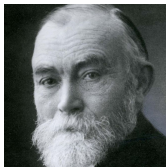


Syntax-Semantics Interface

K. Kogkalidis

Logic & Language 2020

Two Key Ideas



Gottlob Frege

Principle of Compositionality

The meaning of a complex expression is derived by its constituent components and their interactions.

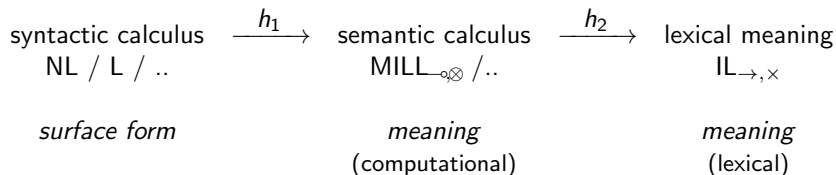
Universal Grammar

Homomorphism between a syntactic and a semantic algebra associates syntactic operations to semantic operations.



Richard Montague

In our case



For each h_i we will need:

- ▶ A map to associate source types to target types
- ▶ A map to associate source terms to target terms

remember: $\mathcal{L}, \eta, \theta$ of ACGs

Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\frac{\Gamma \vdash f : B/A \quad \Delta \vdash x : A}{(\Gamma, \Delta) \vdash f \triangleleft x : B} /E \quad \frac{\Gamma \vdash x : B \quad \Delta \vdash f : A \setminus B}{(\Gamma, \Delta) \vdash x \triangleright f : B} \setminus E$$
$$\frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x^r. f : B/A} /I \quad \frac{x : A, \Gamma \vdash f : B}{\Gamma \vdash \lambda x^l. f : A \setminus B} \setminus I$$

Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\begin{array}{c}
 \frac{\Gamma \vdash f : B/A \quad \Delta \vdash x : A}{(\Gamma, \Delta) \vdash f \triangleleft x : B} /E \quad \frac{\Gamma \vdash x : B \quad \Delta \vdash f : A \setminus B}{(\Gamma, \Delta) \vdash x \triangleright f : B} \setminus E \\
 \\
 \frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x^r. f : B/A} /I \quad \frac{x : A, \Gamma \vdash f : B}{\Gamma \vdash \lambda x^l. f : A \setminus B} \setminus I \\
 \\
 \frac{\frac{\text{the}}{np/n} \quad \frac{\text{worker}}{n}}{(the \cdot worker) \vdash np} /E \quad \frac{\frac{\text{liberates}}{(np \setminus s)/np} \quad \frac{\text{herself}}{((np \setminus s)/np) \setminus (np \setminus s)}}{(liberates \cdot herself) \vdash np \setminus s} \setminus E \\
 \hline
 ((the \cdot worker) \cdot (liberates \cdot herself)) \vdash s \quad \setminus E
 \end{array}$$

Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\begin{array}{c}
 \frac{\Gamma \vdash f : B/A \quad \Delta \vdash x : A}{(\Gamma, \Delta) \vdash f \triangleleft x : B} /E \quad \frac{\Gamma \vdash x : B \quad \Delta \vdash f : A \setminus B}{(\Gamma, \Delta) \vdash x \triangleright f : B} \setminus E \\
 \\
 \frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x^r. f : B/A} /I \quad \frac{x : A, \Gamma \vdash f : B}{\Gamma \vdash \lambda x^l. f : A \setminus B} \setminus I \\
 \\
 \frac{\frac{\text{the}}{np/n} \quad \frac{\text{worker}}{n}}{(the \cdot worker) \vdash np} /E \quad \frac{\frac{\text{liberates}}{(np \setminus s)/np} \quad \frac{\text{herself}}{((np \setminus s)/np) \setminus (np \setminus s)}}{(liberates \cdot herself) \vdash np \setminus s} \setminus E \\
 \hline
 ((the \cdot worker) \cdot (liberates \cdot herself)) \vdash s \quad \setminus E
 \end{array}$$

- Syntactic Term: $(the \triangleleft worker) \triangleright (liberates \triangleright herself)$

Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\begin{array}{c}
 \frac{\Gamma \vdash f : B/A \quad \Delta \vdash x : A}{(\Gamma, \Delta) \vdash f \triangleleft x : B} /E \quad \frac{\Gamma \vdash x : B \quad \Delta \vdash f : A \setminus B}{(\Gamma, \Delta) \vdash x \triangleright f : B} \setminus E \\
 \\
 \frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x^r. f : B/A} /I \quad \frac{x : A, \Gamma \vdash f : B}{\Gamma \vdash \lambda x^l. f : A \setminus B} \setminus I \\
 \\
 \frac{\frac{\frac{\text{the}}{np/n} \quad \frac{\text{worker}}{n}}{(the \cdot worker) \vdash np} /E \quad \frac{\frac{\text{liberates}}{(np \setminus s)/np} \quad \frac{\text{herself}}{((np \setminus s)/np) \setminus (np \setminus s)}}{(liberates \cdot herself) \vdash np \setminus s} \setminus E}{((the \cdot worker) \cdot (liberates \cdot herself)) \vdash s} \setminus E
 \end{array}$$

- ▶ Syntactic Term: $(\text{the} \triangleleft \text{worker}) \triangleright (\text{liberates} \triangleright \text{herself})$
- ▶ Computational Term: $(\text{herself liberates}) (\text{the worker})$

Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\begin{array}{c}
 \frac{\Gamma \vdash f : B/A \quad \Delta \vdash x : A}{(\Gamma, \Delta) \vdash f \triangleleft x : B} /E \quad \frac{\Gamma \vdash x : B \quad \Delta \vdash f : A \setminus B}{(\Gamma, \Delta) \vdash x \triangleright f : B} \setminus E \\
 \\
 \frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x^r. f : B/A} /I \quad \frac{x : A, \Gamma \vdash f : B}{\Gamma \vdash \lambda x^l. f : A \setminus B} \setminus I \\
 \\
 \frac{\frac{\frac{\text{the}}{np/n} \quad \frac{\text{worker}}{n}}{(the \cdot worker) \vdash np} /E \quad \frac{\frac{\text{liberates}}{(np \setminus s)/np} \quad \frac{\text{herself}}{((np \setminus s)/np) \setminus (np \setminus s)}}{(liberates \cdot herself) \vdash np \setminus s} \setminus E}{((the \cdot worker) \cdot (liberates \cdot herself)) \vdash s} \setminus E
 \end{array}$$

- ▶ Syntactic Term: $(\text{the} \triangleleft \text{worker}) \triangleright (\text{liberates} \triangleright \text{herself})$
- ▶ Computational Term: $(\text{herself liberates}) (\text{the worker})$
- ▶ Lexical Term: $(\text{liberates } (\text{the worker})) (\text{the worker})$
 assigning $\text{herself} := \lambda f x. ((f \ x) \ x)$ **non-linear!**

Interpretation Domains

What is the meaning of life?

Interpretation Domains

What is the meaning of life?

life

Interpretation Domains

What is the meaning of life?

life

Available Options:

- ▶ Truth-Conditional
Sentences as truth-values, properties as predicates, entities as set elements . . .
- ▶ Distributional/Statistical
Words and phrases as tensors populated by co-occurrence counts
- ▶ Neural
Words and phrases as tensors populated by optimization of objective function
- ▶ ..

Truth-Conditional Semantics: take 1

Truth-conditional semantics

consist of two *semantic primitives*: e (for entities) and t (for truth-values)
with \multimap as the type-forming operator, shorthand xy for $x \multimap y$ ¹

¹left-associative

Truth-Conditional Semantics: take 1

Truth-conditional semantics

consist of two *semantic primitives*: e (for entities) and t (for truth-values) with \multimap as the type-forming operator, shorthand xy for $x \multimap y$ ¹

Simple Translation

Let $\llbracket \cdot \rrbracket$ a mapping, s.t. $\llbracket np \rrbracket = e$, $\llbracket s \rrbracket = t$, $\llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap \llbracket B \rrbracket$

¹left-associative

Truth-Conditional Semantics: take 1

Truth-conditional semantics

consist of two *semantic primitives*: e (for entities) and t (for truth-values) with \multimap as the type-forming operator, shorthand xy for $x \multimap y$ ¹

Simple Translation

Let $\llbracket \cdot \rrbracket$ a mapping, s.t. $\llbracket np \rrbracket = e$, $\llbracket s \rrbracket = t$, $\llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap \llbracket B \rrbracket$

☺ Fine for simple clauses:

Joseph hates Leon $\rightsquigarrow ((\text{hates}^{eet} \text{Leon}^e)^{et} \text{Joseph}^e)^t$

¹left-associative

Truth-Conditional Semantics: take 1

Truth-conditional semantics

consist of two *semantic primitives*: e (for entities) and t (for truth-values) with \multimap as the type-forming operator, shorthand xy for $x \multimap y$ ¹

Simple Translation

Let $\llbracket \cdot \rrbracket$ a mapping, s.t. $\llbracket np \rrbracket = e$, $\llbracket s \rrbracket = t$, $\llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap \llbracket B \rrbracket$

☺ Fine for simple clauses:

$$\text{Joseph hates Leon} \rightsquigarrow ((\text{hates}^{eet} \text{Leon}^e)^{et} \text{Joseph}^e)^t$$

☹ ..less so for quantifiers:

$$\text{Joseph hates everybody} \overset{?}{\rightsquigarrow} (\forall \lambda x. \text{hates } x \text{ Joseph})^t$$

Impossible to derive with simple translation!

¹left-associative

Truth-Conditional Semantics: Three Translations

We examine three translations:

- ▶ Flexible Interpretation (Hendriks 1993)
- ▶ Incremental CPS Interpretation (Barker 2004)
- ▶ Plotkin's CPS Interpretation (Lebedeva 2012)

(1) Flexible Interpretation

The type map η is lifted from a function to a *relation*: Each syntactic source type is associated with a *set* of semantic target types.

$$\eta(A) = \{B \mid B \in \text{shift}(\lceil A \rceil)\}$$

where shift the reflexive, transitive closure of $\lceil \cdot \rceil$ under the laws:

► Value Raising

From $f : \vec{A} \multimap B$ derive:

$$\lambda \vec{x} w. (w (f \vec{x})) : \vec{A} \multimap (B \multimap D) \multimap D$$

► Argument Raising

From $f : \vec{A} \multimap B \multimap \vec{C} \multimap D$ derive:

$$\lambda \vec{x} w \vec{y}. (w \lambda z. (f \vec{x} z \vec{y})) : \vec{A} \multimap ((B \multimap D) \multimap D) \multimap \vec{C} \multimap D$$

► Argument Lowering

From $f : \vec{A} \multimap ((B \multimap D) \multimap D) \multimap \vec{C} \multimap E$ derive:

$$\lambda \vec{x} w \vec{y}. (f \vec{x} (\lambda z. (z w)) \vec{y}) : \vec{A} \multimap B \multimap \vec{C} \multimap D$$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: <i>np</i>	Joseph :: <i>e</i>
hates :: $(np \backslash s) / np$	hates :: <i>eet</i>
everybody :: <i>np</i>	\forall :: $(et)t$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

everybody: np

$[np] = e$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

Value Raising

From $f : \vec{A} \multimap B$ derive:

$$\lambda \vec{x} w. (w (f \vec{x})) : \vec{A} \multimap (B \multimap D) \multimap D$$

everybody: np

$$[np] = e$$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

Value Raising

From $f : \vec{A} \multimap B$ derive:

$$\lambda \vec{x} w. (w (f \vec{x})) : \vec{A} \multimap (B \multimap D) \multimap D$$

everybody: np

$$[np] = e \xRightarrow{vr} \forall : (et)t$$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

everybody: np

$[np] = e \xRightarrow{vr} \forall : (et)t$

hates: $(np \backslash s) / np$

$[(np \backslash s) / np] = eet$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

everybody: np

$[np] = e \xRightarrow{vr} \forall : (et)t$

hates: $(np \backslash s) / np$

$[(np \backslash s) / np] = eet$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

Argument Raising

From $f : \vec{A} \multimap B \multimap \vec{C} \multimap D$ derive:

$$\lambda \vec{x} w \vec{y}. (w \lambda z. (f \vec{x} z \vec{y})) : \vec{A} \multimap ((B \multimap D) \multimap D) \multimap \vec{C} \multimap D$$

everybody: np

$$[np] = e \xRightarrow{vr} \forall : (et)t$$

hates: $(np \backslash s) / np$

$$[(np \backslash s) / np] = eet$$

(1) Flexible Interpretation: Example

Lexical Translation

Source	Target
Joseph :: np	Joseph :: e
hates :: $(np \backslash s) / np$	hates :: eet
everybody :: np	$\forall :: (et)t$

Argument Raising

From $f : \vec{A} \multimap B \multimap \vec{C} \multimap D$ derive:

$$\lambda \vec{x} w \vec{y}. (w \lambda z. (f \vec{x} z \vec{y})) : \vec{A} \multimap ((B \multimap D) \multimap D) \multimap \vec{C} \multimap D$$

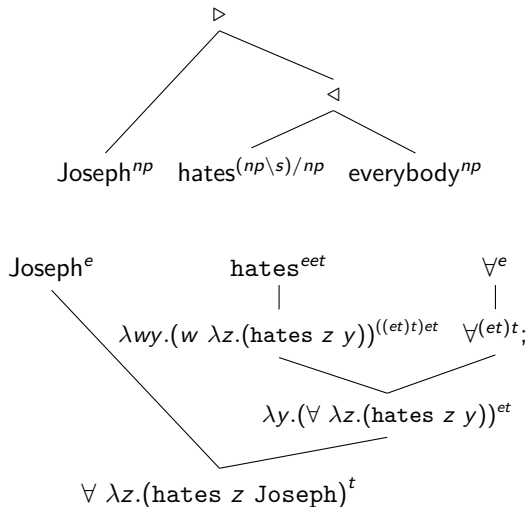
everybody: np

$$[np] = e \xRightarrow{vr} \forall : (et)t$$

hates: $(np \backslash s) / np$

$$[(np \backslash s) / np] = eet \xRightarrow{ar} \lambda w y. (w \lambda z. (\text{hates } z y)) : ((et)t)et$$

(1) Flexible Interpretation: Example



(2) Incremental CPS

Two translations left-to-right $(.)^{\rightsquigarrow}$ and right-to-left incremental $(.)^{\leftarrow}$

For A atomic, $A^{\rightsquigarrow} = A^{\leftarrow} = (\lceil A \rceil \multimap \perp) \multimap \perp^2$

For complex types, we have:

► Left-to-right $(.)^{\rightsquigarrow}$

$$(N \triangleright M)^{\rightsquigarrow} = \lambda k. (N^{\rightsquigarrow} \lambda n. (M^{\rightsquigarrow} \lambda m. (k (m n))))$$

$$(M \triangleleft N)^{\rightsquigarrow} = \lambda k. (M^{\rightsquigarrow} \lambda m. (N^{\rightsquigarrow} \lambda n. (k (m n))))$$

► Right-to-left $(.)^{\leftarrow}$

$$(N \triangleright M)^{\leftarrow} = \lambda k. (M^{\leftarrow} \lambda m. (N^{\leftarrow} \lambda n. (k (m n))))$$

$$(M \triangleleft N)^{\leftarrow} = \lambda k. (N^{\leftarrow} \lambda n. (M^{\leftarrow} \lambda m. (k (m n))))$$

² \perp the *response* type, here t

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: np	$\forall :: (et)t$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: np	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m n))))$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: np	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m n))))$$

$$(h \triangleleft e)^{\sim} = \lambda k_1.(hates^{\sim} \lambda n_1.(everybody^{\sim} \lambda m_1.(k_1 (m_1 n_1))))$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: np	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m \ n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m \ n))))$$

$$\begin{aligned} (h \triangleleft e)^{\sim} &= \lambda k_1.(\text{hates}^{\sim} \lambda n_1.(\text{everybody}^{\sim} \lambda m_1.(k_1 (m_1 \ n_1)))) \\ &= \lambda k_1.(\lambda k_0.(\text{hates}) \lambda n_1.(\forall \lambda m_1.(k_1 (m_1 \ n_1)))) \end{aligned}$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m n))))$$

$$\begin{aligned}
 (h \triangleleft e)^{\sim} &= \lambda k_1.(\text{hates}^{\sim} \lambda n_1.(\text{everybody}^{\sim} \lambda m_1.(k_1 (m_1 n_1)))) \\
 &= \lambda k_1.(\lambda k_0.(\text{hates}) \lambda n_1.(\forall \lambda m_1.(k_1 (m_1 n_1)))) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_1.(\lambda n_1.(\forall \lambda m_1.(k_1 (m_1 n_1))) \text{hates})
 \end{aligned}$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m n))))$$

$$\begin{aligned} (h \triangleleft e)^{\sim} &= \lambda k_1.(\text{hates}^{\sim} \lambda n_1.(\text{everybody}^{\sim} \lambda m_1.(k_1 (m_1 n_1)))) \\ &= \lambda k_1.(\lambda k_0.(\text{hates}) \lambda n_1.(\forall \lambda m_1.(k_1 (m_1 n_1)))) \end{aligned}$$

$$\xrightarrow{\beta} \lambda k_1.(\lambda n_1.(\forall \lambda m_1.(k_1 (m_1 n_1))) \text{ hates})$$

$$\xrightarrow{\beta} \lambda k_1.(\forall \lambda m_1.(k_1 (\text{hates } m_1)))$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: np	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m \ n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m \ n))))$$

$$(J \triangleright (h \triangleleft e))^{\sim} = \lambda k_2.(\text{Joseph}^{\sim} \lambda n_2.((\text{hates} \triangleleft \text{everybody})^{\sim} \lambda m_2.(k_2 (m_2 \ n_2))))$$

(2) Incremental CPS: Example

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda k.(k \text{ hates}) :: ((et)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Left-to-right translation

$$(N \triangleright M)^{\sim} = \lambda k.(N^{\sim} \lambda n.(M^{\sim} \lambda m.(k (m n))))$$

$$(M \triangleleft N)^{\sim} = \lambda k.(M^{\sim} \lambda m.(N^{\sim} \lambda n.(k (m n))))$$

$$\begin{aligned}
 (J \triangleright (h \triangleleft e))^{\sim} &= \lambda k_2.(\text{Joseph}^{\sim} \lambda n_2.((\text{hates} \triangleleft \text{everybody})^{\sim} \lambda m_2.(k_2 (m_2 n_2)))) \\
 &= \lambda k_2.(\lambda k_3.(\text{Joseph}) \lambda n_2.((\text{hates} \triangleleft \text{everybody})^{\sim} \lambda m_2.(k_2 (m_2 n_2)))) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2.(\lambda n_2.((\text{hates} \triangleleft \text{everybody})^{\sim} \lambda m_2.(k_2 (m_2 n_2))) \text{Joseph}) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2.((\text{hates} \triangleleft \text{everybody})^{\sim} \lambda m_2.(k_2 (m_2 \text{Joseph}))) \\
 &= \lambda k_2.(\lambda k_1.(\forall \lambda m_1.(k_1 (\text{hates } m_1))) \lambda m_2.(k_2 (m_2 \text{Joseph}))) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2.(\forall \lambda m_1.(\lambda m_2.(k_2 (m_2 \text{Joseph})) (\text{hates } m_1))) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2.(\forall \lambda m_1.(k_2 (\text{hates } m_1 \text{Joseph}))) \xrightarrow{\text{eval}} \forall \lambda m_1.(\text{hates } m_1 \text{Joseph})
 \end{aligned}$$

(3) Plotkin's CPS Interpretation

New *value* translation $\llbracket \cdot \rrbracket$:

$$\llbracket np \rrbracket = e, \llbracket s \rrbracket = t, \llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap (\llbracket B \rrbracket \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\llbracket A \rrbracket \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\llbracket \cdot \rrbracket$	Target
Joseph :: np	e	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$?$	$?$
everybody :: np	e	$\forall :: (et)t$

(3) Plotkin's CPS Interpretation

New *value* translation $\lceil \cdot \rceil$:

$$\lceil np \rceil = e, \lceil s \rceil = t, \lceil B/A \rceil = \lceil A \backslash B \rceil = \lceil A \rceil \multimap (\lceil B \rceil \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\lceil A \rceil \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\lceil \cdot \rceil$	Target
Joseph :: np	e	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$?$	$?$
everybody :: np	e	$\forall :: (et)t$

$$\lceil np \backslash s \rceil = \lceil np \rceil \multimap (\lceil s \rceil \multimap \perp) \multimap \perp = e \multimap (t \multimap t) \multimap t = e(tt)t$$

(3) Plotkin's CPS Interpretation

New *value* translation $\llbracket \cdot \rrbracket$:

$$\llbracket np \rrbracket = e, \llbracket s \rrbracket = t, \llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap (\llbracket B \rrbracket \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\llbracket A \rrbracket \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\llbracket \cdot \rrbracket$	Target
Joseph :: <i>np</i>	<i>e</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$e((e(tt)t)t)t$?
everybody :: <i>np</i>	<i>e</i>	$\forall :: (et)t$

$$\begin{aligned} \llbracket np \backslash s \rrbracket &= \llbracket np \rrbracket \multimap (\llbracket s \rrbracket \multimap \perp) \multimap \perp = e \multimap (t \multimap t) \multimap t = e(tt)t \\ \llbracket (np \backslash s)/np \rrbracket &= \llbracket np \rrbracket \multimap (\llbracket np \backslash s \rrbracket \multimap \perp) \multimap \perp = e((e(tt)t)t)t \end{aligned}$$

(3) Plotkin's CPS Interpretation

New *value* translation $\llbracket \cdot \rrbracket$:

$$\llbracket np \rrbracket = e, \llbracket s \rrbracket = t, \llbracket B/A \rrbracket = \llbracket A \backslash B \rrbracket = \llbracket A \rrbracket \multimap (\llbracket B \rrbracket \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\llbracket A \rrbracket \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\llbracket \cdot \rrbracket$	Target
Joseph :: np	e	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s)/np$	$e((e(tt)t)t)t$	$? :: ((e((e(tt)t)t)t)t)t$
everybody :: np	e	$\forall :: (et)t$

$$\begin{aligned} \llbracket np \backslash s \rrbracket &= \llbracket np \rrbracket \multimap (\llbracket s \rrbracket \multimap \perp) \multimap \perp = e \multimap (t \multimap t) \multimap t = e(tt)t \\ \llbracket (np \backslash s)/np \rrbracket &= \llbracket np \rrbracket \multimap (\llbracket np \backslash s \rrbracket \multimap \perp) \multimap \perp = e((e(tt)t)t)t \\ \overline{(np \backslash s)/np} &= ((e((e(tt)t)t)t)t)t \end{aligned}$$

(3) Plotkin's CPS Interpretation

New *value* translation $\llbracket \cdot \rrbracket$:

$$\llbracket np \rrbracket = e, \llbracket s \rrbracket = t, \llbracket B/A \rrbracket = \llbracket A \setminus B \rrbracket = \llbracket A \rrbracket \multimap (\llbracket B \rrbracket \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\llbracket A \rrbracket \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\llbracket \cdot \rrbracket$	Target
Joseph :: <i>np</i>	<i>e</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \setminus s)/np$	$e((e(tt)t)t)t$	$? :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	<i>e</i>	$\forall :: (et)t$

Term Assignment

$$\underbrace{\left(\left(\underbrace{\left(\underbrace{\overbrace{e}^x \left(\underbrace{\left(\underbrace{\overbrace{e}^y \left(\overbrace{(tt)}^\tau t \right) t}_{h} \right) t}_{f} \right) t}_{t} \right) t \right) t \right) t}_{t}$$

(3) Plotkin's CPS Interpretation

New *value* translation $\llbracket \cdot \rrbracket$:

$$\llbracket np \rrbracket = e, \llbracket s \rrbracket = t, \llbracket B/A \rrbracket = \llbracket A \setminus B \rrbracket = \llbracket A \rrbracket \multimap (\llbracket B \rrbracket \multimap \perp) \multimap \perp$$

And *computation* translation $\overline{A} = (\llbracket A \rrbracket \multimap \perp) \multimap \perp$ (here: $\perp = t$).

Source	Lexical Translation $\llbracket \cdot \rrbracket$	Target
Joseph :: <i>np</i>	<i>e</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \setminus s)/np$	$e((e(tt)t)t)t$	$? :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	<i>e</i>	$\forall :: (et)t$

Term Assignment

$$\underbrace{\left(\left(\underbrace{\left(\underbrace{\overbrace{e}^x \left(\underbrace{\left(\underbrace{\overbrace{e}^y \left(\overbrace{(tt)}^\tau t \right)}_h t \right)}_f t \right)}_f t \right)}_f t \right)}_f t \right)}_f t \quad \lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x y))))$$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: np	$\forall :: (et)t$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Term Translation

$$\overline{M \triangleleft N} = \overline{N \triangleright M} = \lambda k.(\overline{M} \lambda m.(\overline{N} \lambda n.(m \ n \ k)))$$

$$\begin{aligned}\overline{h \triangleleft e} &= \lambda k.(\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \lambda m.(\forall \lambda n.(m \ n \ k))) \\ &= \lambda k.(\lambda m.(\forall \lambda n.(m \ n \ k)) \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \\ &= \lambda k.(\forall \lambda n.(\lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y))) \ n \ k)) \\ &= \lambda k.(\forall \lambda n.(k \lambda y \tau.(\tau (\text{hates } n \ y))))\end{aligned}$$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Term Translation

$$\overline{M \triangleleft N} = \overline{N \triangleright M} = \lambda k.(\overline{M} \lambda m.(\overline{N} \lambda n.(m \ n \ k)))$$

$$\begin{aligned}
 \overline{h \triangleleft e} &= \lambda k.(\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \lambda m.(\forall \lambda n.(m \ n \ k))) \\
 &= \lambda k.(\lambda m.(\forall \lambda n.(m \ n \ k)) \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \\
 &= \lambda k.(\forall \lambda n.(\lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y))) \ n \ k)) \\
 &= \lambda k.(\forall \lambda n.(k \lambda y \tau.(\tau (\text{hates } n \ y))))
 \end{aligned}$$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Term Translation

$$\overline{M \triangleleft N} = \overline{N \triangleright M} = \lambda k.(\overline{M} \lambda m.(\overline{N} \lambda n.(m \ n \ k)))$$

$$\begin{aligned}\overline{h \triangleleft e} &= \lambda k.(\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \lambda m.(\forall \lambda n.(m \ n \ k))) \\ &= \lambda k.(\lambda m.(\forall \lambda n.(m \ n \ k)) \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \\ &= \lambda k.(\forall \lambda n.(\lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y))) \ n \ k)) \\ &= \lambda k.(\forall \lambda n.(k \lambda y \tau.(\tau (\text{hates } n \ y))))\end{aligned}$$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: <i>np</i>	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: <i>np</i>	$\forall :: (et)t$

Term Translation

$$\overline{M \triangleleft N} = \overline{N \triangleright M} = \lambda k.(\overline{M} \lambda m.(\overline{N} \lambda n.(m \ n \ k)))$$

$$\begin{aligned}\overline{h \triangleleft e} &= \lambda k.(\lambda f.(f \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \lambda m.(\forall \lambda n.(m \ n \ k))) \\ &= \lambda k.(\lambda m.(\forall \lambda n.(m \ n \ k)) \lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y)))) \\ &= \lambda k.(\forall \lambda n.(\lambda x h.(h \lambda y \tau.(\tau (\text{hates } x \ y))) \ n \ k)) \\ &= \lambda k.(\forall \lambda n.(k \lambda y \tau.(\tau (\text{hates } n \ y))))\end{aligned}$$

(3) Plotkin's CPS Interpretation

Lexical Translation

Source	Target
Joseph :: np	$\lambda k.(k \text{ Joseph}) :: (et)t$
hates :: $(np \backslash s) / np$	$\lambda f.(f \lambda x.h.(h \lambda y\tau.(\tau (\text{hates } x \ y)))) :: ((e((e(tt)t)t)t)t)t$
everybody :: np	$\forall :: (et)t$

Term Translation

$$\overline{M \triangleleft N} = \overline{N \triangleright M} = \lambda k.(\overline{M} \lambda m.(\overline{N} \lambda n.(m \ n \ k)))$$

$$\begin{aligned}
 \overline{J \triangleright (h \triangleleft e)} &= \lambda k_2. (\overline{h \triangleleft e} \lambda m_2. (\lambda k_3. (k_3 \text{ Joseph}) \lambda n_2. (m_2 \ n_2 \ k_2))) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2. (\overline{h \triangleleft e} \lambda m_2. (\lambda n_2. (m_2 \ n_2 \ k_2) \text{ Joseph})) \\
 &\stackrel{\beta}{\rightsquigarrow} \lambda k_2. (\overline{h \triangleleft e} \lambda m_2. (m_2 \text{ Joseph } k_2)) \\
 &= \lambda k_2. (\lambda k. (\forall \lambda n. (k \lambda y\tau. (\tau (\text{hates } n \ y)))) \lambda m_2. (m_2 \text{ Joseph } k_2)) \\
 &= \lambda k_2. (\forall \lambda n. (\lambda m_2. (m_2 \text{ Joseph } k_2) \lambda y\tau. (\tau (\text{hates } n \ y)))) \\
 &= \lambda k_2. (\forall \lambda n. (\lambda y\tau. (\tau (\text{hates } n \ y)) \text{ Joseph } k_2)) \\
 &= \lambda k_2. (\forall \lambda n. (k_2 (\text{hates } n \text{ Joseph}))) \xrightarrow{\text{eval}} \forall \lambda n. (\text{hates } n \text{ Joseph})
 \end{aligned}$$