Constraining scope ambiguity in LFG+Glue

Matthew Gotham
University of Oxford

24th International LFG Conference, Australian National University
8–10 July 2019

Outline

Outline

Outline

Outline

Scope (non-)ambiguity in LFG+Glue

Scope ambiguity in English

(1) A police officer guards every exit.

$\Rightarrow\ \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy$ (surface scope)

$\Rightarrow\ \forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy$ (inverse scope)

Scope ambiguity in English

(1) A police officer guards every exit.

$$\Rightarrow \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy \qquad \text{(surface scope)}$$

$$\Rightarrow \forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy \qquad \text{(inverse scope)}$$

$$F : \begin{bmatrix} \text{PRED} & \text{`guard'} \\ \text{SUBJ} & G : \begin{bmatrix} \text{PRED} & \text{`police officer'} \\ \text{SPEC} & I : \begin{bmatrix} \text{PRED} & \text{`a'} \end{bmatrix} \end{bmatrix} \\ \text{OBJ} & H : \begin{bmatrix} \text{PRED} & \text{`exit'} \\ \text{SPEC} & J : \begin{bmatrix} \text{PRED} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\text{a} \rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx$$
$$: ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%A) \multimap \%A)$$
$$\%A = (\textsc{gf}^{*}\uparrow)$$
$$\text{police officer} \rightsquigarrow \text{officer}' : (\textsc{spec}\uparrow) \multimap \uparrow$$
$$\text{guards} \rightsquigarrow \text{guard}' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow)$$
$$\text{every} \rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy$$
$$: ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%B) \multimap \%B)$$
$$\%B = (\textsc{gf}^{*}\uparrow)$$
$$\text{exit} \rightsquigarrow \text{exit}' : (\textsc{spec}\uparrow) \multimap \uparrow$$

$$
\begin{aligned}
\text{a} &\rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx \\
&\quad : (G \multimap I) \multimap ((G \multimap F) \multimap F) \\
&\quad \%A := F \\
\text{police officer} &\rightsquigarrow \text{officer}' : G \multimap I \\
\text{guards} &\rightsquigarrow \text{guard}' : G \multimap (H \multimap F) \\
\text{every} &\rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy \\
&\quad : (H \multimap J) \multimap ((H \multimap F) \multimap F) \\
&\quad \%B := F \\
\text{exit} &\rightsquigarrow \text{exit}' : H \multimap J
\end{aligned}
$$

$$
\dfrac{
\dfrac{
\dfrac{[G]^1 \quad \text{guard}' : G \multimap (H \multimap F)}{H \multimap F}
\qquad
\dfrac{\text{every}' : (H \multimap J) \multimap ((H \multimap F) \multimap F) \quad \text{exit}' : H \multimap J}{(H \multimap F) \multimap F}
}{F}
}{G \multimap F}\,1
\qquad
\dfrac{\text{a}' : (G \multimap I) \multimap ((G \multimap F) \multimap F) \quad \text{officer}' : G \multimap I}{(G \multimap F) \multimap F}
$$

$$
\text{a}'\text{officer}'(\lambda x.\text{every}'\text{exit}'(\text{guard}'x)) : F
$$

$$
\equiv \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy : F
$$

$$
\dfrac{
\dfrac{
\dfrac{
[H]^2 \quad \dfrac{[G]^1 \quad \text{guard}' : G \multimap (H \multimap F)}{H \multimap F}
}{F}
}{G \multimap F}\,1
\qquad
\dfrac{\text{a}' : (G \multimap I) \multimap ((G \multimap F) \multimap F) \quad \text{officer}' : G \multimap I}{(G \multimap F) \multimap F}
}{
\dfrac{F}{H \multimap F}\,2
}
\qquad
\dfrac{\text{every}' : (H \multimap J) \multimap ((H \multimap F) \multimap F) \quad \text{exit}' : H \multimap J}{(H \multimap F) \multimap F}
$$

$$
\dfrac{H \multimap F \qquad (H \multimap F) \multimap F}{\begin{array}{c}\text{every}'\text{exit}'(\lambda y.\text{a}'\text{officer}'(\lambda x.\text{guard}' x y)) : F \\ \equiv \forall y.\text{exit}' y \rightarrow \exists x.\text{officer}' x \wedge \text{guard}' x y : F\end{array}}
$$

Scope rigidity in other languages

(2) *Ein Polizist bewacht jeden Ausgang.*
A police officer guards every exit
(German)

(3) *Yi-ming jingcha kanshou meige chukou.*
One-CL police officer guards every exit
(Chinese)

$\Rightarrow \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy$

$\not\Rightarrow \forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy$ (surface scope only)

Scope rigidity in English

(4) Hilary gave a student every grade.

$\Rightarrow \exists y.\text{student}'y \wedge \forall x.\text{grade}'x \rightarrow \text{give}'\text{hilary}'xy$

$\not\Rightarrow \forall x.\text{grade}'x \rightarrow \exists y.\text{student}'y \wedge \text{give}'\text{hilary}'xy$

(surface scope only within the double object)

Not scope ‘islands’

(5) If every student passes, the lecturer will be happy.

$\Rightarrow$ $(\forall y.\text{student}'y \rightarrow \text{pass}'y) \rightarrow \text{happy}'(\iota x.\text{lecturer}'x)$

$\not\Rightarrow$ $\forall y.\text{student}'y \rightarrow (\text{pass}'y \rightarrow \text{happy}'(\iota x.\text{lecturer}'x))$

Constraining the path

(5) If every student passes, the lecturer will be happy.

$$F : \begin{bmatrix} \textsc{pred} & \text{'happy'} \\ \textsc{subj} & [\text{"the lecturer"}] \\ \textsc{adj} & \left\{ G : \begin{bmatrix} \textsc{pred} & \text{'pass'} \\ \textsc{compform} & \text{'if'} \\ \textsc{subj} & H : \begin{bmatrix} \textsc{pred} & \text{'student'} \\ \textsc{spec} & I : \begin{bmatrix} \textsc{pred} & \text{'every'} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\} \end{bmatrix}$$

$$\text{every} \rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy$$
$$: ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%B) \multimap \%B)$$
$$\%B = (\textsc{path}\uparrow)$$

(5) If every student passes, the lecturer will be happy.

$$F : \begin{bmatrix} \textsc{pred} & \text{`happy'} \\ \textsc{subj} & [\text{``the lecturer''}] \\ \textsc{adj} & \left\{ G : \begin{bmatrix} \textsc{pred} & \text{`pass'} \\ \textsc{compform} & \text{`if'} \\ \textsc{subj} & H : \begin{bmatrix} \textsc{pred} & \text{`student'} \\ \textsc{spec} & I : \begin{bmatrix} \textsc{pred} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\} \end{bmatrix}$$

$$\begin{aligned} \text{every} \rightsquigarrow\ & \lambda P.\lambda Q.\forall y.Py \rightarrow Qy \\ & : (H \multimap I) \multimap ((H \multimap \%B) \multimap \%B) \\ & \%B = (\textsc{path}\ \uparrow) \end{aligned}$$

(where PATH is such that %B can be *G* but not *F*)

Not an available strategy here

(2) Ein Polizist bewacht jeden Ausgang.

$$F : \begin{bmatrix} \text{PRED} & \text{`guard'} \\ \text{TOPIC} & G : [\text{``Ein Polizist''}] \\ \text{SUBJ} & \\ \text{OBJ} & H : \begin{bmatrix} \text{PRED} & \text{`exit'} \\ \text{SPEC} & J : \begin{bmatrix} \text{PRED} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\text{jeden} \rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy :$$

$$: ((\text{SPEC} \uparrow) \multimap \uparrow) \multimap (((\text{SPEC} \uparrow) \multimap \%B) \multimap \%B)$$

$$\%B = (\text{PATH} \uparrow)$$

Not an available strategy here

(2) Ein Polizist bewacht jeden Ausgang.

$$F : \begin{bmatrix} \textsc{pred} & \text{`guard'} \\ \textsc{topic} & G : [\text{``Ein Polizist''}] \\ \textsc{subj} & \\ \textsc{obj} & H : \begin{bmatrix} \textsc{pred} & \text{`exit'} \\ \textsc{spec} & J : \begin{bmatrix} \textsc{pred} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\text{jeden} \rightsquigarrow \lambda P.\lambda Q.\forall y.Py \to Qy :$$
$$: (H \multimap I) \multimap ((H \multimap \%B) \multimap \%B)$$
$$\%B = (\textsc{path} \uparrow)$$

We have $\%B := F$ for **both** the surface scope **and** the inverse scope interpretation.

A previous proposal

Node orderings

Crouch & van Genabith (1999) propose to analzye scope rigidity like this:

bewacht V

$$\text{guard}' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow)$$

$$(\uparrow \textsc{subj}) = (\uparrow \textsc{topic}) \Rightarrow (\uparrow \textsc{subj}) \succ (\uparrow \textsc{obj})$$

Node orderings

Crouch & van Genabith (1999) propose to analzye scope rigidity like this:

bewacht V

$$\text{guard}' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow)$$

$$(\uparrow \textsc{subj}) = (\uparrow \textsc{topic}) \Rightarrow (\uparrow \textsc{subj}) \succ (\uparrow \textsc{obj})$$

- The last line is a **node ordering**: a constraint on linear logic proofs.

Node orderings

Crouch & van Genabith (1999) propose to analzye scope rigidity like this:

bewacht V

$$\text{guard}' : (\uparrow \text{SUBJ}) \multimap ((\uparrow \text{OBJ}) \multimap \uparrow)$$

$$(\uparrow \text{SUBJ}) = (\uparrow \text{TOPIC}) \Rightarrow (\uparrow \text{SUBJ}) \succ (\uparrow \text{OBJ})$$

- The last line is a **node ordering**: a constraint on linear logic proofs.
- Roughly, $\alpha \succ \beta$ means that in every licit linear logic proof, no instance of β occurs strictly lower down than every instance of α.

Node orderings in action

(2) Ein Polizist bewacht jeden Ausgang.

$$F : \begin{bmatrix} \textsc{pred} & \text{`guard'} \\ \textsc{topic} & G : [\text{``Ein Polizist''}] \\ \textsc{subj} & \\ \textsc{obj} & H : [\text{``jeden Ausgang''}] \end{bmatrix}$$

bewacht V

$$\text{guard}' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow)$$

$$(\uparrow \textsc{subj}) = (\uparrow \textsc{topic}) \Rightarrow (\uparrow \textsc{subj}) \succ (\uparrow \textsc{obj})$$

Node orderings in action

(2) Ein Polizist bewacht jeden Ausgang.

$$F : \begin{bmatrix} \textsc{pred} & \text{‘guard’} \\ \textsc{topic} & G : [\text{“Ein Polizist”}] \\ \textsc{subj} & \\ \textsc{obj} & H : [\text{“jeden Ausgang”}] \end{bmatrix}$$

bewacht V

$$\text{guard}' : G \multimap (H \multimap F)$$

$$G = G \Rightarrow G \succ H$$

$G \succ H$

$$\dfrac{\dfrac{\dfrac{\dfrac{[G]^1 \quad G \multimap (H \multimap F)}{H \multimap F} \quad \overset{\textit{jeden Ausgang}}{\overset{\Downarrow}{(H \multimap F) \multimap F}}}{F}}{G \multimap F}\,1 \quad \overset{\textit{ein Polizit}}{\overset{\Downarrow}{(G \multimap F) \multimap F}}}{F}$$

Surface scope

✓

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[H]^2 \quad \dfrac{[G]^1 \quad G \multimap (H \multimap F)}{H \multimap F}}{F}}{G \multimap F}\,1 \quad \overset{\textit{ein Polizist}}{\overset{\Downarrow}{(G \multimap F) \multimap F}}}{F}}{{\color{red}H} \multimap F}\,2 \quad \overset{\textit{jeden Ausgang}}{\overset{\Downarrow}{(H \multimap F) \multimap F}}}{F}$$

Inverse scope

×

What is a proof?

Node orderings are defined over derivations

> *A derivation is a tree-like structure of sequents [...] Represent derivations $\mathcal{D}$ as triples $\langle S, >_S, \$\rangle$ where S is the set of points in the tree, $>_S$ is a transitive , asymmetric ordering over them, and \$ is a function mapping the points onto their corresponding sequents.*
>
> (Crouch & van Genabith 1999: 131)

What is a proof?

Node orderings are defined over derivations

> *A derivation is a tree-like structure of sequents [...] Represent derivations $\mathcal{D}$ as triples $\langle S, >_S, \$\rangle$ where S is the set of points in the tree, $>_S$ is a transitive , asymmetric ordering over them, and \$ is a function mapping the points onto their corresponding sequents.*

(Crouch & van Genabith 1999: 131)

But (natural deduction) derivations are representations of proofs, not the proofs themselves.

> *Gentzen calculus, labelled and unlabelled natural deductions, proof nets, categorical calculus, etc. are all of repute, all have their respective advantages and disadvantages, and are* ***all notations for the same theory****.*

(Corbalán & Morrill 2016: fn. 4), emphasis mine

Sequent calculus

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{G \vdash G} \quad \dfrac{}{H \multimap F \vdash H \multimap F}}{G, G \multimap (H \multimap F) \vdash H \multimap F}\,{\multimap}_L \quad \dfrac{}{F \vdash F}}{G, G \multimap (H \multimap F), (H \multimap F) \multimap F \vdash F}\,{\multimap}_L}{G \multimap (H \multimap F), (H \multimap F) \multimap F \vdash G \multimap F}\,{\multimap}_R \quad \dfrac{}{F \vdash F}}{(G \multimap F) \multimap F, G \multimap (H \multimap F), (H \multimap F) \multimap F \vdash F}\,{\multimap}_L
$$

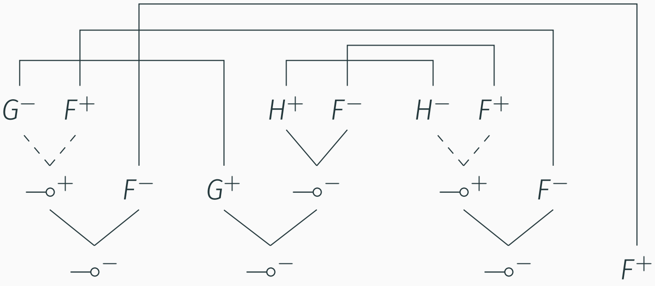
Surface scope

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{G \vdash G} \quad \dfrac{\dfrac{}{H \vdash H} \quad \dfrac{}{F \vdash F}}{H, H \multimap F \vdash F}\,{\multimap}_L}{G, H, G \multimap (H \multimap F) \vdash F}\,{\multimap}_L}{H, G \multimap (H \multimap F) \vdash G \multimap F}\,{\multimap}_R \quad \dfrac{}{F \vdash F}}{H, (G \multimap F) \multimap F, G \multimap (H \multimap F) \vdash F}\,{\multimap}_L}{(G \multimap F) \multimap F, G \multimap (H \multimap F) \vdash H \multimap F}\,{\multimap}_R \quad \dfrac{}{F \vdash F}}{(G \multimap F) \multimap F, G \multimap (H \multimap F), (H \multimap F) \multimap F \vdash F}\,{\multimap}_L
$$

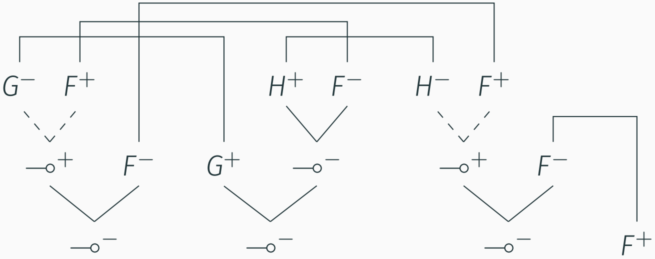
Inverse scope

Proof nets I

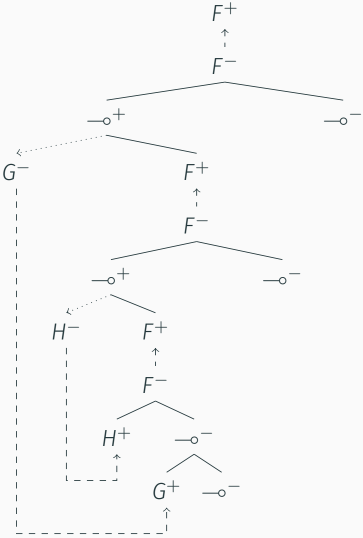
(Moot 2002: Chapter 5)



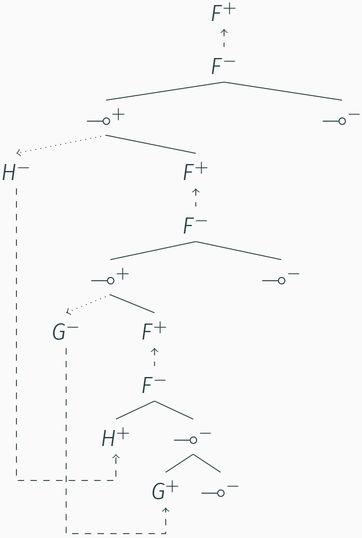
Surface scope



Inverse scope



Surface scope



Inverse scope

- The point is not that an equivalent notion of node ordering couldn’t be defined for these other proof formats.

  (In face, I’ve actually done this in adapting the definition that Crouch & van Genabith (1999) give for a slightly different proof format.)

- The point is not that an equivalent notion of node ordering couldn't be defined for these other proof formats.

  (In face, I've actually done this in adapting the definition that Crouch & van Genabith (1999) give for a slightly different proof format.)

- The point is that if we have properly linguistic constraint on the form of derivations, we're not doing logic any more.

- The point is not that an equivalent notion of node ordering couldn't be defined for these other proof formats.

  (In face, I've actually done this in adapting the definition that Crouch & van Genabith (1999) give for a slightly different proof format.)

- The point is that if we have properly linguistic constraint on the form of derivations, we're not doing logic any more.

> *Rather than make such nonlogical restrictions on our proof theory, I turn to an alternative approach*
>
> (Carpenter 1998: 203)

My proposal

The name of the game

- Assign linear logic formula to lexical items such that all and only the desired interpretations **have** a corresponding proof.

The name of the game

- Assign linear logic formula to lexical items such that all and only the desired interpretations **have** a corresponding proof.
- I.e., **not** filtering out proofs by non-logical means.

In a bit more detail

Expand the fragment of linear logic used such that

In a bit more detail

Expand the fragment of linear logic used such that

- f-structure nodes are linear logic predicates (not formulae),

In a bit more detail

Expand the fragment of linear logic used such that

- f-structure nodes are linear logic predicates (not formulae),
- the arguments to those predicates 'keep track' of the order of application of quantifiers,

In a bit more detail

Expand the fragment of linear logic used such that

- f-structure nodes are linear logic predicates (not formulae),
- the arguments to those predicates ‘keep track’ of the order of application of quantifiers, and
- set things up so that only by applying quantifiers in the desired order can a valid proof be constructed.

Provenance

The approach is inspired by work in Abstract Categorial Grammar (Pogodalla & Pompigne 2012, Kanazawa 2015).

Provenance

The approach is inspired by work in Abstract Categorial Grammar (Pogodalla & Pompigne 2012, Kanazawa 2015).

> *A crude characterisation would be that glue semantics is like categorial grammar and its semantics, but without the categorial grammar.*
>
> (Crouch & van Genabith 2000: 91)

Linear logic fragment

Given a set P of predicates (f-structure nodes) and a set V of variables, the fragment of linear logic used is:

Linear logic fragment

Given a set P of predicates (f-structure nodes) and a set V of variables, the fragment of linear logic used is:

$$
\begin{array}{rcll}
n & ::= & \mathsf{V} \mid 0 \mid \mathtt{s}\, n & \text{(terms)} \\
\phi, \psi & ::= & \mathsf{P}\, n \mid \phi \multimap \psi \mid \forall \mathsf{V}.\phi & \text{(formulae)}
\end{array}
$$

Linear logic fragment

Given a set P of predicates (f-structure nodes) and a set V of variables, the fragment of linear logic used is:

$$\begin{array}{rcll} n & ::= & \mathsf{V} \mid 0 \mid \mathtt{s}\, n & \text{(terms)} \\ \phi, \psi & ::= & \mathsf{P}\, n \mid \phi \multimap \psi \mid \forall \mathsf{V}.\phi & \text{(formulae)} \end{array}$$

(where $\mathtt{s}$ is the successor function)

Our German example

$$\text{bewacht} \rightsquigarrow \text{guard}' : \forall i.\forall j.(\uparrow \textsc{subj})\, i \multimap ((\uparrow \textsc{obj})\, j \multimap \uparrow j)$$

$$\text{det} \rightsquigarrow \text{det}' : \forall i.[(\textsc{spec} \uparrow)\, 0 \multimap \uparrow 0] \multimap$$
$$([(\textsc{spec} \uparrow)(\mathtt{s}\, i) \multimap \%A\, (\mathtt{s}\, i)] \multimap \%A\, i)$$

$$\%A = (\textsc{gf}^* \uparrow)$$

Our German example

$$\text{bewacht} \rightsquigarrow \text{guard}' : \forall i.\forall j.(\uparrow \textsc{subj})\, i \multimap ((\uparrow \textsc{obj})\, j \multimap \uparrow j)$$

$$\text{det} \rightsquigarrow \text{det}' : \forall i.[(\textsc{spec}\uparrow)\, 0 \multimap \uparrow 0] \multimap$$
$$([(\textsc{spec}\uparrow)(\mathbb{s}\, i) \multimap \%A\, (\mathbb{s}\, i)] \multimap \%A\, i)$$
$$\%A = (\textsc{gf}^* \uparrow)$$

$$\Downarrow$$

$$\text{bewacht} \rightsquigarrow \text{guard}' : \forall i.\forall j.Gi \multimap (Hj \multimap Fj)$$
$$\text{ein Polizist} \rightsquigarrow \lambda P.\exists x.\text{officer}'x \wedge Px : \forall i.(G(\mathbb{s}i) \multimap F(\mathbb{s}i)) \multimap Fi$$
$$\text{jeden Ausgang} \rightsquigarrow \lambda Q.\forall y.\text{exit}'y \rightarrow Qy : \forall i.(H(\mathbb{s}i) \multimap F(\mathbb{s}i)) \multimap Fi$$
$$\%A := F$$

How it works

- There's a 'counter'.

How it works

- There's a 'counter'.
- Applying a quantifier reduces the counter by one: $[(\textsc{spec}\uparrow)(\mathtt{s}\,i) \multimap \%A\,(\mathtt{s}\,i)] \multimap \%A\,i$

How it works

- There's a 'counter'.
- Applying a quantifier reduces the counter by one: $[(\textsc{spec}\uparrow)(\mathtt{s}\ i) \multimap \%A\,(\mathtt{s}\ i)] \multimap \%A\ i$
- So if Q_1 immediately outscopes Q_2, then you have to set the counter for Q_1 to one lower than for Q_2.

How it works

- There's a 'counter'.
- Applying a quantifier reduces the counter by one: $[(\text{SPEC}\uparrow)(\mathtt{s}\ i) \multimap \%A\ (\mathtt{s}\ i)] \multimap \%A\ i$
- So if Q_1 immediately outscopes Q_2, then you have to set the counter for Q_1 to one lower than for Q_2.
- So to get the inverse scope reading, you'd have to set the counter for the subject position one higher than for the object position.

How it works

- There's a 'counter'.
- Applying a quantifier reduces the counter by one: $[(\text{SPEC}\uparrow)(\mathtt{s}\,i) \multimap \%A\,(\mathtt{s}\,i)] \multimap \%A\,i$
- So if Q_1 immediately outscopes Q_2, then you have to set the counter for Q_1 to one lower than for Q_2.
- So to get the inverse scope reading, you'd have to set the counter for the subject position one higher than for the object position.
- But the lexical entry for the verb guarantees that if you do that, no proof can be constructed: $(\uparrow\text{SUBJ})\,i \multimap ((\uparrow\text{OBJ})\,j \multimap \uparrow j)$

The inverse scope reading is underivable

$$\dfrac{\dfrac{\text{ein Polizist}}{(G2 \multimap F2) \multimap F1}\Downarrow \qquad \dfrac{\dfrac{[H1]^2 \qquad \dfrac{[G2]^1 \qquad \dfrac{\text{guard}' : \forall i.\forall j.Gi \multimap (Hj \multimap Fj)}{G2 \multimap (H1 \multimap F1)}\ \forall_E \times 2}{H1 \multimap F1}}{\dfrac{F1}{G2 \multimap F1}\ 1}}{\ }\ *$$

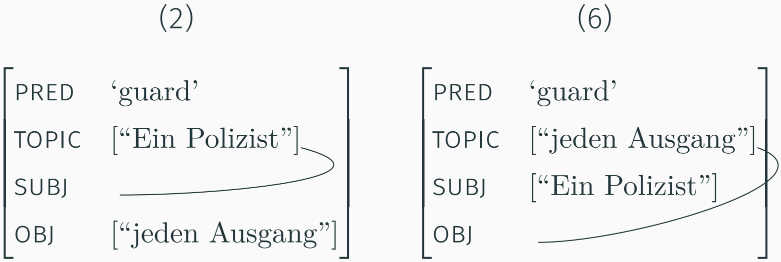
back

The surface scope reading is derivable

$$
\cfrac{
\begin{array}{c}\textit{Ein Polizist}\\ \Downarrow \\ (G1 \multimap F1) \multimap F0\end{array}
\qquad
\cfrac{\cfrac{\begin{array}{c}\textit{jeden Ausgang}\\ \Downarrow\\ (H2 \multimap F2)\multimap F1\end{array} \qquad \cfrac{[G1]^1 \qquad \cfrac{\begin{array}{c}\text{guard}':\\ \forall i.\forall j.Gi \multimap (Hj \multimap Fj)\end{array}}{G1 \multimap (H2 \multimap F2)}\ \forall_E \times 2}{H2 \multimap F2}}{F1}}{G1 \multimap F1}\ 1
}{F0}
$$

The relevance of topicalization

(2) Ein Polizist bewacht jeden Ausgang.
(6) Jeden Ausgang bewacht ein Polizist.



(6), unlike (2), has both the surface scope and inverse scope readings.

Conditional meaning constructors

It seems that we want something like this:

bewacht V

$(\uparrow \textsc{pred}) = \text{`guard'}$

$(\uparrow \textsc{subj}) = (\uparrow \textsc{topic}) \Rightarrow \text{guard}' :$

$\forall i.\forall j.(\uparrow \textsc{subj})\, i \multimap ((\uparrow \textsc{obj})\, j \multimap \uparrow (\text{£}\, i\, j))$

$(\uparrow \textsc{subj}) \neq (\uparrow \textsc{topic}) \Rightarrow \text{guard}' :$

$\forall i.\forall j.\forall k.(\uparrow \textsc{subj})\, i \multimap ((\uparrow \textsc{obj})\, j \multimap \uparrow k)$

But this is an abuse of notation, since meaning constructors aren't defining equations.

bewacht V

$(\uparrow \text{PRED}) = \text{`guard'}$

$\text{guard}' : \forall i.\forall j.(\uparrow \text{SUBJ})\, i \multimap ((\uparrow \text{OBJ})\, j \multimap \uparrow (\text{f}\, i\, j))$

$(@\text{RESET})$

where

$\text{RESET} := (\uparrow \text{SUBJ}) \neq (\uparrow \text{TOPIC})$

$\lambda p.p : \forall i.\forall j.\uparrow i \multimap \uparrow j$

- If the subject is the topic, calling RESET will cause failure. So, the scope is frozen.

- If the subject is the topic, calling RESET will cause failure. So, the scope is frozen.
- If the subject is not the topic, then RESET may or may not be called. If it is, then both scope ordering are possible since the counter can be changed.

Deriving the inverse scope reading with RESET

Remember this derivation?

Remember this derivation? With reset it can be completed.

$$
\dfrac{\begin{array}{c}\textit{jeden Ausgang}\\ \Downarrow\\ (H1 \multimap F1) \multimap F0\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\textit{ein Polizist}\\ \Downarrow\\ (G2 \multimap F2) \multimap F1\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}[H1]^2, [G2]^1, \mathrm{guard}'\\ \vdots\\ F1\end{array} \qquad \dfrac{\forall i.\forall j.Fi \multimap Fj}{F1 \multimap F2}\;\forall_E \times 2}{F2}}{G2 \multimap F2}\;1}{F1}}{H1 \multimap F1}\;2}{F0}
$$

The English double object construction

(7) Most teachers gave a student every grade.

most $\rangle\rangle$ a $\rangle\rangle$ every	a $\rangle\rangle$ most $\rangle\rangle$ every	~~every $\rangle\rangle$ most $\rangle\rangle$ a~~
~~most $\rangle\rangle$ every $\rangle\rangle$ a~~	a $\rangle\rangle$ every $\rangle\rangle$ most	~~every $\rangle\rangle$ a $\rangle\rangle$ most~~

(Bruening 2001)

The English double object construction

(7) Most teachers gave a student every grade.

most ⟩⟩ a ⟩⟩ every	a ⟩⟩ most ⟩⟩ every	~~every ⟩⟩ most ⟩⟩ a~~
~~most ⟩⟩ every ⟩⟩ a~~	a ⟩⟩ every ⟩⟩ most	~~every ⟩⟩ a ⟩⟩ most~~

(Bruening 2001)

The only way for the secondary object <u>not</u> to take narrowest scope is for both objects to scope over the subject (in surface order).

(7) Most teachers gave a student every grade.

most ⟩⟩ a ⟩⟩ every	a ⟩⟩ most ⟩⟩ every	~~every ⟩⟩ most ⟩⟩ a~~
~~most ⟩⟩ every ⟩⟩ a~~	a ⟩⟩ every ⟩⟩ most	~~every ⟩⟩ a ⟩⟩ most~~

(Bruening 2001)

The only way for the secondary object not to take narrowest scope is for both objects to scope over the subject (in surface order).

gave ⇝

$\text{give}' : \forall i.\forall j.\forall k.(\uparrow \text{SUBJ})\, i \multimap ((\uparrow \text{OBJ})\, j \multimap ((\uparrow \text{OBJ}_\theta)\, k \multimap \uparrow(\text{£}ijk)))$

where £ is the function such that $\text{£}ijk = \begin{cases} i \text{ if } j < k < i \\ k \text{ otherwise} \end{cases}$

Reflections

Features on this account

Scope rigidity because

Features on this account

Scope rigidity because

- quantifiers are not modifiers on the linear logic side

Features on this account

Scope rigidity because

- quantifiers are not modifiers on the linear logic side, and
- verb forms can specify which argument takes narrowest scope.

Features on this account

Scope rigidity because

- quantifiers are not modifiers on the linear logic side, and
- verb forms can specify which argument takes narrowest scope.

This has been stated as particular to verb lexical entries, but of course we'd want to generalize to every transitive/ditransitive verb in the language.

Possible alternatives

Possible alternatives

- Provide f-/s-structure with more internal structure (cf. Andrews (2018) on the relative scope of adjectives).

Possible alternatives

- Provide f-/s-structure with more internal structure (cf. Andrews (2018) on the relative scope of adjectives).
- Read linear logic formulae off c-structure instead.

Possible alternatives

- Provide f-/s-structure with more internal structure (cf. Andrews (2018) on the relative scope of adjectives).
- Read linear logic formulae off c-structure instead.

I can't seen either of these options being popular.

Thanks!

This research is funded by the

LEVERHULME TRUST

References

Andrews, Avery D. 2010. Propositional glue and the projection architecture of LFG. *Linguistics and Philosophy* 33. 141–170. `https://doi.org/10.1007/s10988-010-9079-9`.

Andrews, Avery D. 2018. Sets, heads, and spreading in LFG. *Journal of Language Modelling* 6(1). 131–174. `https://doi.org/10.15398/jlm.v6i1.175`.

Bruening, Benjamin. 2001. QR obeys superiority: frozen scope and ACD. *Linguistic Inquiry* 32(2). 233–273.

Carpenter, Bob. 1998. *Type-logical semantics*. Cambridge, MA: MIT Press.

Corbalán, María Inés & Glyn Morrill. 2016. Overtly anaphoric control in type logical grammar. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald & Sylvain Pogodalla (eds.), *Formal grammar: FG 2015, FG 2016* (Lecture Notes in Computer Science 9804), 183–199. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-662-53042-9_11.

Crouch, Richard & Josef van Genabith. 1999. Context change, underspecification and the structure of Glue language derivations. In Mary Dalrymple (ed.), *Semantics and syntax in Lexical Functional Grammar: The resource logic approach*, 117–189. Cambridge, MA: MIT Press.

Crouch, Richard & Josef van Genabith. 2000. Linear logic for linguists. ESSLLI 2000 course notes. Archived 2006-10-19 in the Internet Archive at http://web.archive.org/web/20061019004949/http://www2.parc.com/istl/members/crouch/esslli00_notes.pdf.

Kanazawa, Makoto. 2015. Syntactic features for regular constraints and an approximation of directional slashes in abstract categorial grammars. In Yusuke Kubota & Robert Levine (eds.), *Empirical advances in categorial grammar: Proceedings of the ESSLLI 2015 workshop (CG 2015)*, 34–70. http://www.u.tsukuba.ac.jp/~kubota.yusuke.fn/cg2015-proceedings.pdf. Last accessed 2019-02-12.

Moot, Richard. 2002. *Proof nets for linguistic analysis*. University of Utrecht dissertation.

Pogodalla, Sylvain & Florent Pompigne. 2012. Controlling extraction in abstract categorial grammars. In Philippe de Groote & Mark-Jan Nederhof (eds.), *Formal grammar: FG 2010, FG 2011* (Lecture Notes in Computer Science 7395), 162–177. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-32024-8_11.