Cette adorable personne c'est toi, sous le grand chapeau cosplay.
A Type-Theoretic View of Dynamic Logic

Philippe de Groote
LORIA & Inria-Lorraine
A Type-Theoretic Reconstruction of DRT
A Type-Theoretic Reconstruction of DRT

Motivation:

• to formalize DRT within Church’s simple theory of type (aka, Higher-Order Logic), which will allow DRT and Montague semantics to rest on the same logical foundations.
A Type-Theoretic Reconstruction of DRT

Motivation:

- to formalize DRT within Church’s simple theory of type (aka, Higher-Order Logic), which will allow DRT and Montague semantics to rest on the same logical foundations.

Challenge:

- to express dynamics using “static” primitives (in particular, to avoid the “destructive assignment” problem, which necessitates a LISP-like gensym operator).
A Type-Theoretic Reconstruction of DRT

Motivation:

• to formalize DRT within Church’s simple theory of type (aka, Higher-Order Logic), which will allow DRT and Montague semantics to rest on the same logical foundations.

Challenge:

• to express dynamics using “static” primitives (in particular, to avoid the “destructive assignment” problem, which necessitates a LISP-like gensym operator).

Proposed solution:

• to interpret a sentence according to both its left and right contexts;

• to abstract these two kinds of contexts over the meaning of the sentences.
Typing the left and the right contexts
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- $\textit{\iota}$, the type of individuals (a.k.a. entities).
- $\textit{o}$, the type of propositions (a.k.a. truth values).
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- $\iota$, the type of individuals (a.k.a. entities).
- $\o$, the type of propositions (a.k.a. truth values).

We add a third atomic type, $\gamma$, which stands for the type of the left contexts.
**Typing the left and the right contexts**

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- $\iota$, the type of individuals (a.k.a. entities).
- $\omicron$, the type of propositions (a.k.a. truth values).

We add a third atomic type, $\gamma$, which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( \iota \), the type of individuals (a.k.a. entities).
- \( \omicron \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- $\tau$, the type of individuals (a.k.a. entities).
- $o$, the type of propositions (a.k.a. truth values).

We add a third atomic type, $\gamma$, which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( t \), the type of individuals (a.k.a. entities).
- \( o \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( \iota \), the type of individuals (a.k.a. entities).
- \( o \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( \iota \), the type of individuals (a.k.a. entities).
- \( \omicron \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
**Typing the left and the right contexts**

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( \iota \), the type of individuals (a.k.a. entities).
- \( \omega \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
Typing the left and the right contexts

Montague semantics is based on Church’s simple type theory, which provides a full hierarchy of functional types built upon two atomic types:

- \( t \), the type of individuals (a.k.a. entities).
- \( o \), the type of propositions (a.k.a. truth values).

We add a third atomic type, \( \gamma \), which stands for the type of the left contexts.

What about the type of the right contexts?
Semantic interpretation of the sentences
Semantic interpretation of the sentences

Let $s$ be the syntactic category of sentences. Remember that we intend to abstract our notions of left and right contexts over the meaning of the sentences.
Semantic interpretation of the sentences

Let $s$ be the syntactic category of sentences. Remember that we intend to abstract our notions of left and right contexts over the meaning of the sentences.

$$[s] = \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$$
Semantic interpretation of the sentences

Let $s$ be the syntactic category of sentences. Remember that we intend to abstract our notions of left and right contexts over the meaning of the sentences.

$$[s] = \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$$

Composition of two sentence interpretations
Semantic interpretation of the sentences

Let \( s \) be the syntactic category of sentences. Remember that we intend to abstract our notions of left and right contexts over the meaning of the sentences.

\[
[s] = \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o
\]

Composition of two sentence interpretations

\[
[S_1. S_2] = \lambda e \phi. [S_1] e (\lambda e' \cdot [S_2] e' \phi)
\]
Semantic interpretation of the syntactic categories
Semantic interpretation of the syntactic categories

Montague’s interpretation

\[
\begin{align*}
[s] &= o \\
[n] &= \iota \to o \\
[np] &= (\iota \to o) \to o
\end{align*}
\]
Semantic interpretation of the syntactic categories

Montague’s interpretation

\[
\begin{align*}
[s] &= o \\
[n] &= \iota \rightarrow o \\
[np] &= (\iota \rightarrow o) \rightarrow o
\end{align*}
\]

may be rephrased as follows:

\[
\begin{align*}
[s] &= o \\
[n] &= \iota \rightarrow [s] \\
[np] &= (\iota \rightarrow [s]) \rightarrow [s]
\end{align*}
\]
Semantic interpretation of the syntactic categories

Montague’s interpretation

\[
\begin{align*}
[s] &= o \\
[n] &= \iota \rightarrow o \\
[np] &= (\iota \rightarrow o) \rightarrow o
\end{align*}
\]

may be rephrased as follows:

\[
\begin{align*}
[s] &= o \quad (1) \\
[n] &= \iota \rightarrow [s] \quad (2) \\
[np] &= (\iota \rightarrow [s]) \rightarrow [s] \quad (3)
\end{align*}
\]

Replacing (1) with:

\[
\begin{align*}
[s] &= \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o
\end{align*}
\]
Semantic interpretation of the syntactic categories

Montague’s interpretation

\[
\begin{align*}
[s] & = o \\
[n] & = \iota \rightarrow o \\
[np] & = (\iota \rightarrow o) \rightarrow o
\end{align*}
\]

may be rephrased as follows:

\[
\begin{align*}
[s] & = o & (1) \\
[n] & = \iota \rightarrow [s] & (2) \\
[np] & = (\iota \rightarrow [s]) \rightarrow [s] & (3)
\end{align*}
\]

Replacing (1) with:

\[
\begin{align*}
[s] & = \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o
\end{align*}
\]

we obtain:

\[
\begin{align*}
[n] & = \iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o \\
[np] & = (\iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o) \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o
\end{align*}
\]
This interpretation results in handcrafted lexical semantics such as the following:
This interpretation results in handcrafted lexical semantics such as the following:

\[
\begin{align*}
&[\text{farmer}] = \lambda x e \phi. \text{farmer} \ x \land \phi e \\
&[\text{donkey}] = \lambda x e \phi. \text{donkey} \ x \land \phi e \\
&[\text{owns}] = \lambda o s. \lambda x. o (\lambda y e \phi. \text{own} \ x \ y \land \phi e) \\
&[\text{beats}] = \lambda o s. \lambda x. o (\lambda y e \phi. \text{beat} \ x \ y \land \phi e) \\
&[\text{who}] = \lambda r n x e \phi. n x e (\lambda e. r (\lambda \psi. \psi x) e \phi) \\
&[a] = \lambda n \psi e \phi. \exists x n x e (\lambda e. \psi x (x::e) \phi) \\
&[\text{every}] = \lambda n \psi e \phi. (\forall x. \neg (n x e (\lambda e. \neg (\psi x (x::e) (\lambda e. \top))))\land \phi e) \\
&[\text{it}] = \lambda \psi e \phi. \psi (\text{sel} e) e \phi
\end{align*}
\]
This interpretation results in handcrafted lexical semantics such as the following:

\[
\begin{align*}
\text{[farmer]} & = \lambda x e \phi. \text{farmer} \ x \land \phi \ e \\
\text{[donkey]} & = \lambda x e \phi. \text{donkey} \ x \land \phi \ e \\
\text{[owns]} & = \lambda o s. (\lambda x. o (\lambda y e \phi. \text{own} \ x \ y \land \phi \ e)) \\
\text{[beats]} & = \lambda o s. (\lambda x. o (\lambda y e \phi. \text{beat} \ x \ y \land \phi \ e)) \\
\text{[who]} & = \lambda r n x e \phi. n \ x \ e (\lambda e. r (\lambda \psi. \psi \ x) e \phi) \\
\text{[a]} & = \lambda n \psi e \phi. \exists x. n \ x \ e (\lambda e. \psi \ x (x::e) \phi) \\
\text{[every]} & = \lambda n \psi e \phi. (\forall x. \neg(n \ x \ e (\lambda e. \neg(\psi \ x (x::e) (\lambda e. \top)))}) \land \phi \ e \\
\text{[it]} & = \lambda \psi e \phi. \psi (\text{sel} \ e) e \phi
\end{align*}
\]

...which might seem a little bit involved.
Questions:
Questions:

- Is there a systematic way of obtaining the new lexical semantics from Montague’s?
Questions:

- is there a systematic way of obtaining the new lexical semantics from Montague’s?
- can we find any “modular” presentation of the approach?
Questions:

- is there a systematic way of obtaining the new lexical semantics from Montague’s?
- can we find any “modular” presentation of the approach?
- is there some dynamic logic hidden in the approach?
A Dynamic Logic
A Dynamic Logic

Let $\Omega \triangleq \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$. We intend to design a logic acting on propositions of type $\Omega$. 
A Dynamic Logic

Let $\Omega \triangleq \gamma \to (\gamma \to o) \to o$. We intend to design a logic acting on propositions of type $\Omega$

We share with DRT the two following assumptions:

- discourse composition is mainly conjunctive (roughly speaking, a discourse consists in the conjunction of its sentences);
- the main form of quantification is existential (it introduces referential markers).
A Dynamic Logic

Let $\Omega \triangleq \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$. We intend to design a logic acting on propositions of type $\Omega$

We share with DRT the two following assumptions:

- discourse composition is mainly conjunctive (roughly speaking, a discourse consists in the conjunction of its sentences);
- the main form of quantification is existential (it introduces referential markers).

Consequently, our logic will be based on conjunction and existential quantification (defined as primitives). The other connectives will be obtained using negation (a third primitive) and de Morgan’s laws.
Formal Framework

We consider a simply-typed $\lambda$-calculus, the terms of which are built upon a signature including the following constants:
Formal Framework

We consider a simply-typed $\lambda$-calculus, the terms of which are built upon a signature including the following constants:

FIRST-ORDER LOGIC

\[
\begin{align*}
\top &: \ o \\
\neg &: \ o \rightarrow o \\
\wedge &: \ o \rightarrow o \rightarrow o \\
\exists &: \ (\iota \rightarrow o) \rightarrow o
\end{align*}
\]

\begin{itemize}
\item $\top$ \hspace{1cm} (truth)
\item $\neg$ \hspace{1cm} (negation)
\item $\wedge$ \hspace{1cm} (conjunction)
\item $\exists$ \hspace{1cm} (existential quantification)
\end{itemize}
Formal Framework

We consider a simply-typed λ-calculus, the terms of which are built upon a signature including the following constants:

FIRST-ORDER LOGIC

\[ \top : o \] (truth)
\[ \neg : o \rightarrow o \] (negation)
\[ \land : o \rightarrow o \rightarrow o \] (conjunction)
\[ \exists : (\iota \rightarrow o) \rightarrow o \] (existential quantification)

DYNAMIC PRIMITIVES

\[ :: : \iota \rightarrow \gamma \rightarrow \gamma \] (context updating)
\[ \text{sel} : \gamma \rightarrow \iota \] (choice operator)
Conjunction
Conjunction

Conjunction is nothing but sentence composition. We therefore define:

\[ A \sqcap B \triangleq \lambda e \phi. A e (\lambda e. B e \phi) \]
Conjunction

Conjunction is nothing but sentence composition. We therefore define:

\[ A \sqcap B \triangleq \lambda e \phi. A e (\lambda e. B e \phi) \]

Existential quantification
Conjunction

Conjunction is nothing but sentence composition. We therefore define:

\[ A \sqcap B \triangleq \lambda e \phi. A e (\lambda e. B e \phi) \]

Existential quantification

Existential quantification introduces “reference markers”. It is therefore responsible for context updating:

\[ \Sigma x. P x \triangleq \lambda e \phi. \exists x. P x (x::e) \phi \]
Conjunction

Conjunction is nothing but sentence composition. We therefore define:

\[ A \sqcap B \triangleq \lambda e \phi. A e (\lambda e. B e \phi) \]

Existential quantification

Existential quantification introduces “reference markers”. It is therefore responsible for context updating:

\[ \Sigma x. P x \triangleq \lambda e \phi. \exists x. P x (x::e) \phi \]

Negation
Conjunction

Conjunction is nothing but sentence composition. We therefore define:

\[ A \sqcap B \triangleq \lambda e.\phi. (\lambda e. B e \phi) \]

Existential quantification

Existential quantification introduces “reference markers”. It is therefore responsible for context updating:

\[ \Sigma x. \ P x \triangleq \lambda e.\phi. \exists x. \ P (x::e) \phi \]

Negation

We do not want the continuation of the discourse to fall into the scope of the negation. Consequently, negation must be defined as follows:
$\sim A \triangleq \lambda e \phi. \neg (A e (\lambda e. \top)) \land \phi e$
Implication and Universal Quantification
Implication and Universal Quantification

These are defined using de Morgan’s laws:

\[ A \sqsupset B \triangleq \sim (A \sqcap \sim B) \]
\[ \Pi x. P x \triangleq \sim \Sigma x. \sim (P x) \]
Implication and Universal Quantification

These are defined using de Morgan’s laws:

\[ A \sqcup B \triangleq \sim(A \sqcap \sim B) \]
\[ \Pi x. P x \triangleq \sim\Sigma x. \sim(P x) \]

Embedding of first-order logic into dynamic logic
Implication and Universal Quantification

These are defined using de Morgan’s laws:

\[ A \sqsubseteq B \triangleq \sim (A \sqcap \sim B) \]
\[ \Pi x. P x \triangleq \sim \Sigma x. \sim (P x) \]

Embedding of first-order logic into dynamic logic

\[ \overline{R t_1 \ldots t_n} = \lambda e \phi. R t_1 \ldots t_n \land \phi e \]
\[ \overline{\neg A} = \sim A \]
\[ \overline{A \land B} = \overline{A} \land \overline{B} \]
\[ \overline{\exists x. A} = \Sigma x. \overline{A} \]
Implication and Universal Quantification

These are defined using de Morgan’s laws:

\[ A \sqsubseteq B \triangleq \sim(A \sqcap \sim B) \]
\[ \Pi x. P x \triangleq \sim \Sigma x. \sim(P x) \]

Embedding of first-order logic into dynamic logic

\[ Rt_1 \ldots t_n = \lambda e. \phi. R t_1 \ldots t_n \land \phi e \]
\[ \neg A = \sim A \]
\[ A \land B = \overline{A} \cap \overline{B} \]
\[ \exists x. A = \Sigma x. \overline{A} \]

This embedding is such that, for every term \( e \) of type \( \gamma \):

\[ A \equiv \overline{A} e (\lambda e. \top) \]
Donkey Sentence Revisited
Donkey Sentence Revisited

Montague-like semantic interpretation:

\[
\begin{align*}
[farmer] &= \text{farmer} \\
[\text{donkey}] &= \text{donkey} \\
[\text{owns}] &= \lambda OS. S (\lambda x. O (\lambda y. \text{own} x y)) \\
[\text{beats}] &= \lambda OS. S (\lambda x. O (\lambda y. \text{beat} x y)) \\
[\text{who}] &= \lambda R Q x. Q x \land R (\lambda P. P x) \\
[a] &= \lambda P Q. \exists x. P x \land Q x \\
[every] &= \lambda P Q. \forall x. P x \supset Q x \\
[it] &= ???
\end{align*}
\]
Donkey Sentence Revisited

Montague-like semantic interpretation:

\[
\begin{align*}
\text{[farmer]} & = \text{farmer} \\
\text{[donkey]} & = \text{donkey} \\
\text{[owns]} & = \lambda OS. S (\lambda x. O (\lambda y. \text{own} x y)) \\
\text{[beats]} & = \lambda OS. S (\lambda x. O (\lambda y. \text{beat} x y)) \\
\text{[who]} & = \lambda R Q x. Q x \land R (\lambda P. P x) \\
\text{[a]} & = \lambda P Q. \exists x. P x \land Q x \\
\text{[every]} & = \lambda P Q. \forall x. P x \supset Q x \\
\text{[it]} & = ???
\end{align*}
\]
Dynamic interpretation:

\[
\begin{align*}
\text{[farmer]} & = \underline{\text{farmer}} \\
\text{[donkey]} & = \underline{\text{donkey}} \\
\text{[owns]} & = \lambda OS. S (\lambda x . O (\lambda y . \underline{\text{own}} x y)) \\
\text{[beats]} & = \lambda OS. S (\lambda x . O (\lambda y . \underline{\text{beat}} x y)) \\
\text{[who]} & = \lambda RQ . Q x \sqcap R (\lambda P . P x) \\
\text{[a]} & = \lambda PQ . \Sigma x . P x \sqcap Q x \\
\text{[every]} & = \lambda PQ . \Pi x . P x \sqsupset Q x \\
\text{[it]} & = \lambda Pe\phi . P (\text{sel} e) e \phi
\end{align*}
\]
With the dynamic interpretation we have that:

\[
\text{beats} \ [\text{it}] \ (\text{every} \ (\text{who} \ (\text{owns} \ (\text{a} \ (\text{donkey})))) \ (\text{farmer})))
\]
With the dynamic interpretation we have that:

\[
\text{[beats]} \text{[it]} (\text{[every]} (\text{[who]} (\text{[owns]} (\text{[a]} \text{[donkey]})) \text{[farmer]})))
\]

\(\beta\)-reduces to the following term (modulo de Morgan’s laws):

\[
\lambda e \phi. (\forall x. \text{farmer } x \supset (\forall y. \text{donkey } y \supset (\text{own } x y \supset \text{beat } x (\text{sel } (x::y::e)))))) \land \phi e
\]
With the dynamic interpretation we have that:

\[
\text{[beats][it]} (\text{[every]} (\text{[who]} (\text{[owns]} ([a] \text{[donkey]]) [\text{farmer}])))
\]

\(\beta\)-reduces to the following term (modulo de Morgan’s laws):

\[
\lambda e \phi. (\forall x. \text{farmer } x \supset (\forall y. \text{donkey } y \supset (\text{own } x y \supset \text{beat } x (\text{sel}(x::y::e)))))) \land \phi e
\]

that is, assuming that \text{sel} is a “perfect” anaphora resolution operator:

\[
\lambda e \phi. (\forall x. \text{farmer } x \supset (\forall y. \text{donkey } y \supset (\text{own } x y \supset \text{beat } x y))) \land \phi e
\]
The Higher-Order Case
The Higher-Order Case

Define type “dynamization” as follows:

\[
D \iota = \iota \\
D \Omega = \Omega \\
D(\alpha \rightarrow \beta) = D\alpha \rightarrow D\beta
\]
The Higher-Order Case

Define type “dynamization” as follows:

\[
\begin{align*}
D_\ell &= \ell \\
D_\emptyset &= \Omega \\
D(\alpha \to \beta) &= D\alpha \to D\beta
\end{align*}
\]

Then, define term “dynamization” as follows:

\[
\begin{align*}
D t &= \lambda x_1 \ldots x_n. t (R_{\text{nil}}x_1) \ldots (R_{\text{nil}}x_n) \quad \text{at type } \alpha_1 \to \ldots \alpha_n \to \ell \\
D t &= \lambda x_1 \ldots x_n e \phi. t (R_e x_1) \ldots (R_e x_n) \land (\phi e) \quad \text{at type } \alpha_1 \to \ldots \alpha_n \to o \\
R_e t &= \lambda x_1 \ldots x_n. t (D x_1) \ldots (D x_n) \quad \text{at type } D(\alpha_1 \to \ldots \alpha_n \to \ell) \\
R_e t &= \lambda x_1 \ldots x_n. t (D x_1) \ldots (D x_n) e (\lambda e. \top) \quad \text{at type } D(\alpha_1 \to \ldots \alpha_n \to o)
\end{align*}
\]
Finally, define $\lambda$-term translation as follows:

$$
\begin{align*}
\bar{x} & = x \\
\bar{\land} & = \land \\
\bar{\lor} & = \lor \\
\bar{\exists} & = \Sigma \\
\bar{\neg} & = \sim \\
\bar{k} & = Dk \quad \text{for the other constants} \\
\bar{\lambda x. t} & = \lambda x. \bar{t} \\
\bar{t u} & = \bar{t} \bar{u}
\end{align*}
$$
Finally, define λ-term translation as follows:

\[
\begin{align*}
    \overline{x} &= x \\
    \overline{\land} &= \land \\
    \overline{\exists} &= \Sigma \\
    \overline{\neg} &= \neg \\
    \overline{k} &= Dk \quad \text{for the other constants} \\
    \overline{\lambda x. t} &= \overline{\lambda x. \overline{t}} \\
    \overline{tu} &= \overline{t\overline{u}}
\end{align*}
\]

Then, for every closed term \( t \) of type \( o \), and every context \( e \), we have that:

\[
\overline{te (\overline{\lambda e. \top})} \equiv \overline{t}
\]
Comparison with existing works
Comparison with existing works

Most existing works on dynamics (DRT, Muskens’, Groenendijk & Stokhof’s) interpret dynamic propositions as binary relations on states (a.k.a., assignments or environments).
Comparison with existing works

Most existing works on dynamics (DRT, Muskens’, Groenendijk & Stokhof’s) interpret dynamic propositions as binary relations on states (a.k.a., assignments or environments).

In our setting, these would be terms of type:

\[ \gamma \rightarrow \gamma \rightarrow o, \]
Comparison with existing works

Most existing works on dynamics (DRT, Muskens’, Groenendijk & Stokhof’s) interpret dynamic propositions as binary relations on states (a.k.a., assignments or environments).

In our setting, these would be terms of type:

$$\gamma \rightarrow \gamma \rightarrow o,$$

and the semantics of Groenendijk & Stokhof’s DPL would be rephrased as follows:

$$A_d \triangleq \lambda gh. h=g \land A$$  \hspace{1cm} \text{(atomic proposition)}

$$\neg P_d \triangleq \lambda gh. h=g \land \neg(\exists k. P_d hk)$$  \hspace{1cm} \text{(negation)}

$$(P \land Q)_d \triangleq \lambda gh. \exists k. P_d gk \land Q_d kh$$  \hspace{1cm} \text{(conjunction)}

$$(\exists x. P)_d \triangleq \lambda gh. \exists k. k[x]g \land P_d kh$$  \hspace{1cm} \text{(existential)}
There exists a canonical embedding $[\cdot]$ from $\gamma \to \gamma \to o$ into $\gamma \to (\gamma \to o) \to o$:

$$[R] \triangleq \lambda e \phi. \exists e'. \phi e' \wedge Re e'$$
There exists a canonical embedding $\llbracket \cdot \rrbracket$ from $\gamma \to \gamma \to o$ into $\gamma \to (\gamma \to o) \to o$:

$$\llbracket R \rrbracket \triangleq \lambda e. \exists e'. \phi e' \land Re e'$$

Then, we have:

$$\llbracket A_d \rrbracket \equiv \overline{A}$$
$$\llbracket (\neg P)_d \rrbracket \equiv \sim \llbracket P_d \rrbracket$$
$$\llbracket (P \land Q)_d \rrbracket \equiv \llbracket P_d \rrbracket \sqcap \llbracket Q_d \rrbracket$$
There exists a canonical embedding $\llbracket \cdot \rrbracket$ from $\gamma \to \gamma \to o$ into $\gamma \to (\gamma \to o) \to o$:

$$\llbracket R \rrbracket \triangleq \lambda e \phi. \exists e'. \phi e' \land Re e'$$

Then, we have:

$$\llbracket A_d \rrbracket \equiv \overline{A}$$

$$\llbracket (\neg P)_d \rrbracket \equiv \sim \llbracket P_d \rrbracket$$

$$\llbracket (P \land Q)_d \rrbracket \equiv [P_d] \sqcap [Q_d]$$

As for the existential quantifier:

$$\llbracket (\exists x. P)_d \rrbracket = \lambda e \phi. \exists e'. \phi e' \land \exists k. k[x]e \land \llbracket P_d \rrbracket \land e'$$

$$\Sigma x. \llbracket [P_d] \rrbracket = \lambda e \phi. \exists e'. \phi e' \land \exists x. \llbracket P_d \rrbracket (x :: e) e'$$
Conclusions
Conclusions

What do we gain?
Conclusions

What do we gain?

- No destructive assignment.
Conclusions

What do we gain?

- No destructive assignment.
- Parametric in $\gamma$. 
Conclusions

What do we gain?

- No destructive assignment.
- Parametric in $\gamma$.
- Relatively independent of the underlying logic:
Conclusions

What do we gain?

- No destructive assignment.
- Parametric in $\gamma$.
- Relatively independent of the underlying logic:
  - generalizes to H.O.L.;
Conclusions

What do we gain?

- No destructive assignment.
- Parametric in $\gamma$.

- Relatively independent of the underlying logic:
  - generalizes to H.O.L.;
  - an intuitionistic version could be worked out.
Conclusions

What do we gain?

- No destructive assignment.
- Parametric in $\gamma$.
- Relatively independent of the underlying logic:
  - generalizes to H.O.L.;
  - an intuitionistic version could be worked out.
- Relies on well-established existing theories (proof-theory and model theory for free).
Conclusions

What do we gain?

• No destructive assignment.

• Parametric in $\gamma$.

• Relatively independent of the underlying logic:
  
  – generalizes to H.O.L.;
  
  – an intuitionistic version could be worked out.

• Relies on well-established existing theories (proof-theory and model theory for free).

• Deduction is replaced by computation.