Glue Semantics for Minimalist Syntax

Matthew Gotham

Department of Linguistics
University College London

Annual Meeting of the LAGB
18 September 2015

Slides available at
http://www.ucl.ac.uk/~ucjtmgg/docs/LAGB2015-slides.pdf

Glue semantics

(Slides available at http://www.ucl.ac.uk/~ucjtmgg/docs/LAGB2015-slides.pdf)

- A theory of the syntax/semantics interface.
- Originally developed for LFG, and now the mainstream view of the syntax/semantics interface within LFG (Dalrymple, 1999).
- Implementations also exist for HPSG (Asudeh and Crouch, 2002) and LTAG (Frank and van Genabith, 2001).

Key ideas:

- Syntax+lexicon produces a multiset of premises in a fragment of linear logic (Girard, 1987).
- Semantic interpretation consists in finding a proof to a specified type of conclusion from those premises.

  (like in categorial grammar)

Aims of this talk

- To give an implementation of Glue in Minimalism.
- To show that it has some potential advantages over more conventional approaches to the syntax/semantics interface.

What I mean by 'Minimalist syntax'

- Syntactic theories in the ST→EST→REST→GB→. . . 'Chomskyan' tradition, i.e. as opposed to LFG, HPSG etc.
- So nothing especially cutting-edge. But:
- The factoring together of subcategorization and structure building (in the mechanism of feature-checking) is, if not crucial to this analysis, then certainly useful.

Plan for the rest of the talk

Linear logic

(Girard, 1987)

- Often called a 'logic of resources' (Crouch and van Genabith, 2000, p. 5).
- Key difference from classical logic: for

  premise$_1, \ldots,$ premise$_n \vdash$ conclusion

  to be valid, each premise must be 'used' exactly once. So
- $A \vdash A$ but $A, A \nvdash A$
- We will only be concerned with a small fragment in this talk: implication and the universal quantifier only connectives that will be used.

Interpretation as deduction

Linear implication and functional types

Rules of inference for the $\multimap$ fragment of intuitionistic propositional linear logic and their images under the Curry-Howard correspondence.

$\multimap$ elimination (linear modus ponens) corresponds to application

$$\frac{f : A \multimap B \qquad x : A}{f(x) : B} \multimap_E$$

$\multimap$ introduction (linear conditional proof) corresponds to abstraction

$$\frac{\begin{array}{c}[x : A]^n \\ \vdots \\ \Phi : B\end{array}}{\lambda x.\Phi : A \multimap B} \multimap_I{}^n$$

A simple example

(1) John loves Mary.

John loves Mary

$j' : B$ $\quad \lambda y.\lambda x.\text{love}'(x, y) : A \multimap (B \multimap C)$ $\quad m' : A$

$$\frac{\dfrac{\lambda y.\lambda x.\text{love}'(x,y) : A \multimap (B \multimap C) \qquad m' : A}{\lambda x.\text{love}'(x, m') : B \multimap C} \multimap_E \qquad j' : B}{\text{love}'(j', m') : C} \multimap_E$$

Quantifier scope ambiguity

(2) Someone loves everyone.

<u>someone</u> <u>loves</u> <u>everyone</u>

$\lambda P.\text{some}'(\text{person}', P) : (B \multimap C) \multimap C$ $\quad \lambda y.\lambda x.\text{love}'(x, y) : A \multimap (B \multimap C)$ $\quad \lambda Q.\text{every}'(\text{person}', Q) : (A \multimap C) \multimap C$

Surface scope interpretation

$$\dfrac{\begin{array}{c}\lambda P.\text{some}'(\text{person}', P)\\ : (B \multimap C) \multimap C\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\lambda Q.\text{every}'(\text{person}', Q)\\ : (A \multimap C) \multimap C\end{array} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\lambda z.\lambda v.\text{love}'(v, z)\\ : A \multimap (B \multimap C)\end{array} \quad \begin{bmatrix} y \\ : A\end{bmatrix}^1}{\begin{array}{c}\lambda v.\text{love}'(v, y)\\ : B \multimap C\end{array}} \multimap_E \quad \begin{bmatrix} x \\ : B\end{bmatrix}^2}{\begin{array}{c}\text{love}'(x, y)\\ : C\end{array}} \multimap_E}{\begin{array}{c}\lambda y.\text{love}'(x, y)\\ : A \multimap C\end{array}} \multimap_I{}^1}{\begin{array}{c}\text{every}'(\text{person}', (\lambda y.\text{love}'(x, y)))\\ : C\end{array}} \multimap_E}{\begin{array}{c}\lambda x.\text{every}'(\text{person}', (\lambda y.\text{love}'(x, y)))\\ : B \multimap C\end{array}} \multimap_I{}^2}{\text{some}'(\text{person}', \lambda x.\text{every}'(\text{person}', \lambda y.\text{love}'(x, y))) : C} \multimap_E$$

Inverse scope interpretation

$$\dfrac{\begin{array}{c}\lambda Q.\text{every}'(\text{person}', Q)\\ : (A \multimap C) \multimap C\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\lambda P.\text{some}'(\text{person}', P)\\ : (B \multimap C) \multimap C\end{array} \qquad \dfrac{\begin{array}{c}\lambda z.\lambda v.\text{love}'(v, z)\\ : A \multimap (B \multimap C)\end{array} \quad \begin{bmatrix} y \\ : A\end{bmatrix}^1}{\begin{array}{c}\lambda v.\text{love}'(v, y)\\ : B \multimap C\end{array}} \multimap_E}{\begin{array}{c}\text{some}'(\text{person}', \lambda x.\text{love}'(x, y))\\ : C\end{array}} \multimap_E}{\begin{array}{c}\lambda y.\text{some}'(\text{person}', \lambda x.\text{love}'(x, y))\\ : A \multimap C\end{array}} \multimap_I{}^1}{\text{every}'(\text{person}', \lambda y.\text{some}'(\text{person}', \lambda x.\text{love}'(x, y))) : C} \multimap_E$$

Meaning constructors

Following Kokkonidis (2008), I'll use a fragment of (monadic) first-order linear logic as the glue language.

- Predicates: e and t
- Constants: $1, 2, 3 \ldots$
- Variables: $X, Y, Z \ldots$
- Connectives: $\multimap$ and $\forall$

I'll use subscript notation, e.g. $e_1 \multimap t_X$ instead of $e(1) \multimap t(X)$.

	LL	λ calculus	
propositions as types	implicational proposition	functional type	
rules as operations	$\multimap$ elimination	application	$\dfrac{f : A \multimap B \qquad x : A}{f(x) : B} \multimap_E$
	$\multimap$ introduction	abstraction	$\dfrac{\begin{array}{c}[x : A]^n\\ \vdots\\ \Phi : B\end{array}}{\lambda x.\Phi : A \multimap B} \multimap_I{}^n$
	$\forall$ elimination	—	$\dfrac{\Phi : \forall X.A}{\Phi : A[X \leftarrow c]} \forall_E$ c free for X
	$\forall$ introduction	—	$\dfrac{\Phi : A}{\Phi : \forall X.A} \forall_I$ X not free in any open leaf

Basic ideas

- Syntactic objects have features.
- The structure-building operation(s) (Merge) is/are based on the matching of features.
- Every feature bears an index, and when two features match their indices must also match.
- Those indices are used to label linear logic formulae paired with interpretations, thereby providing the syntax/semantics connection.

Features

Largely based on Adger (2003) and Adger (2010)

- Some features describe what an LI *is*.
- Some features describe what an LI *needs* (uninterpretable features). Those can be strong(*) or weak.

V	T
$\langle u\mathrm{D}, u\mathrm{D}\rangle$	$\langle u\mathrm{D}^*\rangle$
|	|
love	-s

(I'm going to ignore morphosyntactic features and agreement.)

Structure-building operation(s)

Merge.

- Hierarchy of Projections-driven.
- Selectional features-driven.
  - External.
  - Internal.

Hierarchy of projections

Adger (2003) has:

Clausal: $\mathrm{C} \rangle \mathrm{T} \rangle (\mathrm{Neg}) \rangle (\mathrm{Perf}) \rangle (\mathrm{Prog}) \rangle (\mathrm{Pass}) \rangle v \rangle \mathrm{V}$

Nominal: $\mathrm{D} \rangle (\mathrm{Poss}) \rangle n \rangle \mathrm{N}$

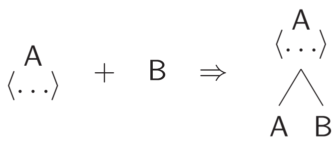
Adjectival: $(\mathrm{Deg}) \rangle \mathrm{A}$

We'll use:

Clausal: $\mathrm{C} \rangle \mathrm{T} \rangle \mathrm{V}$

Nominal: $\mathrm{D} \rangle \mathrm{N}$

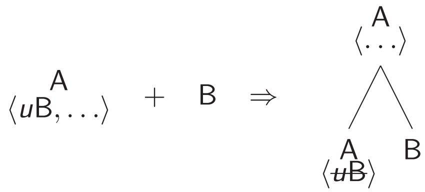
HoPs merge



Where A and B are in the same hierarchy of projections (HoPs) and A is higher on that HoPs than B

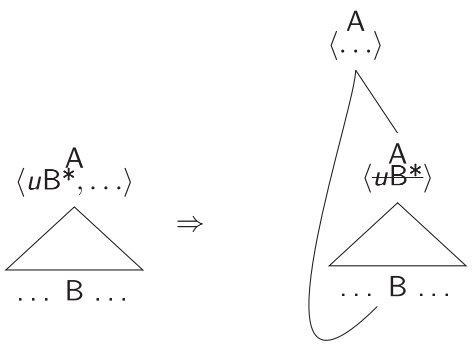
Select merge

External



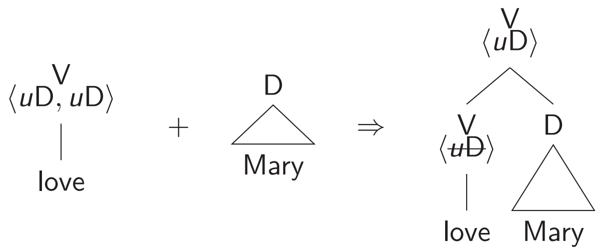
Select merge

Internal



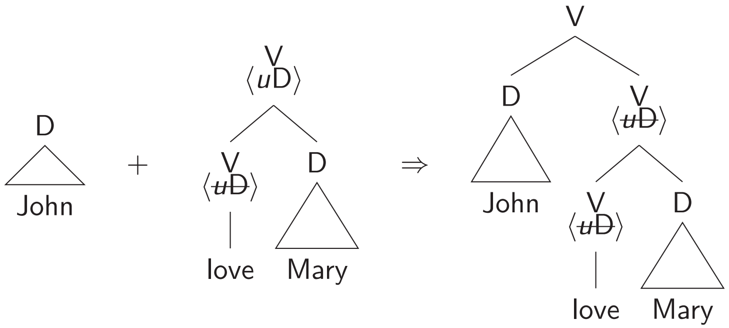
External merge

An example



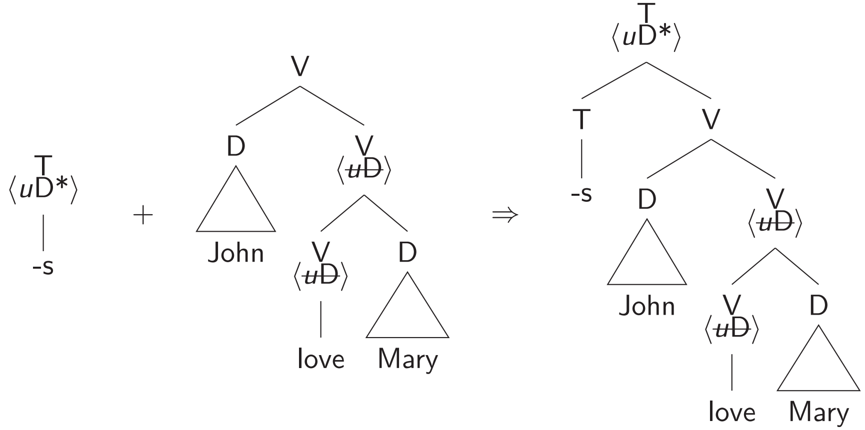
External merge

An example



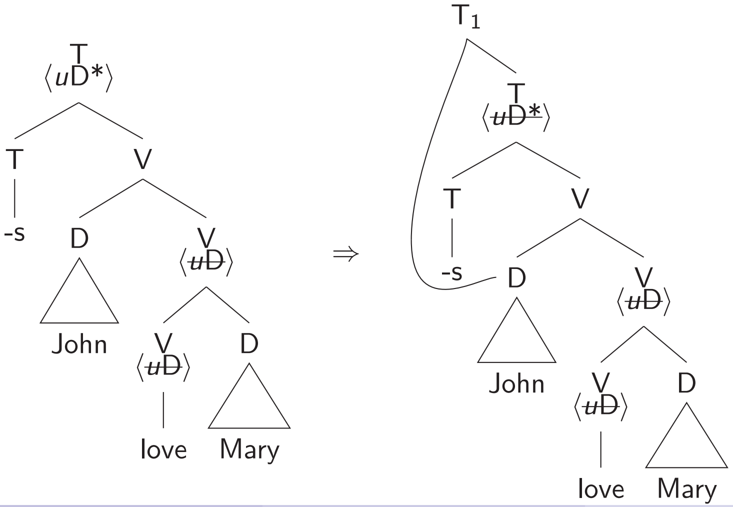
HoPs merge

An example

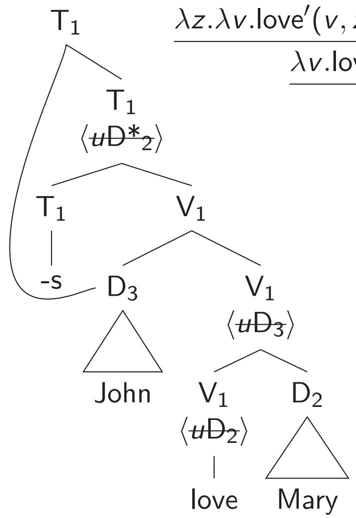


Internal merge

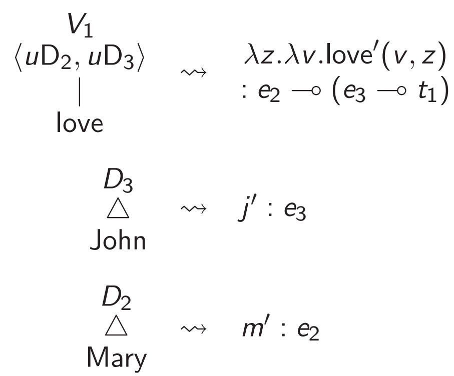
An example

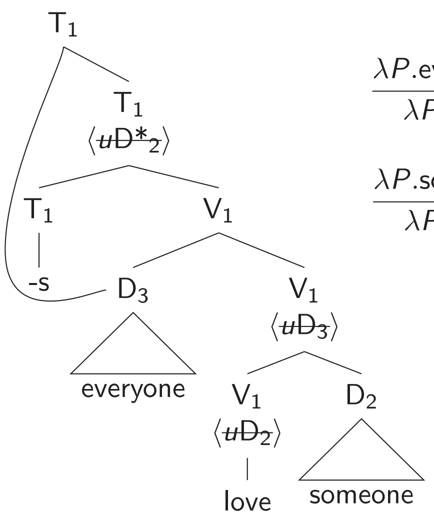


Indices



$$\dfrac{\dfrac{\lambda z.\lambda v.\text{love}'(v,z) : e_2 \multimap (e_3 \multimap t_1) \qquad m' : e_2}{\lambda v.\text{love}'(v,m') : e_3 \multimap t_1}\multimap_E \qquad j' : e_3}{\text{love}'(j',m') : t_1}\multimap_E$$





$$\dfrac{\lambda P.\text{every}'(\text{person}', P) : \forall X.(e_3 \multimap t_X) \multimap t_X}{\lambda P.\text{every}'(\text{person}', P) : (e_3 \multimap t_1) \multimap t_1}\;\forall_E$$

$$\dfrac{\lambda P.\text{some}'(\text{person}', P) : \forall X.(e_2 \multimap t_X) \multimap t_X}{\lambda P.\text{some}'(\text{person}', P) : (e_2 \multimap t_1) \multimap t_1}\;\forall_E$$

$$\underset{\text{everyone}}{\overset{D_3}{\triangle}} \rightsquigarrow \begin{array}{l}\lambda P.\text{every}'(\text{person}', P) \\ : \forall X.(e_3 \multimap t_X) \multimap t_X\end{array}$$

$$\underset{\text{someone}}{\overset{D_2}{\triangle}} \rightsquigarrow \begin{array}{l}\lambda P.\text{some}'(\text{person}', P) \\ : \forall X.(e_2 \multimap t_X) \multimap t_X\end{array}$$

Surface scope interpretation

$$\dfrac{\begin{array}{l}\lambda P.\text{some}'(\text{person}', P) \\ : (e_3 \multimap t_1) \multimap t_1\end{array} \quad \dfrac{\dfrac{\begin{array}{l}\lambda Q.\text{every}'(\text{person}', Q) \\ : (e_2 \multimap t_1) \multimap t_1\end{array} \quad \dfrac{\dfrac{\dfrac{\begin{array}{l}\lambda z.\lambda v.\text{love}'(v, z) \\ : e_2 \multimap (e_3 \multimap t_1)\end{array} \quad \begin{bmatrix} y \\ : e_2\end{bmatrix}^1}{\begin{array}{c}\lambda v.\text{love}'(v, y) \\ : e_3 \multimap t_1\end{array}}\multimap_E \quad \begin{bmatrix} x \\ : e_3\end{bmatrix}^2}{\begin{array}{c}\text{love}'(x, y) \\ : t_1\end{array}}\multimap_E}{\begin{array}{c}\lambda y.\text{love}'(x, y) \\ : e_2 \multimap t_1\end{array}}\multimap_I{}^1}{\begin{array}{c}\text{every}'(\text{person}', (\lambda y.\text{love}'(x, y))) \\ : t_1\end{array}}\multimap_E}{\begin{array}{c}\lambda x.\text{every}'(\text{person}', (\lambda y.\text{love}'(x, y))) \\ : e_3 \multimap t_1\end{array}}\multimap_I{}^2}{\text{some}'(\text{person}', \lambda x.\text{every}'(\text{person}', \lambda y.\text{love}'(x, y))) : t_1}\multimap_E$$

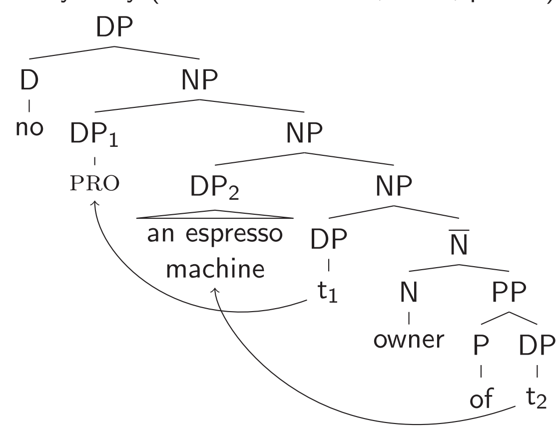
Inverse scope interpretation

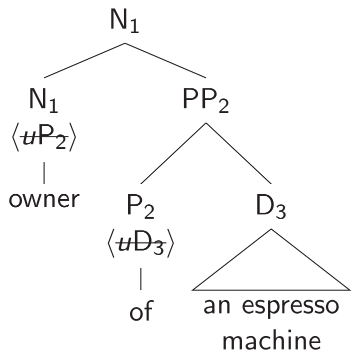
$$\dfrac{\begin{array}{l}\lambda Q.\text{every}'(\text{person}', Q) \\ : (e_2 \multimap t_1) \multimap t_1\end{array} \quad \dfrac{\dfrac{\begin{array}{l}\lambda P.\text{some}'(\text{person}', P) \\ : (e_3 \multimap t_1) \multimap t_1\end{array} \quad \dfrac{\begin{array}{l}\lambda z.\lambda v.\text{love}'(v, z) \\ : e_2 \multimap (e_3 \multimap t_1)\end{array} \quad \begin{bmatrix} y \\ : e_2\end{bmatrix}^1}{\begin{array}{c}\lambda v.\text{love}'(v, y) \\ : e_3 \multimap t_1\end{array}}\multimap_E}{\begin{array}{c}\text{some}'(\text{person}', \lambda x.\text{love}'(x, y)) \\ : t_1\end{array}}\multimap_E}{\begin{array}{c}\lambda y.\text{some}'(\text{person}', \lambda x.\text{love}'(x, y)) \\ : e_2 \multimap t_1\end{array}}\multimap_I{}^1}{\text{every}'(\text{person}', \lambda y.\text{some}'(\text{person}', \lambda x.\text{love}'(x, y))) : t_1}\multimap_E$$

Embedded QPs

(3) No owner of an espresso machine drinks tea.

Analysis by (Heim and Kratzer, 1998, p. 229) of the surface scope reading:

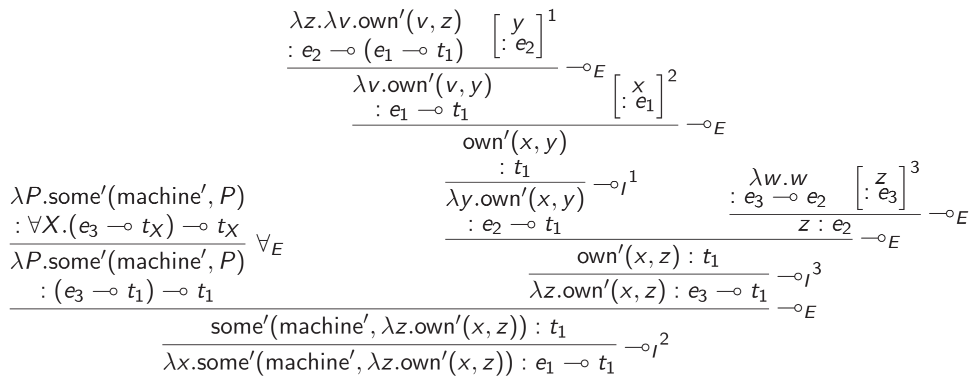




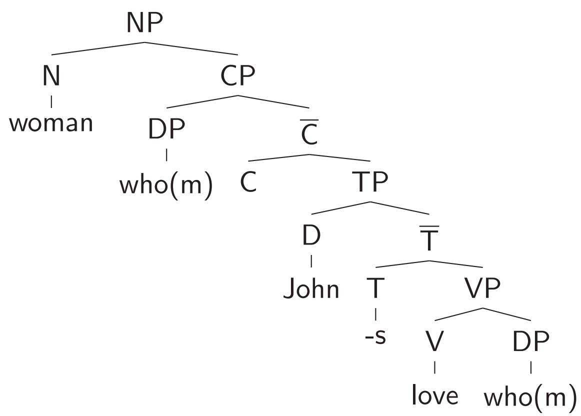
N_1 $\langle uP_2 \rangle$ | owner $\rightsquigarrow$ $\lambda z.\lambda v.\text{own}'(v,z)$ $: e_2 \multimap (e_1 \multimap t_1)$

P_2 $\langle uD_3 \rangle$ | of $\rightsquigarrow$ $\lambda w.w : e_3 \multimap e_2$

D_3 $\triangle$ an e.m. $\rightsquigarrow$ $\lambda P.\text{some}'(\text{machine}', P)$ $: \forall X.(e_3 \multimap t_X) \multimap t_X$



(Overt) movement

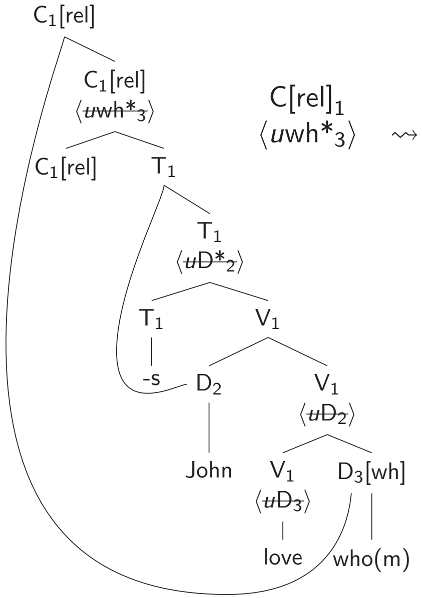


Trace theory was abandoned in early minimalism in favor of the so-called copy theory of movement. Indices were deemed incompatible with the principle of Inclusiveness, which restricts the content of tree structures to information originating in the lexicon. Because indices of phrases cannot be traced back to any lexical entry, they are illegitimate syntactic objects.

(Neeleman and van de Koot, 2010, p. 331)

If we assume that the computational system of syntax doesn't use variables, variables are introduced at the point where the LF-structure of a sentence is translated into a semantic representation.

(Sauerland, 1998, p. 196)



$$\begin{array}{c}\mathrm{C[rel]}_1\\ \langle u\mathrm{wh}^*{}_3\rangle\end{array} \rightsquigarrow \begin{array}{c}\lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x) :\\ (e_3 \multimap t_1) \multimap ((e_1 \multimap t_1) \multimap (e_1 \multimap t_1))\end{array}$$

$$\dfrac{\begin{array}{c}\lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)\\ : (e_3 \multimap t_1) \multimap\\ ((e_1 \multimap t_1) \multimap (e_1 \multimap t_1))\end{array} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\lambda z.\lambda v.\mathrm{love}'(v,z)\\ : e_3 \multimap (e_2 \multimap t_1) \quad [y : e_3]^1\end{array}}{\begin{array}{c}\lambda v.\mathrm{love}'(v,y)\\ : e_2 \multimap t_1\end{array}}\multimap_E \qquad j' : e_2}{\mathrm{love}'(j',y) : t_1}\multimap_E}{\lambda y.\mathrm{love}'(j',y) : e_3 \multimap t_1}\multimap_I{}^1}{\lambda Q.\lambda x.\mathrm{love}'(j',x) \wedge Q(x) : (e_1 \multimap t_1) \multimap (e_1 \multimap t_1)}\multimap_E$$

References I

Adger, David (2003). *Core Syntax: A Minimalist Approach*. Core Linguistics. Oxford: Oxford University Press.

– (2010). "A Minimalist theory of feature structure". In: *Features: Perspectives on a Key Notion in Linguistics*. Ed. by Anna Kibort and Greville G. Corbett. Oxford: Oxford University Press, pp. 185–218.

Asudeh, Ash and Richard Crouch (2002). "Glue Semantics for HPSG". In: *Proceedings of the 8th International HPSG Conference*. (Norwegian University of Science and Technology, Trondheim). Ed. by Frank van Eynde, Lars Hellan, and Dorothee Beermann. Stanford, CA: CSLI Publications.

References II

Crouch, Richard and 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`.

Dalrymple, Mary, ed. (1999). *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. Cambridge, MA: MIT Press.

Frank, Anette and Josef van Genabith (2001). "GlueTag: Linear Logic-based Semantics for LTAG—and what it teaches us about LFG and LTAG". In: *Proceedings of the LFG01 Conference*. (University of Hong Kong). Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications.

Girard, Jean-Yves (1987). "Linear Logic". In: *Theoretical Computer Science* 50.1, pp. 1–101. ISSN: 0304-3975.

References III

Heim, Irene and Angelika Kratzer (1998). *Semantics in Generative Grammar*. Blackwell Textbooks in Linguistics 13. Oxford: Wiley-Blackwell.

Kokkonidis, Miltiadis (2008). "First-Order Glue". In: *Journal of Logic, Language and Information* 17.1, pp. 43–68.

Neeleman, Ad and Hans van de Koot (2010). "A Local Encoding of Syntactic Dependencies and Its Consequences for the Theory of Movement". In: *Syntax* 13.4, pp. 331–372.

Sauerland, Uli (1998). "The Meaning of Chains". PhD thesis. Massachusetts Institute of Technology.