# Symmetric categorial grammar

## Thursday, Part One

Raffaella Bernardi & Michael Moortgat

# Contents

# 1. The plan for today

First part We go through our test suite of scope construal:

- Scope flexibility: local
- Local vs. non-local scope
- Bridge predicates vs. scope islands
- First order vs. higher-order predicates (finds vs. seeks)
- Generalized coordination.

Second part Relations between types: derivability versus similarity. We have a separate set of slides for this.

# 2. Scope ambiguity, local

Everyone teases someone Type uniformity: $(s \oslash s) \oslash np$ fits both the subject and the object role. The scope ambiguity arises from the nondeterministic choice between the subject or object ($\otimes L$) rule as the last step of the derivation.

$$\frac{\vdots}{((s \oslash s) \oslash np) \circ (tv \circ ((s \oslash s) \oslash np)) \longrightarrow s}\ (\otimes L)$$

$$\lambda c.(\llbracket\forall\rrbracket\ \lambda\langle q, y\rangle.(\llbracket\exists\rrbracket\ \lambda\langle p, z\rangle.(y\ \langle c, \lambda c'.(z\ \langle c', \lambda c''.(\llbracket\mathsf{teases}\rrbracket\ \langle p, \langle c'', q\rangle\rangle)\rangle)\rangle)\rangle)\rangle)$$
$$\lambda c.(\llbracket\exists\rrbracket\ \lambda\langle p, z\rangle.(\llbracket\forall\rrbracket\ \lambda\langle q, y\rangle.(z\ \langle c, \lambda c'.(y\ \langle c', \lambda c''.(\llbracket\mathsf{teases}\rrbracket\ \langle p, \langle c'', q\rangle\rangle)\rangle)\rangle)\rangle)\rangle)$$

Substituting the $\llbracket\cdot\rrbracket$ definitions For $q, p$ we substitute $\lambda k.(k\ x)$, for $y, z$ the `LIFT` combinator. After reduction we obtain

$$\lambda c.(\forall\ \lambda x.(\exists\ \lambda y.(c\ ((\mathsf{teases}\ y)\ x))))$$
$$\lambda c.(\exists\ \lambda y.(\forall\ \lambda x.(c\ ((\mathsf{teases}\ y)\ x))))$$

## 2.1. Independence of lexical solution

$$\mathsf{teases}^{e\to e\to t} \xrightarrow{[\![\mathsf{teases}]\!]}_{\lambda\to} \lfloor (np\backslash s)/np \rfloor$$

For the lifting of the $e \to e \to t$ constant, we have considered two solutions, depending on whether one lexically represents surface scope or inverted scope.

1. $[\![\mathsf{teases}]\!] = \lambda\langle q, \langle c, q'\rangle\rangle.(q'\ \lambda x.(q\ \lambda y.(c\ ((\mathsf{teases}\ y)\ x))))$
2. $[\![\mathsf{teases}]\!] = \lambda\langle q, \langle c, q'\rangle\rangle.(q\ \lambda y.(q'\ \lambda x.(c\ ((\mathsf{teases}\ y)\ x))))$

Under both choices the subterm below

$$\lambda c.([\![\mathsf{teases}]\!]\ \langle \lambda k.(k\ y), \langle c, \lambda k.(k\ x)\rangle\rangle)$$

reduces to

$$\lambda c.(c\ (\mathsf{teases}\ y)\ x)$$

i.e. our analysis of scope construal is fully determined by the derivational non-determinism in the choice of the active QP.

## 2.2. Enforcing surface scope construal

The class of QP phrases is not uniform in its scopal behaviour. For QP's allowing only rigid surface scope construal, we have the type assignment $(s \oslash (np \otimes s))$, instead of $(s \oslash s) \otimes np$. Both satisfy the type uniformity requirement.

| | type uniformity | flexibility |
|---|---|---|
| $s \oslash (np \otimes s)$ | ✓ | |
| $(s \oslash s) \otimes np$ | ✓ | ✓ |

Example: "Noone noticed anything"

$$s \vdash ((((s/np)\backslash s) \oplus (np \otimes (s \oslash np))) \oplus (np/(s\backslash s)))$$

$$\lambda c.(\llbracket \mathsf{noone} \rrbracket\ \lambda\langle q, y\rangle.(y\ \langle c, \lambda c'.(\llbracket \mathsf{noticed} \rrbracket\ \langle \lambda c''.(\llbracket \mathsf{anything} \rrbracket\ \lambda\langle v, q'\rangle.(v\ \langle q', c''\rangle)), \langle c', q\rangle\rangle)\rangle)))$$

## Exercise

The subterm $\lambda c''.(\llbracket \text{anything} \rrbracket\ \lambda\langle v, q'\rangle.(v\ \langle q', c''\rangle))$ is the result of the CPS transformation of the proof term corresponding to the lowering of $(s \oslash (np \oslash s))$ to $np$.

Can you provide an appropriate term for $\llbracket \text{anything} \rrbracket$ defined in terms of $\exists^{(e\to t)\to t}$?

# 3. Local vs. non-local construal

Molly thinks someone left The ambiguity arises here from the fact that the QP can non-deterministically select the embedded or the main clause as its scope domain: local versus non-local scope readings.

1. THINK $>$ SOMEONE [local]
2. SOMEONE $>$ THINK [non-local]

$$\lambda c.(\llbracket\textsf{thinks}\rrbracket\ \langle\lambda c'.(\llbracket\exists\rrbracket\ \lambda\langle q, y\rangle.(y\ \langle c', \lambda c''.(\llbracket\textsf{left}\rrbracket\ \langle c'', q\rangle)\rangle)), \langle c, \llbracket\textsf{m}\rrbracket\rangle\rangle)$$

$$\lambda c.(\llbracket\exists\rrbracket\ \lambda\langle q, y\rangle.(y\ \langle c, \lambda c'.(\llbracket\textsf{thinks}\rrbracket\ \langle\lambda c''.(\llbracket\textsf{left}\rrbracket\ \langle c'', q\rangle), \langle c', \llbracket\textsf{m}\rrbracket\rangle\rangle)\rangle))$$

Substituting the definitions for $\llbracket\cdot\rrbracket$, we would like these to reduce to:

$$\lambda c.(c\ ((\textsf{thinks}\ (\exists\ \textsf{left}))\ \textsf{m}))$$
$$\lambda c.(\exists\ \lambda y.(c\ ((\textsf{thinks}\ (\textsf{left}\ y))\ \textsf{m})))$$

## 3.1. Sentential complements

The lexical constant for the verb 'thinks' is of type $((np\backslash s)/s)' = t \to (e \to t)$. We want to lift this constant to the CBN level:

$$\mathsf{thinks}^{t\to e\to t} \xrightarrow{[\![\mathsf{thinks}]\!]}_{\lambda_\to} \lfloor (np\backslash s)/s \rfloor$$

where

$$\begin{aligned} \lfloor (np\backslash s)/s \rfloor &= R^{\lfloor s \rfloor} \times \lfloor iv \rfloor \\ &= R^{\lfloor s \rfloor} \times (\lfloor s \rfloor \times R^{\lfloor np \rfloor}) \end{aligned}$$

## 3.2. Two solutions

$$\mathsf{thinks}^{t\to e\to t} \xrightarrow{[\![\mathsf{thinks}]\!]}_{\lambda\to} \lfloor (np\backslash s)/s \rfloor$$

The solution for this type transition is not unique. Let us compare the effect of the following two possibilities on scope construal.

1. $[\![\mathsf{think}]\!] = \lambda\langle p, \langle c, q\rangle\rangle.(p\ \lambda v.(q\ \lambda x.(c\ ((\mathsf{thinks}\ v)\ x))))$
2. $[\![\mathsf{think}]\!] = \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.(c\ ((\mathsf{thinks}\ (p\ \mathtt{ID}))\ x)))$
   $= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((\mathtt{RESET}\ p)\ \lambda v.(c\ ((\mathsf{thinks}\ v)\ x))))$

where $\mathtt{RESET} = \lambda m\lambda c.(c\ (m\ \mathtt{ID}))$
$(C_t \to (K_t \to t))$.

## 3.3. The effect of RESET

$$
\begin{aligned}
[\![\mathsf{think}]\!] &= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((\mathtt{RESET}\ p)\ \lambda v.(c\ ((\mathsf{thinks}\ v)\ x)))) \\
&= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((\lambda m.\lambda c'.c'(m\ \mathtt{ID}))\ p)\ \lambda v.(c\ ((\mathsf{thinks}\ v)\ x))) \\
&= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((\lambda c'.c'\ (p\ \mathtt{ID}))\ \lambda v.(c\ ((\mathsf{thinks}\ v)\ x)))) \\
&= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((\lambda v.(c\ ((\mathsf{thinks}\ v)\ x)))\ (p\ \mathtt{ID}))) \\
&= \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.((c\ ((\mathsf{thinks}\ (p\ \mathtt{ID})\ x)))))
\end{aligned}
$$

## Scope sieves

CPS image of the proofs:

$$\lambda c.(\llbracket\mathsf{thinks}\rrbracket\ \langle\lambda c'.(\llbracket\exists\rrbracket\ \lambda\langle q, y\rangle.(y\ \langle c', \lambda c''.(\llbracket\mathsf{left}\rrbracket\ \langle c'', q\rangle)\rangle)), \langle c, \llbracket\mathsf{m}\rrbracket\rangle\rangle)$$

$$\lambda c.(\llbracket\exists\rrbracket\ \lambda\langle q, y\rangle.(y\ \langle c, \lambda c'.(\llbracket\mathsf{thinks}\rrbracket\ \langle\lambda c''.(\llbracket\mathsf{left}\rrbracket\ \langle c'', q\rangle), \langle c', \llbracket\mathsf{m}\rrbracket\rangle\rangle)\rangle))$$

Lexical options:

$$1.\quad \llbracket\mathsf{think}\rrbracket\ =\ \lambda\langle p, \langle c, q\rangle\rangle.(p\ \lambda v.(q\ \lambda x.(c\ ((\mathsf{thinks}\ v)\ x))))$$

$$2.\quad =\ \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.(c\ ((\mathsf{thinks}\ (p\ \mathtt{ID}))\ x)))$$

Comparison Comparing the interaction with embedded QP for "Molly thinks someone left" we observe

- the sequent has two proofs: local, non-local construal
- solution 2. associates them with the required readings
- solution 1. transforms the local reading to the non-local one: it turns the predicate into a scope sieve

# 4. Blocking non-local scope construal

Some predicates force the QP to have only the local scope reading:

- Molly thinks someone left
  1. THINK $>$ SOMEONE [Local]
  2. SOMEONE $>$ THINK [Non-local]
- Molly shouts someone left
  1. SHOUT $>$ SOMEONE [Local]
  2. *SOMEONE $>$ SHOUT

How can we capture this difference?

Friday we will present on going work on this.

# 5. Like vs. Need: the problem

- Everyone likes someone
  1. SOMEONE $>$ EVERYONE
  2. EVERYONE $>$ SOMEONE
- Everyone needs someone.
  1. SOMEONE $>$ EVERYONE $>$ NEED
  2. EVERYONE $>$ SOMEONE $>$ NEED
  3. EVERYONE $>$ NEED $>$ SOMEONE

"Like": $(np\backslash s)/np$ vs. "Need": $(np\backslash s)/(s/(np\backslash s))$

## 5.1. Likes vs. Need: CPS solution

a. $\lambda c'.(\llbracket\text{needs}\rrbracket\ \langle\lambda\langle p, v\rangle.(p\ \langle v, q\rangle), \langle c', q'\rangle\rangle)$ $(= M : C_s)$

1. $\lambda c.(\llbracket\text{somebody}\rrbracket\ \lambda\langle q, y\rangle.(\llbracket\text{everyone}\rrbracket\ \lambda\langle q', y'\rangle.(y\ \langle c, \lambda c'.(y'\ \langle c', M_s\rangle)))))$
2. $\lambda c.(\llbracket\text{everyone}\rrbracket\ \lambda\langle q', y'\rangle.(\llbracket\text{somebody}\rrbracket\ \lambda\langle q, y\rangle.(y'\ \langle c, \lambda c'.(y\ \langle c', M_s\rangle))))$

b. $\lambda c'.(\llbracket\text{needs}\rrbracket\ \langle\lambda\langle p, v\rangle.(\llbracket\text{somebody}\rrbracket\ \lambda\langle q, y\rangle.(y\ \langle v, \lambda v'.(p\ \langle v', q\rangle)))), \langle c', q'\rangle\rangle)$
$(= N : C_s)$

3. $\lambda c.(\llbracket\text{everyone}\rrbracket\ \lambda\langle q', y'\rangle.(y'\ \langle c, N_s\rangle))$

With lexical substitution, we want these to reduce to:

1. $\lambda c.(\exists\ \lambda y.(\forall\ \lambda x.(c\ ((\text{needs}\ \lambda k.(k\ y))\ x))))$
2. $\lambda c.(\forall\ \lambda x.(\exists\ \lambda y.(c\ ((\text{needs}\ \lambda k.(k\ y))\ x))))$
3. $\lambda c.(\forall\ \lambda x.(c\ ((\text{needs}\ \exists)\ x)))$

## 5.2. Likes vs. Need: Lifted Term solution

Given by Arno Bastenhof, BSc Thesis, July 2007, Utrecht University.

The lexical constant for 'needs' is of type $(np\backslash s)/(s/(np\backslash s)) = ((e \to t) \to t) \to (e \to t)$.

We want to lift this constant to the CBN level:

$$\mathsf{needs}^{((e\to t)\to t)\to(e\to t)} \xrightarrow{[\![\mathsf{needs}]\!]}_{\lambda_{\to}} \lfloor (np\backslash s)/(s/(np\backslash s)) \rfloor$$

where

$$\begin{aligned} \lfloor (np\backslash s)/(s/(np\backslash s)) \rfloor \;=\; & R^{R^{\lfloor s/(np\backslash s)\rfloor}\times(\lfloor s\rfloor\times R^{\lfloor np\rfloor})} \\ & R^{\lfloor s/(np\backslash s)\rfloor} = R^{R^{\lfloor iv\rfloor}\times\lfloor s\rfloor} \\ & R^{\lfloor iv\rfloor} = R^{\lfloor s\rfloor\times R^{\lfloor np\rfloor}} \end{aligned}$$

$[\![\mathsf{need}]\!] = \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.(c\ (\mathsf{need}\ \lambda k.(p\ \langle\lambda\langle c', q'\rangle.q'\ (\lambda y.c'\ (k\ y)), \lambda y.y\rangle)\ x)))$

## 5.3. Generalized Coordination: the problem

John sought and found someone

1. SOMEONE $>$ SOUGHT & FOUND
2. SOUGHT & FOUND $>$ SOMEONE
3. $^*$SOUGHT $>$ SOMEONE $>$ FOUND
4. $^*$FOUND $>$ SOMEONE $>$ SOUGHT

## 5.4. Remark

First, note that

$$(a)\ np \vdash s/(np\backslash s) \quad \text{and} \quad (b)\ (s \oslash s) \oslash np \vdash s/(np\backslash s).$$

$$\dfrac{\dfrac{\dfrac{\vdots}{np \circ np\backslash s \longrightarrow s \oslash s \circ s}}{(s \oslash s) \oslash np \circ np\backslash s \longrightarrow s}}{(s \oslash s) \oslash np \circ np\backslash s \longrightarrow s}}{(s \oslash s) \oslash np \longrightarrow s/(np\backslash s)}$$

Recall, $A/B \vdash A/C$, if $C \vdash B$.

$$(a)\ \underbrace{iv/((s/(np\backslash s)))}_{TV_{seek}} \vdash \underbrace{iv/np}_{TV_{find}} \quad \text{and} \quad (b)\ \underbrace{iv/((s/(np\backslash s)))}_{TV_{seek}} \vdash \underbrace{iv/((s \oslash s) \oslash np)}_{TV}$$

Furthermore, note that:

$$(c)\ \underbrace{(np\backslash s)/np}_{TV_{find}} \vdash \underbrace{(np\backslash s)/(((s \oslash s) \oslash np))}_{TV}$$

$$\vdots$$

$$\dfrac{\dfrac{\dfrac{\dfrac{np \circ ((np\backslash s)/np \circ np) \longrightarrow s \oslash s \circ s}{np \circ ((np\backslash s)/np \circ ((s \oslash s) \oslash np)) \longrightarrow s}}{np \circ ((np\backslash s)/np \circ (((s \oslash s) \oslash np))) \longrightarrow s}}{(np\backslash s)/np \circ ((s \oslash s) \oslash np) \longrightarrow np\backslash s}}{(np\backslash s)/np \longrightarrow (np\backslash s)/(((s \oslash s) \oslash np))}$$

Hence, both $TV_{find}$ and $TV_{seek}$ derive $TV$.

Summary: $(a)\ TV_{seek} \vdash TV_{find}$, $(b)\ TV_{seek} \vdash TV$, and $(c)\ TV_{find} \vdash TV$.

Let, $(X \backslash X)/X$ be the polymorphic type assigned to conjunction. It can become either $(TV_{find} \backslash TV_{find})/TV_{find}$ (abb. $CONJ_{find}$) or, $(TV \backslash TV)/TV$ (abb. $CONJ$)

$$TV_{seek} \circ (CONJ \circ TV_{find}) \vdash TV \qquad TV_{seek} \circ (CONJ \circ TV_{find}) \vdash TV_{find}$$

Recall,

$$\frac{\Gamma[C] \longrightarrow A}{\Gamma[B] \longrightarrow A}$$

$$\frac{TV \circ ((TV \backslash TV)/TV \circ TV) \longrightarrow TV}{TV_{seek} \circ ((TV \backslash TV)/TV \circ TV_{find}) \longrightarrow TV} \quad \text{since } (b) \text{ and } (c)$$

$$\frac{TV_{find} \circ (TV_{find} \backslash TV_{find})/TV_{find} \circ TV_{find}) \longrightarrow TV_{find}}{TV_{seek} \circ (TV_{find} \backslash TV_{find})/TV_{find} \circ TV_{find}) \longrightarrow TV_{find}} \quad \text{since } (a)$$

Recall, "John sought and found someone" has two readings:

(a) SOMEONE > SOUGHT > FOUND (b) SOUGHT & FOUND > SOMEONE

$TV_{find} = iv/np$ vs. $TV = iv/((s \oslash s) \otimes np)$

$$\underbrace{NP}_{john} \circ (\underbrace{(\underbrace{TV_{seek}}_{sought} \circ (\underbrace{CONJ}_{and} \circ \underbrace{TV_{find}}_{found}))}_{TV_{find}} \circ \underbrace{QP}_{someone}) \vdash s$$

corresponds to (a).

$$\underbrace{NP}_{john} \circ (\underbrace{(\underbrace{TV_{seek}}_{sought} \circ (\underbrace{CONJ}_{and} \circ \underbrace{TV_{find}}_{found}))}_{TV} \circ \underbrace{QP}_{someone}) \vdash s$$

corresponds to (b).

# 6. Comparison: type shifting principles

We compare our approach with Hendriks' (1993).

Flexible Montague Grammar

- Syntax: slash elimination only (function application)
- The mapping from syntactic to semantic types is weakened to a relation
- To resolve type mismatches for application, a set of type-shifting principles is postulated: Value Raising (VR), Argument Lowering (AL), Argument Raising (AR).

LG

- Syntax: rules of use + rules of proof for all connectives.
- The mappings $\lfloor\cdot\rfloor, \lceil\cdot\rceil$ from syntactic types to their CPS interpretations are functional.
- The type-shifting principles are derived rules.

# Deriving VR, AL, AR in LG

Value Raising, Argument Lowering These principles are valid already in **NL**, the pure residuation logic. By Monotonicity, if $A \to B/(A\backslash B)$, then

$$A/C \to (B/(A\backslash B))/C \qquad (B/(A\backslash B))\backslash C \to A\backslash C$$

An example: $(np\backslash s)/(s/(np\backslash s)) \to (np\backslash s)/np$ for a de re reading of 'seek'.

Argument Raising AR is invalid in **NL**. But the version for the quantifier type $(s \oslash s) \oslash np$ is derivable in **LG**. For example:

$$(np\backslash s)/np \to (np\backslash s)/((s \oslash s) \oslash np)$$