Glue semantics for Universal Dependencies

Matthew Gotham and Dag Haug

Centre for Advanced Study
at the Norwegian Academy of Science and Letters

Gothenburg, 8 March 2018

Introduction

- Universal Dependencies (UD) is a *de facto* annotation standard for cross-linguistic annotation of syntactic structure

Introduction

- Universal Dependencies (UD) is a *de facto* annotation standard for cross-linguistic annotation of syntactic structure
- $\rightarrow$ interest in deriving semantic representations from UD structures, ideally in a language-independent way

Introduction

- Universal Dependencies (UD) is a *de facto* annotation standard for cross-linguistic annotation of syntactic structure
- $\rightarrow$ interest in deriving semantic representations from UD structures, ideally in a language-independent way
- Our approach: adapt and exploit techniques from LFG + Glue semantics
  - dependency structures $\approx$ f-structures
  - LFG inheritance in UD (via Stanford dependencies)
  - Glue offers a syntax-semantics interace where syntax can underspecify semantics

Introduction

- Universal Dependencies (UD) is a *de facto* annotation standard for cross-linguistic annotation of syntactic structure
- $\rightarrow$ interest in deriving semantic representations from UD structures, ideally in a language-independent way
- Our approach: adapt and exploit techniques from LFG + Glue semantics
  - dependency structures $\approx$ f-structures
  - LFG inheritance in UD (via Stanford dependencies)
  - Glue offers a syntax-semantics interace where syntax can underspecify semantics
- Postpone the need for language-specific, lexical resources

Outline

Plan

Target representations

- Our target representations for sentence meanings are DRSs.
- The format of these DRSs is inspired by Boxer (Bos, 2008).

Target representations

- Our target representations for sentence meanings are DRSs.
- The format of these DRSs is inspired by Boxer (Bos, 2008).
- We assume discourse referents (drefs) of three sorts: entities (x_n), eventualities (e_n) and propositions (p_n).

Target representations

- Our target representations for sentence meanings are DRSs.
- The format of these DRSs is inspired by Boxer (Bos, 2008).
- We assume discourse referents (drefs) of three sorts: entities (x_n), eventualities (e_n) and propositions (p_n).
- The predicates *ant* means that its argument has an antecedent (it's a presupposed dref).

  $\rightarrow$ Also applies to the predicates beginning *pron.*_
- The connective ∂ marks presupposed conditions—it maps TRUE to TRUE and is otherwise undefined.

  $\rightarrow$ Unlike Boxer, which has separate DRSs for presupposed and asserted material.

An example

(1) Abrams persuaded the dog to bark.

Boxer:

$$\left(\begin{array}{|c|}\hline x_2 \\ \hline dog(x_2) \\ \hline\end{array} + \begin{array}{|l|}\hline x_1\ \ e_1\ \ p_1 \\ \hline named(x_1, abrams) \\ persuade(e_1) \\ agent(e_1, x_1) \\ theme(e_1, x_2) \\ content(e_1, p_1) \\ p_1 : \begin{array}{|l|}\hline e_2 \\ \hline bark(e_2) \\ agent(e_2, x_2) \\ \hline\end{array} \\ \hline\end{array}\right)$$

Us:

$$\begin{array}{|l|}\hline x_1\ \ x_2\ \ e_1\ \ p_1 \\ \hline named(x_1, abrams) \\ ant(x_2) \\ \partial(dog(x_2)) \\ persuade(e_1) \\ agent(e_1, x_1) \\ theme(e_1, x_2) \\ content(e_1, p_1) \\ p_1 : \begin{array}{|l|}\hline e_2 \\ \hline bark(e_2) \\ agent(e_2, x_2) \\ \hline\end{array} \\ \hline\end{array}$$

Other running examples

(taken from the CCS development suite)

(2) He hemmed and hawed.

x_1 e_1 e_2
$pron.he(x_1)$ $hem(e_1)$ $agent(e_1, x_1)$ $haw(e_2)$ $agent(e_2, x_1)$

(3) The dog they thought we admired barks.

$x_1\ x_2\ x_3\ e_1\ e_2\ p_1$

$ant(x_1), \partial(dog(x_1))$
$pron.they(x_2), pron.we(x_3)$
$bark(e_1), agent(e_1, x_1)$
$\partial(think(e_2)), \partial(agent(e_2, x_2))$
$\partial(content(e_2, p_1))$

p_1 : [| $admire(e_3)$, $agent(e_3, x_3)$, $theme(e_3, x_1)$]

Underlying logic

- The Glue approach relies on meanings being put together by application and abstraction, so we need some form of compositional or λ-DRT for meaning construction.

$$\text{someone} \rightsquigarrow \lambda P.\begin{array}{|c|}\hline x_1 \\ \hline \mathit{person}(x_1) \\ \hline \end{array} \; ; \; P(x_1)$$

Underlying logic

- The Glue approach relies on meanings being put together by application and abstraction, so we need some form of compositional or λ-DRT for meaning construction.

$$\text{someone} \rightsquigarrow \lambda P.\boxed{\begin{array}{c} x_1 \\ \hline person(x_1) \end{array}} \; ; \; P(x_1)$$

- Conceptually, we are assuming PCDRT (Haug, 2014), which has a definition of the *ant* predicate and (relatedly) a treatment of so-far-unresolved anaphora that doesn't require indexing.
- This specific assumption is not crucial, though.

Plan

What is Glue?

- A theory of the syntax/semantics interface, originally developed for LFG, and now the mainstream in LFG (Dalrymple et al., 1993, 1999).

What is Glue?

- A theory of the syntax/semantics interface, originally developed for LFG, and now the mainstream in LFG (Dalrymple et al., 1993, 1999).
- Has been applied to other frameworks: HPSG (Asudeh & Crouch, 2002), LTAG (Frank & van Genabith, 2001) and Minimalism (Gotham, 2018).

What is Glue?

- A theory of the syntax/semantics interface, originally developed for LFG, and now the mainstream in LFG (Dalrymple et al., 1993, 1999).
- Has been applied to other frameworks: HPSG (Asudeh & Crouch, 2002), LTAG (Frank & van Genabith, 2001) and Minimalism (Gotham, 2018).
- Interpretations of constituents are paired with formulae of a fragment of linear logic (Girard, 1987), and semantic composition is deduction in that logic mediated by the Curry-Howard correspondence (Howard, 1980).

What is Glue?

- A theory of the syntax/semantics interface, originally developed for LFG, and now the mainstream in LFG (Dalrymple et al., 1993, 1999).
- Has been applied to other frameworks: HPSG (Asudeh & Crouch, 2002), LTAG (Frank & van Genabith, 2001) and Minimalism (Gotham, 2018).
- Interpretations of constituents are paired with formulae of a fragment of linear logic (Girard, 1987), and semantic composition is deduction in that logic mediated by the Curry-Howard correspondence (Howard, 1980).

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)

Scope ambiguity as an example

(4) Someone sees everything.

Two interpretations:

1. There is someone who sees everything. (surface scope, $\exists > \forall$)
2. Everything is seen. (inverse scope, $\forall > \exists$)

Q: Where is the ambiguity?

Montague Grammar

(Montague, 1973; Dowty et al., 1981)

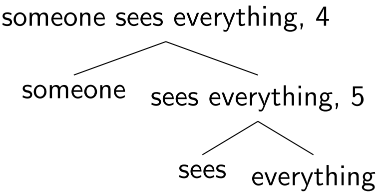
Ambiguity of syntactic derivation:

Montague Grammar

(Montague, 1973; Dowty et al., 1981)

Ambiguity of syntactic derivation:

Surface scope

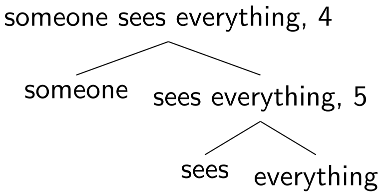


Montague Grammar

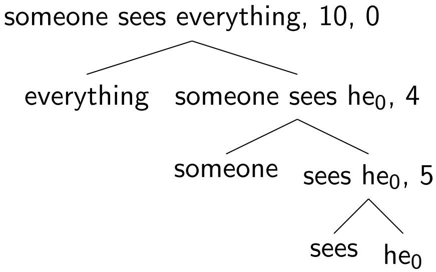
(Montague, 1973; Dowty et al., 1981)

Ambiguity of syntactic derivation:

Surface scope



Inverse scope



Mainstream Minimalism
(May, 1977, 1985; Heim & Kratzer, 1998)

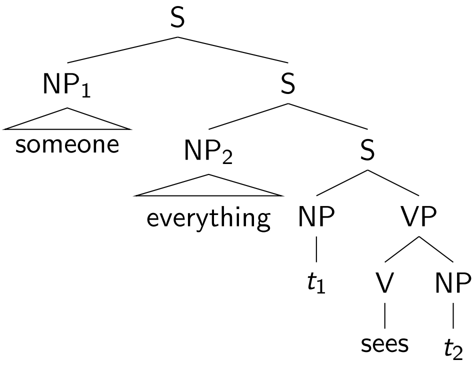
Ambiguity of syntactic structure:

Mainstream Minimalism

(May, 1977, 1985; Heim & Kratzer, 1998)

Ambiguity of syntactic structure:

Surface scope

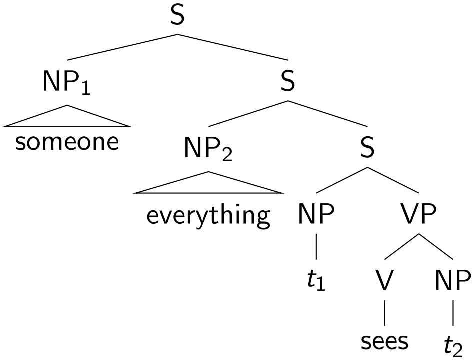


Mainstream Minimalism

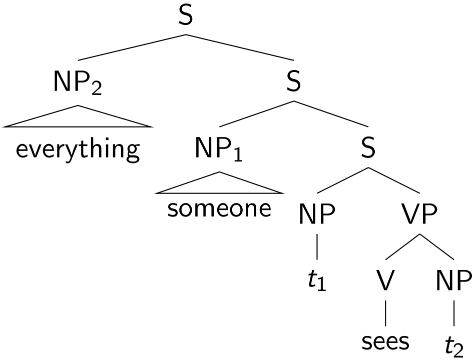
(May, 1977, 1985; Heim & Kratzer, 1998)

Ambiguity of syntactic structure:

Surface scope



Inverse scope



Another way

- The approaches just mentioned have in common is the view that syntactic structure plus lexical semantics *determines* interpretation.

Another way

- The approaches just mentioned have in common is the view that syntactic structure plus lexical semantics *determines* interpretation.
- From this it follows that if a sentence is ambiguous, such as (4), then that ambiguity must be either lexical or syntactic.

Another way

- The approaches just mentioned have in common is the view that syntactic structure plus lexical semantics *determines* interpretation.
- From this it follows that if a sentence is ambiguous, such as (4), then that ambiguity must be either lexical or syntactic.
- The Glue approach is that syntax *constrains* what can combine with what, and how.

  (to this extent there is a similarity with Cooper storage (Cooper, 1983))

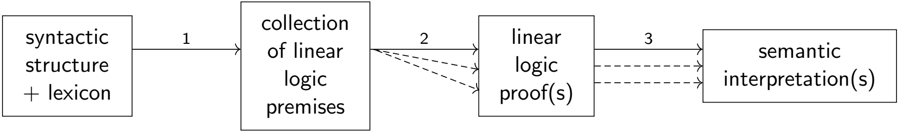
- Totally informal statement of what the constraints look like in (4):

- Totally informal statement of what the constraints look like in (4):
  - ⟦*sees*⟧ applies to A, then B, to form C.
  - ⟦*someone*⟧ applies to (something that applies to B to form C) to form C.
  - ⟦*everything*⟧ applies to (something that applies to A to form C) to form C.

- Totally informal statement of what the constraints look like in (4):
  - ⟦*sees*⟧ applies to A, then B, to form C.
  - ⟦*someone*⟧ applies to (something that applies to B to form C) to form C.
  - ⟦*everything*⟧ applies to (something that applies to A to form C) to form C.
- There's more than one way to put ⟦*someone*⟧, ⟦*sees*⟧ and ⟦*everything*⟧ together, while obeying these constraints, to form C.

- Totally informal statement of what the constraints look like in (4):
  - ⟦*sees*⟧ applies to A, then B, to form C.
  - ⟦*someone*⟧ applies to (something that applies to B to form C) to form C.
  - ⟦*everything*⟧ applies to (something that applies to A to form C) to form C.
- There's more than one way to put ⟦*someone*⟧, ⟦*sees*⟧ and ⟦*everything*⟧ together, while obeying these constraints, to form C.
- The different ways:
  - Give the different interpretations of (4).
  - Correspond to different proofs from the same premises in Linear Logic.

The syntax-semantics interface according to Glue



1. Function, given by Glue implementation
2. Relation, given by linear logic proof theory
3. Function, given by Curry-Howard correspondence

Linear logic

Linear logic is often called a 'logic of resources'(Crouch & van Genabith, 2000, 5).

Linear logic

Linear logic is often called a 'logic of resources'(Crouch & van Genabith, 2000, 5). The reason for this is that, in linear logic, for a sequent

$$\text{premise(s)} \vdash \text{conclusion}$$

to be valid, every premise in premise(s) must be 'used' exactly once.

Linear logic

Linear logic is often called a 'logic of resources'(Crouch & van Genabith, 2000, 5). The reason for this is that, in linear logic, for a sequent

$$\text{premise(s)} \vdash \text{conclusion}$$

to be valid, every premise in premise(s) must be 'used' exactly once. So for example,

$$\begin{array}{lcl} A \vdash A & \text{and} & A, A \multimap B \vdash B, \text{ but} \\ A, A \nvdash A & \text{and} & A, A \multimap (A \multimap B) \nvdash B \end{array}$$

($\multimap$ is linear implication)

Interpretation as deduction

In Glue,

Interpretation as deduction

In Glue,

- expressions of a meaning language (in this case, λ-DRT) are paired with formulae in a fragment of linear logic (the glue language)

Interpretation as deduction

In Glue,

- expressions of a meaning language (in this case, λ-DRT) are paired with formulae in a fragment of linear logic (the glue language), and
- steps of deduction carried out using those formulae correspond to operations performed on the meaning terms, according to the Curry-Howard correspondence.

Linear implication

Rules for $\multimap$	
Elimination. . .	Introduction. . .
$\dfrac{X \multimap Y \qquad X}{Y} \multimap_E$	$\dfrac{\begin{array}{c}[X]^n \\ \vdots \\ Y\end{array}}{X \multimap Y} \multimap_{I,n}$ Exactly one hypothesis must be discharged in the introduction step.

Linear implication and functional types

Rules for $\multimap$ and their images under the Curry-Howard correspondence	
Elimination. . .	Introduction. . .
$\dfrac{f : X \multimap Y \quad a : X}{f(a) : Y} \multimap_E$	$\dfrac{\begin{array}{c}[v : X]^n \\ \vdots \\ f : Y\end{array}}{\lambda v.f : X \multimap Y} \multimap_{I,n}$ Exactly one hypothesis must be discharged in the introduction step.
. . . corresponds to . . .	
. . . application.	. . . abstraction.

Linear implication and functional types

<table>
<tr><td colspan="2">Rules for $\multimap$ and their images under the Curry-Howard correspondence</td></tr>
<tr><td>Elimination...</td><td>Introduction...</td></tr>
<tr><td>$$\frac{f : X \multimap Y \quad a : X}{f(a) : Y} \multimap_E$$</td><td>$$\frac{\begin{array}{c}[v : X]^n \\ \vdots \\ f : Y\end{array}}{\lambda v.f : X \multimap Y} \multimap_{I,n}$$ Exactly one hypothesis must be discharged in the introduction step.</td></tr>
<tr><td colspan="2">...corresponds to ...</td></tr>
<tr><td>...application.</td><td>...abstraction.</td></tr>
</table>

Propositions as types:

$$type(X \multimap Y) := type(X) \rightarrow type(Y)$$

What you need from syntax

label	A	B	C
assigned to	the object argument of *sees*	the subject argument of *sees*	the sentence as a whole
	everything	*someone*	(where *someone* takes scope)
			(where *everything* takes scope)

What you need from syntax

label	A	B	C
assigned to	the object argument of *sees*	the subject argument of *sees*	the sentence as a whole
	everything	*someone*	(where *someone* takes scope)
			(where *everything* takes scope)

$\Downarrow$

$$\lambda Q.[x_1 \mid person(x_1)]; Q(x_1) : (B \multimap C) \multimap C \qquad \text{type } (e\to t)\to t$$

$$\lambda v.\lambda u.[\ \mid see(u, v)] : A \multimap (B \multimap C) \qquad \text{type } e\to(e\to t)$$

$$\lambda P.[\ \mid [x_1 \mid\ ] \Rightarrow P(x_1)] : (A \multimap C) \multimap C \qquad \text{type } (e\to t)\to t$$

Surface scope interpretation

$$\frac{[\![\mathit{sees}]\!] : A \multimap (B \multimap C) \qquad [z : A]^1}{[\![\mathit{sees}]\!](z) : B \multimap C} \multimap_E$$

$$\frac{[\![\mathit{sees}]\!](z) : B \multimap C \qquad [w : B]^2}{[\![\mathit{sees}]\!](z)(w) : C} \multimap_E$$

$$\frac{[\![\mathit{sees}]\!](z)(w) : C}{\lambda z.[\![\mathit{sees}]\!](z)(w) : A \multimap C} \multimap_{I,1}$$

$$\frac{[\![\mathit{everything}]\!] : (A \multimap C) \multimap C \qquad \lambda z.[\![\mathit{sees}]\!](z)(w) : A \multimap C}{[\![\mathit{everything}]\!](\lambda z.[\![\mathit{sees}]\!](z)(w)) : C} \multimap_E$$

$$\frac{[\![\mathit{everything}]\!](\lambda z.[\![\mathit{sees}]\!](z)(w)) : C}{\lambda w.[\![\mathit{everything}]\!](\lambda z.[\![\mathit{sees}]\!](z)(w)) : B \multimap C} \multimap_{I,2}$$

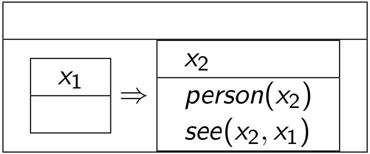
$$\frac{[\![\mathit{someone}]\!] : (B \multimap C) \multimap C \qquad \lambda w.[\![\mathit{everything}]\!](\lambda z.[\![\mathit{sees}]\!](z)(w)) : B \multimap C}{[\![\mathit{someone}]\!](\lambda w.[\![\mathit{everything}]\!](\lambda z.[\![\mathit{sees}]\!](z)(w))) : C} \multimap_E$$

$$\rightsquigarrow_\beta \begin{array}{|l|}\hline x_1 \\ \hline \mathit{person}(x_1) \\ \begin{array}{|c|}\hline x_2 \\ \hline \\ \hline\end{array} \Rightarrow \begin{array}{|c|}\hline \\ \hline \mathit{see}(x_1, x_2) \\ \hline\end{array} \\ \hline\end{array}$$

Inverse scope interpretation

$$\dfrac{\begin{array}{c}[\![everything]\!] : \\ (A \multimap C) \multimap C\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}[\![someone]\!] : \\ (B \multimap C) \multimap C\end{array} \qquad \dfrac{\begin{array}{c}[\![sees]\!] : \\ A \multimap (B \multimap C)\end{array} \quad [z : A]^1}{[\![sees]\!](z) : B \multimap C}\multimap_E}{[\![someone]\!]([\![sees]\!](z)) : C}\multimap_E}{\lambda z.[\![someone]\!]([\![sees]\!](z)) : A \multimap C}\multimap_{I,1}}{[\![everything]\!](\lambda z.[\![someone]\!]([\![sees]\!](z))) : C}\multimap_E$$

$\rightsquigarrow_\beta$



Plan

Theoretical considerations

- Dependency grammars have severe expressivity constraints
  - Unique head constraint
  - Overt token constraint

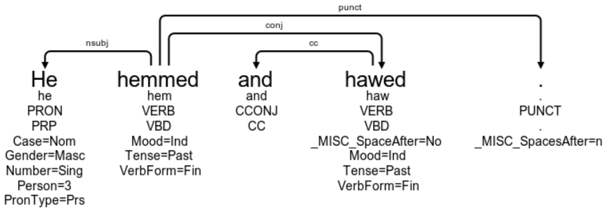
Theoretical considerations

- Dependency grammars have severe expressivity constraints
  - Unique head constraint
  - Overt token constraint
- There are also some UD-specific choices
  - No argument/adjunct distinction

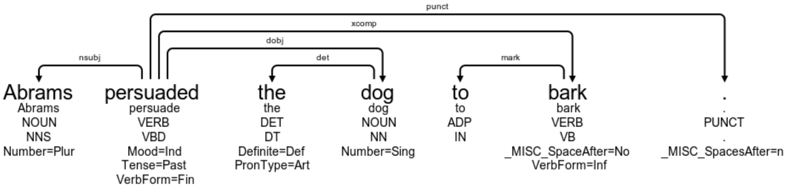
Theoretical considerations

- Dependency grammars have severe expressivity constraints
  - Unique head constraint
  - Overt token constraint
- There are also some UD-specific choices
  - No argument/adjunct distinction
- Some of this will be alleviated through enhanced dependencies but those are not yet widely available

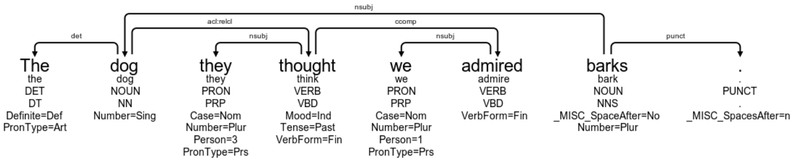
Coordination structure



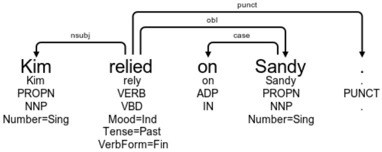
Control structure

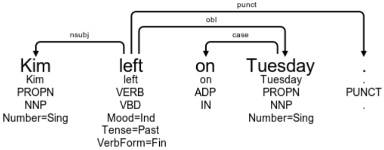


Relative clause structure



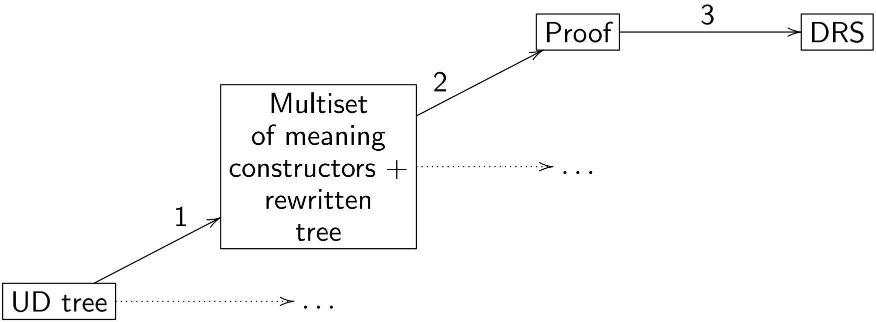
No argument/adjunct distinction





Plan

Overview


UD tree
1
Multiset of meaning constructors + rewritten tree
2
Proof
3
DRS
...
...

Overview

- Traversal of the UD tree, matching each node against a rule file
- For each matched rule, a meaning constructor is produced...
- ...and then instantiated non-deterministically, possibly rewriting the UD tree in the process
- The result is a set of pairs $\langle M, T\rangle$ where M is a multiset of meaning constructors and T is a rewritten UD tree
- Each multiset is fed into a linear logic prover (by Miltiadis Kokkonidis) and beta reduction software (from Johan Bos' Boxer)

Example

ROOT

arrived
pos=VERB
index=2

NSUBJ

Peter
pos=PROPN
index=1

$$\text{pos} = \text{PROPN} \rightarrow$$
$$\lambda P.[x|named(x, :lemma:)] \; ; \; P(x) :$$
$$(e_{\downarrow} \multimap t_{\%R}) \multimap t_{\%R}$$

Example

ROOT

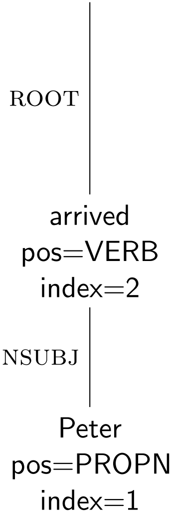
arrived
pos=VERB
index=2

NSUBJ

Peter
pos=PROPN
index=1

pos = PROPN $\rightarrow$
$\lambda P.[x|named(x, Peter)]$; $P(x)$:
$(e_1 \multimap t_2) \multimap t_2$

Example



$$\text{pos} = \text{VERB} \rightarrow$$
$$\lambda F, [e | \mathit{:lemma:}(e)]; \mathit{:DEP:}(e); F(e) :$$
$$(v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}$$

Example

ROOT

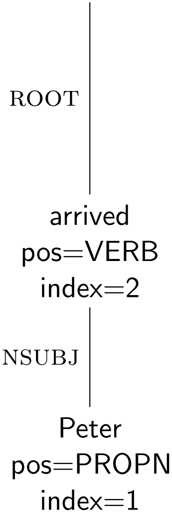
arrived
pos=VERB
index=2

NSUBJ

Peter
pos=PROPN
index=1

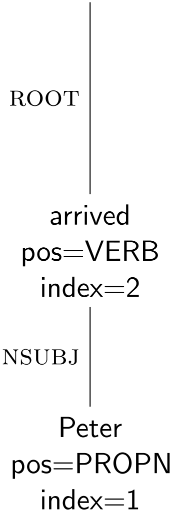
$$\text{pos} = \text{VERB} \rightarrow$$
$$\lambda x.\lambda F, [e|arrive(e), nsubj(e, x)] \; ; F(e) :$$
$$e_{\downarrow nsubj} \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}$$

Example



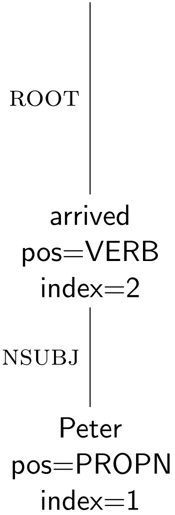
$\text{pos} = \text{VERB} \rightarrow$
$\lambda x.\lambda F, [e|arrive(e), nsubj(e,x)] \,;\, F(e) :$
$e_1 \multimap (v_2 \multimap t_2) \multimap t_2$

Example



$$\text{relation} = \text{ROOT} \rightarrow$$
$$\lambda\_.[\ |\ ] : v(\downarrow) \multimap t(\downarrow)$$

Example



$$\text{relation} = \text{ROOT} \rightarrow$$
$$\lambda\_.[\,|\,] : v_2 \multimap t_2$$

Example

ROOT

arrived
pos=VERB
index=2

NSUBJ

Peter
pos=PROPN
index=1

$$\lambda P.[x|\mathit{named}(x, \mathit{Peter})] \ ; \ P(x) : (e_1 \multimap t_2) \multimap t_2$$

$$\lambda x.\lambda F, [e|\mathit{arrive}(e), \mathit{nsubj}(e, x)] \ ; F(e) : e_1 \multimap (v_2 \multimap t_2) \multimap t_2$$

$$\lambda\_.[\ |\ ] : v_2 \multimap t_2$$

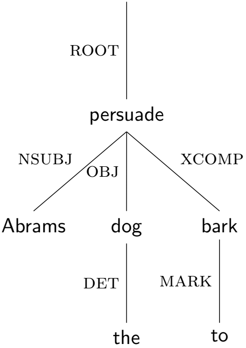
Interpretation in Glue

$$\dfrac{[\![Peter]\!] : (e_1 \multimap t_2) \multimap t_2 \qquad \dfrac{\dfrac{\dfrac{[\![arrived]\!] : e_1 \multimap (v_2 \multimap t_2) \multimap t_2 \qquad [y : e_1]^1}{[\![arrived]\!](y) : (v_2 \multimap t_2) \multimap t_2} \multimap_E \qquad [\![\texttt{root}]\!] : v_2 \multimap t_2}{[\![arrived]\!](y)([\![\texttt{root}]\!]) : t_2} \multimap_E}{\lambda y.[\![arrived]\!](y)([\![\texttt{root}]\!]) : e_1 \multimap t_2} \multimap_{I,1}}{[\![Peter]\!](\lambda y.[\![arrived]\!](y)([\![\texttt{root}]\!])) : t_2} \multimap_E$$

$$\left(\lambda P.\begin{array}{|l|}\hline x_1 \\ \hline named(x_1, Peter) \\ \hline \end{array}; P(x_1)\right) \left(\lambda y.\left(\lambda x.\lambda F.\begin{array}{|l|}\hline e_1 \\ \hline arrive(e_1) \\ nsubj(e_1, x) \\ \hline \end{array}; F(e_1)\right)(y)\left(\lambda V.\begin{array}{|l|}\hline \\ \hline \\ \hline \end{array}\right)\right)$$

$$\rightsquigarrow_\beta \begin{array}{|l|}\hline x_1 \;\; e_1 \\ \hline named(x_1, Peter) \\ arrive(e_1) \\ nsubj(e_1, x_1) \\ \hline \end{array}$$

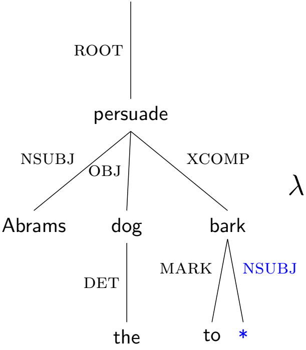
Control



$$\lambda P.\lambda y.\lambda x.\lambda F. \left[\begin{array}{l} e\ q\ {\color{blue}z} \\ \hline persuade(e) \\ controldep(e, {\color{blue}z}) \\ xcomp(e, q) \\ obj(e, y) \\ nsubj(e, x) \\ q : P({\color{blue}z})(\lambda.[\ |\ ]) \end{array}\right] ; F(e)$$

$$(e_{\downarrow\text{XCOMP}\ {\color{blue}\text{NSUBJ}}} \multimap (v_{\downarrow\text{XCOMP}} \multimap t_{\downarrow\text{XCOMP}}) \multimap t_{\downarrow\text{XCOMP}})$$
$$\multimap (e_{\downarrow\text{NSUBJ}}) \multimap (e_{\downarrow\text{OBJ}}) \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}$$

Control



$$\lambda P.\lambda y.\lambda x.\lambda F. \boxed{\begin{array}{l} e\ q\ z \\ \hline persuade(e) \\ controldep(e, z) \\ xcomp(e, q) \\ obj(e, y) \\ nsubj(e, x) \\ q : P(z)(\lambda.[\ |\ ]) \end{array}} \ ; F(e)$$

$$(e_8 \multimap (v_6 \multimap t_6) \multimap t_6) \\ \multimap e_4 \multimap e_1 \multimap (v_2 \multimap t_2) \multimap t_2$$

$$\frac{\llbracket persuade \rrbracket : ((v_6 \multimap t_6) \multimap t_6) \multimap e_4 \multimap e_1 \multimap (v_2 \multimap t_2) \multimap t_2 \qquad \llbracket bark \rrbracket : (v_6 \multimap t_6) \multimap t_6}{\llbracket persuade \rrbracket(\llbracket bark \rrbracket) : e_4 \multimap e_1 \multimap (v_2 \multimap t_2) \multimap t_2}$$

$$\frac{\llbracket persuade \rrbracket(\llbracket bark \rrbracket) : e_4 \multimap e_1 \multimap (v_2 \multimap t_2) \multimap t_2 \qquad [u : e_4]^1}{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u) : e_1 \multimap (v_2 \multimap t_2) \multimap t_2}$$

$$\frac{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u) : e_1 \multimap (v_2 \multimap t_2) \multimap t_2 \qquad [v : e_1]^2}{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v) : (v_2 \multimap t_2) \multimap t_2}$$

$$\frac{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v) : (v_2 \multimap t_2) \multimap t_2 \qquad \llbracket \texttt{root} \rrbracket : v_2 \multimap t_2}{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket) : t_2}$$

$$\frac{\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket) : t_2}{\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket) : e_4 \multimap t_2}\ 1$$

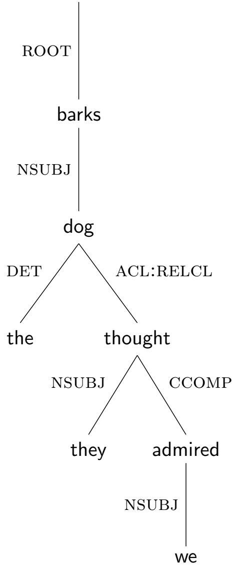
$$\frac{\begin{matrix}\vdots \\ \llbracket the \rrbracket(\llbracket dog \rrbracket) : (e_4 \multimap t_2) \multimap t_2\end{matrix} \qquad \lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket) : e_4 \multimap t_2}{\llbracket the \rrbracket(\llbracket dog \rrbracket)(\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket)) : t_2}$$

$$\frac{\llbracket the \rrbracket(\llbracket dog \rrbracket)(\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket)) : t_2}{\lambda v.\llbracket the \rrbracket(\llbracket dog \rrbracket)(\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket)) : e_1 \multimap t_2}\ 2$$

$$\frac{\llbracket Abrams \rrbracket : (e_1 \multimap t_2) \multimap t_2 \qquad \lambda v.\llbracket the \rrbracket(\llbracket dog \rrbracket)(\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket)) : e_1 \multimap t_2}{\llbracket Abrams \rrbracket(\lambda v.\llbracket the \rrbracket(\llbracket dog \rrbracket)(\lambda u.\llbracket persuade \rrbracket(\llbracket bark \rrbracket)(u)(v)(\llbracket \texttt{root} \rrbracket))) : t_2}$$

$$\rightsquigarrow_\beta \begin{array}{|l|}\hline x_1 \;\; x_2 \;\; x_3 \;\; e_1 \;\; p_1 \\ \hline named(x_1, abrams), ant(x_2) \\ \partial(dog(x_2)), persuade(e_1) \\ nsubj(e_1, x_1), obj(e_1, x_2) \\ controldep(e_1, x_3), xcomp(e_1, p_1) \\ p_1 : \begin{array}{|l|}\hline e_2 \\ \hline bark(e_2) \\ nsubj(e_2, x_3) \\ \hline \end{array} \\ \hline \end{array}$$

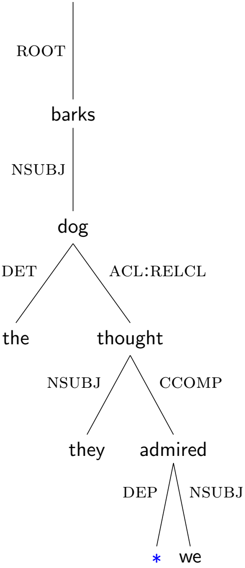
Relative clauses



$$\lambda P.\lambda V.\lambda x.P(x);V(x)(\lambda\_.[\ |\ ])$$

$$(e_{\uparrow} \multimap t_{\uparrow}) \multimap$$
$$(e_{\downarrow dep^{*}\ dep\{PType=Rel\}} \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}) \multimap$$
$$e_{\uparrow} \multimap t_{\uparrow}$$

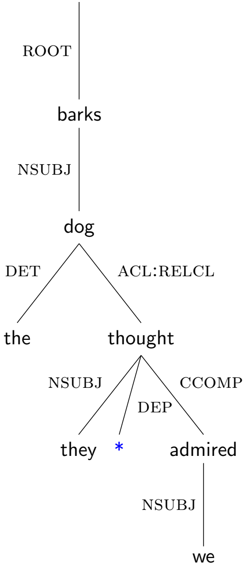
Relative clauses



$$\lambda P.\lambda V.\lambda x.P(x); V(x)(\lambda\_.[\ |\ ])$$

$(e_2 \multimap t_2) \multimap$
$(e_9 \multimap (v_4 \multimap t_4) \multimap t_4) \multimap$
$e_2 \multimap t_2$

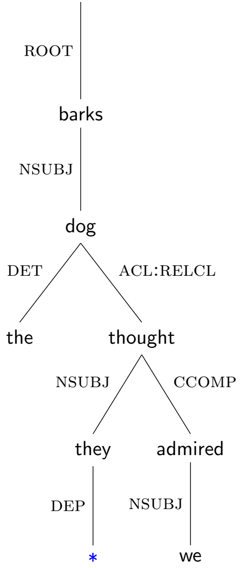
Relative clauses



$$\lambda P.\lambda V.\lambda x.P(x); V(x)(\lambda\_.[\;|\;])$$

$$(e_2 \multimap t_2) \multimap$$
$$(e_9 \multimap (v_4 \multimap t_4) \multimap t_4) \multimap$$
$$e_2 \multimap t_2$$

Relative clauses



$$\lambda P.\lambda V.\lambda x.P(x); V(x)(\lambda\_.[\ |\ ])$$

$$(e_2 \multimap t_2) \multimap$$
$$(e_{\color{blue}9} \multimap (v_4 \multimap t_4) \multimap t_4) \multimap$$
$$e_2 \multimap t_2$$

Plan

Discussion of output

$x_1 \; e_1$
$named(x_1, Peter)$
$arrive(e_1)$
$nsubj(e_1, x_1)$

- What kind of θ-role is 'nsubj'?
  - A syntactic name, lifted from the arc label.
  - In and of itself, uninformative.

Discussion of output

x_1 e_1
$named(x_1, Peter)$ $arrive(e_1)$ $nsubj(e_1, x_1)$

- What kind of θ-role is 'nsubj'?
  - A syntactic name, lifted from the arc label.
  - In and of itself, uninformative.
- What we have in the DRS above is as much information as can be extracted from the UD tree alone, without lexical knowledge.
- Lexical knowledge in the form of meaning postulates such as (5) can be harnessed to further specify the meaning representation.

(5) $\forall e \forall x((arrive(e) \wedge nsubj(e, x)) \rightarrow theme(e, x))$

Discussion of output

x_1 e_1
$named(x_1, Peter)$
$arrive(e_1)$
$theme(e_1, x_1)$

- What kind of θ-role is ‘nsubj’?
  - A syntactic name, lifted from the arc label.
  - In and of itself, uninformative.
- What we have in the DRS above is as much information as can be extracted from the UD tree alone, without lexical knowledge.
- Lexical knowledge in the form of meaning postulates such as (5) can be harnessed to further specify the meaning representation.

(5) $$\forall e \forall x ((arrive(e) \wedge nsubj(e, x)) \rightarrow theme(e, x))$$

$$
\begin{array}{|l|}
\hline
x_1 \;\; x_2 \;\; x_3 \;\; e_1 \;\; p_1 \\
\hline
\ldots \\
persuade(e_1), obj(e_1, x_2), controldep(e_1, x_3), xcomp(e_1, p_1) \\
p_1 : \begin{array}{|l|}
\hline
e_2 \\
\hline
\ldots, nsubj(e_2, x_3) \\
\hline
\end{array} \\
\hline
\end{array}
$$

- The *persuade* + `xcomp` meaning constructor has
  - introduced an *xcomp* relation between the persuading event e_1 and the proposition p_1 that there is a barking event e_2, and
  - introduced an individual x_3 as the *nsubj* of e_2 and the *controldep* of e_1.

$x_1 \; x_2 \; x_3 \; e_1 \; p_1$
...
$persuade(e_1), obj(e_1, x_2), controldep(e_1, x_3), xcomp(e_1, p_1)$
p_1 : [e_2 | $\ldots, nsubj(e_2, x_3)$]

- The *persuade* + `xcomp` meaning constructor has
  - introduced an *xcomp* relation between the persuading event e_1 and the proposition p_1 that there is a barking event e_2, and
  - introduced an individual x_3 as the *nsubj* of e_2 and the *controldep* of e_1.
- But the information that *persuade* is an object control verb can again be encoded in a meaning postulate:

$$\forall e \forall x ((persuade(e) \wedge controldep(e, x)) \rightarrow obj(e, x))$$

$x_1 \; x_2 \; x_3 \; e_1 \; p_1$
$\ldots$ $persuade(e_1), obj(e_1, x_2), obj(e_1, x_3), xcomp(e_1, p_1)$ $p_1 : \begin{array}{\|l\|} \hline e_2 \\ \hline \ldots, nsubj(e_2, x_3) \\ \hline \end{array}$

- The *persuade* + `xcomp` meaning constructor has
  - introduced an *xcomp* relation between the persuading event e_1 and the proposition p_1 that there is a barking event e_2, and
  - introduced an individual x_3 as the *nsubj* of e_2 and the *controldep* of e_1.
- But the information that *persuade* is an object control verb can again be encoded in a meaning postulate:

$$\forall e \forall x((persuade(e) \wedge controldep(e, x)) \rightarrow obj(e, x))$$

x_1 x_2 x_3 e_1 p_1
$\ldots$
$persuade(e_1), obj(e_1, x_2), obj(e_1, x_3), xcomp(e_1, p_1)$
p_1 : [e_2 | $\ldots, nsubj(e_2, x_3)$]

- The *persuade* + `xcomp` meaning constructor has
    - introduced an *xcomp* relation between the persuading event e_1 and the proposition p_1 that there is a barking event e_2, and
    - introduced an individual x_3 as the *nsubj* of e_2 and the *controldep* of e_1.
- But the information that *persuade* is an object control verb can again be encoded in a meaning postulate:

  $$\forall e \forall x ((persuade(e) \wedge controldep(e, x)) \rightarrow obj(e, x))$$

- With thematic uniqueness, we get $x_2 = x_3$ in this case.
- Blurs the distinction between lexical syntax and semantics.

VP/Sentence coordination: He hemmed and hawed

x_1 e_2 e_3
$pron.he(x_1)$ $hem(e_2)$ $nsubj(e_2, x_1)$ $haw(e_3)$

- No way to distinguish V/VP/S coordination in DG because of the overt token constraint
- No argument sharing because of the unique head constraint

NP Coordination: Abrams and/or Browne danced

$e_1 \; x_2 \; x_3 \; x_4$

$dance(e_1)$
$nsubj(e_{,}x_2)$
$named(x_3, browne)$
$named(x_4, abrams)$
$x_3 \sqsubseteq x_2$
$x_4 \sqsubseteq x_2$

$e_1 \; x_2 \; x_3 \; x_4$

$dance(e_1)$
$nsubj(e_{,}x_2)$
$named(x_3, browne)$
$named(x_4, abrams)$

$[\; x_3 \sqsubseteq x_2 \;] \vee [\; x_4 \sqsubseteq x_2 \;]$

Argument/adjunct distinction

e_1 x_2 x_3	e_1 x_2 x_3
$rely(e_1)$ $named(x_2, kim)$ $named(x_3, sandy)$ $on(x_3, e_1)$	$leave(e_1)$ $named(x_2, kim)$ $named(x_3, tuesday)$ $on(x_3, e_1)$

- Again, we will have to rely on meaning postulates to resolve the *on* relation to a thematic role in one case and a temporal relation in the other

Evaluation

- What we have so far is a proof of concept tested on carefully crafted examples
  - application of LFG techniques (functional uncertainties) to enrich underspecified UD syntax
  - application of glue semantics to dependency structures

Evaluation

- What we have so far is a proof of concept tested on carefully crafted examples
  - application of LFG techniques (functional uncertainties) to enrich underspecified UD syntax
  - application of glue semantics to dependency structures
- Very far from something practically useful
  - Basic coverage of UD relations except `vocative`, `dislocated`, `clf`, `list`, `parataxis`, `orphan`
  - Little or no work on interactions, special constructions, real data noise

Pros and cons of glue semantics

- No need for binarization
- Flexible approach to scoping yield different readings
- Hard to restrict unwanted/non-existing scopings
- Computing lots of uninteresting scope differences

Unwanted scopings

$$\lambda F. \begin{array}{|l|}\hline e \\ \hline \mathit{arrive}(e) \\ \hline \end{array} ; F(e) : (v_1 \multimap t_1) \multimap t_1$$

$$\lambda\_. \begin{array}{|l|}\hline \\ \hline \\ \hline \end{array} : v_1 \multimap t_1$$

It is clear which DRS sentence-level operators (negation, auxiliaries etc.) should target!

- Modalities in the linear logic
- Different types for the two DRSs

Efficient scoping

- Two parameters:
  - level of scope
  - order of combination of quantifiers at each level
- We currently naively compute everything with a light-weight prover $\rightarrow$ obvious performance problems
- Disallow intermediate scopings?
- Structure sharing across derivations (building on work in an LFG context)

Conclusions

- Theoretical achievement: application of glue to dependency grammar

Conclusions

- Theoretical achievement: application of glue to dependency grammar
- Practical achievement: an interesting proof of concept

Conclusions

- Theoretical achievement: application of glue to dependency grammar
- Practical achievement: an interesting proof of concept
- But lots of work remains
  - Support for partial proofs
  - Axiomatization of lexical knowledge
  - Ambiguity management

References I

Asudeh, Ash & Richard Crouch. 2002. Glue Semantics for HPSG. In Frank van Eynde, Lars Hellan & Dorothee Beermann (eds.), *Proceedings of the 8th international HPSG conference*, Stanford, CA: CSLI Publications.

Bos, Johan. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 conference on semantics in text processing* STEP '08, 277–286. Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1626481.1626503.

Cooper, Robin. 1983. *Quantification and syntactic theory* (Studies in Linguistics and Philosophy 21). Dordrecht: D. Reidel.

Crouch, Richard & Josef van Genabith. 2000. Linear logic for linguists. ESSLLI 2000 course notes.

References II

Dalrymple, Mary, Vineet Gupta, John Lamping & Vijay Saraswat. 1999. Relating resource-based semantics to categorial semantics. In Mary Dalrymple (ed.), *Semantics and syntax in Lexical Functional Grammar*, 261–280. Cambridge, MA: MIT Press.

Dalrymple, Mary, John Lamping & Vijay Saraswat. 1993. LFG semantics via constraints. In Steven Krauwer, Michael Moortgat & Louis des Tombe (eds.), *EACL 1993*, 97–105.

Dowty, David R., Robert E. Wall & Stanley Peters. 1981. *Introduction to Montague semantics*. Dordrecht: D. Reidel.

Frank, Anette & Josef van Genabith. 2001. GlueTag: Linear logic-based semantics for LTAG—and what it teaches us about LFG and LTAG. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG01 conference*, Stanford, CA: CSLI Publications.

References III

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science* 50(1). 1–101. doi:10.1016/0304-3975(87)90045-4.

Gotham, Matthew. 2018. Making Logical Form type-logical. *Linguistics and Philosophy* doi:10.1007/s10988-018-9229-z. In press.

Haug, Dag Trygve Truslew. 2014. Partial dynamic semantics for anaphora. *Journal of Semantics* 31. 457–511.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar* (Blackwell Textbooks in Linguistics 13). Oxford: Wiley-Blackwell.

Howard, W.A. 1980. The formulae-as-types notion of construction. In J.P. Seldin & J.R. Hindley (eds.), *To H.B. Curry*, 479–490. New York: Academic Press.

May, Robert. 1977. *The grammar of quantification*: Massachusetts Institute of Technology dissertation.

References IV

May, Robert. 1985. *Logical form* (Linguistic Inquiry Monographs 12). Cambridge, MA: MIT Press.

Montague, Richard. 1973. The proper treatment of quantification in ordinary English. In Patrick Suppes, Julius Moravcsik & Jaakko Hintikka (eds.), *Approaches to natural language*, 221–242. Dordrecht: D. Reidel.