New Progress in Continuation-Based Dynamic Logic

Philippe de Groote

Inria Nancy - Grand Est

Logic and Algorithms in Computational Linguistics
Stockholm, August 28–31 2018

Outline

1. A type-theoretic reconstruction of dynamic logic
2. Need for more flexibility
3. A more flexible framework

Outline

The problem of dynamic binding

The problem of dynamic binding

An old problem:

A man enters the room. He smiles.

$[\![\text{A man enters the room}]\!] = \exists x.\mathbf{man}(x) \land \mathbf{enters\_the\_room}(x)$. x is bound.

$[\![\text{He smiles}]\!] = \mathbf{smiles}(x)$. x is free.

The problem of dynamic binding

An old problem:

A man enters the room. He smiles.

$[\![$A man enters the room$]\!] = \exists x.\mathbf{man}(x) \wedge \mathbf{enters_the_room}(x)$. x is bound.

$[\![$He smiles$]\!] = \mathbf{smiles}(x)$. x is free.

How can we get from these:

$$[\![\text{A man enters the room. He smiles}]\!] = \exists x.\mathbf{man}(x) \wedge \mathbf{enters\_the\_room}(x) \wedge \mathbf{smiles}(x).$$

The problem of dynamic binding

An old problem:

A man enters the room. He smiles.

$[\![\text{A man enters the room}]\!] = \exists x.\mathbf{man}(x) \wedge \mathbf{enters\_the\_room}(x)$. x is bound.

$[\![\text{He smiles}]\!] = \mathbf{smiles}(x)$. x is free.

How can we get from these:

$$[\![\text{A man enters the room. He smiles}]\!] = \exists x.\mathbf{man}(x) \wedge \mathbf{enters\_the\_room}(x) \wedge \mathbf{smiles}(x).$$

A well known solution: DRT.

- The reference markers of DRT act as existential quantifiers.
- Nevertheless, from a technical point of view, they must be considered as free variables.

Expressing propositions in context

Expressing propositions in context

"The key idea behind (...) Discourse Representation Theory is that each new sentence of a discourse is interpreted in the context provided by the sentences preceding it."

van Eijck and Kamp.
Representing Discourse in Context.
In *Handbook of Logic and Language*.
Elsevier, 1997.

Expressing propositions in context

"The key idea behind (...) Discourse Representation Theory is that each new sentence of a discourse is interpreted in the context provided by the sentences preceding it."

van Eijck and Kamp.
Representing Discourse in Context.
In *Handbook of Logic and Language*.
Elsevier, 1997.

We go two steps further:

Expressing propositions in context

"The key idea behind (...) Discourse Representation Theory is that each new sentence of a discourse is interpreted in the context provided by the sentences preceding it."

van Eijck and Kamp.
Representing Discourse in Context.
In *Handbook of Logic and Language*.
Elsevier, 1997.

We go two steps further:

- We will interpret a sentence according to both its left and right contexts.

Expressing propositions in context

"The key idea behind (...) Discourse Representation Theory is that each new sentence of a discourse is interpreted in the context provided by the sentences preceding it."

van Eijck and Kamp.
Representing Discourse in Context.
In *Handbook of Logic and Language*.
Elsevier, 1997.

We go two steps further:

- We will interpret a sentence according to both its left and right contexts.
- These two kinds of contexts will be abstracted 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?

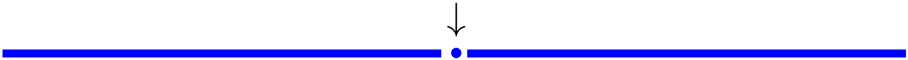
Typing the left and the right contexts

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?



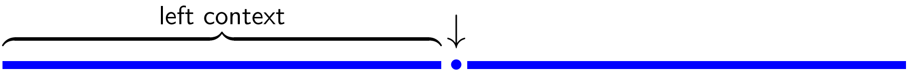
Typing the left and the right contexts

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?



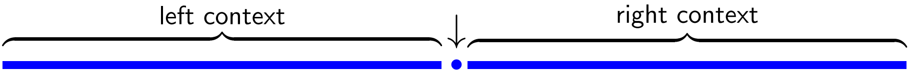
Typing the left and the right contexts

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?



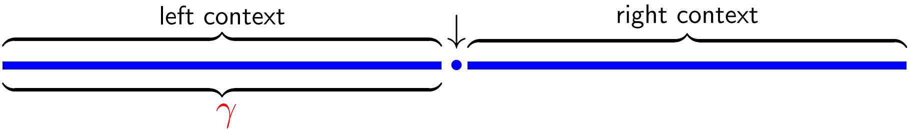
Typing the left and the right contexts

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?



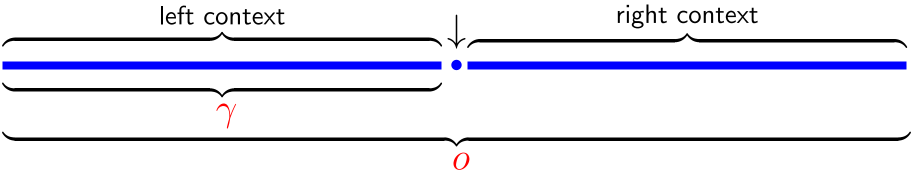
Typing the left and the right contexts

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?



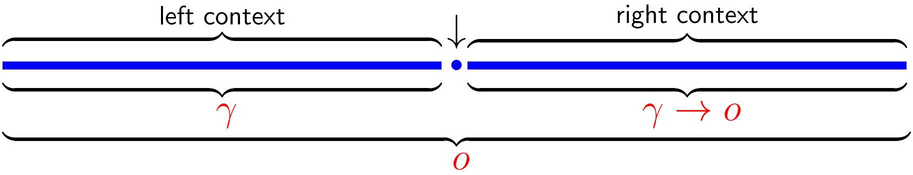
Typing the left and the right contexts

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.

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

- 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.

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

- Composition of two sentence interpretations:

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

Updating and accessing the context

Updating and accessing the context

- The empty context:

Updating and accessing the context

- The empty context:

$$\mathsf{nil} : \gamma$$

Updating and accessing the context

- The empty context:

$$\mathsf{nil} : \gamma$$

- Updating the context (adding a reference marker to the context):

Updating and accessing the context

- The empty context:

$$\mathsf{nil} : \gamma$$

- Updating the context (adding a reference marker to the context):

$$\_ :: \_ : \iota \rightarrow \gamma \rightarrow \gamma$$

Updating and accessing the context

- The empty context:

$$\mathsf{nil} : \gamma$$

- Updating the context (adding a reference marker to the context):

$$\_ :: \_ : \iota \to \gamma \to \gamma$$

- Accessing the context (anaphora resolution):

Updating and accessing the context

- The empty context:

$$\mathsf{nil} : \gamma$$

- Updating the context (adding a reference marker to the context):

$$\_ :: \_ : \iota \to \gamma \to \gamma$$

- Accessing the context (anaphora resolution):

$$\mathsf{sel} : \gamma \to \iota$$

- *A man enters the room*. (S_1)
  He smiles. (S_2)

- *A man enters the room.* (S_1)
  He smiles. (S_2)

- $[\![S_1]\!] = \lambda e\phi.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e))$

- *A man enters the room.* (S_1)
  He smiles. (S_2)

- $[\![S_1]\!] = \lambda e\phi.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e))$
- $[\![S_2]\!] = \lambda e\phi.\, (\mathbf{smiles}\, (\mathsf{sel}\, e)) \wedge (\phi\, e)$

- *A man enters the room.* (S_1)
  He smiles. (S_2)

- $[\![S_1]\!] = \lambda e\phi.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e))$
- $[\![S_2]\!] = \lambda e\phi.\, (\mathbf{smiles}\, (\mathsf{sel}\, e)) \wedge (\phi\, e)$

- $[\![S_1 \,.\, S_2]\!] = \lambda e_1\phi.\, [\![S_1]\!]\, e_1\, (\lambda e_2.\, [\![S_2]\!]\, e_2\, \phi)$

$[\![$A man enters the room. He smiles $]\!]$

$[\![$A man enters the room. He smiles $]\!]$

$$= \quad [\![S_1 . S_2]\!]$$

$[\![$A man enters the room. He smiles $]\!]$

$$= \; [\![S_1 . \, S_2]\!]$$

$$= \; \lambda e_1 \phi_1 . \, [\![S_1]\!] \, e_1 \, (\lambda e_2 . \, [\![S_2]\!] \, e_2 \, \phi_1)$$

$[\![$A man enters the room. He smiles $]\!]$

$$
\begin{aligned}
&= [\![S_1 \,.\, S_2]\!] \\
&= \lambda e_1 \phi_1 .\, [\![S_1]\!]\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
&= \lambda e_1 \phi_1 .\, (\lambda e \phi .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1)
\end{aligned}
$$

$[\![$A man enters the room. He smiles $]\!]$

$$
\begin{aligned}
&= [\![S_1 . S_2]\!] \\
&= \lambda e_1 \phi_1 . [\![S_1]\!]\, e_1\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1) \\
&= \lambda e_1 \phi_1 . (\lambda e \phi . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1) \\
&\to_\beta \lambda e_1 \phi_1 . (\lambda \phi . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e_1)))\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1)
\end{aligned}
$$

$[\![$A man enters the room. He smiles $]\!]$

$$
\begin{array}{ll}
= & [\![S_1 \,.\, S_2]\!] \\
= & \lambda e_1 \phi_1 .\, [\![S_1]\!]\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
= & \lambda e_1 \phi_1 .\, (\lambda e \phi .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 .\, (\lambda \phi .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e_1)))\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ((\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1)\, (x :: e_1))
\end{array}
$$

$[\![$A man enters the room. He smiles $]\!]$

$$
\begin{array}{ll}
= & [\![S_1 \,.\, S_2]\!] \\
= & \lambda e_1\phi_1.\, [\![S_1]\!]\, e_1\, (\lambda e_2.\, [\![S_2]\!]\, e_2\, \phi_1) \\
= & \lambda e_1\phi_1.\, (\lambda e\phi.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2.\, [\![S_2]\!]\, e_2\, \phi_1) \\
\to_\beta & \lambda e_1\phi_1.\, (\lambda \phi.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e_1)))\, (\lambda e_2.\, [\![S_2]\!]\, e_2\, \phi_1) \\
\to_\beta & \lambda e_1\phi_1.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ((\lambda e_2.\, [\![S_2]\!]\, e_2\, \phi_1)\, (x :: e_1)) \\
\to_\beta & \lambda e_1\phi_1.\, \exists x.\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ([\![S_2]\!]\, (x :: e_1)\, \phi_1)
\end{array}
$$

[[A man enters the room. He smiles]]

$$
\begin{aligned}
&= \ [\![S_1 \,.\, S_2]\!] \\
&= \ \lambda e_1 \phi_1 .\, [\![S_1]\!]\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
&= \ \lambda e_1 \phi_1 .\, (\lambda e \phi .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
&\rightarrow_\beta \ \lambda e_1 \phi_1 .\, (\lambda \phi .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e_1)))\, (\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1) \\
&\rightarrow_\beta \ \lambda e_1 \phi_1 .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ((\lambda e_2 .\, [\![S_2]\!]\, e_2\, \phi_1)\, (x :: e_1)) \\
&\rightarrow_\beta \ \lambda e_1 \phi_1 .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ([\![S_2]\!]\, (x :: e_1)\, \phi_1) \\
&= \ \lambda e_1 \phi_1 .\, \exists x .\, (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \\
&\qquad\qquad \wedge ((\lambda e \phi .\, (\mathbf{smiles}\, (\mathsf{sel}\, e)) \wedge (\phi\, e))\, (x :: e_1)\, \phi_1)
\end{aligned}
$$

$[\![$A man enters the room. He smiles $]\!]$

$$
\begin{aligned}
&= [\![S_1\,.\,S_2]\!] \\
&= \lambda e_1\phi_1.\,[\![S_1]\!]\,e_1\,(\lambda e_2.\,[\![S_2]\!]\,e_2\,\phi_1) \\
&= \lambda e_1\phi_1.\,(\lambda e\phi.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x)\wedge(\phi\,(x::e)))\,e_1\,(\lambda e_2.\,[\![S_2]\!]\,e_2\,\phi_1) \\
&\to_\beta \lambda e_1\phi_1.\,(\lambda\phi.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x)\wedge(\phi\,(x::e_1)))\,(\lambda e_2.\,[\![S_2]\!]\,e_2\,\phi_1) \\
&\to_\beta \lambda e_1\phi_1.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x)\wedge((\lambda e_2.\,[\![S_2]\!]\,e_2\,\phi_1)\,(x::e_1)) \\
&\to_\beta \lambda e_1\phi_1.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x)\wedge([\![S_2]\!]\,(x::e_1)\,\phi_1) \\
&= \lambda e_1\phi_1.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x) \\
&\qquad \wedge((\lambda e\phi.\,(\mathbf{smiles}\,(\mathsf{sel}\,e))\wedge(\phi\,e))\,(x::e_1)\,\phi_1) \\
&\to_\beta \lambda e_1\phi_1.\,\exists x.\,(\mathbf{man}\,x)\wedge(\mathbf{enters}\,x) \\
&\qquad \wedge((\lambda\phi.\,(\mathbf{smiles}\,(\mathsf{sel}\,(x::e_1)))\wedge(\phi\,(x::e_1)))\,\phi_1)
\end{aligned}
$$

⟦A man enters the room. He smiles ⟧

$$
\begin{array}{ll}
= & [\![S_1 . S_2]\!] \\
= & \lambda e_1 \phi_1 . [\![S_1]\!]\, e_1\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1) \\
= & \lambda e_1 \phi_1 . (\lambda e \phi . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e)))\, e_1\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 . (\lambda \phi . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\phi\, (x :: e_1)))\, (\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ((\lambda e_2 . [\![S_2]\!]\, e_2\, \phi_1)\, (x :: e_1)) \\
\rightarrow_\beta & \lambda e_1 \phi_1 . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge ([\![S_2]\!]\, (x :: e_1)\, \phi_1) \\
= & \lambda e_1 \phi_1 . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \\
 & \qquad \wedge ((\lambda e \phi . (\mathbf{smiles}\, (\mathsf{sel}\, e)) \wedge (\phi\, e))\, (x :: e_1)\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \\
 & \qquad \wedge ((\lambda \phi . (\mathbf{smiles}\, (\mathsf{sel}\, (x :: e_1))) \wedge (\phi\, (x :: e_1)))\, \phi_1) \\
\rightarrow_\beta & \lambda e_1 \phi_1 . \exists x . (\mathbf{man}\, x) \wedge (\mathbf{enters}\, x) \wedge (\mathbf{smiles}\, (\mathsf{sel}\, (x :: e_1))) \wedge (\phi_1\, (x :: e_1))
\end{array}
$$

Assigning a semantics to the lexical entries

Assigning a semantics to the lexical entries

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

Assigning a semantics to the lexical entries

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

$$
[\![s]\!] = o \tag{1}
$$

$$
[\![n]\!] = \iota \rightarrow [\![s]\!] \tag{2}
$$

$$
[\![np]\!] = (\iota \rightarrow [\![s]\!]) \rightarrow [\![s]\!] \tag{3}
$$

Assigning a semantics to the lexical entries

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

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

Replacing (1) with:

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

Assigning a semantics to the lexical entries

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

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

Replacing (1) with:

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

we obtain:

$$\begin{array}{rcl} [\![n]\!] & = & \iota \to \gamma \to (\gamma \to o) \to o \\ [\![np]\!] & = & (\iota \to \gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o \end{array}$$

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

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

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

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

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

...which might seem a little bit involved.

Dynamic propositions

Dynamic propositions

- Let $\Omega \triangleq \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$.

Dynamic propositions

- Let $\Omega \triangleq \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$.
- Ω may be seen as the type of dynamic propositions.

Dynamic propositions

- Let $\Omega \triangleq \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$.
- Ω may be seen as the type of dynamic propositions.
- We therefore intend to design a logic acting on propositions of type Ω

Formal framework

Formal framework

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

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

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

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

$$\top \;:\; o \qquad \textit{(truth)}$$
$$\neg \;:\; o \to o \qquad \textit{(negation)}$$
$$\wedge \;:\; o \to o \to o \qquad \textit{(conjunction)}$$
$$\exists \;:\; (\iota \to o) \to o \qquad \textit{(existential quantification)}$$

Dynamic primitives

$$:: \;:\; \iota \to \gamma \to \gamma \qquad \textit{(context updating)}$$
$$\mathsf{sel} \;:\; \gamma \to \iota \qquad \textit{(choice operator)}$$

Dynamic connectives

Dynamic connectives

Conjunction

Dynamic connectives

Conjunction

Conjunction is nothing but sentence composition. We therefore define:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e\phi.\, P\, e\, (\lambda e.\, Q\, e\, \phi)$$

Dynamic connectives

Conjunction

Conjunction is nothing but sentence composition. We therefore define:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e\phi.\, P\, e\, (\lambda e.\, Q\, e\, \phi)$$

Existential quantification

Dynamic connectives

Conjunction

Conjunction is nothing but sentence composition. We therefore define:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e\phi.\, P\, e\, (\lambda e.\, Q\, e\, \phi)$$

Existential quantification

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

$$\exists\!\!\mathsf{l} x.\, P \triangleq \lambda e\phi.\, \exists x.\, P\, x\, (x :: e)\, \phi$$

Dynamic connectives

Conjunction

Conjunction is nothing but sentence composition. We therefore define:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e\phi.\, P\, e\, (\lambda e.\, Q\, e\, \phi)$$

Existential quantification

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

$$\exists\!\!\mathsf{I} x.\, P \triangleq \lambda e\phi.\, \exists x.\, P\, x\, (x :: e)\, \phi$$

Negation

Dynamic connectives

Conjunction

Conjunction is nothing but sentence composition. We therefore define:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e\phi.\, P\, e\, (\lambda e.\, Q\, e\, \phi)$$

Existential quantification

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

$$\exists\!\!\!\mid x.\, P \triangleq \lambda e\phi.\, \exists x.\, P\, x\, (x :: e)\, \phi$$

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:

$$\neg\!\!\!\mid P \triangleq (\lambda e\phi.\, \neg(P\, e\, (\lambda e.\, \top))) \wedge (\phi\, e)$$

Disjunction, Implication, and Universal Quantification

Disjunction, Implication, and Universal Quantification

These are defined using de Morgan's laws:

$$\begin{aligned} P \mathbin{\mathbb{V}} Q &\triangleq \neg\!\!\!\mid (\neg\!\!\!\mid P \mathbin{/\!\!\!\wedge} \neg\!\!\!\mid Q) \\ P \Rightarrow Q &\triangleq \neg\!\!\!\mid (P \mathbin{/\!\!\!\wedge} \neg\!\!\!\mid Q) \\ \mathbb{V}x.\, P &\triangleq \neg\!\!\!\mid (\exists\!\!\mid x.\, \neg\!\!\!\mid P) \end{aligned}$$

Translation of first-order logic

Translation of first-order logic

Embedding of first-order logic into dynamic logic

Translation of first-order logic

Embedding of first-order logic into dynamic logic

$$\overline{A} \;=\; \lambda e\phi.\, A \wedge (\phi\, e)$$

Translation of first-order logic

Embedding of first-order logic into dynamic logic

$$\begin{aligned}
\overline{A} &= \lambda e\phi.\, A \wedge (\phi\, e) \\
\overline{\neg P} &= \neg\!\!\!\neg\, \overline{P} \\
\overline{P \wedge Q} &= \overline{P} \mathbin{/\!\!\!\wedge} \overline{Q} \\
\overline{P \vee Q} &= \overline{P} \mathbin{\vee\!\!\!\vee} \overline{Q} \\
\overline{P \rightarrow Q} &= \overline{P} \Rrightarrow \overline{Q} \\
\overline{\exists x.\, P} &= \exists\!\!\!\exists x.\, \overline{P} \\
\overline{\forall x.\, P} &= \forall\!\!\!\forall x.\, \overline{P}
\end{aligned}$$

Translation of first-order logic

Embedding of first-order logic into dynamic logic

$$
\begin{aligned}
\overline{A} &= \lambda e\phi.\, A \wedge (\phi\, e) \\
\overline{\neg P} &= \neg\!\!\!\neg\, \overline{P} \\
\overline{P \wedge Q} &= \overline{P} \mathbin{/\!\!\!\wedge} \overline{Q} \\
\overline{P \vee Q} &= \overline{P} \mathbin{\vee\!\!\!\vee} \overline{Q} \\
\overline{P \to Q} &= \overline{P} \Rightarrow \overline{Q} \\
\overline{\exists x.\, P} &= \exists\!\!\exists x.\, \overline{P} \\
\overline{\forall x.\, P} &= \forall\!\!\!\forall x.\, \overline{P}
\end{aligned}
$$

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

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

Application to donkey sentences

Application to donkey sentences

Montague-like semantic interpretation:

Application to donkey sentences

Montague-like semantic interpretation:

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

Dynamic interpretation:

Dynamic interpretation:

$$\begin{aligned}
[\![\text{farmer}]\!] &= \lambda x.\, \overline{\mathbf{farmer}\, x} \\
[\![\text{donkey}]\!] &= \lambda x.\, \overline{\mathbf{donkey}\, x} \\
[\![\text{owns}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{own}\, x\, y})) \\
[\![\text{beats}]\!] &= \lambda OS.\, S\,(\lambda x.\, O\,(\lambda y.\, \overline{\mathbf{beat}\, x\, y})) \\
[\![\text{who}]\!] &= \lambda RQx.\, (Q\, x) \mathbin{/\!\!\!\wedge} (R\,(\lambda P.\, P\, x)) \\
[\![\text{a}]\!] &= \lambda PQ.\, \exists\!\!\!\exists x.\, (P\, x) \mathbin{/\!\!\!\wedge} (Q\, x) \\
[\![\text{every}]\!] &= \lambda PQ.\, \forall\!\!\!\forall x.\, (P\, x) \Rrightarrow (Q\, x) \\
[\![\text{it}]\!] &= \lambda Pe\phi.\, P\,(\mathsf{sel}\, e)\, e\, \phi
\end{aligned}$$

Outline

2. Need for more flexibility
   - Accessibility constraints
   - Limitations

Accessibility constraints

DRT like accessibility constraints:

Accessibility constraints

DRT like accessibility constraints:

Bill doesn't have a car.
**It is black.*
(Karttunen 1976)

Accessibility constraints

DRT like accessibility constraints:

Bill doesn't have a car.
**It is black.*
(Karttunen 1976)

Harvey courts a girl at every convention. (non specific reading)
**She is very pretty.*
(Karttunen 1976)

Accessibility constraints

DRT like accessibility constraints:

Bill doesn't have a car.
**It is black.*
(Karttunen 1976)

Harvey courts a girl at every convention. (non specific reading)
**She is very pretty.*
(Karttunen 1976)

You must write a letter to your parents.
**They are expecting the letter.*
(Karttunen 1976)

But...

But...

I don't have a microwave oven. I wouldn't know what to do with it.
(Frank 1996)

Harvey courts a girl at every convention. She always comes to the banquet with him. The girl is usually also very pretty.
(Karttunen 1976)

You must write a letter to your parents. It has to be sent by airmail. The letter must get there by tomorrow
(Karttunen 1976)

A thief might break into the house. He would take the silver.
(Roberts 1989)

Limitations

Limitations

- Accessibility constraints are wired into the formalism.

Limitations

- Accessibility constraints are wired into the formalism.
- Only one mode of composition.

Limitations

- Accessibility constraints are wired into the formalism.
- Only one mode of composition.
- No systematic way of defining the logical connectives.

Limitations

- Accessibility constraints are wired into the formalism.
- Only one mode of composition.
- No systematic way of defining the logical connectives.
- The obtained logic relies heavily on classical logic.

Limitations

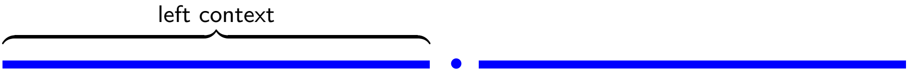
- Accessibility constraints are wired into the formalism.
- Only one mode of composition.
- No systematic way of defining the logical connectives.
- The obtained logic relies heavily on classical logic.
- No methodology for extending the frameork (generalized quantifiers, plurality, temporality, intentionality, ...)

Outline

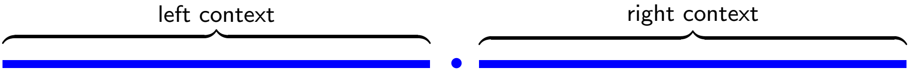
3. A more flexible framework
   - Revisiting the picture
   - Accomodating the connectives
   - Specifying accessibility constraints
   - Taking polarity into account
   - Putting everything together

Revisiting the picture

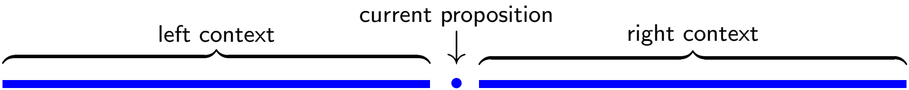
Revisiting the picture



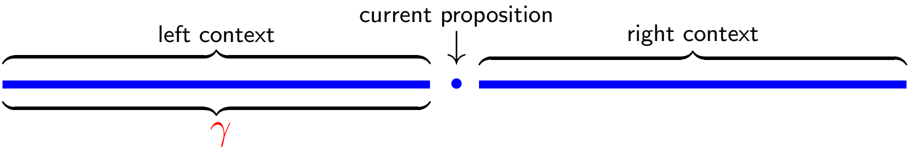
Revisiting the picture



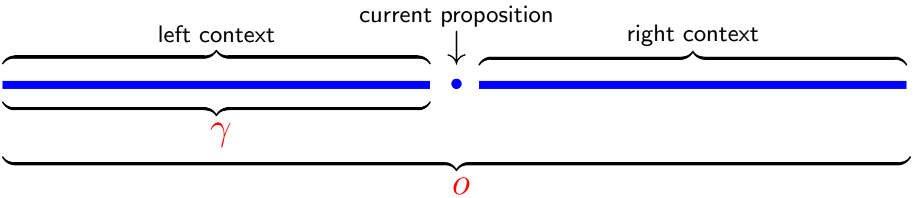
Revisiting the picture



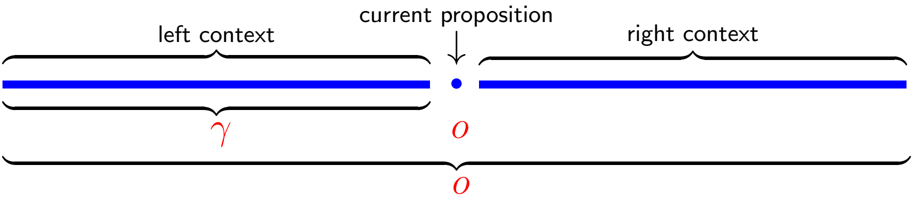
Revisiting the picture



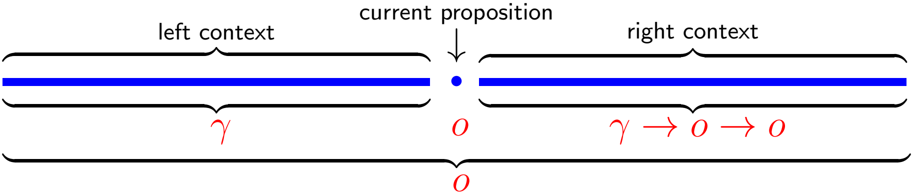
Revisiting the picture



Revisiting the picture



Revisiting the picture



Accomodating the connectives

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Variants

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Variants

$$\lambda e_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Variants

$$\lambda e_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_0\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Variants

$$\lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_0\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi.\, Q\, e_0\, (\lambda e_1 p.\, P\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

Variants

$$\lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_0\, (\lambda e_2 q.\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi.\, Q\, e_0\, (\lambda e_1 p.\, P\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p \wedge q)))$$

$$\lambda e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, p \wedge (\phi\, e_2\, q)))$$

Accomodating the connectives

Conjunction

$$P \mathbin{/\!\!\!\wedge} Q \triangleq \lambda e_0 \phi .\, P\, e_0\, (\lambda e_1 p .\, Q\, e_1\, (\lambda e_2 q .\, \phi\, e_2\, (p \wedge q)))$$

Variants

$$\lambda e_0 \phi .\, P\, e_0\, (\lambda e_1 p .\, Q\, e_1\, (\lambda e_2 q .\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi .\, P\, e_0\, (\lambda e_1 p .\, Q\, e_0\, (\lambda e_2 q .\, \phi\, e_0\, (p \wedge q)))$$

$$\lambda e_0 \phi .\, Q\, e_0\, (\lambda e_1 p .\, P\, e_1\, (\lambda e_2 q .\, \phi\, e_2\, (p \wedge q)))$$

$$\lambda e_0 \phi .\, P\, e_0\, (\lambda e_1 p .\, Q\, e_1\, (\lambda e_2 q .\, p \wedge (\phi\, e_2\, q)))$$

$$\vdots$$

Generalizing

Generalizing

Let $* : o \to o \to o$ be a (static) binary connective. We define:

$$[*]^{++} \triangleq \lambda PQe_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))$$

Generalizing

Let $* : o \to o \to o$ be a (static) binary connective. We define:

$$[*]^{++} \triangleq \lambda PQe_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))$$

Then, we have:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq P\, [\wedge]^{++}\, Q$$

Generalizing

Let $* : o \to o \to o$ be a (static) binary connective. We define:

$$[*]^{++} \triangleq \lambda PQe_0\phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))$$

Then, we have:

$$P \mathbin{/\!\!\!\wedge} Q \triangleq P\, [\wedge]^{++}\, Q$$

Similarly, let $\triangleright : o \to o$ be a unary connective. We define:

$$[\triangleright]^{+} \triangleq \lambda Pe_0\phi.\, P\, e_0\, (\lambda e_1 p.\, \phi\, e_1\, (\triangleright p))$$

Finally, let $\mathcal{Q} : (\iota \to o) \to o$ be a first-order quantifier. We define:

$$[\mathcal{Q}] \triangleq \lambda P e_0 \phi.\, \mathcal{Q} x.\, P\, x\, e_0\, (\lambda e_1 p.\, \phi\, (x :: e_1)\, p)$$

Specifying accessibility constraints

Specifying accessibility constraints

The following lambda-term:

$$\lambda e p.\, p : \gamma \rightarrow o \rightarrow o$$

discards the current left context and returns the current proposition. It corresponds to the empty continuation.

Specifying accessibility constraints

The following lambda-term:

$$\lambda ep.\, p : \gamma \rightarrow o \rightarrow o$$

discards the current left context and returns the current proposition. It corresponds to the empty continuation.

We may then define the following closure operator:

$$\diamond P \triangleq \lambda e\phi.\, \phi\, e\, (P\, e\, (\lambda ep.\, p))$$

Specifying accessibility constraints

The following lambda-term:

$$\lambda ep.\, p : \gamma \to o \to o$$

discards the current left context and returns the current proposition. It corresponds to the empty continuation.

We may then define the following closure operator:

$$\diamond P \triangleq \lambda e\phi.\, \phi\, e\, (P\, e\, (\lambda ep.\, p))$$

This allows, for instance, the "usual" dynamic disjunction to be defined as follows:

$$P \Vvee Q \triangleq \diamond (\diamond P\, [\vee]^{++} \diamond Q)$$

Taking polarity into account

Taking polarity into account

The part played by classical negation

Taking polarity into account

The part played by classical negation

$$\forall x. (\exists y. A[x, y]) \rightarrow B[x]$$

Taking polarity into account

The part played by classical negation

$$\forall x.\, (\exists y.\, A[x,y]) \rightarrow B[x] \equiv \forall x.\, \neg(\exists y.\, A[x,y]) \vee B[x]$$

Taking polarity into account

The part played by classical negation

$$\begin{aligned}\forall x.\,(\exists y.\,A[x,y]) \to B[x] &\equiv \forall x.\,\neg(\exists y.\,A[x,y]) \lor B[x]\\ &\equiv \forall x.\,(\forall y.\,\neg A[x,y]) \lor B[x]\end{aligned}$$

Taking polarity into account

The part played by classical negation

$$\begin{aligned}\forall x.\,(\exists y.\,A[x,y]) \to B[x] &\equiv \forall x.\,\neg(\exists y.\,A[x,y]) \vee B[x]\\ &\equiv \forall x.\,(\forall y.\,\neg A[x,y]) \vee B[x]\\ &\equiv \forall x.\,\forall y.\,\neg A[x,y] \vee B[x]\end{aligned}$$

Taking polarity into account

The part played by classical negation

$$
\begin{aligned}
\forall x.\,(\exists y.\,A[x,y]) \rightarrow B[x] &\equiv \forall x.\,\neg(\exists y.\,A[x,y]) \vee B[x] \\
&\equiv \forall x.\,(\forall y.\,\neg A[x,y]) \vee B[x] \\
&\equiv \forall x.\,\forall y.\,\neg A[x,y] \vee B[x] \\
&\equiv \forall x.\,\forall y.\,A[x,y] \rightarrow B[x]
\end{aligned}
$$

Taking polarity into account

The part played by classical negation

$$\begin{aligned}\forall x.\,(\exists y.\,A[x,y]) \rightarrow B[x] &\equiv \forall x.\,\neg(\exists y.\,A[x,y]) \vee B[x]\\ &\equiv \forall x.\,(\forall y.\,\neg A[x,y]) \vee B[x]\\ &\equiv \forall x.\,\forall y.\,\neg A[x,y] \vee B[x]\\ &\equiv \forall x.\,\forall y.\,A[x,y] \rightarrow B[x]\end{aligned}$$

Connective polarity

Taking polarity into account

The part played by classical negation

$$\begin{aligned}\forall x.\,(\exists y.\,A[x,y]) \rightarrow B[x] &\equiv \forall x.\,\neg(\exists y.\,A[x,y]) \vee B[x]\\ &\equiv \forall x.\,(\forall y.\,\neg A[x,y]) \vee B[x]\\ &\equiv \forall x.\,\forall y.\,\neg A[x,y] \vee B[x]\\ &\equiv \forall x.\,\forall y.\,A[x,y] \rightarrow B[x]\end{aligned}$$

Connective polarity

$$(\neg P^{-})^{+}$$

$$(P^{+} \vee Q^{+})^{+}$$

$$(P^{+} \wedge Q^{+})^{+}$$

$$(P^{-} \rightarrow Q^{+})^{+}$$

Adding a dualizing operator

$$\cdot^{\perp} : o \rightarrow o$$

Adding a dualizing operator

$$.^{\perp} : o \rightarrow o$$

such that:

$$\begin{aligned} P^{\perp\perp} &= P \\ (\neg P)^{\perp} &= \neg\, P^{\perp} \\ (P \vee Q)^{\perp} &= P^{\perp} \wedge Q^{\perp} \\ (P \wedge Q)^{\perp} &= P^{\perp} \vee Q^{\perp} \\ (\exists x.\, P)^{\perp} &= \forall x.\, P^{\perp} \\ (\forall x.\, P)^{\perp} &= \exists x.\, P^{\perp} \end{aligned}$$

Remeber that we defined:

$$[\triangleright]^{+} \triangleq \lambda P e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, \phi\, e_1\, (\triangleright p))$$

and

$$[*]^{++} \triangleq \lambda P Q e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))$$

Remeber that we defined:

$$[\triangleright]^{+} \triangleq \lambda P e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, \phi\, e_1\, (\triangleright\, p))$$

and

$$[*]^{++} \triangleq \lambda P Q e_0 \phi.\, P\, e_0\, (\lambda e_1 p.\, Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))$$

Similarly, we define:

$$[\triangleright]^{-} \triangleq \lambda P e_0 \phi.\, (P\, e_0\, (\lambda e_1 p.\, (\phi\, e_1\, (\triangleright\, p))^{\perp}))^{\perp}$$

and

$$[*]^{-+} \triangleq \lambda P Q e_0 \phi.\, (P\, e_0\, (\lambda e_1 p.\, (Q\, e_1\, (\lambda e_2 q.\, \phi\, e_2\, (p * q)))^{\perp}))^{\perp}$$

Putting everything together

Putting everything together

Definition of the dynamic connectives

Putting everything together

Definition of the dynamic connectives

$$P \mathbin{/\!\!\wedge} Q \triangleq P\,[\wedge]^{++}\,Q$$

Putting everything together

Definition of the dynamic connectives

$$P \mathbin{/\!\!\!\wedge} Q \triangleq P\,[\wedge]^{++}\,Q$$
$$P \mathbin{\backslash\!\!\!\vee} Q \triangleq \diamond(\diamond P\,[\vee]^{++} \diamond Q)$$

Putting everything together

Definition of the dynamic connectives

$$P \mathbin{/\!\!\!\wedge} Q \triangleq P\,[\wedge]^{++}\,Q$$
$$P \mathbin{\backslash\!\!\!\vee} Q \triangleq \diamond(\diamond P\,[\vee]^{++} \diamond Q)$$
$$P \Rightarrow Q \triangleq \diamond(P\,[\rightarrow]^{-+}\,Q)$$

Putting everything together

Definition of the dynamic connectives

$$
\begin{aligned}
P \mathbin{/\!\!\!\wedge} Q &\triangleq P\,[\wedge]^{++}\,Q \\
P \mathbin{\backslash\!\!\!\vee} Q &\triangleq \diamond(\diamond P\,[\vee]^{++}\diamond Q) \\
P \Rightarrow Q &\triangleq \diamond(P\,[\rightarrow]^{-+}\,Q) \\
\neg\!\shortmid P &\triangleq \diamond([\neg]^{-}\,P)
\end{aligned}
$$

Putting everything together

Definition of the dynamic connectives

$$
\begin{aligned}
P \mathbin{/\!\!\!\wedge} Q &\triangleq P\,[\wedge]^{++}\,Q \\
P \mathbin{\setminus\!\!\!\vee} Q &\triangleq \diamond(\diamond P\,[\vee]^{++} \diamond Q) \\
P \Rightarrow Q &\triangleq \diamond(P\,[\rightarrow]^{-+}\,Q) \\
\neg P &\triangleq \diamond([\neg]^{-}\,P) \\
\exists x.\,P &\triangleq [\exists]\,(\lambda x.\,P)
\end{aligned}
$$

Putting everything together

Definition of the dynamic connectives

$$
\begin{aligned}
P \mathbin{/\!\!\!\wedge} Q &\triangleq P\,[\wedge]^{++}\,Q \\
P \mathbin{\backslash\!\!\!\vee} Q &\triangleq \diamond(\diamond P\,[\vee]^{++} \diamond Q) \\
P \Rightarrow Q &\triangleq \diamond(P\,[\rightarrow]^{-+}\,Q) \\
\neg\!\!\shortmid P &\triangleq \diamond([\neg]^{-}\,P) \\
\exists\!\!\shortmid x.\,P &\triangleq [\exists]\,(\lambda x.\,P) \\
\forall\!\!\!\forall x.\,P &\triangleq \diamond([\forall]\,(\lambda x.\,P))
\end{aligned}
$$

Embedding of first-order logic into dynamic logic

Embedding of first-order logic into dynamic logic

$$\overline{A} \;=\; \lambda e\phi.\, \phi\, e\, A$$

Embedding of first-order logic into dynamic logic

$$\begin{aligned}
\overline{A} &= \lambda e\phi.\, \phi\, e\, A \\
\overline{\neg P} &= \neg\!\!\shortmid\, \overline{P} \\
\overline{P \wedge Q} &= \overline{P} \mathbin{/\!\!\wedge} \overline{Q} \\
\overline{P \vee Q} &= \overline{P} \mathbin{\vee\!\!\!\vee} \overline{Q} \\
\overline{P \to Q} &= \overline{P} \Rightarrow \overline{Q} \\
\overline{\exists x.\, P} &= \exists\!\!\shortmid x.\, \overline{P} \\
\overline{\forall x.\, P} &= \forall\!\!\!\forall x.\, \overline{P}
\end{aligned}$$