi))$

Note:  $ST$  translates to a predicate on  $x$ .

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_\iota \exists y_\iota ((\bar{R} x y) \wedge (HST(\varphi) y))$

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_\iota \exists y_\iota ((\bar{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_\iota \forall y_\iota ((\bar{R} x y) \supset (HST(\varphi) y))$

# Multi-Modal Logic

Translation to Higher-Order (map to type  $o\iota$ ):

- Associate each  $P \in PROP$  with some predicate  $\bar{P}_{o\iota}$ .
- Associate each  $R \in MOD$  with some relation  $\bar{R}_{o\iota\iota}$ .
- $HST(P) = \bar{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_\iota \exists y_\iota ((\bar{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_\iota \forall y_\iota ((\bar{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_\iota \exists y_\iota ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_\iota \forall y_\iota ((\overline{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\lambda x_\iota \neg (HST(\varphi)x))$
- $HST(\varphi \vee \psi) = (\lambda x_\iota ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_\iota \exists y_\iota ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_\iota \forall y_\iota ((\overline{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{oi}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{oui}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (\lambda x_i ((HST(\varphi)x) \vee (HST(\psi) x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_i \exists y_i ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

$\neg_{oi(oi)}$  is  $(\lambda U_{oi} \lambda x_i \neg(Ux))$  (Complement)

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (\lambda x_i ((HST(\varphi)x) \vee (HST(\psi)x)))$
- $HST(\langle R \rangle \varphi) = \lambda x_i \exists y_i ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{oi}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{oui}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = \lambda x_i \exists y_i ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

$\vee_{oi(oi)}$  is  $(\lambda U_{oi} \lambda V_{oi} \lambda x_i . (Ux) \vee (Vx))$  (Union)

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = \lambda x_i \exists y_i ((\overline{R} x y) \wedge (HST(\varphi) y))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{oi}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{oui}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = (\diamond \overline{R} HST(\varphi))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

$\diamond_{oi(oi)(oui)}$  is  $(\lambda R_{oui} \lambda U_{oi} \lambda x_i \exists y_i . R x y \wedge U y)$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = (\diamond \overline{R} HST(\varphi))$
- $HST([R]\varphi) = \lambda x_i \forall y_i ((\overline{R} x y) \supset (HST(\varphi) y))$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{oi}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{oui}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = (\diamond \overline{R} HST(\varphi))$
- $HST([R]\varphi) = (\square \overline{R} HST(\varphi))$

$\square_{oi(oi)(oui)}$  is  $(\lambda R_{oui} \lambda U_{oi} \lambda x_i \forall y_i . R x y \supset U y)$

# Encoding Multi-Modal Logic

- Associate each  $P \in PROP$  with some predicate  $\overline{P}_{ou}$ .
- Associate each  $R \in MOD$  with some relation  $\overline{R}_{ou}$ .
- $HST(P) = \overline{P}$
- $HST(\neg\varphi) = (\neg HST(\varphi))$
- $HST(\varphi \vee \psi) = (HST(\varphi) \vee HST(\psi))$
- $HST(\langle R \rangle \varphi) = (\diamond \overline{R} HST(\varphi))$
- $HST([R]\varphi) = (\square \overline{R} HST(\varphi))$

Using these definitions, the translation is trivial.

# Multi-Modal Fragment of Higher-Order

$$PROP = \{P, Q, \dots\} \subseteq \mathcal{V}_{ol} \cup \mathcal{P}_{ol}$$

# Multi-Modal Fragment of Higher-Order

$$PROP = \{P, Q, \dots\} \subseteq \mathcal{V}_{oi} \cup \mathcal{P}_{oi}$$

$$MOD = \{R, S, \dots\} \subseteq \mathcal{V}_{ou} \cup \mathcal{P}_{oi}$$

# Multi-Modal Fragment of Higher-Order

$$PROP = \{P, Q, \dots\} \subseteq \mathcal{V}_{oi} \cup \mathcal{P}_{oi}$$

$$MOD = \{R, S, \dots\} \subseteq \mathcal{V}_{ou} \cup \mathcal{P}_{oi}$$

Multi-Modal Formulas are  
certain terms  $(\varphi, \psi, \dots)$  of type  $oi$ :

$$P | \neg \varphi | (\varphi \vee \psi) | (\diamond R \varphi) | (\square R \varphi)$$

# Multi-Modal Fragment of Higher-Order

$$PROP = \{P, Q, \dots\} \subseteq \mathcal{V}_{oi} \cup \mathcal{P}_{oi}$$

$$MOD = \{R, S, \dots\} \subseteq \mathcal{V}_{ou} \cup \mathcal{P}_{oi}$$

Multi-Modal Formulas are  
certain terms  $(\varphi, \psi, \dots)$  of type  $oi$ :

$$P | \neg \varphi | (\varphi \vee \psi) | (\diamond R \varphi) | (\square R \varphi)$$

No inductive translation is required.

# Next...

Two directions:

1. Generalize the types.
2. Extend to Hybrid Logic.

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

$\alpha$ -Multi-Modal Formulas  $MMF^\alpha$  (terms of type  $o\alpha$ ):

$$P_{o\alpha} \mid \neg^\alpha \varphi \mid (\varphi \vee^\alpha \psi) \mid (\diamond^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta}) \mid (\square^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta})$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

$\alpha$ -Multi-Modal Formulas  $MMF^\alpha$  (terms of type  $o\alpha$ ):

$$P_{o\alpha} \mid \neg^\alpha \varphi \mid (\varphi \vee^\alpha \psi) \mid (\diamond^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta}) \mid (\square^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta})$$

$$\neg^\alpha \text{ is } (\lambda U_{o\alpha} \lambda x_\alpha \neg(Ux))$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

$\alpha$ -Multi-Modal Formulas  $MMF^\alpha$  (terms of type  $o\alpha$ ):

$$P_{o\alpha} \mid \neg^\alpha \varphi \mid (\varphi \vee^\alpha \psi) \mid (\diamond^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta}) \mid (\square^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta})$$

$$\vee^\alpha \text{ is } (\lambda U_{o\alpha} \lambda V_{o\alpha} \lambda x_\iota . (Ux) \vee (Vx))$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

$\alpha$ -Multi-Modal Formulas  $MMF^\alpha$  (terms of type  $o\alpha$ ):

$$P_{o\alpha} \mid \neg^\alpha \varphi \mid (\varphi \vee^\alpha \psi) \mid (\diamond^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta}) \mid (\square^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta})$$

$$\diamond^{\alpha,\beta} \text{ is } (\lambda R_{o\beta\alpha} \lambda U_{o\beta} \lambda x_\alpha \exists y_\beta . R x y \wedge U y)$$

# Generalizing the Types

(Properties on type  $\alpha$ )

$$PROP^\alpha = \{P_{o\alpha}, Q_{o\alpha}, \dots\} \subseteq \mathcal{V}_{o\alpha} \cup \mathcal{P}_{o\alpha}$$

(Relations between  $\alpha$  and  $\beta$ )

$$MOD^{\alpha,\beta} = \{R_{o\beta\alpha}, S_{o\beta\alpha}, \dots\} \subseteq \mathcal{V}_{o\beta\alpha} \cup \mathcal{P}_{o\beta\alpha}$$

$\alpha$ -Multi-Modal Formulas  $MMF^\alpha$  (terms of type  $o\alpha$ ):

$$P_{o\alpha} \mid \neg^\alpha \varphi \mid (\varphi \vee^\alpha \psi) \mid (\diamond^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta}) \mid (\square^{\alpha,\beta} R_{o\beta\alpha} \varphi_{o\beta})$$

$$\square^{\alpha,\beta} \text{ is } (\lambda R_{o\beta\alpha} \lambda U_{o\beta} \lambda x_\alpha \forall y_\beta . R x y \supset U y)$$

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

Let  $P_{o\alpha} \in PROP^{\alpha}$ .

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

Let  $P_{o\alpha} \in PROP^{\alpha}$ .

- $[\bar{\epsilon}]P$  is in  $MMF^{o\alpha}$

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

Let  $P_{o\alpha} \in PROP^{\alpha}$ .

- $[\bar{\epsilon}]P$  is in  $MMF^{o\alpha}$

This is true at  $Q_{o\alpha}$  iff  $Q$  is a subset of  $P$ . TPs can prove equivalence automatically (expanding definitions and working in higher-order logic).

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_{\alpha}. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

Let  $P_{o\alpha} \in PROP^{\alpha}$ .

- $[\bar{\epsilon}]P$  is in  $MMF^{o\alpha}$
- $[\bar{\epsilon}](P \wedge \neg P)$

# Examples

Let  $\bar{\epsilon}_{o\alpha(o\alpha)}$  stand for  $[\lambda U_{o\alpha} \lambda x_\alpha. U x]$ .

Intuitively,  $(\bar{\epsilon} U x)$  means  $x \in U$ .

Let  $P_{o\alpha} \in PROP^\alpha$ .

- $[\bar{\epsilon}]P$  is in  $MMF^{o\alpha}$

- $[\bar{\epsilon}](P \wedge \neg P)$

This is true at  $Q_{o\alpha}$  iff  $Q$  is empty. Automatic Proof by TPS

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

Intuitively,  $(\in x U)$  means  $x \in U$ .

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

Let  $OPEN_{o(o\alpha)} \in PROP^{o\alpha}$ .

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

Let  $OPEN_{o(o\alpha)} \in PROP^{o\alpha}$ .

•  $\langle \in \rangle OPEN$  is in  $MMF^\alpha$

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. Ux]$ .

Let  $OPEN_{o(o\alpha)} \in PROP^{o\alpha}$ .

•  $\langle \in \rangle OPEN$  is in  $MMF^\alpha$

This is true at  $x_\alpha$  iff  $x$  is in some  $P$  in  $OPEN$ .

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

Let  $OPEN_{o(o\alpha)} \in PROP^{o\alpha}$ .

•  $\langle \in \rangle OPEN$  is in  $MMF^\alpha$

This is true at  $x_\alpha$  iff  $x$  is in some  $P$  in  $OPEN$ .

Represents union of a collection

# Examples

Let  $\in_{o\alpha(o\alpha)}$  stand for  $[\lambda x_\alpha \lambda U_{o\alpha}. U x]$ .

Let  $OPEN_{o(o\alpha)} \in PROP^{o\alpha}$ .

•  $\langle \in \rangle OPEN$  is in  $MMF^\alpha$

This is true at  $x_\alpha$  iff  $x$  is in some  $P$  in  $OPEN$ .

Represents union of a collection

(Automatic proof of equivalence by TPs)

# Hybrid Logic

Multi-Modal Plus Nominals:

$$NOM = \{i, j, \dots\}$$

# Hybrid Logic

Multi-Modal Plus Nominals:

$$NOM = \{i, j, \dots\}$$

“At” operator:

$$@_i \varphi$$

# Hybrid Logic

Multi-Modal Plus Nominals:

$$NOM = \{i, j, \dots\}$$

“At” operator:

$$@_i \varphi$$

Maybe downarrow binder:

$$\downarrow x. \varphi(x)$$

# First-Order Translation

- Associate nominals  $i$  with variable  $\bar{i}$ .
- $ST_x(i) = (x = \bar{i})$

# First-Order Translation

- Associate nominals  $i$  with variable  $\bar{i}$ .
- $ST_x(i) = (x = \bar{i})$
- $ST_x(@_i\varphi) = [\bar{i}/x]ST_x(\varphi)$

# First-Order Translation

- Associate nominals  $i$  with variable  $\bar{i}$ .
- $ST_x(i) = (x = \bar{i})$
- $ST_x(@_i\varphi) = [\bar{i}/x]ST_x(\varphi)$
- $ST_x(y) = (x = y)$  ( $y$  is a “nominal variable”)

# First-Order Translation

- Associate nominals  $i$  with variable  $\bar{i}$ .
- $ST_x(i) = (x = \bar{i})$
- $ST_x(@_i\varphi) = [\bar{i}/x]ST_x(\varphi)$
- $ST_x(y) = (x = y)$  ( $y$  is a “nominal variable”)
- $ST_x(\downarrow y.\varphi) = [x/y]ST_x(\varphi)$

# Encoding Hybrid Logic with General Types

Nominals at type  $\alpha$ :

$$NOM^\alpha = \{i_\alpha, j_\alpha, \dots\} \subseteq \mathcal{V}_\alpha \cup \mathcal{P}_\alpha$$

# Encoding Hybrid Logic with General Types

Nominals at type  $\alpha$ :

$$NOM^\alpha = \{i_\alpha, j_\alpha, \dots\} \subseteq \mathcal{V}_\alpha \cup \mathcal{P}_\alpha$$

$\alpha$ -Hybrid Formula  $HF^\alpha$  is a term of type  $o\alpha$  constructed as  $\alpha$ -Multi-Modal Plus:

# Encoding Hybrid Logic with General Types

Nominals at type  $\alpha$ :

$$NOM^\alpha = \{i_\alpha, j_\alpha, \dots\} \subseteq \mathcal{V}_\alpha \cup \mathcal{P}_\alpha$$

$\alpha$ -Hybrid Formula  $HF^\alpha$  is a term of type  $o_\alpha$  constructed as  $\alpha$ -Multi-Modal Plus:

•  $(\mathcal{U}^\alpha i_\alpha)$  where  $\mathcal{U}^\alpha$  is  $(\lambda x_\alpha \lambda y_\alpha (x = y))$

# Encoding Hybrid Logic with General Types

Nominals at type  $\alpha$ :

$$NOM^\alpha = \{i_\alpha, j_\alpha, \dots\} \subseteq \mathcal{V}_\alpha \cup \mathcal{P}_\alpha$$

$\alpha$ -Hybrid Formula  $HF^\alpha$  is a term of type  $o_\alpha$  constructed as  $\alpha$ -Multi-Modal Plus:

- $(U^\alpha i_\alpha)$  where  $U^\alpha$  is  $(\lambda x_\alpha \lambda y_\alpha (x = y))$
- $(@^{\alpha,\beta} j_\beta \varphi_{o\beta})_{o\alpha}$  where  $@^{\alpha,\beta}$  is  $\lambda z_\beta \lambda V_{o\beta} \lambda x_\alpha . V z$

(Note this does not depend  $x$ .)

# Encoding Hybrid Logic with General Types

Nominals at type  $\alpha$ :

$$NOM^\alpha = \{i_\alpha, j_\alpha, \dots\} \subseteq \mathcal{V}_\alpha \cup \mathcal{P}_\alpha$$

$\alpha$ -Hybrid Formula  $HF^\alpha$  is a term of type  $o_\alpha$  constructed as  $\alpha$ -Multi-Modal Plus:

- $(\mathcal{U}^\alpha i_\alpha)$  where  $\mathcal{U}^\alpha$  is  $(\lambda x_\alpha \lambda y_\alpha (x = y))$
- $(@^{\alpha,\beta} j_\beta \varphi_{o\beta})_{o_\alpha}$  where  $@^{\alpha,\beta}$  is  $\lambda z_\beta \lambda V_{o\beta} \lambda x_\alpha . V z$
- $(\downarrow^\alpha (\lambda i_\alpha . \varphi_{o_\alpha}))$  ( $i_\alpha$  is a “nominal variable”)

where  $\downarrow^\alpha$  is  $\lambda W_{o_\alpha \alpha} . \lambda x_\alpha (W x x)$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

$$\bullet (\mathcal{U}^{o\alpha} e) \supset [\bar{\epsilon}](P \wedge \neg P)$$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

•  $(\mathcal{U}^{o\alpha} e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

- $(\mathcal{U}^{o\alpha}e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

- $(\mathcal{U}^{o\alpha}e) \equiv [\bar{\epsilon}](P \wedge \neg P)$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

- $(\mathcal{U}^{o\alpha}e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

- $(\mathcal{U}^{o\alpha}e) \equiv [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is the *unique* empty set (extensionality).

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

- $(\mathcal{U}^{o\alpha} e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

- $(\mathcal{U}^{o\alpha} e) \equiv [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is the *unique* empty set (extensionality).

- $@^{\beta, (o\alpha)} e [\bar{\epsilon}](P \wedge \neg P)$

# Examples

$$P_{o\alpha} \in PROP^\alpha$$

$$e_{o\alpha} \in NOM^{o\alpha}$$

- $(\mathcal{U}^{o\alpha} e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

- $(\mathcal{U}^{o\alpha} e) \equiv [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is the *unique* empty set (extensionality).

- $@^{\beta, (o\alpha)} e [\bar{\epsilon}](P \wedge \neg P)$

True at  $b_\beta$  iff  $e$  is empty (does not depend on  $b$ ).

# Examples

$P_{o\alpha} \in PROP^\alpha$

$e_{o\alpha} \in NOM^{o\alpha}$

•  $(\mathcal{U}^{o\alpha} e) \supset [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is empty.

•  $(\mathcal{U}^{o\alpha} e) \equiv [\bar{\epsilon}](P \wedge \neg P)$

Valid if  $e$  is the *unique* empty set (extensionality).

•  $@^{\beta, (o\alpha)} e [\bar{\epsilon}](P \wedge \neg P)$

True at  $b_\beta$  iff  $e$  is empty (does not depend on  $b$ ).

Automatic proofs in TPS

# Conclusion

- Using  $\lambda$ -terms, the standard translation from Multi-Modal (and Hybrid) Logic to first-order logic becomes an easier translation into higher-order logic.

# Conclusion

- Using  $\lambda$ -terms, the standard translation from Multi-Modal (and Hybrid) Logic to first-order logic becomes an easier translation into higher-order logic.
- The translation shows Hybrid Logic is a natural fragment of higher-order logic.

# Conclusion

- Using  $\lambda$ -terms, the standard translation from Multi-Modal (and Hybrid) Logic to first-order logic becomes an easier translation into higher-order logic.
- The translation shows Hybrid Logic is a natural fragment of higher-order logic.
- By generalizing over types, we obtain a typed form of Hybrid Logic.