LING5702: Lecture Notes 10
Broad-coverage Compositional Semantics

Recall from the last lecture notes that we can assemble meanings from sub-parts by:

1. composing words into phrases and sentences using grammar rules; then
2. associating these words and rules with meanings via function application in lambda calculus.

Previously we focused on point 1; this lecture will focus on point 2.

Contents

10.1 We can build meanings using beta reduction [Montague, 1973]

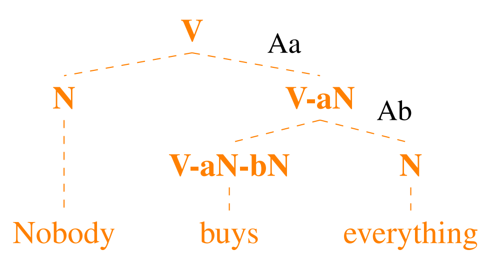
Recall we can assign sentence meanings to lambda calculus expressions (*Nobody buys everything*):

$$\begin{array}{l} \mathsf{None}\ (\lambda_x\ \mathsf{Person}\ x) \\ \quad (\lambda_x\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y) \\ \qquad (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e) \\ \qquad\qquad (\lambda_e\ \mathsf{True}))) \end{array}$$

These can be carved up and assigned to words and rules:

$$\begin{array}{l} \mathsf{None}\ (\lambda_x\ \mathsf{Person}\ x) \quad \leftarrow \textit{nobody} \\ \quad (\lambda_x\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y) \quad \leftarrow \textit{everything} \\ \qquad (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e) \quad \leftarrow \textit{buys} \\ \qquad\qquad (\lambda_e\ \mathsf{True}))) \end{array}$$

And associated with words and rules on the syntactic analysis tree:



If we make semantic argument structure mirror syntactic argument structure, we can define words:

$$\begin{aligned}
\textit{nobody} &\stackrel{\text{def}}{=} \lambda_r\, \lambda_s\, \mathsf{None}\ (\lambda_x\, \mathsf{Person}\ x \wedge r\ x) \\
&\qquad\qquad\qquad (\lambda_x\, \mathsf{True} \wedge s\ x) \\
\textit{everything} &\stackrel{\text{def}}{=} \lambda_r\, \lambda_s\, \mathsf{All}\ (\lambda_y\, \mathsf{Thing}\ y \wedge r\ y) \\
&\qquad\qquad\qquad (\lambda_y\, \mathsf{True} \wedge s\ y) \\
\textit{buys} &\stackrel{\text{def}}{=} \lambda_p\, \lambda_q\, \lambda_r\, \lambda_s\, q\ (\lambda_x\, \mathsf{True}) \\
&\qquad\qquad (\lambda_x\, p\ (\lambda_y\, \mathsf{True}) \\
&\qquad\qquad\quad (\lambda_y\, \mathsf{Some}\ (\lambda_e\, \mathsf{Buy}\ y\ x\ e \wedge r\ e) \\
&\qquad\qquad\qquad\quad (\lambda_e\, \mathsf{True} \wedge s\ e)))
\end{aligned}$$

and we can define compositional semantic functions for basic argument attachment rules:

(Aa) $\lambda_{f:\beta}\, \lambda_{g:\alpha\text{-}\mathbf{a}\beta}\, (g\ f) : \alpha$ (applies the right child to the left child)

(Ab) $\lambda_{f:\alpha\text{-}\mathbf{b}\beta}\, \lambda_{g:\beta}\, (f\ g) : \alpha$ (applies the left child to the right child)

This translates the **V-aN** into (Ab *buys* *everything*) and **V** into (Aa *nobody* (Ab *buys* *everything*)).

We can then substitute the rule and word definitions and simplify using **beta reduction**:

$$(\lambda_\chi\ \ldots\chi\ldots)\ \varphi = \ldots\varphi\ldots$$

This removes the first λ_χ and replaces its variable χ with expression φ everywhere in $\ldots\chi\ldots$ (like how variables are replaced with arguments in a normal function).

The result of the first function application (of *buys* to *everything*), replacing $f, g, p, r, \ldots$ is:

$$\begin{aligned}
\text{Ab}\ \textit{buys}\ \textit{everything} &= (\lambda_f\, \lambda_g\, f\ g) && \text{substitute definitions} \\
&\quad (\lambda_p\, \lambda_q\, \lambda_r\, \lambda_s\, q\ (\lambda_x\, \mathsf{True}) \\
&\qquad\qquad (\lambda_x\, p\ (\lambda_y\, \mathsf{True}) \\
&\qquad\qquad\quad (\lambda_y\, \mathsf{Some}\ (\lambda_e\, \mathsf{Buy}\ y\ x\ e \wedge r\ e) \\
&\qquad\qquad\qquad\quad (\lambda_e\, \mathsf{True} \wedge s\ e)))) \\
&\quad (\lambda_r\, \lambda_s\, \mathsf{All}\ (\lambda_y\, \mathsf{Thing}\ y \wedge r\ y) \\
&\qquad\qquad (\lambda_y\, \mathsf{True} \wedge s\ y)) \\
&= (\lambda_g\, (\lambda_p\, \lambda_q\, \lambda_r\, \lambda_s\, q\ (\lambda_x\, \mathsf{True}) && \text{replace } f \\
&\qquad\qquad (\lambda_x\, p\ (\lambda_y\, \mathsf{True}) \\
&\qquad\qquad\quad (\lambda_y\, \mathsf{Some}\ (\lambda_e\, \mathsf{Buy}\ y\ x\ e \wedge r\ e) \\
&\qquad\qquad\qquad\quad (\lambda_e\, \mathsf{True} \wedge s\ e))))\ g) \\
&\quad (\lambda_r\, \lambda_s\, \mathsf{All}\ (\lambda_y\, \mathsf{Thing}\ y \wedge r\ y) \\
&\qquad\qquad (\lambda_y\, \mathsf{True} \wedge s\ y)) \\
&= (\lambda_p\, \lambda_q\, \lambda_r\, \lambda_s\, q\ (\lambda_x\, \mathsf{True}) && \text{replace } g \\
&\qquad\qquad (\lambda_x\, p\ (\lambda_y\, \mathsf{True}) \\
&\qquad\qquad\quad (\lambda_y\, \mathsf{Some}\ (\lambda_e\, \mathsf{Buy}\ y\ x\ e \wedge r\ e) \\
&\qquad\qquad\qquad\quad (\lambda_e\, \mathsf{True} \wedge s\ e)))) \\
&\quad (\lambda_r\, \lambda_s\, \mathsf{All}\ (\lambda_y\, \mathsf{Thing}\ y \wedge r\ y) \\
&\qquad\qquad (\lambda_y\, \mathsf{True} \wedge s\ y))
\end{aligned}$$

$$
\begin{aligned}
&= \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x \mathsf{True}) && \text{replace } p\\
&\qquad (\lambda_x\ (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge r\ y)\\
&\qquad\qquad\qquad (\lambda_y\ \mathsf{True} \wedge s\ y))\\
&\qquad\quad (\lambda_y \mathsf{True})\\
&\qquad\quad (\lambda_y \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e)))\\
&= \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x \mathsf{True}) && \text{replace } r\\
&\qquad (\lambda_x\ (\lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge (\lambda_y\ \mathsf{True})\ y)\\
&\qquad\qquad\qquad (\lambda_y\ \mathsf{True} \wedge s\ y))\\
&\qquad\quad (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e)))\\
&= \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x \mathsf{True}) && \text{replace } y\\
&\qquad (\lambda_x\ (\lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge \mathsf{True})\\
&\qquad\qquad\qquad (\lambda_y\ \mathsf{True} \wedge s\ y))\\
&\qquad\quad (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e)))\\
&= \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \mathsf{True}) && \text{replace } s\\
&\qquad (\lambda_x\ \ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge \mathsf{True})\\
&\qquad\qquad (\lambda_y\ \mathsf{True} \wedge (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e))\ y))\\
&= \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \mathsf{True}) && \text{replace } y\\
&\qquad (\lambda_x\ \ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge \mathsf{True})\\
&\qquad\qquad (\lambda_y\ \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e)))
\end{aligned}
$$

The result of the second function application (of *buys everything* to *nobody*) is similar:

$$
\begin{aligned}
\mathrm{Aa}\ \textit{nobody}\ (\mathrm{Ab}\ \textit{buys everything}) &= \lambda_r\ \lambda_s\ \mathsf{None}\ (\lambda_x\ \mathsf{Person}\ x \wedge \mathsf{True})\\
&\qquad (\lambda_x\ \mathsf{True} \wedge \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge \mathsf{True})\\
&\qquad\qquad (\lambda_y\ \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ \mathsf{True} \wedge s\ e)))\\
&= \lambda_r\ \lambda_s\ \mathsf{None}\ (\lambda_x\ \mathsf{Person}\ x) && \text{conjunction with } \mathsf{True}\\
&\qquad (\lambda_x\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y)\\
&\qquad\qquad (\lambda_y\ \mathsf{Some}\ (\lambda_e\ \mathsf{Buy}\ y\ x\ e \wedge r\ e)\\
&\qquad\qquad\qquad (\lambda_e\ s\ e)))
\end{aligned}
$$

The leftover λ_r will be needed if the clause is modified:

- *Now nobody buys everything* adds $\mathsf{Now}\ e$ to the restriction.

The leftover λ_s will be needed if the clause is used as a complement:

- *I see nobody buys everything* adds $\mathsf{See}\ e\ \mathsf{Me}$ to the nuclear scope.

If that's the end of the derivation, we add a context, which can be empty: $(\lambda_z\ \mathsf{True})\ (\lambda_z\ \mathsf{True})$.

All primitive categories are modified and quantified, so we give them all this type: (e→t)→(e→t)→t.

Practice 10.1:

Beta reduce the following expression:

$$(\lambda_s \ \mathsf{Count} \ (\lambda_x \ s \ x \wedge \mathsf{Employee} \ x)) \ (\lambda_y \ \mathsf{Temporary} \ y)$$

10.2 Modifier schemata

Unlike arguments, modifiers add constraints to the *restrictors* of their modificands:

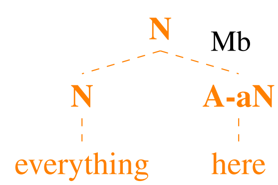
$$\text{(Ma)} \quad \overbrace{\lambda_{f:\sigma\text{-}\mathbf{a}\tau}}^{\text{modifier}} \ \overbrace{\lambda_{g:\upsilon\varphi_{1..N}}}^{\text{modificand}} \ \overbrace{(\lambda_{h_{N..1},r,s}}^{\text{local arguments}} \ g \ \overbrace{h_N \ldots h_1}^{\text{pass to modificand}} \ (\lambda_x \ r \ x \wedge \overbrace{f \ (\lambda_{t,u} \ t \ x \wedge u \ x) \ (\lambda_z \ \mathsf{True}) \ (\lambda_z \ \mathsf{True})) \ s}^{\text{modifier constrains restrictor of modificand}}) : \upsilon\varphi_{1..N}$$

$$\text{(Mb)} \quad \underbrace{\lambda_{f:\upsilon\varphi_{1..N}}}_{\text{modificand}} \ \underbrace{\lambda_{g:\sigma\text{-}\mathbf{a}\tau}}_{\text{modifier}} \ \underbrace{(\lambda_{h_{N..1},r,s}}_{\text{local arguments}} \ f \ \underbrace{h_N \ldots h_1}_{\text{pass to modificand}} \ (\lambda_x \ r \ x \wedge \underbrace{g \ (\lambda_{t,u} \ t \ x \wedge u \ x) \ (\lambda_z \ \mathsf{True}) \ (\lambda_z \ \mathsf{True})) \ s}_{\text{modifier constrains restrictor of modificand}}) : \upsilon\varphi_{1..N}$$

For example, with the above definition for *everyone* and this definition for *here*:

$$\begin{aligned} \mathit{here} \stackrel{\text{def}}{=} \lambda_q \ \lambda_r \ \lambda_s \ q \ & (\lambda_x \ \mathsf{True}) \\ & (\lambda_x \ \mathsf{Some} \ (\lambda_e \ \mathsf{Here} \ x \ e \wedge r \ e) \\ & \qquad\qquad (\lambda_e \ \mathsf{True} \wedge s \ e)) \end{aligned}$$

and with this syntactic analysis of the noun phrase *everything here*:



we generate this meaning (using $N = 0$ in the Mb rule, because the modificand has no arguments):

$$\begin{aligned} \mathsf{Mb} \ \mathit{everything} \ \mathit{here} = \ & (\lambda_f \ \lambda_g \ \lambda_r \ \lambda_s \ f \ (\lambda_x \ r \ x \wedge g \ (\lambda_t \ \lambda_u \ t \ x \wedge u \ x) \\ & \qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z \ \mathsf{True}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z \ \mathsf{True})) \\ & \qquad\qquad\qquad s) \\ & (\lambda_r \ \lambda_s \ \mathsf{All} \ (\lambda_y \ \mathsf{Thing} \ y \wedge r \ y) \\ & \qquad\qquad\quad (\lambda_y \ \mathsf{True} \wedge s \ y)) \\ & (\lambda_q \ \lambda_r \ \lambda_s \ q \ (\lambda_x \ \mathsf{True}) \\ & \qquad\qquad\quad (\lambda_x \ \mathsf{Some} \ (\lambda_e \ \mathsf{Here} \ x \ e \wedge r \ e) \\ & \qquad\qquad\qquad\qquad\quad (\lambda_e \ \mathsf{True} \wedge s \ e))) \end{aligned}$$

$$
\begin{aligned}
= \ & (\lambda_g\ \lambda_r\ \lambda_s\ (\lambda_r\ \lambda_s\ \text{All}\ (\lambda_y\ \text{Thing}\ y \wedge r\ y) \\
& \qquad\qquad\qquad\qquad (\lambda_y\ \text{True} \wedge s\ y)) \\
& \qquad\qquad (\lambda_x\ r\ x \wedge g\ (\lambda_t\ \lambda_u\ t\ x \wedge u\ x) \\
& \qquad\qquad\qquad\qquad (\lambda_z\ \text{True}) \\
& \qquad\qquad\qquad\qquad (\lambda_z\ \text{True})) \\
& \qquad\qquad s) \\
& (\lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \text{True}) \\
& \qquad\qquad (\lambda_x\ \text{Some}\ (\lambda_e\ \text{Here}\ x\ e \wedge r\ e) \\
& \qquad\qquad\qquad\qquad (\lambda_e\ \text{True} \wedge s\ e))) \\
= \ & (\lambda_g\ \lambda_r\ \lambda_s\ (\lambda_s\ \text{All}\ (\lambda_y\ \text{Thing}\ y \wedge\ (\lambda_x\ r\ x \wedge g\ (\lambda_t\ \lambda_u\ t\ x \wedge u\ x) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True})) \\
& \qquad\qquad\qquad\qquad\qquad\qquad y) \\
& \qquad\qquad\qquad\quad (\lambda_y\ \text{True} \wedge s\ y)) \\
& \qquad\qquad s) \\
& (\lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \text{True}) \\
& \qquad\qquad (\lambda_x\ \text{Some}\ (\lambda_e\ \text{Here}\ x\ e \wedge r\ e) \\
& \qquad\qquad\qquad\qquad (\lambda_e\ \text{True} \wedge s\ e))) \\
= \ & (\lambda_g\ \lambda_r\ \lambda_s\ (\lambda_s\ \text{All}\ (\lambda_y\ \text{Thing}\ y \wedge\ r\ y \wedge g\ (\lambda_t\ \lambda_u\ t\ y \wedge u\ y) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True})) \\
& \qquad\qquad\qquad\quad (\lambda_y\ \text{True} \wedge s\ y)) \\
& \qquad\qquad s) \\
& (\lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \text{True}) \\
& \qquad\qquad (\lambda_x\ \text{Some}\ (\lambda_e\ \text{Here}\ x\ e \wedge r\ e) \\
& \qquad\qquad\qquad\qquad (\lambda_e\ \text{True} \wedge s\ e))) \\
= \ & (\lambda_g\ \lambda_r\ \lambda_s\ \text{All}\ (\lambda_y\ \text{Thing}\ y \wedge\ r\ y \wedge g\ (\lambda_t\ \lambda_u\ t\ y \wedge u\ y) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True})) \\
& \qquad\qquad\quad (\lambda_y\ \text{True} \wedge s\ y)) \\
& (\lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \text{True}) \\
& \qquad\qquad (\lambda_x\ \text{Some}\ (\lambda_e\ \text{Here}\ x\ e \wedge r\ e) \\
& \qquad\qquad\qquad\qquad (\lambda_e\ \text{True} \wedge s\ e))) \\
= \ & (\lambda_r\ \lambda_s\ \text{All}\ (\lambda_y\ \text{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \text{True}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_x\ \text{Some}\ (\lambda_e\ \text{Here}\ x\ e \wedge r\ e) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_e\ \text{True} \wedge s\ e))) \\
& \qquad\qquad\qquad\qquad\qquad\qquad (\lambda_t\ \lambda_u\ t\ y \wedge u\ y) \\
& \qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad (\lambda_z\ \text{True})) \\
& \qquad\qquad (\lambda_y\ \text{True} \wedge s\ y))
\end{aligned}
$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_r\ \lambda_s\ (\lambda_t\ \lambda_u\ t\ y \wedge u\ y) \\
\hspace{16em} (\lambda_x\ \mathsf{True}) \\
\hspace{16em} (\lambda_x\ \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ x\ e \wedge r\ e) \\
\hspace{23em} (\lambda_e\ \mathsf{True} \wedge s\ e))) \\
\hspace{12em} (\lambda_z\ \mathsf{True}) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_r\ \lambda_s\ (\lambda_u\ (\lambda_x\ \mathsf{True})\ y \wedge u\ y) \\
\hspace{16em} (\lambda_x\ \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ x\ e \wedge r\ e) \\
\hspace{23em} (\lambda_e\ \mathsf{True} \wedge s\ e))) \\
\hspace{12em} (\lambda_z\ \mathsf{True}) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_r\ \lambda_s\ (\lambda_u\ \mathsf{True} \wedge u\ y) \\
\hspace{16em} (\lambda_x\ \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ x\ e \wedge r\ e) \\
\hspace{23em} (\lambda_e\ \mathsf{True} \wedge s\ e))) \\
\hspace{12em} (\lambda_z\ \mathsf{True}) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_r\ \lambda_s\ \mathsf{True} \wedge (\lambda_x\ \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ x\ e \wedge r\ e) \\
\hspace{28em} (\lambda_e\ \mathsf{True} \wedge s\ e)) \\
\hspace{21em} y \\
\hspace{12em} (\lambda_z\ \mathsf{True}) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_r\ \lambda_s\ \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ y\ e \wedge r\ e) \\
\hspace{25em} (\lambda_e\ \mathsf{True} \wedge s\ e)) \\
\hspace{12em} (\lambda_z\ \mathsf{True}) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_s\ \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ y\ e \wedge (\lambda_z\ \mathsf{True})\ e) \\
\hspace{23em} (\lambda_e\ \mathsf{True} \wedge s\ e)) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge\ r\ y \wedge\ (\lambda_s\ \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ y\ e \wedge \mathsf{True})) \\
\hspace{23em} (\lambda_e\ \mathsf{True} \wedge s\ e)) \\
\hspace{12em} (\lambda_z\ \mathsf{True})) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$\begin{array}{l}
= (\lambda_r\ \lambda_s\ \mathsf{All}\ (\lambda_y\ \mathsf{Thing}\ y \wedge r\ y \wedge \mathsf{True} \wedge \mathsf{Some}\ (\lambda_e\ \mathsf{Here}\ y\ e \wedge \mathsf{True}) \\
\hspace{20em} (\lambda_e\ \mathsf{True} \wedge (\lambda_z\ \mathsf{True})\ e)) \\
\hspace{5em} (\lambda_y\ \mathsf{True} \wedge s\ y))
\end{array}$$

$$= (\lambda_r\ \lambda_s\ \textsf{All}\ (\lambda_y\ \textsf{Thing}\ y \wedge r\ y \wedge \textsf{True} \wedge \textsf{Some}\ (\lambda_e\ \textsf{Here}\ y\ e \wedge \textsf{True})\ (\lambda_e\ \textsf{True} \wedge \textsf{True}))\ (\lambda_y\ \textsf{True} \wedge s\ y))$$

$$= (\lambda_r\ \lambda_s\ \textsf{All}\ (\lambda_y\ \textsf{Thing}\ y \wedge r\ y \wedge \textsf{Some}\ (\lambda_e\ \textsf{Here}\ y\ e)\ (\lambda_e\ \textsf{True}))\ (\lambda_y\ s\ y))$$

Practice 10.2:

Given the above composition rules and the following word definitions:

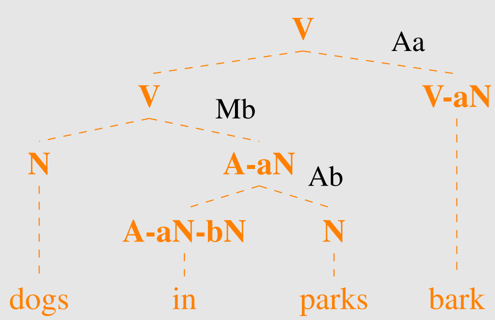
$$\textit{dogs} \stackrel{\text{def}}{=} (\lambda_r\ \lambda_s\ \textsf{Most}\ (\lambda_x\ \textsf{Dog}\ x \wedge r\ x)\ (\lambda_x\ \textsf{True} \wedge s\ x)) : \mathbf{N}$$

$$\textit{in} \stackrel{\text{def}}{=} (\lambda_p\ \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \textsf{True})\ (\lambda_x\ p\ (\lambda_y\ \textsf{True})\ (\lambda_y\ \textsf{Some}\ (\lambda_e\ \textsf{In}\ y\ x\ e \wedge r\ e)\ (\lambda_e\ \textsf{True} \wedge s\ e)))) : \mathbf{A\text{-}aN\text{-}bN}$$

$$\textit{parks} \stackrel{\text{def}}{=} (\lambda_r\ \lambda_s\ \textsf{Most}\ (\lambda_y\ \textsf{Park}\ y \wedge r\ y)\ (\lambda_y\ \textsf{True} \wedge s\ y)) : \mathbf{N}$$

$$\textit{bark} \stackrel{\text{def}}{=} \lambda_q\ \lambda_r\ \lambda_s\ q\ (\lambda_x\ \textsf{True})\ (\lambda_x\ \textsf{Some}\ (\lambda_e\ \textsf{Bark}\ x\ e \wedge r\ e)\ (\lambda_e\ \textsf{True} \wedge s\ e))) : \mathbf{V\text{-}aN}$$

what is the lambda calculus translation of the following syntactic analysis tree:



(You do not have to provide the entire beta-reduction, just the result.)

10.3 Non-local arguments

Composition rules for non-local arguments are much the same as rules for local arguments.

1. **Introduction rules** for simple non-local arguments are just identity functions:

   (Ea) $\lambda_{f:\gamma\{\textbf{-a},\textbf{-b}\}\delta\psi_{1..M}}\ f{:}\gamma\{\textbf{-g}, \textbf{-h}, \textbf{-v}\}\delta\psi_{1..M}$

Introduction rules for complex non-local arguments are treated as modifiers:

$$\text{(Em)}\quad \lambda_{f:\tau\varphi_{1..N}\psi_{2..M}}\,(\lambda_{k_{M..1},h_{N..1},r,s}\; f\; k_{M..2}\; h_{N..1}\; (\lambda_x\; r\; x \wedge k_1\; \begin{array}[t]{l}(\lambda_{t,u}\; t\; x \wedge u\; x)\\ (\lambda_z\; \text{True})\\ (\lambda_z\; \text{True}))\; s):\tau\varphi_{1..N}\{\textbf{-g}, \textbf{-h}, \textbf{-v}\}(\sigma\textbf{-a}\tau)\psi_{2..M}\end{array}$$

2. Composition rules for arguments and modifiers must also **propagate** non-local arguments.

   Here are the argument rules, with non-local propagation in red:

$$\text{(Aa)}\quad \lambda_{f:\beta\psi_{1..m}}\; \lambda_{g:\alpha\textbf{-a}\beta\psi_{m+1..M}}\; (\lambda_{k_{M..1}}\; g\; k_{M..m+1}\; (f\; k_{m..1})) : \alpha\psi_{1..M}$$

$$\text{(Ab)}\quad \lambda_{f:\alpha\textbf{-b}\beta\psi_{1..m}}\; \lambda_{g:\beta\psi_{m+1..M}}\; (\lambda_{k_{M..1}}\; f\; k_{m..1}\; (g\; k_{M..m+1})) : \alpha\psi_{1..M}$$

   Here are the modifier rules, with non-local propagation in red:

$$\text{(Ma)}\quad \lambda_{f:\sigma\textbf{-a}\tau\psi_{1..m}}\; \lambda_{g:\upsilon\varphi_{1..N}\psi_{m+1..M}}\; (\lambda_{k_{M..1},h_{N..1},r,s}\; g\; k_{m..1}\; h_{N..1}\; (\lambda_x\; r\; x \wedge f\; k_{M..m+1}\; \begin{array}[t]{l}(\lambda_{t,u}\; t\; x \wedge u\; x)\\ (\lambda_z\; \text{True})\\ (\lambda_z\; \text{True}))\; s):\upsilon\varphi_{1..N}\psi_{1..M}\end{array}$$

$$\text{(Mb)}\quad \lambda_{f:\upsilon\varphi_{1..N}\psi_{1..m}}\; \lambda_{g:\sigma\textbf{-a}\tau\psi_{m+1..M}}\; (\lambda_{k_{M..1},h_{N..1},r,s}\; f\; k_{M..m+1}\; h_{N..1}\; (\lambda_x\; r\; x \wedge g\; k_{m..1}\; \begin{array}[t]{l}(\lambda_{t,u}\; t\; x \wedge u\; x)\\ (\lambda_z\; \text{True})\\ (\lambda_z\; \text{True}))\; s):\upsilon\varphi_{1..N}\psi_{1..M}\end{array}$$

3. **Attachment rules** G and H for non-local arguments are the same as Aa and Ab:

$$\text{(G)}\quad \lambda_{f:\beta\psi_{1..m}}\; \lambda_{g:\alpha\textbf{-g}\beta\psi_{m+1..M}}\; \lambda_{k_{M..1}}\; (g\; k_{M..m+1}\; (f\; k_{m..1})) : \alpha\psi_{1..M}$$

$$\text{(H)}\quad \lambda_{f:\alpha\textbf{-h}\beta\psi_{1..m}}\; \lambda_{g:\beta\psi_{m+1..M}}\; \lambda_{k_{M..1}}\; (f\; k_{m..1}\; (g\; k_{M..m+1})) : \alpha\psi_{1..M}$$

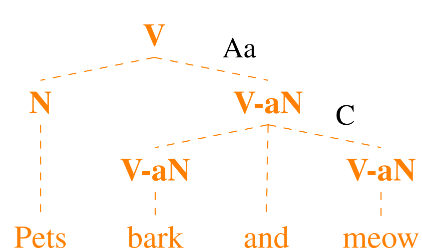
   and rules for Ra and Rb are the same as for Ma and Mb:

$$\text{(Ra)}\quad \lambda_{f:\sigma\textbf{-r}\tau\psi_{1..m}}\; \lambda_{g:\upsilon\varphi_{1..N}\psi_{m+1..M}}\; (\lambda_{k_{M..1}h_{N..1},r,s}\; g\; k_{m..1}\; h_{N..1}\; (\lambda_x\; r\; x \wedge f\; k_{M..m+1}\; \begin{array}[t]{l}(\lambda_{t,u}\; t\; x \wedge u\; x)\\ (\lambda_z\; \text{True})\\ (\lambda_z\; \text{True}))\; s):\upsilon\varphi_{1..N}\psi_{1..M}\end{array}$$

$$\text{(Rb)}\quad \lambda_{f:\upsilon\varphi_{1..N}\psi_{1..m}}\; \lambda_{g:\sigma\textbf{-r}\tau\psi_{m+1..M}}\; (\lambda_{k_{M..1}h_{N..1},r,s}\; f\; k_{M..m+1}\; h_{N..1}\; (\lambda_x\; r\; x \wedge g\; k_{m..1}\; \begin{array}[t]{l}(\lambda_{t,u}\; t\; x \wedge u\; x)\\ (\lambda_z\; \text{True})\\ (\lambda_z\; \text{True}))\; s):\upsilon\varphi_{1..N}\psi_{1..M}\end{array}$$

10.4 Conjunction schemata [Partee & Rooth, 1983]

We need a compositional semantics for conjunctions:



Here's a definition that's pretty simple – it just passes all its arguments down to its conjuncts:

(C) $\lambda_{f:\tau\varphi_{1..N}\psi_{1..M}}\ \lambda_{g:\tau\varphi_{1..N}\psi_{1..M}}\ (\lambda_{k_{M..1},h_{N..1},r,s}\ (f\ k_{M..1}\ h_{N..1}\ r\ s) \wedge (g\ k_{M..1}\ h_{N..1}\ r\ s)){:}\tau\varphi_{1..N}\psi_{1..M}$

This duplicates any quantifiers in $h_{N..1}$, which is a relatively weak reading:

$$\begin{array}{ll}\mathsf{Some} & (\lambda_x\ \mathsf{Pet}\ x) \\ & (\lambda_x\ \mathsf{Bark}\ x) \wedge \\ \mathsf{Some} & (\lambda_x\ \mathsf{Pet}\ x) \\ & (\lambda_x\ \mathsf{Meow}\ x)\end{array}$$

The duplicate reading can be avoided with quantifying attachment rules for primitive arguments:

(Aa′) $\lambda_{f:\upsilon\psi_{1..m}}\ \lambda_{g:\tau\varphi_{1..N}\text{-}\mathbf{a}\upsilon\psi_{m+1..M}}\ (\lambda_{k_{M..1},h_{N..1},r,s}\ f\ k_{m..1}\ (\lambda_x\ \mathsf{True})\ (\lambda_x\ g\ k_{M..m+1}\ (\lambda_{t,u}\ t\ x \wedge u\ x)\ h_{N..1}\ r\ s)){:}\tau\varphi_{1..N}\psi_{1..M}$

(Ab′) $\lambda_{f:\tau\varphi_{1..N}\text{-}\mathbf{b}\upsilon\psi_{1..m}}\ \lambda_{g:\upsilon\psi_{m+1..M}}\ (\lambda_{k_{M..1},h_{N..1},r,s}\ g\ k_{M..m+1}\ (\lambda_x\ \mathsf{True})\ (\lambda_x\ f\ k_{m..1}\ (\lambda_{t,u}\ t\ x \wedge u\ x)\ h_{N..1}\ r\ s)){:}\tau\varphi_{1..N}\psi_{1..M}$

(G′) $\lambda_{f:\upsilon\psi_{1..m}}\ \lambda_{g:\tau\varphi_{1..N}\text{-}\mathbf{g}\upsilon\psi_{m+1..M}}\ (\lambda_{k_{M..1},h_{N..1},r,s}\ f\ k_{m..1}\ (\lambda_x\ \mathsf{True})\ (\lambda_x\ g\ k_{M..m+1}\ (\lambda_{t,u}\ t\ x \wedge u\ x)\ h_{N..1}\ r\ s)){:}\tau\varphi_{1..N}\psi_{1..M}$

(H′) $\lambda_{f:\tau\varphi_{1..N}\text{-}\mathbf{h}\upsilon\psi_{1..m}}\ \lambda_{g:\upsilon\psi_{m+1..M}}\ (\lambda_{k_{M..1},h_{N..1},r,s}\ g\ k_{M..m+1}\ (\lambda_x\ \mathsf{True})\ (\lambda_x\ f\ k_{m..1}\ (\lambda_{t,u}\ t\ x \wedge u\ x)\ h_{N..1}\ r\ s)){:}\tau\varphi_{1..N}\psi_{1..M}$

Here's the result of that:

$$\begin{array}{ll}\mathsf{Some} & (\lambda_x\ \mathsf{Pet}\ x) \\ & (\lambda_x\ \mathsf{Bark}\ x \wedge \\ & \quad \mathsf{Meow}\ x)\end{array}$$

10.5 Argument re-organization schemata

The passive rule re-assigns the passive non-local dependency to be the subject:

(V) $\lambda_{f:\mathbf{L\text{-}aN\text{-}vN}}\ (\lambda_{q,r,s}\ f\ q\ \mathsf{Some}\ r\ s){:}\mathbf{A\text{-}aN}$

Re-ordering rules swap the first and second arguments:

(O) $\lambda_{f:\tau'\psi'\varphi'}\ (\lambda_{p,q,r,s}\ f\ q\ p\ r\ s){:}\tau\varphi\psi$

Zero-head rules attach the noun phrase child as the nuclear scope of the subject:

(Z) $\lambda_{p:\mathbf{N}}\ (\lambda_{q,r,s}\ q\ r\ (\lambda_x\ p\ (\lambda_y\ \mathsf{Equal}\ x\ y)\ s)){:}\mathbf{A\text{-}aN}$

References

[Montague, 1973] Montague, R. (1973). The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik, & P. Suppes (Eds.), *Approaches to Natural Langauge* (pp.

221–242). Dordrecht: D. Riedel. Reprinted in R. H. Thomason ed., *Formal Philosophy*, Yale University Press, 1994.

[Partee & Rooth, 1983] Partee, B. & Rooth, M. (1983). Generalized conjunction and type ambiguity. In R. Bauerle, C. Schwarze, & A. von Stechow (Eds.), *Meaning, Use and Interpretation of Language* (pp. 361–383). Berlin: Walter de Gruyter.