Towards Glue Semantics for Minimalist Syntax

Matthew Gotham

Department of Linguistics
University College London

Cambridge Syntax Cluster event:
'Interactions between syntax and semantics across frameworks'
8 January 2015

Plan for today

Glue semantics

Minimalist syntax

Putting the two together

So what?

Glue

An approach to the syntax-semantics interface

- Compatible with different syntactic theories: LFG (Dalrymple, 1999), HPSG (Asudeh and Crouch, 2002), LTAG (Frank and Genabith, 2001), CG...
- Compatible with different meaning representations: IL, DRT, situation semantics...

So this talk should really be called 'towards glue semantics for minimalist syntax and the lambda calculus'

A good introduction is provided by Lev (2007, Ch. 3).

Basic points

- Lexicon+syntax produces premises in a fragment of linear logic (the glue language), each of which is paired with a lambda term
- Semantic interpretation uses deduction to assemble final meaning from these premises

Linear logic

Classical logic proof rules: conclusion follows from some subset of the **set** of premises

$$P \to Q, P \to (Q \to R) \vdash P \to R$$
$$P, Q \vdash Q$$

Linear logic proof rules: conclusion follows from *the* **multiset** of premises

$$P \multimap Q, P \multimap (Q \multimap R) \nvdash P \multimap R$$
$$P, Q \nvdash Q$$

Linear logic keeps a strict accounting of the number of times a premise is used in a proof. But it doesn't care about how they are ordered or grouped.

$$P, P \multimap Q \vdash Q$$
$$P \multimap Q, P \vdash Q$$

The Curry-Howard Correspondence

There's a mapping between (intuitionistic) linear logic inference rules and operations on meaning terms.

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

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

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

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

A simple example

(1) Sam trains Rover

Sam trains Rover

$$s' : B \qquad \lambda y.\lambda x.\text{train}'(x, y) : A \multimap (B \multimap C) \qquad r' : A$$

$$\cfrac{\cfrac{\lambda y.\lambda x.\text{train}'(x, y) : A \multimap (B \multimap C) \quad r' : A}{\lambda x.\text{train}'(x, r') : B \multimap C} \multimap_E \qquad s' : B}{\text{train}'(s', r') : C} \multimap_E$$

An ambiguous sentence

(2) A boy trains every dog

A boy trains every dog

$\lambda P.\text{some}'(\text{boy}', P) : (B \multimap C) \multimap C$

$\lambda y.\lambda x.\text{train}'(x, y) : A \multimap (B \multimap C)$

$\lambda Q.\text{every}'(\text{dog}', Q) : (A \multimap C) \multimap C$

Surface scope

$$\dfrac{\lambda y.\lambda x.\mathrm{train}'(x,y) : A \multimap (B \multimap C) \qquad [y : A]^1}{B \multimap C}\multimap_E$$

$$\dfrac{B \multimap C \qquad [x : B]^2}{C}\multimap_E$$

$$\dfrac{C}{A \multimap C}\multimap_I{}^1$$

$$\dfrac{A \multimap C \qquad \lambda Q.\mathrm{every}'(\mathrm{dog}', Q) : (A \multimap C) \multimap C}{\mathrm{every}'(\mathrm{dog}', (\lambda y.\mathrm{train}'(x,y))) : C}\multimap_E$$

$$\dfrac{\mathrm{every}'(\mathrm{dog}', (\lambda y.\mathrm{train}'(x,y))) : C}{B \multimap C}\multimap_I{}^2$$

$$\dfrac{\lambda P.\mathrm{some}'(\mathrm{boy}', P) : (B \multimap C) \multimap C \qquad B \multimap C}{\mathrm{some}'(\mathrm{boy}', \lambda x.\mathrm{every}'(\mathrm{dog}', \lambda y.\mathrm{train}'(x,y))) : C}\multimap_E$$

Inverse scope

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\lambda P.\text{some}'(\text{boy}', P)\\ : (B \multimap C) \multimap C\end{array} \qquad \dfrac{\begin{array}{c}\lambda y.\lambda x.\text{train}'(x, y)\\ : A \multimap (B \multimap C)\end{array} \quad [A]^1}{B \multimap C}\multimap_E}{\text{some}'(\text{boy}', \lambda x.\text{train}'(x, y)) : C}\multimap_E}{A \multimap C}\multimap_I{}^1 \qquad \begin{array}{c}\lambda Q.\text{every}'(\text{dog}', Q)\\ : (A \multimap C) \multimap C\end{array}}{\text{every}'(\text{dog}', \lambda y.\text{some}'(\text{boy}', \lambda x.\text{train}'(x, y))) : C}\multimap_E$$

Lexical items

- Lexical items (LIs) are bundles of features.
- Some features describe what an LI *is*.
- Some features describe what an LI *needs* (uninterpretable features). Those can be strong(*) or weak.

V	T
$\langle u\text{D}, u\text{D}\rangle$	$\langle u\text{D*}\rangle$
|	
train	

(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

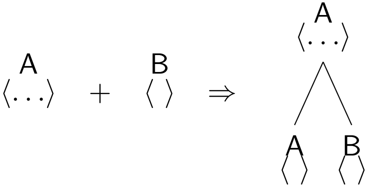
Adger (2003) has:

Clausal:	C $\rangle$ T $\rangle$ (Neg) $\rangle$ (Perf) $\rangle$ (Prog) $\rangle$ (Pass) $\rangle$ v $\rangle$ V
Nominal:	D $\rangle$ (Poss) $\rangle$ n $\rangle$ N
Adjectival:	(Deg) $\rangle$ A

We'll use:

Clausal:	T $\rangle$ V
Nominal:	D $\rangle$ N

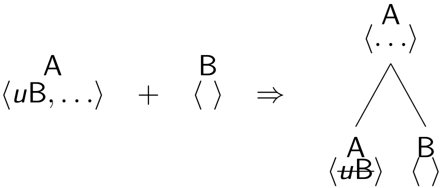
HoPs merge



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

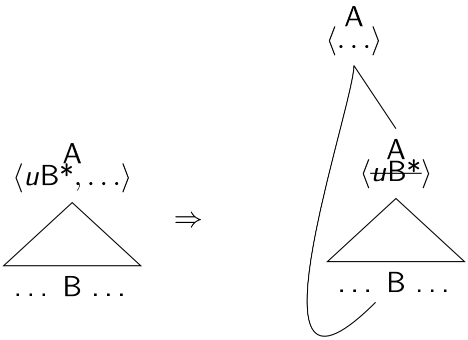
Select merge

External



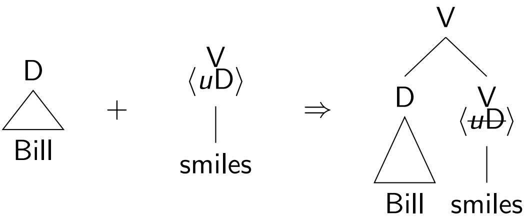
Select merge

Internal



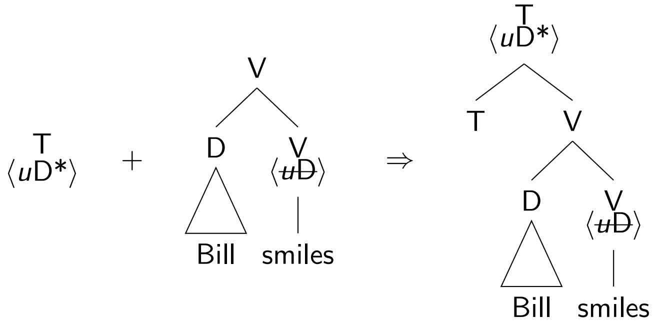
External merge

An example



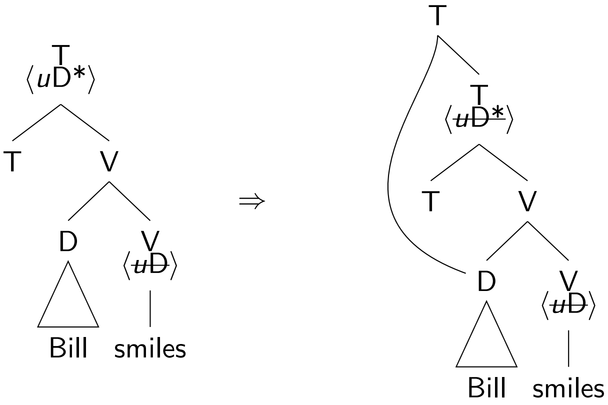
HoPs merge

An example



Internal merge

An example



Indices on features

Every feature (interpretatble or uninterpretable) bears a numerical index subject to the following contstraints:

- The indices assigned to features within a single lexical item must all be distinct.

  E.g. in this:

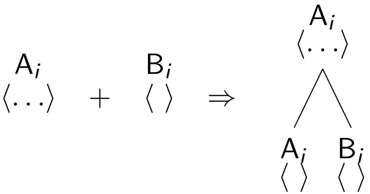
  $$\begin{array}{c} \mathrm{V}_i \\ \langle u\mathrm{D}_j, u\mathrm{D}_k \rangle \\ | \\ \text{train} \end{array}$$

  We must have $i \neq j \neq k$

- Structure-building operations are sensitive to indices in the following ways:

HoPs merge

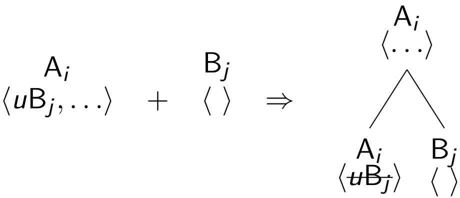
with indices



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

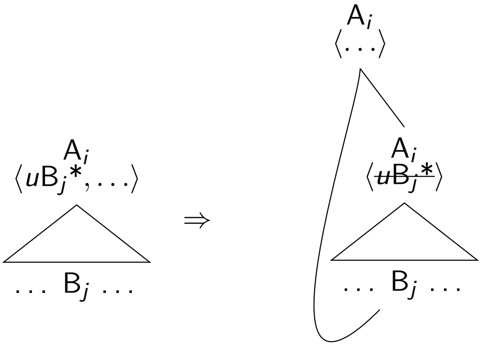
External merge

with indices



Internal merge

with indices



Meaning constructors

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

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

We need a new rule of inference:

$$\frac{f : \forall X(P)}{f : P[a/X]} \, \forall_E$$

Lexical items

Some examples

Sam	
Syntax	N_{i}
Semantics	$s' : e(i)$

trains	
Syntax	V_{i} $\langle u\text{D}_{j}, u\text{D}_{k} \rangle$
Semantics	$\lambda x.\lambda y.\text{train}'(y, x) : e(j) \multimap (e(k) \multimap t(i))$

Lexical items

Some more examples

every	
Syntax	D_i
Semantics	$\lambda P.\lambda Q.\mathrm{every}'(P, Q)$: $(e_N(i) \multimap t(i)) \multimap \forall X((e(i) \multimap t(X)) \multimap t(X))$

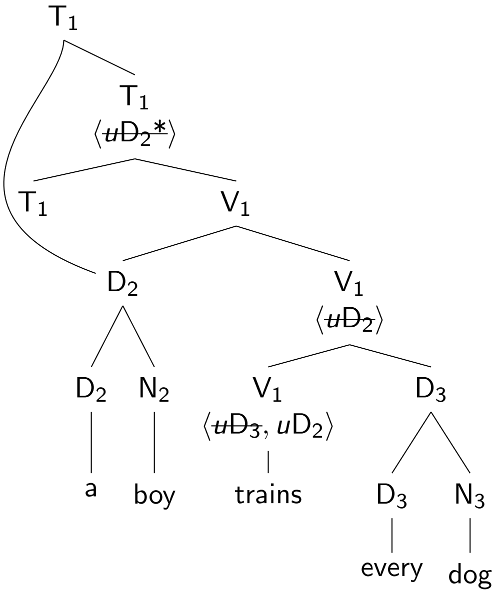
dog	
Syntax	N_i
Semantics	$\lambda x.\mathrm{dog}'(x) : e_N(i) \multimap t(i)$

a

Syntax	D_{i}
Semantics	$\lambda F.\lambda G.\mathrm{some}'(F, G)$: $(e_N(i) \multimap t(i)) \multimap \forall X((e(i) \multimap t(X)) \multimap t(X))$

boy

Syntax	N_{i}
Semantics	$\lambda x.\mathrm{boy}'(x) : e_N(i) \multimap t(i)$


T1
T1
⟨uD2*⟩
T1
V1
D2
V1
⟨uD2⟩
D2
N2
V1
⟨uD3, uD2⟩
D3
a
boy
trains
D3
N3
every
dog

The mapping to interpretation

The multiset of premises:

- $\lambda P.\lambda Q.\text{some}'(P, Q)$:
  $(e_N(2) \multimap t(2)) \multimap \forall X((e(2) \multimap t(X)) \multimap t(X))$
- $\text{boy}' : e_N(2) \multimap t(2)$
- $\lambda x.\lambda y.\text{train}'(y, x) : e(3) \multimap (e(2) \multimap t(1))$
- $\lambda F.\lambda G.\text{every}'(F, G)$:
  $(e_N(3) \multimap t(3)) \multimap \forall Y((e(3) \multimap t(Y)) \multimap t(Y))$
- $\text{dog}' : e_N(3) \multimap t(3)$

Solving from the multiset of premises

$$\cfrac{\cfrac{\begin{array}{cc} \lambda P.\lambda Q.\text{some}'(P, Q) : & \text{boy}' : \\ (e_N(2) \multimap t(2)) \multimap \forall X((e(2) \multimap t(X)) \multimap t(X)) & e_N(2) \multimap t(2) \end{array}}{\lambda Q.\text{some}'(\text{boy}', Q) : \forall X((e(2) \multimap t(X)) \multimap t(X))} \multimap_E}{\lambda Q.\text{some}'(\text{boy}', Q) : (e(2) \multimap t(1)) \multimap t(1)} \forall_E$$

$$\cfrac{\cfrac{\begin{array}{cc} \lambda F.\lambda G.\text{every}'(F, G) : & \text{dog}' : \\ (e_N(3) \multimap t(3)) \multimap \forall X((e(3) \multimap t(X)) \multimap t(X)) & e_N(2) \multimap t(2) \end{array}}{\lambda G.\text{every}'(\text{dog}', G) : \forall X((e(3) \multimap t(X)) \multimap t(X))} \multimap_E}{\lambda G.\text{every}'(\text{dog}', G) : (e(3) \multimap t(1)) \multimap t(1)} \forall_E$$

Surface scope

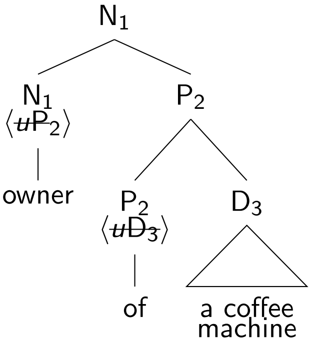
$$
\dfrac{
\begin{array}{c}\lambda P.\text{some}'(\text{boy}', P)\\ : (e(2) \multimap t(1)) \multimap t(1)\end{array}
\quad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}\lambda y.\lambda x.\text{train}'(x, y)\\ : e(3) \multimap (e(2) \multimap t(1))\end{array}
\quad [y : e(3)]^1
}{e(2) \multimap t(1)} \multimap_E
\quad [x : e(2)]^2
}{t(1)} \multimap_E
}{e(3) \multimap t(1)} {\multimap_I}^1
\quad
\begin{array}{c}\lambda Q.\text{every}'(\text{dog}', Q)\\ : (e(3) \multimap t(1)) \multimap t(1)\end{array}
}{\text{every}'(\text{dog}', (\lambda y.\text{train}'(x, y))) : t(1)} \multimap_E
}{e(2) \multimap t(1)} {\multimap_I}^2
}{\text{some}'(\text{boy}', \lambda x.\text{every}'(\text{dog}', \lambda y.\text{train}'(x, y))) : t(1)} \multimap_E
$$

Inverse scope

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\lambda P.\mathrm{some}'(\mathrm{boy}', P)\\ :(e(2)\multimap t(1))\multimap t(1)\end{array} \quad \dfrac{\begin{array}{c}\lambda y.\lambda x.\mathrm{train}'(x,y)\\ : e(3)\multimap(e(2)\multimap t(1))\end{array}\quad [y : e(3)]^1}{e(2)\multimap t(1)}\multimap_E}{\mathrm{some}'(\mathrm{boy}',\lambda x.\mathrm{train}'(x,y)) : t(1)}\multimap_E}{e(3)\multimap t(1)}\multimap_I{}^1 \quad \begin{array}{c}\lambda Q.\mathrm{every}'(\mathrm{dog}', Q)\\ :(e(3)\multimap t(1))\multimap t(1)\end{array}}{\mathrm{every}'(\mathrm{dog}',\lambda y.\mathrm{some}'(\mathrm{boy}',\lambda x.\mathrm{train}'(x,y))) : t(1)}\multimap_E$$

Embedded QNPs

(3) No owner of a coffee machine drinks tea.



$$\lambda x.\lambda y.\mathrm{own}'(y, x) : e(2) \multimap (e_N(1) \multimap t(1))$$

$$\lambda v.v : e(3) \multimap e(2)$$

$$\lambda P.\mathrm{some}'(\text{cof-mach}', P) : \forall X((e(3) \multimap t(X)) \multimap t(X))$$

$$
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}\lambda x.\lambda y.\text{own}'(y,x) :\\ e(2) \multimap (e_N(1) \multimap t(1))\end{array}
\quad
\begin{array}{c}\lambda v.v :\\ e(3) \multimap e(2)\end{array}
}{
\begin{array}{c}\lambda v.\lambda y.\text{own}'(y,v) :\\ e(3) \multimap (e_N(1) \multimap t(1))\end{array}
}\ HS
}{
\begin{array}{c}\lambda y.\lambda v.\text{own}'(y,v) :\\ e_N(1) \multimap (e(3) \multimap t(1))\end{array}
}\ Perm
\qquad
\begin{array}{c}\forall X((e(3) \multimap t(X)) \multimap t(X))\\
\dfrac{\lambda P.\text{some}'(\text{cof-mach}', P) :}{(e(3) \multimap t(1)) \multimap t(1)}\ \forall_E\\
\lambda P.\text{some}'(\text{cof-mach}', P) :\end{array}
}{
\lambda y.\text{some}'(\text{cof-mach}', \lambda v.\text{own}'(y,v)) : e_N(1) \multimap t(1)
}\ HS
$$

Phases

- General idea: at certain points in the tree you must use the multiset of premises you have in a proof with a conclusion of a particular type.
- Scope islands: impossible interpretation would involve failling to compute proof at one of those points.

References

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

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.

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.

Kokkonidis, Miltiadis (2008). "First-Order Glue". In: *Journal of Logic, Language and Information* 17 (1), pp. 43–68.

Lev, Iddo (2007). "Packed Computation of Exact Meaning Representations". PhD thesis. Stanford University.