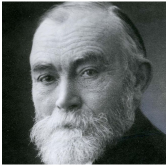
# Syntax-Semantics Interface

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

$$\begin{array}{ccccc} \text{syntactic calculus} & \xrightarrow{h_1} & \text{semantic calculus} & \xrightarrow{h_2} & \text{lexical meaning} \\ \text{NL / L / ..} & & \text{MILL}_{\multimap,\otimes} \text{ /..} & & \text{IL}_{\to,\times} \end{array}$$

| *surface form* | *meaning* (computational) | *meaning* (lexical) |
|---|---|---|

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: $\mathfrak{L}$, $\eta$, $\theta$ of ACGs

# Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\frac{\Gamma \vdash \mathtt{s} : B/A \quad \Delta \vdash \mathtt{t} : A}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleleft \mathtt{t} : B} \ /E \qquad \frac{\Gamma \vdash \mathtt{s} : B \quad \Delta \vdash \mathtt{t} : A \backslash B}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleright \mathtt{t} : B} \ \backslash E$$

$$\frac{\Gamma, \mathtt{x} : A \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^r . \mathtt{s} : B/A} \ /I \qquad \frac{\mathtt{x} : A, \Gamma \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^l . \mathtt{s} : A \backslash B} \ \backslash I$$

# Linear Syntax, Intuitionistic Semantics?

## Minimal term syntax for NL

$$\frac{\Gamma \vdash \mathtt{s} : B/A \quad \Delta \vdash \mathtt{t} : A}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleleft \mathtt{t} : B} \; /E \qquad \frac{\Gamma \vdash \mathtt{s} : B \quad \Delta \vdash \mathtt{t} : A\backslash B}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleright \mathtt{t} : B} \; \backslash E$$

$$\frac{\Gamma, \mathtt{x} : A \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^r.\mathtt{s} : B/A} \; /I \qquad \frac{\mathtt{x} : A, \Gamma \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^l.\mathtt{s} : A\backslash B} \; \backslash I$$

$$\dfrac{\dfrac{\dfrac{\text{the}}{np/n} \quad \dfrac{\text{worker}}{n}}{(\text{the} \cdot \text{worker}) \vdash np} \; /E \qquad \dfrac{\dfrac{\text{liberates}}{(np\backslash s)/np} \quad \dfrac{\text{herself}}{((np\backslash s)/np)\backslash(np\backslash s)}}{(\text{liberates} \cdot \text{herself}) \vdash np\backslash s} \; \backslash E}{((\text{the} \cdot \text{worker}) \cdot (\text{liberates} \cdot \text{herself})) \vdash s} \; \backslash E$$

# Linear Syntax, Intuitionistic Semantics?

## Minimal term syntax for NL

$$\frac{\Gamma \vdash \mathtt{s} : B/A \quad \Delta \vdash \mathtt{t} : A}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleleft \mathtt{t} : B} /E \qquad \frac{\Gamma \vdash \mathtt{s} : B \quad \Delta \vdash \mathtt{t} : A\backslash B}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleright \mathtt{t} : B} \backslash E$$

$$\frac{\Gamma, \mathtt{x} : A \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^r.\mathtt{s} : B/A} /I \qquad \frac{\mathtt{x} : A, \Gamma \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^l.\mathtt{s} : A\backslash B} \backslash I$$

$$\dfrac{\dfrac{\dfrac{\text{the}}{np/n} \quad \dfrac{\text{worker}}{n}}{(\text{the} \cdot \text{worker}) \vdash np} /E \quad \dfrac{\dfrac{\text{liberates}}{(np\backslash s)/np} \quad \dfrac{\text{herself}}{((np\backslash s)/np)\backslash(np\backslash s)}}{(\text{liberates} \cdot \text{herself}) \vdash np\backslash s} \backslash E}{((\text{the} \cdot \text{worker}) \cdot (\text{liberates} \cdot \text{herself})) \vdash s} \backslash E$$

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

# Linear Syntax, Intuitionistic Semantics?

## Minimal term syntax for NL

$$\frac{\Gamma \vdash \mathtt{s} : B/A \quad \Delta \vdash \mathtt{t} : A}{(\Gamma, \Delta) \vdash \mathtt{s} \lhd \mathtt{t} : B} /E \qquad \frac{\Gamma \vdash \mathtt{s} : B \quad \Delta \vdash \mathtt{t} : A\backslash B}{(\Gamma, \Delta) \vdash \mathtt{s} \rhd \mathtt{t} : B} \backslash E$$

$$\frac{\Gamma, \mathtt{x} : A \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^r.\mathtt{s} : B/A} /I \qquad \frac{\mathtt{x} : A, \Gamma \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^l.\mathtt{s} : A\backslash B} \backslash I$$

$$\dfrac{\dfrac{\dfrac{\text{the}}{np/n} \quad \dfrac{\text{worker}}{n}}{(\text{the} \cdot \text{worker}) \vdash np} /E \quad \dfrac{\dfrac{\text{liberates}}{(np\backslash s)/np} \quad \dfrac{\text{herself}}{((np\backslash s)/np)\backslash(np\backslash s)}}{(\text{liberates} \cdot \text{herself}) \vdash np\backslash s} \backslash E}{((\text{the} \cdot \text{worker}) \cdot (\text{liberates} \cdot \text{herself})) \vdash s} \backslash E$$

- Syntactic Term: $\mathtt{(the} \lhd \mathtt{worker)} \rhd \mathtt{(liberates} \rhd \mathtt{herself)}$
- Computational Term: `(herself liberates) (the worker)`

# Linear Syntax, Intuitionistic Semantics?

Minimal term syntax for NL

$$\frac{\Gamma \vdash \mathtt{s} : B/A \quad \Delta \vdash \mathtt{t} : A}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleleft \mathtt{t} : B} \, /E \qquad \frac{\Gamma \vdash \mathtt{s} : B \quad \Delta \vdash \mathtt{t} : A\backslash B}{(\Gamma, \Delta) \vdash \mathtt{s} \triangleright \mathtt{t} : B} \, \backslash E$$

$$\frac{\Gamma, \mathtt{x} : A \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^r.\mathtt{s} : B/A} \, /I \qquad \frac{\mathtt{x} : A, \Gamma \vdash \mathtt{s} : B}{\Gamma \vdash \lambda \mathtt{x}^l.\mathtt{s} : A\backslash B} \, \backslash I$$

$$\dfrac{\dfrac{\dfrac{\text{the}}{np/n} \quad \dfrac{\text{worker}}{n}}{(\text{the} \cdot \text{worker}) \vdash np} \, /E \qquad \dfrac{\dfrac{\text{liberates}}{(np\backslash s)/np} \quad \dfrac{\text{herself}}{((np\backslash s)/np)\backslash(np\backslash s)}}{(\text{liberates} \cdot \text{herself}) \vdash np\backslash s} \, \backslash E}{((\text{the} \cdot \text{worker}) \cdot (\text{liberates} \cdot \text{herself})) \vdash s} \, \backslash E$$

- Syntactic Term: `(the` $\triangleleft$ `worker)` $\triangleright$ `(liberates` $\triangleright$ `herself)`
- Computational Term: `(herself liberates) (the worker)`
- Lexical Term: `(liberates (the worker)) (the worker)`
  assigning `herself` $:= \lambda \mathtt{fx}.((\mathtt{f}\ \mathtt{x})\ \mathtt{x})$ 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 . . .
- Type-Theoretic
  Sentences as judgements, properties as dependent types, entities as simple types . . .
- 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]

[1] right-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 $\lceil . \rceil$ a mapping, s.t. $\lceil np \rceil = e$, $\lceil s \rceil = t$, $\lceil B/A \rceil = \lceil A \backslash B \rceil = \lceil A \rceil \multimap \lceil B \rceil$

[1] right-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 $\lceil . \rceil$ a mapping, s.t. $\lceil np \rceil = e$, $\lceil s \rceil = t$, $\lceil B/A \rceil = \lceil A \backslash B \rceil = \lceil A \rceil \multimap \lceil B \rceil$

☺ Fine for simple clauses:

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

[1] right-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 $\lceil . \rceil$ a mapping, s.t. $\lceil np \rceil = e$, $\lceil s \rceil = t$, $\lceil B/A \rceil = \lceil A \backslash B \rceil = \lceil A \rceil \multimap \lceil B \rceil$

☺ Fine for simple clauses:

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

☹ ..less so for quantifiers:

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

Impossible to derive with simple translation!

[1] right-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 $\eta$ is lifted from a function to a *relation*: Each syntactic source type is associated with a *set* of semantic target types.

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

where $\mathrm{shift}$ the reflexive, transitive closure of $\lceil . \rceil$ 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:
  $$\lambda \vec{x} w \vec{y}.(f\ \vec{x}\ (\lambda z.(z\ w))\ \vec{y}) : \vec{A} \multimap B \multimap \vec{C} \multimap E$$

# (1) Flexible Interpretation: Example

Lexical Translation

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph} :: e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates} :: eet$ |
| everybody :: $np$ | $\forall :: (et)t$ |

# (1) Flexible Interpretation: Example

**Lexical Translation**

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph}$ :: $e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates}$ :: $eet$ |
| everybody :: $np$ | $\forall$ :: $(et)t$ |

everybody: $np$
$\lceil np \rceil = e$

# (1) Flexible Interpretation: Example

Lexical Translation

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph}$ :: $e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates}$ :: $eet$ |
| 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: $np$

$\lceil np \rceil = e$

# (1) Flexible Interpretation: Example

Lexical Translation

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph} :: e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates} :: eet$ |
| 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: $np$

$\lceil np \rceil = e \stackrel{vr}{\Longrightarrow} \forall : (et)t$

# (1) Flexible Interpretation: Example

Lexical Translation

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph} :: e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates} :: eet$ |
| everybody :: $np$ | $\forall :: (et)t$ |

everybody: $np$

$\lceil np \rceil = e \xRightarrow{vr} \forall : (et)t$

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

$\lceil (np\backslash s)/np \rceil = eet$

# (1) Flexible Interpretation: Example

**Lexical Translation**

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph} :: e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates} :: eet$ |
| 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: $np$

$\lceil np \rceil = e \overset{vr}{\Longrightarrow} \forall : (et)t$

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

$\lceil (np\backslash s)/np \rceil = eet$

# (1) Flexible Interpretation: Example

Lexical Translation

| Source | Target |
|---|---|
| Joseph :: $np$ | $\texttt{Joseph}$ :: $e$ |
| hates :: $(np\backslash s)/np$ | $\texttt{hates}$ :: $eet$ |
| everybody :: $np$ | $\forall$ :: $(et)t$ |

Argument Raising

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

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

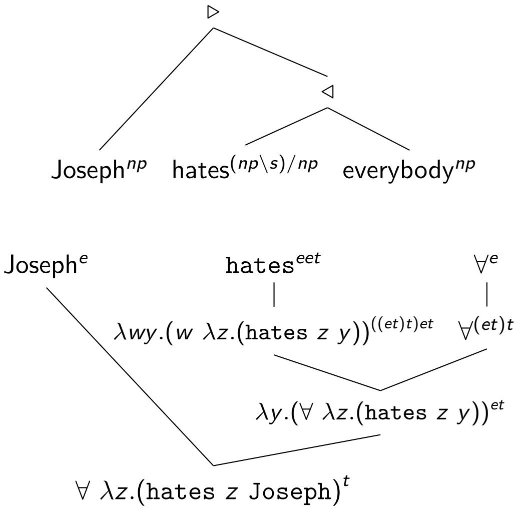
everybody: $np$

$\lceil np \rceil = e \overset{vr}{\Longrightarrow} \forall : (et)t$

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

$\lceil (np\backslash s)/np \rceil = eet \overset{ar0}{\Longrightarrow} \lambda wy.(w\ \lambda z.(\texttt{hates}\ z\ y)) : ((et)t)et$

# (1) Flexible Interpretation: Example



# (2) Incremental CPS

Two translations left-to-right $(.)^{\rightsquigarrow}$ and right-to-left incremental $(.)^{\leftsquigarrow}$
For $A$ atomic, $A^{\rightsquigarrow} = A^{\leftsquigarrow} = (\lceil A \rceil \multimap \bot) \multimap \bot$[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 $(.)^{\leftsquigarrow}$

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

[2] $\bot$ the *response* type, here $t$

# (2) Incremental CPS: Example

**Lexical Translation**

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

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

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

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

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

# (2) Incremental CPS: Example

**Lexical Translation**

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

**Left-to-right translation**

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

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

$$\begin{aligned}(\mathrm{h} \triangleleft \mathrm{e})^{\leadsto} &= \lambda k_1.(\mathrm{hates}^{\leadsto}\ \lambda m_1.(\mathrm{everybody}^{\leadsto}\ \lambda n_1.(k_1\ (m_1\ n_1))))\\ &= \lambda k_1.(\lambda k_0.(k_0\ \texttt{hates})\ \lambda m_1.(\forall\ \lambda n_1.(k_1\ (m_1\ n_1))))\\ &\overset{\beta}{\leadsto} \lambda k_1.(\lambda m_1.(\forall\ \lambda n_1.(k_1\ (m_1\ n_1)))\ \texttt{hates})\end{aligned}$$

# (2) Incremental CPS: Example

| Lexical Translation | |
|---|---|
| Source | Target |
| Joseph :: $np$ | $\lambda k.(k\ \texttt{Joseph})$ :: $(et)t$ |
| hates :: $(np\backslash s)/np$ | $\lambda k.(k\ \texttt{hates})$ :: $((eet)t)t$ |
| everybody :: $np$ | $\forall$ :: $(et)t$ |

Left-to-right translation

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

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

$$\begin{aligned}
(\mathrm{h} \triangleleft \mathrm{e})^{\leadsto} &= \lambda k_1.(\mathrm{hates}^{\leadsto}\ \lambda m_1.(\mathrm{everybody}^{\leadsto}\ \lambda n_1.(k_1\ (m_1\ n_1))))\\
&= \lambda k_1.(\lambda k_0.(k_0\ \texttt{hates})\ \lambda m_1.(\forall\ \lambda n_1.(k_1\ (m_1\ n_1))))\\
&\overset{\beta}{\leadsto} \lambda k_1.(\lambda m_1.(\forall\ \lambda n_1.(k_1\ (m_1\ n_1)))\ \texttt{hates})\\
&\overset{\beta}{\leadsto} \lambda k_1.(\forall\ \lambda n_1.(k_1\ (\texttt{hates}\ n_1))))
\end{aligned}$$

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

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

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

$$\begin{aligned}(\mathrm{J} \triangleright (\mathrm{h} \triangleleft \mathrm{e}))^{\leadsto} &= \lambda k_2.(\mathrm{Joseph}^{\leadsto}\ \lambda n_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\ &= \lambda k_2.\,(\lambda k_3.\,(k_3\ \texttt{Joseph})\ \lambda n_2.\,((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ n_2)))\end{aligned}$$

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

$$\begin{aligned}(\mathrm{J} \triangleright (\mathrm{h} \triangleleft \mathrm{e}))^{\leadsto} &= \lambda k_2.(\mathrm{Joseph}^{\leadsto}\ \lambda n_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\ &= \lambda k_2.\,(\lambda k_3.\,(k_3\ \texttt{Joseph})\ \lambda n_2.\,((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ n_2)))\\ &\stackrel{\beta}{\leadsto} \lambda k_2.\,(\lambda n_2.\,((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ n_2))\ \texttt{Joseph})\end{aligned}$$

# (2) Incremental CPS: Example

**Lexical Translation**

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

**Left-to-right translation**

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

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

$$\begin{aligned}
(\mathsf{J} \triangleright (\mathsf{h} \triangleleft \mathsf{e}))^{\leadsto} &= \lambda k_2.(\mathsf{Joseph}^{\leadsto}\ \lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\
&= \lambda k_2.\,(\lambda k_3.\,(k_3\ \texttt{Joseph})\ \lambda n_2.\,((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ n_2)))\\
&\overset{\beta}{\leadsto} \lambda k_2.\,(\lambda n_2.\,((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ n_2))\ \texttt{Joseph})\\
&\overset{\beta}{\leadsto} \lambda k_2.\,((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.\,(k_2\ (m_2\ \texttt{Joseph}))
\end{aligned}$$

# (2) Incremental CPS: Example

| Lexical Translation | |
|---|---|
| Source | Target |
| Joseph :: $np$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | $\lambda k.(k\ \texttt{hates}) :: ((eet)t)t$ |
| everybody :: $np$ | $\forall :: (et)t$ |

Left-to-right translation

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

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

$$\begin{aligned}
(\mathsf{J} \triangleright (\mathsf{h} \triangleleft \mathsf{e}))^{\rightsquigarrow} &= \lambda k_2.(\mathsf{Joseph}^{\rightsquigarrow}\ \lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\rightsquigarrow}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\
&= \lambda k_2.(\lambda k_3.(k_3\ \texttt{Joseph})\ \lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\rightsquigarrow}\ \lambda m_2.(k_2\ (m_2\ n_2)))\\
&\overset{\beta}{\rightsquigarrow} \lambda k_2.(\lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\rightsquigarrow}\ \lambda m_2.(k_2\ (m_2\ n_2))\ \texttt{Joseph})\\
&\overset{\beta}{\rightsquigarrow} \lambda k_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\rightsquigarrow}\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\\
&= \lambda k_2.(\lambda k_1.(\forall\ \lambda n_1.(k_1\ (\texttt{hates}\ n_1)))\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph})))
\end{aligned}$$

# (2) Incremental CPS: Example

**Lexical Translation**

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

**Left-to-right translation**

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

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

$$\begin{aligned}
(\mathsf{J} \triangleright (\mathsf{h} \triangleleft \mathsf{e}))^{\leadsto} &= \lambda k_2.(\mathsf{Joseph}^{\leadsto}\ \lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\
&= \lambda k_2.(\lambda k_3.(k_3\ \texttt{Joseph})\ \lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))\\
&\overset{\beta}{\leadsto} \lambda k_2.(\lambda n_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2))\ \texttt{Joseph})\\
&\overset{\beta}{\leadsto} \lambda k_2.((\mathsf{hates} \triangleleft \mathsf{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\\
&= \lambda k_2.(\lambda k_1.(\forall\ \lambda n_1.(k_1\ (\texttt{hates}\ n_1)))\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\\
&\overset{\beta}{\leadsto} \lambda k_2.(\forall\ \lambda n_1.(\lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\ (\texttt{hates}\ n_1)))
\end{aligned}$$

# (2) Incremental CPS: Example

Lexical Translation

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

Left-to-right translation

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

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

$$\begin{aligned}
(\mathrm{J} \triangleright (\mathrm{h} \triangleleft \mathrm{e}))^{\leadsto} &= \lambda k_2.(\mathrm{Joseph}^{\leadsto}\ \lambda n_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2)))))\\
&= \lambda k_2.(\lambda k_3.(k_3\ \texttt{Joseph})\ \lambda n_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2))))\\
&\overset{\beta}{\leadsto} \lambda k_2.(\lambda n_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ n_2))\ \texttt{Joseph})\\
&\overset{\beta}{\leadsto} \lambda k_2.((\mathrm{hates} \triangleleft \mathrm{everybody})^{\leadsto}\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\\
&= \lambda k_2.(\lambda k_1.(\forall\ \lambda n_1.(k_1\ (\texttt{hates}\ n_1)))\ \lambda m_2.(k_2\ (m_2\ \texttt{Joseph})))\\
&\overset{\beta}{\leadsto} \lambda k_2.(\forall\ \lambda n_1.(\lambda m_2.(k_2\ (m_2\ \texttt{Joseph}))\ (\texttt{hates}\ n_1)))\\
&\overset{\beta}{\leadsto} \lambda k_2.(\forall\ \lambda n_1.(k_2\ (\texttt{hates}\ n_1\ \texttt{Joseph})))\ \xrightarrow{\text{eval}}\ \forall\lambda n_1.(\texttt{hates}\ n_1\ \texttt{Joseph})
\end{aligned}$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

$$\lceil np \backslash s \rceil =$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

$$\lceil np \backslash s \rceil = \lceil np \rceil \multimap (\lceil s \rceil \multimap \bot) \multimap \bot =$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

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

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

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

$$\lceil (np\backslash s)/np \rceil =$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \mathtt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | ? | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

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

$$\lceil (np\backslash s)/np \rceil = \lceil np \rceil \multimap (\lceil np\backslash s \rceil \multimap \bot) \multimap \bot$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

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

$$\lceil (np\backslash s)/np \rceil = \lceil np \rceil \multimap (\lceil np\backslash s \rceil \multimap \bot) \multimap \bot = e\,((e\,(tt)\,t)\,t)\,t$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np\backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | ? |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

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

$$\lceil (np\backslash s)/np \rceil = \lceil np \rceil \multimap (\lceil np\backslash s \rceil \multimap \bot) \multimap \bot = e\,((e\,(tt)\,t)\,t)\,t$$

$$\overline{(np\backslash s)/np} =$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \mathtt{Joseph})$ :: $(et)t$ |
| hates :: $(np \backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | ? :: $((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall$ :: $(et)t$ |

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

$$\lceil (np \backslash s)/np \rceil = \lceil np \rceil \multimap (\lceil np \backslash s \rceil \multimap \bot) \multimap \bot = e\,((e\,(tt)\,t)\,t)\,t$$

$$\overline{(np \backslash s)/np} = ((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph})$ :: $(et)t$ |
| hates :: $(np\backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | ? :: $((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall$ :: $(et)t$ |

Term Assignment

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

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | $? :: ((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

Term Assignment

$$\underbrace{\Big(\Big(\overbrace{e}^{x}\underbrace{\Big(\Big(\overbrace{e}^{y}\overbrace{(tt)}^{\tau}\,t\Big)\,t\Big)}_{h}\,t\Big)\,t\Big)}_{f}\,t \qquad \lambda f.(f \qquad\qquad )$$

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | $? :: ((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

Term Assignment

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

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | $? :: ((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

Term Assignment

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

# (3) Plotkin's CPS Interpretation

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

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

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

Lexical Translation

| Source | Value $\lceil . \rceil$ | Target |
|---|---|---|
| Joseph :: $np$ | $e$ | $\lambda k.(k\ \texttt{Joseph}) :: (et)t$ |
| hates :: $(np \backslash s)/np$ | $e\,((e\,(tt)\,t)\,t)\,t$ | ? :: $((e\,((e\,(tt)\,t)\,t)\,t)\,t)\,t$ |
| everybody :: $np$ | $e$ | $\forall :: (et)t$ |

Term Assignment

$$\underbrace{\left(\left(\overbrace{e}^{x}\ \underbrace{\left(\left(\overbrace{e}^{y}\ \overbrace{(tt)}^{\tau}\ t\right)t\right)}_{h}\ t\right)t\right)}_{f}\ t \qquad \lambda f.(f\ \lambda xh.(h\ \lambda y\tau.(\tau\ (\texttt{hates}\ x\ y))))$$

# (3) Plotkin's CPS Interpretation

Lexical Translation

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

# (3) Plotkin's CPS Interpretation

Lexical Translation

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

Term Translation

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

# (3) Plotkin's CPS Interpretation

Lexical Translation

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

## Term Translation

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

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

# (3) Plotkin's CPS Interpretation

### Lexical Translation

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

### Term Translation

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

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