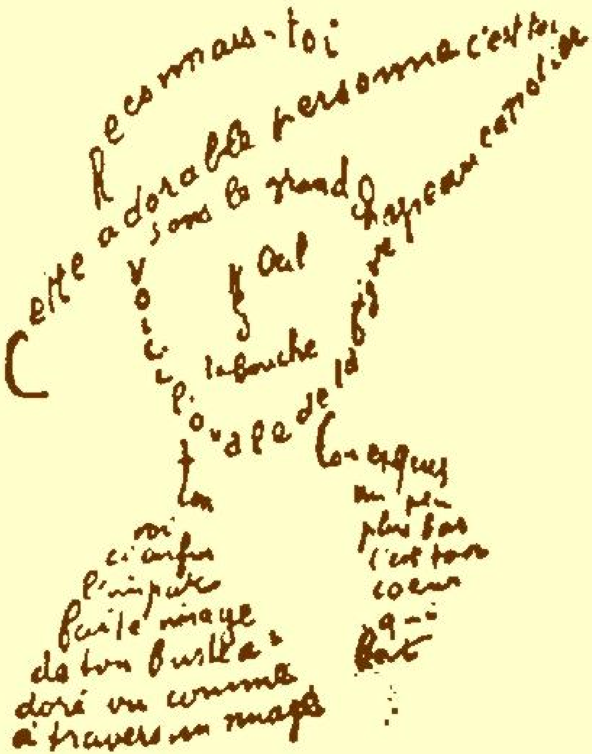

Reconnais-toi
Cette adorable personne c'est toi
sous le grand chapeau canotier
oeil
nez
la bouche
Voici l'ovale de ta figure
ton cou exquis
voici enfin l'imparfaite image de ton buste adoré vu comme à travers un nuage
un peu plus bas c'est ton coeur qui bat

Typing
Binding &
Anaphora

Dynamic Contexts as $\lambda\mu$-Terms

Philippe de Groote

Inria Nancy - Grand Est

Outline

Outline

- A type-theoretic reconstruction of DRT.

Outline

- A type-theoretic reconstruction of DRT.
- A type-theoretic view of dynamic logic.

Outline

- A type-theoretic reconstruction of DRT.
- A type-theoretic view of dynamic logic.
- Contexts as $\lambda\mu$-terms.

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, wich 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, wich 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:

- ι, the type of individuals (a.k.a. entities).
- 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:

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

We add a third atomic type, γ, 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:

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

We add a third atomic type, γ, 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:

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?

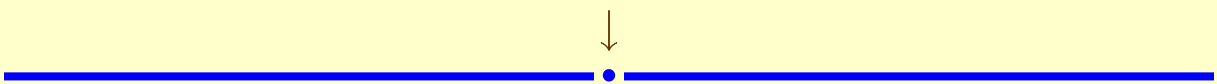
Typing the left and the right contexts

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

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?



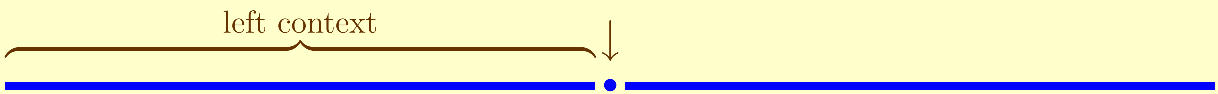
Typing the left and the right contexts

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

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?



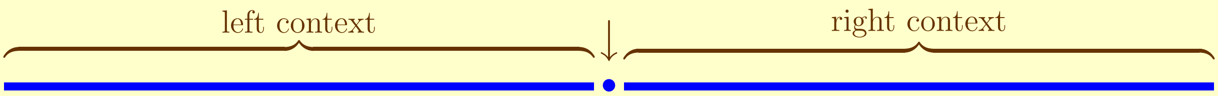
Typing the left and the right contexts

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

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?



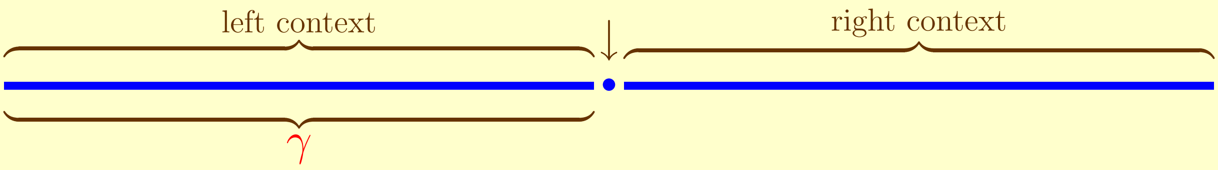
Typing the left and the right contexts

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

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?



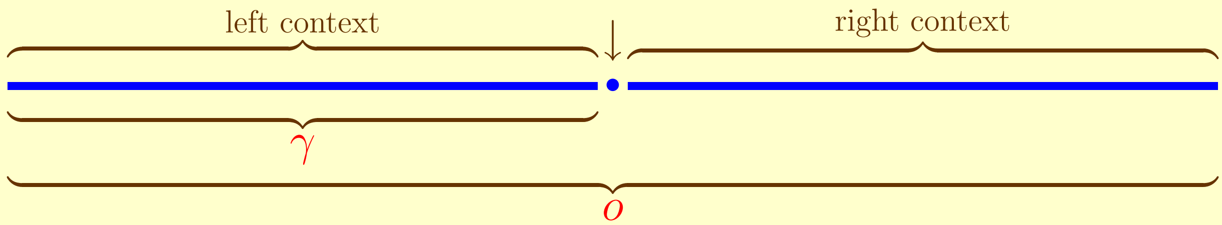
Typing the left and the right contexts

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

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

We add a third atomic type, γ, which stands for the type of the left contexts.

What about the type of the right contexts?



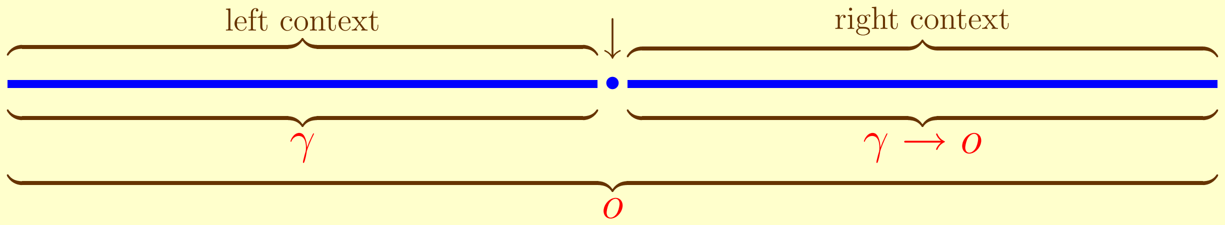
Typing the left and the right contexts

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

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

We add a third atomic type, γ, 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

$$[\![S_1.\, S_2]\!] = \lambda e \phi.\, [\![S_1]\!]\, e\, (\lambda e'.\, [\![S_2]\!]\, e'\, \phi)$$

Semantic interpretation of the syntactic categories

Semantic interpretation of the syntactic categories

Montague's interpretation

$$\begin{aligned} [\![s]\!] &= o \\ [\![n]\!] &= \iota \to o \\ [\![np]\!] &= (\iota \to o) \to o \end{aligned}$$

Semantic interpretation of the syntactic categories

Montague's interpretation

$$\begin{aligned} [\![s]\!] &= o \\ [\![n]\!] &= \iota \rightarrow o \\ [\![np]\!] &= (\iota \rightarrow o) \rightarrow o \end{aligned}$$

may be rephrased as follows:

$$[\![s]\!] = o \quad (1)$$
$$[\![n]\!] = \iota \rightarrow [\![s]\!] \quad (2)$$
$$[\![np]\!] = (\iota \rightarrow [\![s]\!]) \rightarrow [\![s]\!] \quad (3)$$

Semantic interpretation of the syntactic categories

Montague's interpretation

$$
\begin{aligned}
[\![s]\!] &= o \\
[\![n]\!] &= \iota \to o \\
[\![np]\!] &= (\iota \to o) \to o
\end{aligned}
$$

may be rephrased as follows:

$$
\begin{aligned}
[\![s]\!] &= o & (1) \\
[\![n]\!] &= \iota \to [\![s]\!] & (2) \\
[\![np]\!] &= (\iota \to [\![s]\!]) \to [\![s]\!] & (3)
\end{aligned}
$$

Replacing (1) with:

$$
[\![s]\!] = \gamma \to (\gamma \to o) \to o
$$

Semantic interpretation of the syntactic categories

Montague's interpretation

$$\begin{aligned} [\![s]\!] &= o \\ [\![n]\!] &= \iota \rightarrow o \\ [\![np]\!] &= (\iota \rightarrow o) \rightarrow o \end{aligned}$$

may be rephrased as follows:

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

Replacing (1) with:

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

we obtain:

$$\begin{aligned} [\![n]\!] &= \iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o \\ [\![np]\!] &= (\iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o) \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o \end{aligned}$$

This interpretation results in handcrafted lexical semantics such as the following:

This interpretation results in handcrafted lexical semantics such as the following:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \lambda x e \phi.\, \mathbf{farmer}\, x \,\wedge\, \phi\, e \\
[\![\text{donkey}]\!] &= \lambda x e \phi.\, \mathbf{donkey}\, x \,\wedge\, \phi\, e \\
[\![\text{owns}]\!] &= \lambda o s.\, s\,(\lambda x.\, o\,(\lambda y e \phi.\, \mathbf{own}\, x\, y \,\wedge\, \phi\, e)) \\
[\![\text{beats}]\!] &= \lambda o s.\, s\,(\lambda x.\, o\,(\lambda y e \phi.\, \mathbf{beat}\, x\, y \,\wedge\, \phi\, e)) \\
[\![\text{who}]\!] &= \lambda r n x e \phi.\, n\, x\, e\,(\lambda e.\, r\,(\lambda \psi.\, \psi\, x)\, e\, \phi) \\
[\![\text{a}]\!] &= \lambda n \psi e \phi.\, \exists x.\, n\, x\, e\,(\lambda e.\, \psi\, x\,(x{::}e)\, \phi) \\
[\![\text{every}]\!] &= \lambda n \psi e \phi.\, (\forall x.\, \neg(n\, x\, e\,(\lambda e.\, \neg(\psi\, x\,(x{::}e)\,(\lambda e.\, \top))))) \,\wedge\, \phi\, e \\
[\![\text{it}]\!] &= \lambda \psi e \phi.\, \psi\,(\mathtt{sel}\, e)\, e\, \phi
\end{aligned}
$$

This interpretation results in handcrafted lexical semantics such as the following:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \lambda x e \phi.\, \mathbf{farmer}\, x \wedge \phi\, e \\
[\![\text{donkey}]\!] &= \lambda x e \phi.\, \mathbf{donkey}\, x \wedge \phi\, e \\
[\![\text{owns}]\!] &= \lambda o s.\, s\, (\lambda x.\, o\, (\lambda y e \phi.\, \mathbf{own}\, x\, y \wedge \phi\, e)) \\
[\![\text{beats}]\!] &= \lambda o s.\, s\, (\lambda x.\, o\, (\lambda y e \phi.\, \mathbf{beat}\, x\, y \wedge \phi\, e)) \\
[\![\text{who}]\!] &= \lambda r n x e \phi.\, n\, x\, e\, (\lambda e.\, r\, (\lambda \psi.\, \psi\, x)\, e\, \phi) \\
[\![\text{a}]\!] &= \lambda n \psi e \phi.\, \exists x.\, n\, x\, e\, (\lambda e.\, \psi\, x\, (x{::}e)\, \phi) \\
[\![\text{every}]\!] &= \lambda n \psi e \phi.\, (\forall x.\, \neg(n\, x\, e\, (\lambda e.\, \neg(\psi\, x\, (x{::}e)\, (\lambda e.\, \top))))) \wedge \phi\, e \\
[\![\text{it}]\!] &= \lambda \psi e \phi.\, \psi\, (\texttt{sel}\, e)\, e\, \phi
\end{aligned}
$$

...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 Type-Theoretic View of Dynamic Logic

A Type-Theoretic View of Dynamic Logic

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

A Type-Theoretic View of Dynamic Logic

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

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 Type-Theoretic View of Dynamic Logic

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

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 λ-calculus, the terms of which are built upon asignature including the following constants:

Formal Framework

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

FIRST-ORDER LOGIC

$$\top \ : \ o \qquad (truth)$$

$$\neg \ : \ o \rightarrow o \qquad (negation)$$

$$\wedge \ : \ o \rightarrow o \rightarrow o \qquad (conjunction)$$

$$\exists \ : \ (\iota \rightarrow o) \rightarrow o \qquad (existential\ quantification)$$

Formal Framework

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

FIRST-ORDER LOGIC

$$\begin{array}{rcll} \top & : & o & (\textit{truth}) \\ \neg & : & o \rightarrow o & (\textit{negation}) \\ \wedge & : & o \rightarrow o \rightarrow o & (\textit{conjunction}) \\ \exists & : & (\iota \rightarrow o) \rightarrow o & (\textit{existential quantification}) \end{array}$$

DYNAMIC PRIMITIVES

$$\begin{array}{rcll} :: & : & \iota \rightarrow \gamma \rightarrow \gamma & (\textit{context updating}) \\ \mathsf{sel} & : & \gamma \rightarrow \iota & (\textit{choice operator}) \end{array}$$

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:

$$A \sqcap B \triangleq \lambda e \phi .\, A\, e\, (\lambda e .\, B\, e\, \phi)$$

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

We do not want the continuation of the discourse to fall into the scope of the negation. Consequently, negation must be defined as follows:

$$\sim A \triangleq \lambda e \phi .\, \neg\, (A\, e\, (\lambda e .\, \top)) \wedge \phi\, e$$

Implication and Universal Quantification

Implication and Universal Quantification

These are defined using de Morgan's laws:

$$
\begin{aligned}
A \sqsupset B &\triangleq {\sim}(A \sqcap {\sim}B) \\
\Pi x.\, P\, x &\triangleq {\sim}\Sigma x.\, {\sim}(P\, x)
\end{aligned}
$$

Implication and Universal Quantification

These are defined using de Morgan's laws:

$$A \sqsupset B \triangleq {\sim}(A \sqcap {\sim}B)$$
$$\Pi x.\, P\, x \triangleq {\sim}\Sigma x.\, {\sim}(P\, x)$$

Embedding of first-order logic into dynamic logic

Implication and Universal Quantification

These are defined using de Morgan's laws:

$$\begin{aligned} A \sqsupset B &\triangleq {\sim}(A \sqcap {\sim}B) \\ \Pi x.\, P\, x &\triangleq {\sim}\Sigma x.\, {\sim}(P\, x) \end{aligned}$$

Embedding of first-order logic into dynamic logic

$$\begin{aligned} \overline{R\, t_1 \, \ldots \, t_n} &= \lambda e \phi.\, R\, t_1 \, \ldots \, t_n \wedge \phi\, e \\ \overline{\neg A} &= {\sim}\overline{A} \\ \overline{A \wedge B} &= \overline{A} \sqcap \overline{B} \\ \overline{\exists x.\, A} &= \Sigma x.\, \overline{A} \end{aligned}$$

Implication and Universal Quantification

These are defined using de Morgan's laws:

$$\begin{array}{rcl} A \sqsupset B & \triangleq & \sim(A \sqcap \sim B) \\ \Pi x.\, P\, x & \triangleq & \sim\Sigma x.\sim(P\, x) \end{array}$$

Embedding of first-order logic into dynamic logic

$$\begin{array}{rcl} \overline{R\, t_1\, \ldots\, t_n} & = & \lambda e\phi.\, R\, t_1\, \ldots\, t_n \wedge \phi\, e \\ \overline{\neg A} & = & \sim\overline{A} \\ \overline{A \wedge B} & = & \overline{A} \sqcap \overline{B} \\ \overline{\exists x.\, A} & = & \Sigma x.\, \overline{A} \end{array}$$

This embedding is such that, for every term e of type γ:

$$A \equiv \overline{A}\, e\, (\lambda e.\, \top)$$

Donkey Sentence Revisited

Donkey Sentence Revisited

Montague-like semantic interpretation:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \textbf{farmer} \\
[\![\text{donkey}]\!] &= \textbf{donkey} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \textbf{own}\, x\, y)) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \textbf{beat}\, x\, y)) \\
[\![\text{who}]\!] &= \lambda RQx.\, Q\, x \wedge R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] &= \lambda PQ.\, \exists x.\, P\, x \wedge Q\, x \\
[\![\text{every}]\!] &= \lambda PQ.\, \forall x.\, P\, x \supset Q\, x \\
[\![\text{it}]\!] &= ???
\end{aligned}
$$

Donkey Sentence Revisited

Montage-like semantic interpretation:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \textbf{farmer} \\
[\![\text{donkey}]\!] &= \textbf{donkey} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \textbf{own}\, x\, y)) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \textbf{beat}\, x\, y)) \\
[\![\text{who}]\!] &= \lambda RQx.\, Q\, x \wedge R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] &= \lambda PQ.\, \exists x.\, P\, x \wedge Q\, x \\
[\![\text{every}]\!] &= \lambda PQ.\, \forall x.\, P\, x \supset Q\, x \\
[\![\text{it}]\!] &= ???
\end{aligned}
$$

Dynamic interpretation:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \overline{\textbf{farmer}} \\
[\![\text{donkey}]\!] &= \overline{\textbf{donkey}} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\textbf{own}}\, x\, y)) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\textbf{beat}}\, x\, y)) \\
[\![\text{who}]\!] &= \lambda RQx.\, Q\, x \sqcap R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] &= \lambda PQ.\, \Sigma x.\, P\, x \sqcap Q\, x \\
[\![\text{every}]\!] &= \lambda PQ.\, \Pi x.\, P\, x \sqsupset Q\, x \\
[\![\text{it}]\!] &= \lambda Pe\phi.\, P\,(\texttt{sel}\, e)\, e\, \phi
\end{aligned}
$$

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:

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

β-reduces to the following term (modulo de Morgan's laws):

$$\lambda e\phi.\,(\forall x.\,\mathbf{farmer}\,x \supset (\forall y.\,\mathbf{donkey}\,y \supset (\mathbf{own}\,x\,y \supset \mathbf{beat}\,x\,(\mathsf{sel}\,(x{::}y{::}e))))) \wedge \phi\,e$$

With the dynamic interpretation we have that:

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

β-reduces to the following term (modulo de Morgan's laws):

$$\lambda e\phi.\,(\forall x.\,\mathbf{farmer}\,x \supset (\forall y.\,\mathbf{donkey}\,y \supset (\mathbf{own}\,x\,y \supset \mathbf{beat}\,x\,(\mathsf{sel}\,(x{::}y{::}e))))) \wedge \phi\,e$$

that is, assuming that sel is a "perfect" anaphora resolution operator:

$$\lambda e\phi.\,(\forall x.\,\mathbf{farmer}\,x \supset (\forall y.\,\mathbf{donkey}\,y \supset (\mathbf{own}\,x\,y \supset \mathbf{beat}\,x\,y))) \wedge \phi\,e$$

Observations

Observations

- Dynamic propositions are usually seen as contexts transformers.

Observations

- Dynamic propositions are usually seen as contexts transformers.
- Here, we transform contexts (γ) into "type-raised" contexts ($(\gamma \rightarrow o) \rightarrow o$).

Observations

- Dynamic propositions are usually seen as contexts transformers.
- Here, we transform contexts (γ) into "type-raised" contexts ($(\gamma \rightarrow o) \rightarrow o$).
- Interpreting $(\gamma \rightarrow o) \rightarrow o$ as a double negation, we may identify it with γ.

Observations

- Dynamic propositions are usually seen as contexts transformers.
- Here, we transform contexts (γ) into "type-raised" contexts ($(\gamma \to o) \to o$).
- Interpreting $(\gamma \to o) \to o$ as a double negation, we may identify it with γ.
- To this end, we must use a calculus whose type system corresponds to classical logic.

Observations

- Dynamic propositions are usually seen as contexts transformers.
- Here, we transform contexts (γ) into "type-raised" contexts ($(\gamma \to o) \to o$).
- Interpreting $(\gamma \to o) \to o$ as a double negation, we may identify it with γ.
- To this end, we must use a calculus whose type system corresponds to classical logic.
- The $\lambda\mu$-calculus is such a calculus.

Call by Value $\lambda\mu$-calculus

Call by Value $\lambda\mu$-calculus

syntax

Call by Value $\lambda\mu$-calculus

syntax

The set of $\lambda\mu$-terms is built upon two disjoint sets of variables (namely, λ-variables and μ) according to the following rules:

$$
\begin{array}{rcll}
T & ::= & c & \text{(constant)} \\
 & | & x & \text{(λ-variable)} \\
 & | & (\lambda x.\, T) & \text{(λ-abstraction)} \\
 & | & (T\, T) & \text{(application)} \\
 & | & (\mu a.\, T) & \text{(μ-abstraction)} \\
 & | & a\,(T) & \text{(naming)} \\
 & | & \langle T \rangle & \text{(reset)}
\end{array}
$$

Call by Value $\lambda\mu$-calculus

syntax

The set of $\lambda\mu$-terms is built upon two disjoint sets of variables (namely, λ-variables and μ) according to the following rules:

$$
\begin{array}{rcll}
T & ::= & c & \text{(constant)} \\
& | & x & (\lambda\text{-variable}) \\
& | & (\lambda x. T) & (\lambda\text{-abstraction}) \\
& | & (T\,T) & \text{(application)} \\
& | & (\mu a. T) & (\mu\text{-abstraction}) \\
& | & a\,(T) & \text{(naming)} \\
& | & \langle T \rangle & \text{(reset)}
\end{array}
$$

The set of values is defined as follows:

$$V \ ::= \ c\,V \ldots V \ | \ x \ | \ \lambda x. T$$

Intended operational meaning

Intended operational meaning

$$(\mu a. \ldots a\,(t_1) \ldots a\,(t_2) \ldots a\,(t_n) \ldots) \quad u$$

Intended operational meaning

$$\underbrace{(\mu a.\ \ldots a\,(t_1)\ldots a\,(t_2)\ldots a\,(t_n)\ldots)}_{\alpha\to\beta}\quad u$$

Intended operational meaning

$$\underbrace{(\mu a.\ \ldots a\,(t_1)\ldots a\,(t_2)\ldots a\,(t_n)\ldots)}_{\alpha\to\beta}\ \underbrace{u}_{\alpha}$$

Intended operational meaning

$$\underbrace{(\mu a.\ \ldots a\,\overbrace{(t_1)}^{\alpha\to\beta}\ldots a\,\overbrace{(t_2)}^{\alpha\to\beta}\ldots a\,\overbrace{(t_n)}^{\alpha\to\beta}\ldots)}_{\alpha\to\beta}\ \underbrace{u}_{\alpha}$$

Intended operational meaning

$$\underbrace{(\mu a.\ \ldots a\,\overbrace{(t_1)}^{\alpha\to\beta}\ldots a\,\overbrace{(t_2)}^{\alpha\to\beta}\ldots a\,\overbrace{(t_n)}^{\alpha\to\beta}\ldots)}_{\alpha\to\beta}\ \underbrace{u}_{\alpha}$$

$$\to\quad (\mu a.\ \ldots a\,(t_1\,u)\ldots a\,(t_2\,u)\ldots a\,(t_n\,u)\ldots)$$

Intended operational meaning

$$\underbrace{(\mu a.\ \ldots a\,\overbrace{(t_1)}^{\alpha\to\beta} \ldots a\,\overbrace{(t_2)}^{\alpha\to\beta} \ldots a\,\overbrace{(t_n)}^{\alpha\to\beta} \ldots)}_{\alpha\to\beta}\ \underbrace{u}_{\alpha}$$

$$\to \quad (\mu a.\ \ldots a\,(t_1\,u) \ldots a\,(t_2\,u) \ldots a\,(t_n\,u) \ldots)$$

Symmetrically:

$$f\,(\mu a.\ \ldots a\,(t_1) \ldots a\,(t_2) \ldots a\,(t_n) \ldots) \quad \to \quad (\mu a.\ \ldots a\,(f\,t_1) \ldots a\,(f\,t_2) \ldots a\,(f\,t_n) \ldots)$$

Typing rules

Typing rules

$$\Gamma, x : \alpha; \Delta^{\perp} \vdash x : \alpha$$

$$\frac{\Gamma, x : \alpha; \Delta^{\perp} \vdash t : \beta}{\Gamma; \Delta^{\perp} \vdash \lambda x.\, t : \alpha \to \beta} \qquad \frac{\Gamma; \Delta^{\perp} \vdash t : \alpha \to \beta \qquad \Gamma; \Delta^{\perp} \vdash u : \alpha}{\Gamma; \Delta^{\perp} \vdash t\, u : \beta}$$

Typing rules

$$\Gamma,\, x : \alpha;\, \Delta^{\perp} \vdash x : \alpha$$

$$\frac{\Gamma,\, x : \alpha;\, \Delta^{\perp} \vdash t : \beta}{\Gamma;\, \Delta^{\perp} \vdash \lambda x.\, t : \alpha \to \beta} \qquad \frac{\Gamma;\, \Delta^{\perp} \vdash t : \alpha \to \beta \qquad \Gamma;\, \Delta^{\perp} \vdash u : \alpha}{\Gamma;\, \Delta^{\perp} \vdash t\, u : \beta}$$

$$\frac{\Gamma;\, \Delta^{\perp},\, a : \alpha^{\perp} \vdash t : \perp}{\Gamma;\, \Delta^{\perp} \vdash \mu a.\, t : \alpha} \qquad \frac{\Gamma;\, \Delta^{\perp},\, a : \alpha^{\perp} \vdash t : \alpha}{\Gamma;\, \Delta^{\perp},\, a : \alpha^{\perp} \vdash a\, (t) : \perp}$$

Typing rules

$$\Gamma, x : \alpha; \Delta^{\perp} \vdash x : \alpha$$

$$\frac{\Gamma, x : \alpha; \Delta^{\perp} \vdash t : \beta}{\Gamma; \Delta^{\perp} \vdash \lambda x.\, t : \alpha \to \beta} \qquad \frac{\Gamma; \Delta^{\perp} \vdash t : \alpha \to \beta \qquad \Gamma; \Delta^{\perp} \vdash u : \alpha}{\Gamma; \Delta^{\perp} \vdash t\, u : \beta}$$

$$\frac{\Gamma; \Delta^{\perp}, a : \alpha^{\perp} \vdash t : \perp}{\Gamma; \Delta^{\perp} \vdash \mu a.\, t : \alpha} \qquad \frac{\Gamma; \Delta^{\perp}, a : \alpha^{\perp} \vdash t : \alpha}{\Gamma; \Delta^{\perp}, a : \alpha^{\perp} \vdash a\,(t) : \perp}$$

$$\frac{\Gamma; \Delta^{\perp} \vdash t : \perp}{\Gamma; \Delta^{\perp} \vdash \langle t \rangle : \perp}$$

Reduction rules

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \rightarrow t[x{:=}v]$ (v is a value)

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \rightarrow t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \rightarrow \mu b.\, t[a\,(\_){:=}b\,(\_\,u)]$

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \longrightarrow t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \longrightarrow \mu b.\, t[a\,(\_){:=}b\,(\_\, u)]$

μ_V-reduction:

$v\,(\mu a.\, t) \longrightarrow \mu b.\, t[a\,(\_){:=}b\,(v\,\_)]$ (v is a value)

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \rightarrow t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \rightarrow \mu b.\, t[a\,(\_){:=}b\,(\_\, u)]$

μ_V-reduction:

$v\,(\mu a.\, t) \rightarrow \mu b.\, t[a\,(\_){:=}b\,(v\, \_)]$ (v is a value)

Renaming:

$a\,(\mu b.\, t) \rightarrow t[b{:=}a]$

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \rightarrow t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \rightarrow \mu b.\, t[a\,(\_){:=}b\,(\_\, u)]$

μ_V-reduction:

$v\,(\mu a.\, t) \rightarrow \mu b.\, t[a\,(\_){:=}b\,(v\, \_)]$ (v is a value)

Renaming:

$a\,(\mu b.\, t) \rightarrow t[b{:=}a]$

η_μ-reduction:

$\mu a.\, a\,(t) \rightarrow t$ (a does not occur in t)

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \to t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \to \mu b.\, t[a\,(\_){:=}b\,(\_\, u)]$

μ_V-reduction:

$v\,(\mu a.\, t) \to \mu b.\, t[a\,(\_){:=}b\,(v\, \_)]$ (v is a value)

Renaming:

$a\,(\mu b.\, t) \to t[b{:=}a]$

η_μ-reduction:

$\mu a.\, a\,(t) \to t$ (a does not occur in t)

Reset$_\mu$:

$\langle \mu a.\, t\rangle \to \langle t[a\,(\_){:=}\_]\rangle$

Reduction rules

β_V-reduction:

$(\lambda x.\, t)\, v \rightarrow t[x{:=}v]$ (v is a value)

μ-reduction:

$(\mu a.\, t)\, u \rightarrow \mu b.\, t[a\,(\_){:=}b\,(\_\, u)]$

μ_V-reduction:

$v\,(\mu a.\, t) \rightarrow \mu b.\, t[a\,(\_){:=}b\,(v\,\_)]$ (v is a value)

Renaming:

$a\,(\mu b.\, t) \rightarrow t[b{:=}a]$

η_μ-reduction:

$\mu a.\, a\,(t) \rightarrow t$ (a does not occur in t)

Reset$_\mu$:

$\langle \mu a.\, t \rangle \rightarrow \langle t[a\,(\_){:=}\_] \rangle$

Reset$_V$:

$\langle v \rangle \rightarrow v$ (v is a value)

Representig Contexts as $\lambda\mu$-Terms

Representig Contexts as $\lambda\mu$-Terms

Assumptions

Representig Contexts as $\lambda\mu$-Terms

Assumptions

- We take $\perp$ to be o;

Representig Contexts as $\lambda\mu$-Terms

Assumptions

- We take $\perp$ to be o;
- The type of dynamic proposition is defined as $\Omega = \gamma \rightarrow \gamma$

Representig Contexts as $\lambda\mu$-Terms

Assumptions

- We take $\bot$ to be o;
- The type of dynamic proposition is defined as $\Omega = \gamma \rightarrow \gamma$

Dynamic interpretation of atomic propositions

Representig Contexts as $\lambda\mu$-Terms

Assumptions

- We take $\bot$ to be o;
- The type of dynamic proposition is defined as $\Omega = \gamma \to \gamma$

Dynamic interpretation of atomic propositions

$$\overline{R\, t_1 \, \ldots \, t_n} = \lambda e.\, \mu c.\, R\, t_1 \, \ldots \, t_n \wedge c\,(e)$$

Representig Contexts as $\lambda\mu$-Terms

Assumptions

- We take $\bot$ to be o;
- The type of dynamic proposition is defined as $\Omega = \gamma \to \gamma$

Dynamic interpretation of atomic propositions

$$\overline{R\,t_1\,\ldots\,t_n} = \lambda e.\,\mu c.\,R\,t_1\,\ldots\,t_n \wedge c\,(e)$$

Indeed:

$$\dfrac{\dfrac{\dfrac{e:\gamma;\,c:\gamma^o \vdash R\,t_1\,\ldots\,t_n : o \qquad \dfrac{e:\gamma;\,c:\gamma^o \vdash e:\gamma}{e:\gamma;\,c:\gamma^o \vdash c\,(e):o}}{e:\gamma;\,c:\gamma^o \vdash R\,t_1\,\ldots\,t_n \wedge c\,(e):o}}{e:\gamma;\, \vdash \mu c.\,R\,t_1\,\ldots\,t_n \wedge c\,(e):\gamma}}{\vdash \lambda e.\,\mu c.\,R\,t_1\,\ldots\,t_n \wedge c\,(e):\gamma \to \gamma}$$

Composition of two dynamic propositions

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e)$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) \quad = \quad \lambda e.\,(\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e)$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\begin{aligned}\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) &= \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e)\\ &= \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,((\lambda e_1.\,\mu c_1.\,p_1 \wedge c_1\,(e_1))\,e)\end{aligned}$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\begin{array}{rcl}
\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) & = & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e) \\
& = & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,((\lambda e_1.\, \mu c_1.\, p_1 \wedge c_1\,(e_1))\,e) \\
& \longrightarrow & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,(\mu c_1.\, p_1 \wedge c_1\,(e))
\end{array}$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\begin{array}{rcl}
\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) & = & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e) \\
& = & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,((\lambda e_1.\, \mu c_1.\, p_1 \wedge c_1\,(e_1))\,e) \\
& \longrightarrow & \lambda e.\, (\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,(\mu c_1.\, p_1 \wedge c_1\,(e)) \\
& \longrightarrow & \lambda e.\, \mu c_1.\, p_1 \wedge c_1\,((\lambda e_2.\, \mu c_2.\, p_2 \wedge c_2\,(e_2))\,e)
\end{array}$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\begin{aligned}
\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) &= \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e)\\
&= \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,((\lambda e_1.\,\mu c_1.\,p_1 \wedge c_1\,(e_1))\,e)\\
&\longrightarrow \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,(\mu c_1.\,p_1 \wedge c_1\,(e))\\
&\longrightarrow \lambda e.\,\mu c_1.\,p_1 \wedge c_1\,((\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,e)\\
&\longrightarrow \lambda e.\,\mu c_1.\,p_1 \wedge c_1\,(\mu c_2.\,p_2 \wedge c_2\,(e))
\end{aligned}$$

Composition of two dynamic propositions

$$\lambda e.\, B\,(A\,e)$$

Example:

$$\begin{array}{rcl}
\lambda e.\, \overline{p_2}\,(\overline{p_1}\,e) & = & \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,(\overline{p_2}\,e) \\
& = & \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,((\lambda e_1.\,\mu c_1.\,p_1 \wedge c_1\,(e_1))\,e) \\
& \longrightarrow & \lambda e.\,(\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,(\mu c_1.\,p_1 \wedge c_1\,(e)) \\
& \longrightarrow & \lambda e.\,\mu c_1.\,p_1 \wedge c_1\,((\lambda e_2.\,\mu c_2.\,p_2 \wedge c_2\,(e_2))\,e) \\
& \longrightarrow & \lambda e.\,\mu c_1.\,p_1 \wedge c_1\,(\mu c_2.\,p_2 \wedge c_2\,(e)) \\
& \longrightarrow & \lambda e.\,\mu c_1.\,p_1 \wedge (p_2 \wedge c_1\,(e))
\end{array}$$

Reading a dynamic proposition

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$\mathsf{READ}\, e\, \overline{p}$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\mathsf{READ}\, e\, \overline{p} \quad = \quad \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{aligned} \mathsf{READ}\, e\, \overline{p} &= \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\ &= \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\, (e))\, e) \rangle \end{aligned}$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{array}{rcl} \mathsf{READ}\, e\, \overline{p} & = & \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\ & = & \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\, (e))\, e) \rangle \\ & \longrightarrow & \langle (\lambda x.\, \top)\, (\mu c.\, p \wedge c\, (e)) \rangle \end{array}$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{array}{rcl} \mathsf{READ}\, e\, \overline{p} & = & \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\ & = & \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\, (e))\, e) \rangle \\ & \longrightarrow & \langle (\lambda x.\, \top)\, (\mu c.\, p \wedge c\, (e)) \rangle \\ & \longrightarrow & \langle \mu c.\, p \wedge c\, ((\lambda x.\, \top)\, e) \rangle \end{array}$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{array}{rcl}
\mathsf{READ}\, e\, \overline{p} & = & \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\
 & = & \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\,(e))\, e) \rangle \\
 & \longrightarrow & \langle (\lambda x.\, \top)\, (\mu c.\, p \wedge c\,(e)) \rangle \\
 & \longrightarrow & \langle \mu c.\, p \wedge c\,((\lambda x.\, \top)\, e) \rangle \\
 & \longrightarrow & \langle \mu c.\, p \wedge c\,(\top) \rangle
\end{array}$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{aligned}
\mathsf{READ}\, e\, \overline{p} &= \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\
&= \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\, (e))\, e) \rangle \\
&\longrightarrow \langle (\lambda x.\, \top)\, (\mu c.\, p \wedge c\, (e)) \rangle \\
&\longrightarrow \langle \mu c.\, p \wedge c\, ((\lambda x.\, \top)\, e) \rangle \\
&\longrightarrow \langle \mu c.\, p \wedge c\, (\top) \rangle \\
&\longrightarrow p \wedge \top
\end{aligned}$$

Reading a dynamic proposition

$$\mathsf{READ}\, e\, A = \langle (\lambda x.\, \top)\, (A\, e) \rangle$$

Example:

$$\begin{array}{rcl}
\mathsf{READ}\, e\, \overline{p} & = & \langle (\lambda x.\, \top)\, (\overline{p}\, e) \rangle \\
& = & \langle (\lambda x.\, \top)\, ((\lambda e.\, \mu c.\, p \wedge c\, (e))\, e) \rangle \\
& \longrightarrow & \langle (\lambda x.\, \top)\, (\mu c.\, p \wedge c\, (e)) \rangle \\
& \longrightarrow & \langle \mu c.\, p \wedge c\, ((\lambda x.\, \top)\, e) \rangle \\
& \longrightarrow & \langle \mu c.\, p \wedge c\, (\top) \rangle \\
& \longrightarrow & p \wedge \top \\
& \equiv & p
\end{array}$$

Dynamic Logic Revisited

Dynamic Logic Revisited

Conjunction

Dynamic Logic Revisited

Conjunction

Conjunction amounts to functional composition.

$$A \sqcap B \triangleq \lambda e.\, B\,(A\,e)$$

Dynamic Logic Revisited

Conjunction

Conjunction amounts to functional composition.

$$A \sqcap B \triangleq \lambda e.\, B\,(A\,e)$$

Existential quantification

Dynamic Logic Revisited

Conjunction

Conjunction amounts to functional composition.

$$A \sqcap B \triangleq \lambda e.\, B\,(A\,e)$$

Existential quantification

Existential quantification introduces "reference markers" by updating the context:

$$\Sigma x.\, P\,x \triangleq \lambda e.\, \mu c.\, \exists x.\, c\,(P\,x\,(x{::}e))$$

Dynamic Logic Revisited

Conjunction

Conjunction amounts to functional composition.

$$A \sqcap B \triangleq \lambda e.\, B\,(A\,e)$$

Existential quantification

Existential quantification introduces "reference markers" by updating the context:

$$\Sigma x.\, P\,x \triangleq \lambda e.\, \mu c.\, \exists x.\, c\,(P\,x\,(x{::}e))$$

Negation

Dynamic Logic Revisited

Conjunction

Conjunction amounts to functional composition.

$$A \sqcap B \triangleq \lambda e.\, B\,(A\,e)$$

Existential quantification

Existential quantification introduces "reference markers" by updating the context:

$$\Sigma x.\, P\,x \triangleq \lambda e.\, \mu c.\, \exists x.\, c\,(P\,x\,(x{::}e))$$

Negation

The scope of the negation must be restricted to the current proposition and, consequently, "reset" the continuation:

$$\sim A \triangleq \lambda e.\, \mu c.\, \neg(\mathsf{READ}\,e\,A) \wedge c\,(e)$$

Implication and Universal Quantification

Implication and Universal Quantification

They could be defined using de Morgan's laws. Alternative (simpler) definitions are as follows:

Implication and Universal Quantification

They could be defined using de Morgan's laws. Alternative (simpler) definitions are as follows:

$$A \sqsupset B \triangleq \lambda e.\, \mu c.\, \neg\langle(\lambda e.\, \neg(\mathsf{READ}\, e\, B))\, (A\, e)\rangle \wedge c\,(e)$$

$$\Pi x.\, P\, x \triangleq \lambda e.\, \mu c.\, (\forall x.\, \mathsf{READ}\,(x{::}e)\,(P\, x)) \wedge c\,(e)$$

Donkey Sentence Again

Donkey Sentence Again

The dynamic lexical semantics is kept unchanged:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \overline{\textbf{farmer}} \\
[\![\text{donkey}]\!] &= \overline{\textbf{donkey}} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\textbf{own}}\, x\, y)) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\textbf{beat}}\, x\, y)) \\
[\![\text{who}]\!] &= \lambda RQx.\, Q\, x \sqcap R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] &= \lambda PQ.\, \Sigma x.\, P\, x \sqcap Q\, x \\
[\![\text{every}]\!] &= \lambda PQ.\, \Pi x.\, P\, x \sqsupset Q\, x \\
[\![\text{it}]\!] &= \lambda Pe.\, P\,(\texttt{sel}\, e)\, e
\end{aligned}
$$

Donkey Sentence Again

The dynamic lexical semantics is kept unchanged:

$$
\begin{array}{rcl}
[\![\text{farmer}]\!] & = & \overline{\mathbf{farmer}} \\
[\![\text{donkey}]\!] & = & \overline{\mathbf{donkey}} \\
[\![\text{owns}]\!] & = & \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{own}}\, x\, y)) \\
[\![\text{beats}]\!] & = & \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{beat}}\, x\, y)) \\
[\![\text{who}]\!] & = & \lambda RQx.\, Q\, x \sqcap R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] & = & \lambda PQ.\, \Sigma x.\, P\, x \sqcap Q\, x \\
[\![\text{every}]\!] & = & \lambda PQ.\, \Pi x.\, P\, x \sqsupset Q\, x \\
[\![\text{it}]\!] & = & \lambda Pe.\, P\,(\mathtt{sel}\, e)\, e
\end{array}
$$

Then, we have that:

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

Donkey Sentence Again

The dynamic lexical semantics is kept unchanged:

$$
\begin{aligned}
[\![\text{farmer}]\!] &= \overline{\mathbf{farmer}} \\
[\![\text{donkey}]\!] &= \overline{\mathbf{donkey}} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{own}}\, x\, y)) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{beat}}\, x\, y)) \\
[\![\text{who}]\!] &= \lambda RQx.\, Q\, x \sqcap R\,(\lambda P.\, P\, x) \\
[\![\text{a}]\!] &= \lambda PQ.\, \Sigma x.\, P\, x \sqcap Q\, x \\
[\![\text{every}]\!] &= \lambda PQ.\, \Pi x.\, P\, x \sqsupset Q\, x \\
[\![\text{it}]\!] &= \lambda Pe.\, P\,(\mathtt{sel}\, e)\, e
\end{aligned}
$$

Then, we have that:

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

reduces to the following term :

$$\lambda e.\, \mu c.\, (\forall x.\, \mathbf{farmer}\, x \supset (\forall y.\, \mathbf{donkey}\, y \supset (\mathbf{own}\, x\, y \supset \mathbf{beat}\, x\, (\mathsf{sel}\,(x{::}y{::}e))))) \wedge c\,(e)$$

Conclusions

Conclusions

- The $\lambda\mu$-calculus allows propositions (o) and contexts (γ) to be mixed in a same term.

Conclusions

- The $\lambda\mu$-calculus allows propositions (o) and contexts (γ) to be mixed in a same term.
- To this end, the use of the "reset" operator is central.

Conclusions

- The $\lambda\mu$-calculus allows propositions (o) and contexts (γ) to be mixed in a same term.
- To this end, the use of the "reset" operator is central.
- The reset operator we have used is rather "cheap".

Conclusions

- The $\lambda\mu$-calculus allows propositions (o) and contexts (γ) to be mixed in a same term.
- To this end, the use of the "reset" operator is central.
- The reset operator we have used is rather "cheap".
- More powerful versions (w.r.t. typing) should allow other dynamic phenomena (e.g. definite clauses, focus, presuppositions) to be handled similarly.