Universal Typed Semantic Parsing

Siva Reddy

joint work with
Matt Lamm, Sebastian Schuster, Christopher D. Manning

UDepLambda is based on work with
Oscar Täckström, Tom Kwiatkowski, Dipanjan Das, Slav Petrov,
Michael Collins, Mark Steedman, Mirella Lapata

Stanford University

Dependency Tree to Semantics

I'VE CREATED A FLOW CHART OF OUR FLOOR
© 1994 United Feature Syndicate, Inc.
MEANING WHAT EXACTLY?
9-2

Dependencies lack a formal theory of semantics

Existing Syntax Semantics interfaces

CCG [Steedman, 2000; Bos et al., 2004]

HPSG [Copestake et al., 2001]

LFG [Dalrymple et al., 1995]

TAG [Joshi et al., 1995]

CCG

$$\frac{\text{Disney}}{NP} \qquad \frac{\text{acquired}}{S\backslash NP/NP} \qquad \frac{\text{Pixar}}{NP}$$

CCG

$$\begin{array}{ccc}
\text{Disney} & \text{acquired} & \text{Pixar} \\
\overline{NP} & \overline{S\backslash NP/NP} & \overline{NP} \\
\text{Disney} & \lambda y \lambda x \lambda e.\ \text{acquired}(e) & \text{Pixar} \\
 & \wedge\ \text{arg}_1(e, x) & \\
 & \wedge\ \text{arg}_2(e, y) &
\end{array}$$

CCG

Lambda Calculus

$$(\lambda x.M)N = M[x := N]$$

$$\begin{aligned} sum(2,3) &= (\lambda x \lambda y.(+\ \ x\ \ \ y))(2)(3) \\ &= (\lambda y.(+\ \ 2\ \ \ y))(3) \\ &= (+\ \ 2\ \ \ 3) \\ &= 5 \end{aligned}$$

$$\begin{aligned} \textbf{TYPE}[sum] &= int \rightarrow int \rightarrow int \\ sum(4, sum(2,3)) &= 9 \end{aligned}$$

CCG

$$\begin{array}{ccc}
\dfrac{\text{Disney}}{NP} & \dfrac{\text{acquired}}{S\backslash NP/NP} & \dfrac{\text{Pixar}}{NP} \\[1em]
\text{Disney} & \begin{array}{c}\lambda y \lambda x \lambda e.\ \text{acquired}(e) \\ \wedge\ \text{arg}_1(e,x) \\ \wedge\ \text{arg}_2(e,y)\end{array} & \text{Pixar}
\end{array}$$

CCG

$$
\begin{array}{ccc}
\text{Disney} & \text{acquired} & \text{Pixar} \\
\hline
NP & S\backslash NP/NP & NP \\
\text{Disney} & \begin{array}{l}\lambda y\lambda x\lambda e.\ \text{acquired}(e) \\ \quad \wedge\ \text{arg}_1(e,x) \\ \quad \wedge\ \text{arg}_2(e,y)\end{array} & \text{Pixar} \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} \quad >
\end{array}
$$

CCG

$$\begin{array}{cccl}
\text{Disney} & \text{acquired} & \text{Pixar} & \\
\hline
NP & S\backslash NP/NP & NP & \\
 & & & \\
\text{Disney} & \lambda y\lambda x\lambda e.\ \text{acquired}(e) & \text{Pixar} & \\
 & \wedge\ \text{arg}_1(e,x) & & \\
 & \wedge\ \text{arg}_2(e,y) & & \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} & > \\
 & & & \\
 & \multicolumn{2}{c}{\lambda x\lambda e.\ \text{acquired}(e)} & \\
 & \multicolumn{2}{c}{\wedge\ \text{arg}_1(e,x) \wedge\ \text{arg}_2(e,\text{Pixar})} &
\end{array}$$

CCG

$$\begin{array}{cccl}
\text{Disney} & \text{acquired} & \text{Pixar} & \\
\cline{1-1}\cline{2-2}\cline{3-3}
NP & S\backslash NP/NP & NP & \\
\text{Disney} & \lambda y\lambda x\lambda e.\ \text{acquired}(e) & \text{Pixar} & \\
 & \wedge\ \text{arg}_1(e,x) & & \\
 & \wedge\ \text{arg}_2(e,y) & & \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} & > \\
 & \multicolumn{2}{c}{\lambda x\lambda e.\ \text{acquired}(e)} & \\
 & \multicolumn{2}{c}{\wedge\ \text{arg}_1(e,x) \wedge\ \text{arg}_2(e,\text{Pixar})} & \\
\cline{1-3}
\multicolumn{3}{c}{S} & < \\
\multicolumn{3}{c}{\lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e,\text{Disney}) \wedge \text{arg}_2(e,\text{Pixar})} &
\end{array}$$

CCG

$$
\begin{array}{cccc}
\text{Disney} & \text{acquired} & \text{Pixar} & \\
\cline{1-1}\cline{2-2}\cline{3-3}
NP & S\backslash NP/NP & NP & \\
\text{Disney} & \lambda y\lambda x\lambda e.\ \text{acquired}(e) & \text{Pixar} & \\
 & \wedge\ \text{arg}_1(e,x) & & \\
 & \wedge\ \text{arg}_2(e,y) & & \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} & > \\
 & \multicolumn{2}{c}{\lambda x\lambda e.\ \text{acquired}(e)} & \\
 & \multicolumn{2}{c}{\wedge\ \text{arg}_1(e,x) \wedge\ \text{arg}_2(e,\text{Pixar})} & \\
\cline{1-3}
\multicolumn{3}{c}{S} & < \\
\multicolumn{3}{c}{\lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e,\text{Disney}) \wedge \text{arg}_2(e,\text{Pixar})} &
\end{array}
$$

Typing and Combinator Rules allow
Synchronous Syntax-Semantics interface

Dependency Tree to Semantics

Principle of Compositionality: the semantics of a complex expression is determined by the semantics of its constituent expressions and the rules used to combine them

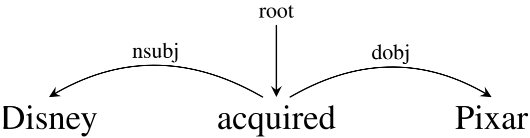
Complex expression is the dependency tree

Constituent expressions are subtrees

Rules are the dependency labels

Composition Order

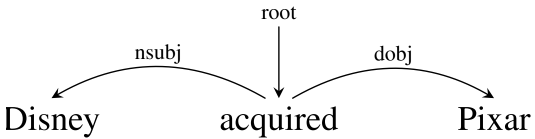
Binarization



$$\lambda z.\exists xy.\text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge$$
$$\text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)$$

Composition Order

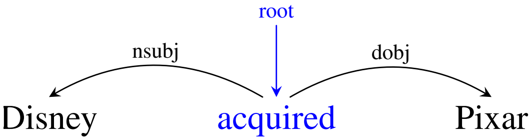
Binarization



Dependency labels drive the composition

Composition Order

Binarization

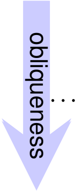


root

nsubj

dobj

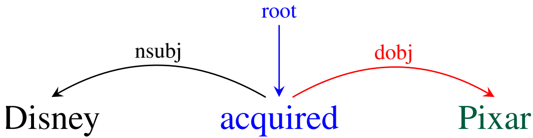
Disney acquired Pixar

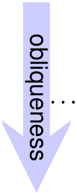


$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

Composition Order

Binarization



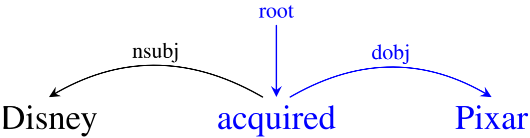


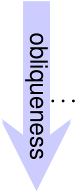
$\ldots > \text{dobj} > \cdots > \text{nsubj} > \ldots$

(dobj acquired Pixar)

Composition Order

Binarization



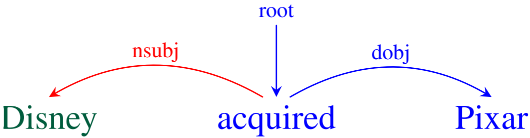


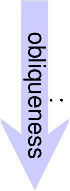
$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

(dobj acquired Pixar)

Composition Order

Binarization



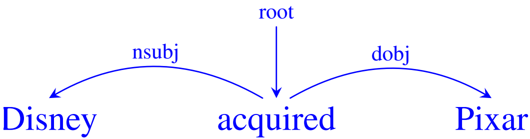


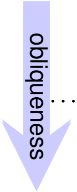
$\ldots > \text{dobj} > \cdots > \text{nsubj} > \ldots$

(nsubj (dobj acquired Pixar) Disney)

Composition Order

Binarization



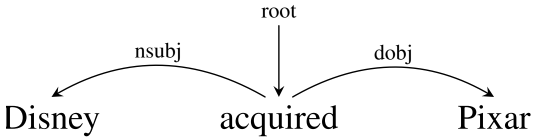


$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

(nsubj (dobj acquired Pixar) Disney)

Composition Order

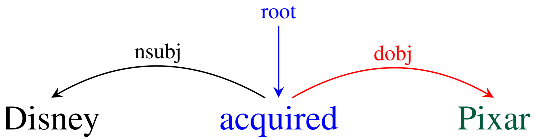
Binarization



(nsubj (dobj acquired Pixar) Disney)

$$\lambda z.\exists xy.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{Disney}(x_a) \wedge \\ \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)$$

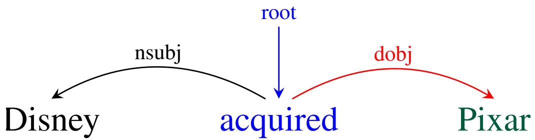
Substitution



Lambda Calculus Basic Types

- Individuals: **Ind** (also denoted by $._a$)
- Events: **Event** (also denoted by $._e$)
- Truth values: **Bool**

Substitution

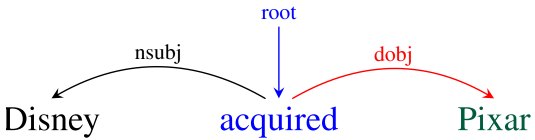


Lambda Expression for words

$$VERB \Rightarrow \lambda x.\,\mathsf{word}(x_e), e.g., \mathsf{acquired} \Rightarrow \lambda x.\,\mathsf{acquired}(x_e)$$

$$PROPN \Rightarrow \lambda x.\,\mathsf{word}(x_a), e.g., \mathsf{Pixar} \Rightarrow \lambda x.\,\mathsf{Pixar}(x_a)$$

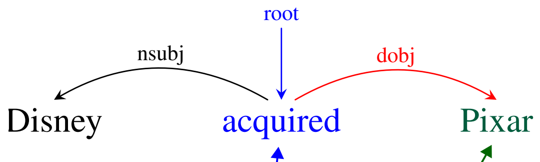
Substitution



Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda \mathbf{f}\ \lambda \mathbf{g}\ \lambda \mathbf{z}\ .\ \exists \mathbf{x}\ .\ \mathbf{f}(\mathbf{z}) \quad \wedge \quad \mathbf{g}(\mathbf{x}) \quad \wedge \quad \mathbf{arg_2}(\mathbf{z_e}, \mathbf{x_a})$$

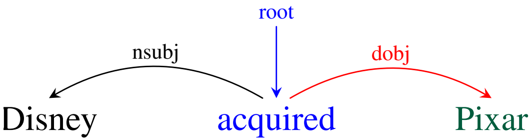
Substitution



Lambda Expression for dependency labels

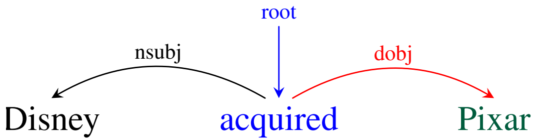
$$\text{dobj} \Rightarrow \lambda \mathbf{f}\ \lambda \mathbf{g}\ \lambda \mathbf{z}\ .\ \exists \mathbf{x}\ .\ \mathbf{f}(\mathbf{z}) \ \wedge\ \mathbf{g}(\mathbf{x}) \ \wedge\ \mathbf{arg_2}(\mathbf{z_e}, \mathbf{x_a})$$

Composition



(**dobj**	**acquired**	**Pixar**)
$\lambda f \lambda g \lambda z . \exists y .$ $f(z) \wedge g(y) \wedge$ $\mathrm{arg}_2(z_e, y_a)$	$\lambda z . \mathrm{acquired}(z_e)$	$\lambda y . \mathrm{Pixar}(y_a)$

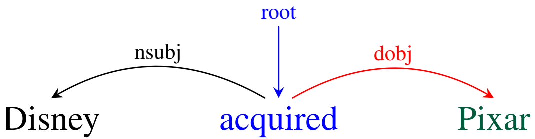
Composition



($\textbf{dobj}$	$\textbf{acquired}$	$\textbf{Pixar}$)
$\lambda f \lambda g \lambda z . \exists y . f(z) \wedge g(y) \wedge \text{arg}_2(z_e, y_a)$	$\lambda z . \text{acquired}(z_e)$	$\lambda y . \text{Pixar}(y_a)$

$$\lambda g \lambda z . \ \exists y . \ \text{acquired}(z_e) \wedge g(y) \wedge \text{arg}_2(z_e, y_a)$$

Composition

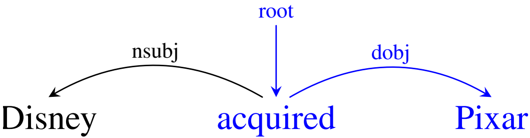


$$\begin{array}{ccc} (\textbf{dobj} & \textbf{acquired} & \textbf{Pixar}) \\ \lambda f \lambda g \lambda z . \exists y . & \lambda z.\text{acquired}(z_e) & \lambda y.\text{Pixar}(y_a) \\ f(z) \wedge g(y) \wedge & & \\ \text{arg}_2(z_e, y_a) & & \end{array}$$

$$\frac{}{\lambda g \lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge g(y) \wedge \text{arg}_2(z_e, y_a)}$$

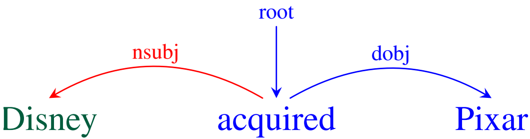
$$\frac{}{\lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{arg}_2(z_e, y_a)}$$

Composition



$$\frac{(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})}{\lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ \wedge\ \text{arg}_2(z_e, y_a)}$$

Composition



root

nsubj

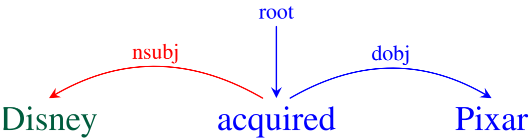
dobj

Disney acquired Pixar

(**nsubj** (**dobj** **acquired** **Pixar**) **Disney**)

(**nsubj**	(**dobj** **acquired** **Pixar**)	**Disney**)
$\lambda f \lambda g \lambda z.\ \exists x.$ $f(z) \wedge g(x) \wedge$ $\mathrm{arg}_1(z_e, x_a)$	$\overline{}$ $\lambda z.\ \exists y.\ \mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a)$ $\wedge\ \mathrm{arg}_2(z_e, y_a)$	$\lambda x.\mathrm{Disney}(x_a)$

Composition



root

nsubj

dobj

Disney acquired Pixar

(nsubj **(dobj** **acquired** **Pixar)** **Disney)**

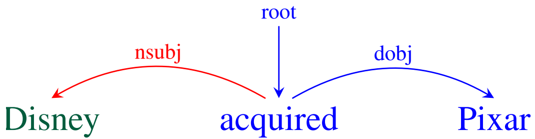
$\lambda f \lambda g \lambda z.\ \exists x.$
$f(z) \wedge g(x) \wedge$
$\mathrm{arg}_1(z_e, x_a)$

$$\frac{}{\lambda z.\ \exists y.\ \mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

$\lambda x.\mathrm{Disney}(x_a)$

$$\frac{}{\lambda g \lambda z.\exists xy.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{g}(x) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

Composition



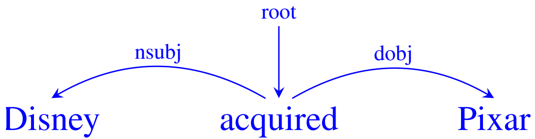
(**nsubj** (**dobj** **acquired** **Pixar**) **Disney**)

$$\lambda f \lambda g \lambda z.\ \exists x.\ f(z) \wedge g(x) \wedge \mathrm{arg}_1(z_e, x_a) \qquad \frac{\quad}{\lambda z.\ \exists y.\ \mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{arg}_2(z_e, y_a)} \qquad \lambda x.\mathrm{Disney}(x_a)$$

$$\frac{\quad}{\lambda g \lambda z.\exists xy.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{g}(x) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

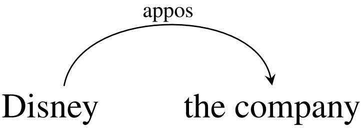
$$\frac{\quad}{\lambda z.\exists xy.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{Disney}(x_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

Composition



$$\frac{(\textbf{nsubj} \quad (\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar}) \quad \textbf{Disney})}{\lambda z.\exists xy.\text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)}$$

Composition

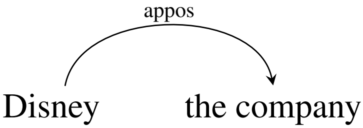


appos

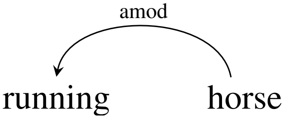
Disney the company

$$\textit{appos} = \lambda f \lambda g \lambda x . f(x) \wedge g(x)$$

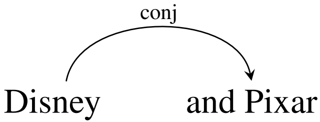
Composition



$$\mathit{appos} = \lambda f \lambda g \lambda x. f(x) \wedge g(x)$$



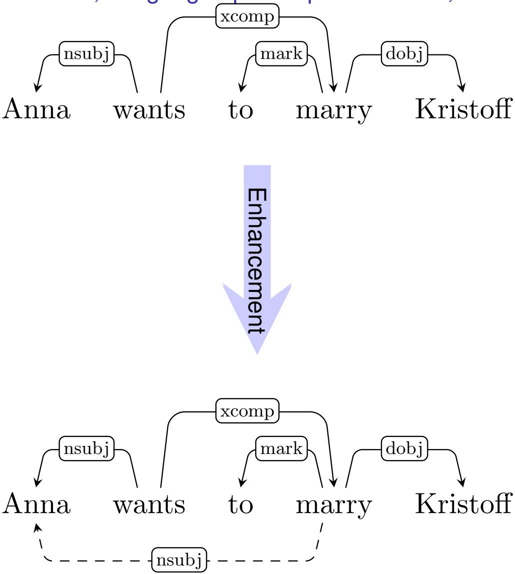
$$\mathit{amod} = \lambda f \lambda g \lambda x. \exists z. f(x) \wedge g(z) \wedge \mathrm{amod}^{i}(z_e, x_a)$$



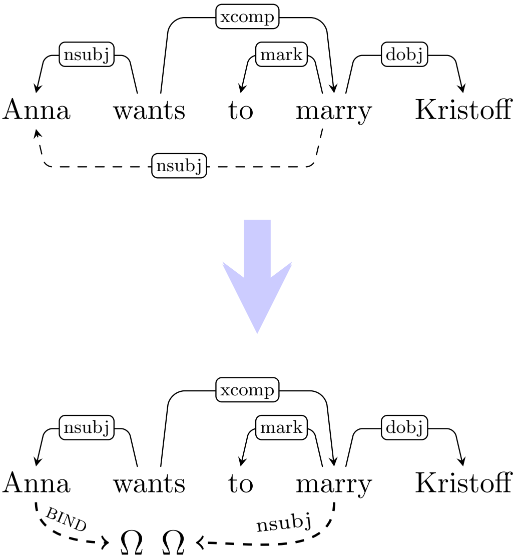
$$\mathit{conj} = \lambda f \lambda g \lambda z. \exists xy. f(x) \wedge g(y) \wedge \mathrm{coord}(z, x, y)$$

Enhancement

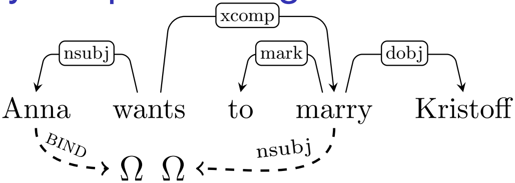
Long distance, Coordination, Language-specific phenomenon, Quantifiers



Dependency Graphs to Logical Forms



Dependency Graphs to Logical Forms



Substitution Expressions

$$
\begin{aligned}
\text{BIND} &= \lambda f \lambda g \lambda x . f(x) \land g(x) \\
\text{xcomp} &= \lambda f g x . \exists y . f(x) \land g(y) \land \text{xcomp}(x_e, y_e) \\
\omega &= \lambda x . \text{EQ}(x, \omega)
\end{aligned}
$$

Final Expression:

$$
\begin{aligned}
\lambda z . \exists x y w . &\ \text{wants}(z_e) \land \text{Anna}(x_a) \land \text{arg}_1(z_e, x_a) \\
&\land \text{marry}(y_e) \land \text{xcomp}(z_e, y_e) \land \text{arg}_1(y_e, x_a) \\
&\land \text{Kristoff}(w_a) \land \text{arg}_2(y_e, w_a) \, .
\end{aligned}
$$

Quantifiers and Negation Scope

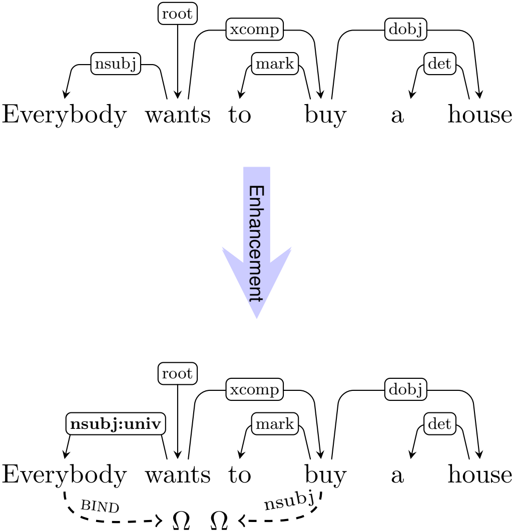
(Fancellu et al. 2017, Reddy et al. 2017)

Higher-order type system

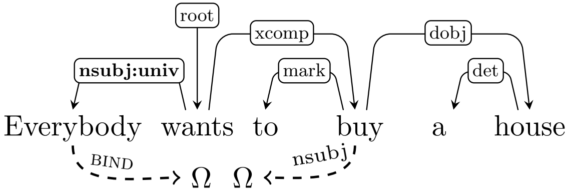
Fine-grained dependency labels

Quantifiers and Negation Scope

Fancellu et al. 2017, Reddy et al. 2017



Quantifiers and Negation Scope



Type System

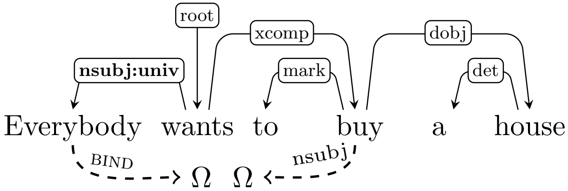
everybody $= \lambda x.\text{everybody}(x_a)$ [Old Type]

$= \lambda f.\forall x.\,\text{person}(x) \rightarrow f(x)$ [New Type]

wants $= \lambda x.\,\text{wants}(x_e)$ [Old Type]

$= \lambda f.\exists x.\,\text{wants}(x_e) \wedge f(x)$ [New Type]

Quantifiers and Negation Scope



Type System

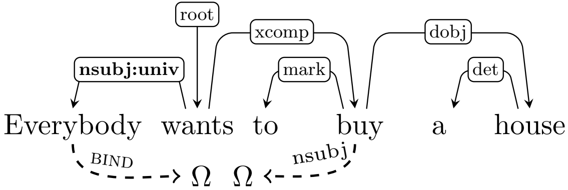
nsubj = $\lambda fgx.\exists y.f(x) \wedge g(y) \wedge \mathrm{arg}_1(x_e, y_a)$ [Old]

nsubj:univ = $\lambda PQf.Q(\lambda y.P(\lambda x.f(x) \wedge \mathrm{arg}_1(x_e, y_a)))$ [New]

dobj = $\lambda fgx.\exists y.f(x) \wedge g(y) \wedge \mathrm{arg}_2(x_e, y_a)$ [Old]

= $\lambda PQf.P(\lambda x.f(x) \wedge Q(\lambda y.\mathrm{arg}_2(x_e, y_a)))$ [New]

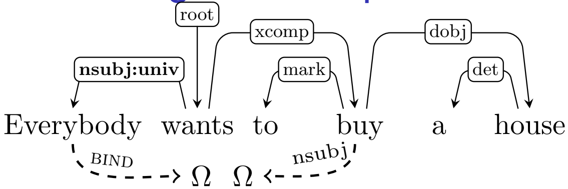
Quantifiers and Negation Scope



Type System

nsubj	$= \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \arg_1(x_e, y_a)$	[Old]
nsubj:univ	$= \lambda PQf.Q(\lambda y.P(\lambda x.f(x) \wedge \arg_1(x_e, y_a)))$	[New]
dobj	$= \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \arg_2(x_e, y_a)$	[Old]
	$= \lambda PQf.P(\lambda x.f(x) \wedge Q(\lambda y.\arg_2(x_e, y_a)))$	[New]

Quantifiers and Negation Scope



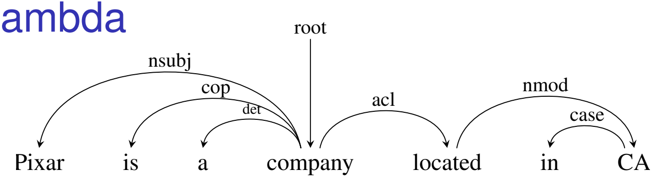
Old Expression:

$$(3)\ \lambda z. \exists xyw. \mathrm{wants}(z_e) \wedge \mathrm{everybody}(x_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{arg}_1(y_e, x_a) \wedge \mathrm{arg}_1(x_e, y_a) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_2(y_e, w_a)\,.$$

New Expression:

$$(6)\ \lambda f. \forall x\ . \mathrm{person}(x_a) \rightarrow [\exists zyw. f(z) \wedge \mathrm{wants}(z_e) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, w_a)]\,.$$

UDepLambda



$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

$$\begin{array}{ccccccccc}
\ldots & (\mathit{acl} & \text{company} & (\mathit{nmod} & \text{located} & (\mathit{case} & \text{CA} & \text{in}) &))\ldots \\
 & \lambda fgx.\exists z. & \lambda x.\text{compay}(x_a) & \lambda fgz.\exists x. & \lambda x.\text{located}(x_e) & \lambda fgx.f(x) & \lambda x.\text{CA}(x_a) & \lambda x.\text{empty}(x) & \\ \cline{6-8}
 & f(x)\wedge g(z)\wedge & & f(z)\wedge g(x) & & \multicolumn{3}{c}{\lambda x.\ \text{CA}(x_a)} & \\
 & \text{arg}_2(z_e, x_a) & & arg_{in}(z_e, x_a) & & & & &
\end{array}$$

lambda expression composition

$$\exists z.\text{company}(\text{Pixar}) \wedge \text{located}(z_e) \wedge \text{arg}_2(z_e, \text{Pixar}) \wedge \text{arg}_{\text{in}}(z_e, \text{CA})$$

UDepLambda in a nutshell

Dependency tree is a series of compositions

Dependency label defines the composition function

Each function takes two typed-semantic sub-expressions

Returns typed-semantics of the larger expression

Limitation 1: Delexicalized Semantics

Context-sensitive semantics of dependency labels, e.g., *nsubj* could mean either agent or patient

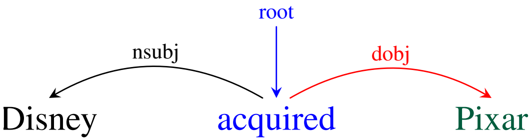
- John broke the window
- The window broke

Delexicalized context is not sufficient, e.g., quantifiers vs determiners

Solution

Learn context-specific semantics from corpus annotated with meaning representation

Limitation 2: Single Type System

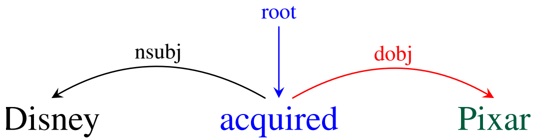


Lambda Expression for words

$$VERB \Rightarrow \lambda x.\,\text{word}(x_e), e.g., \text{acquired} \Rightarrow \lambda x.\,\text{acquired}(x_e) \quad (1)$$

$$PROPN \Rightarrow \lambda x.\,\text{word}(x_a), e.g., \text{Pixar} \Rightarrow \lambda x.\,\text{Pixar}(x_a) \quad (2)$$

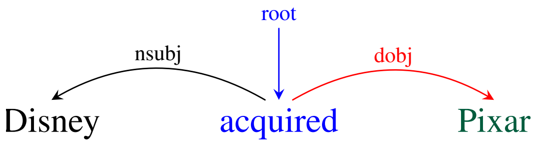
Limitation 2: Single Type System



All **words** have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η

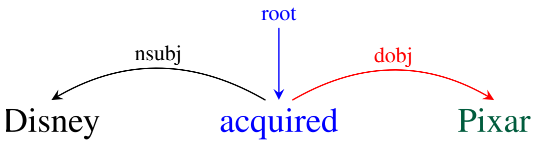
Limitation 2: Single Type System



All **constituents** have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η
- **TYPE**[(dobj acquired Pixar)] = η

Limitation 2: Single Type System



All **constituents** have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η
- **TYPE**[(dobj acquired Pixar)] = η

$\Longrightarrow$ **TYPE**[dobj] = $\eta \rightarrow \eta \rightarrow \eta$

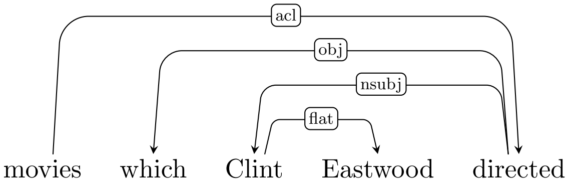
Limitation 2: Single Type System

TYPE[word] = **TYPE**[constituent] = η

Solutions

1. Rely on enhancements and stick to single type *or*
2. Context sensitive types for dependency trees

Context sensitive types



movies = $\mathsf{e} \to \mathsf{t} : \lambda x.\,\mathrm{movies}(x)$

which = $(\mathsf{e}_{\mathrm{obj}} \to \mathsf{X}) \to (\mathsf{e}_{\mathrm{obj}} \to \mathsf{X}) : \lambda f.f$

Clint = $\mathsf{e}_{\mathrm{nsubj}}$: Clint

Eastwood = e : Eastwood

directed = $\mathsf{e}_{\mathrm{nsubj}} \to \mathsf{e}_{\mathrm{obj}} \to \mathsf{t}$
$: \lambda xy.\,\exists e.\,\mathrm{directed}(e) \wedge \mathrm{arg}_1(e,x) \wedge \mathrm{arg}_2(e,y)$

nsubj = Function Application (i.e $\lambda fg.f(g);\lambda gf.g(f)$)

obj = Function Application (i.e $\lambda fg.f(g);\lambda gf.g(f)$)

acl = $(\mathsf{e} \to \mathsf{t}) \to (\mathsf{e} \to \mathsf{t}) \to (\mathsf{e} \to \mathsf{t})$
$: \lambda fgx.f(x) \wedge g(x)$

flat = $\mathsf{e}_x \to \mathsf{e} \to \mathsf{e}_x : \lambda xy.x\_y$ (concatenation)

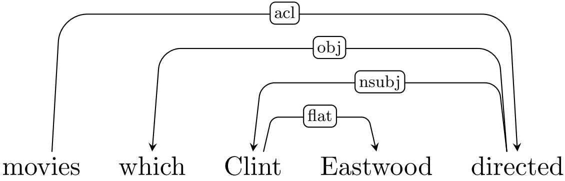
Limitation 3: Binarization

`conj (sentential)` $<$ `nsubj` $<$ `conj (phrasal)` $<$ `obj` $<$ `conj (verbal)`

Solution: Deduction/Derivation

- Use context-sensitive types and dependency tree to guide derivation
- More restricted than Glue

Deduction



1. (acl movies

   (obj (nsubj directed (flat Clint Eastwood)) which))
2. (acl movies

   (nsubj (obj directed which) (flat Clint Eastwood))

Advantage: Universal Meaning Banks

Build this for one language

Natural projection to other languages

Automatic semantic meaning banks for several languages

Community provides lexical level corrections

Questions

- Where can we learn types from?
- What is the target meaning representation?

Options

- AMR is too far away from UD
- GMB is semi-supervised CCG. No UD trees. Small?
- Enhanced UD with semantic inclination

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Substitution:

conj-vp $\Rightarrow \lambda fgx.\exists yz.f(y) \wedge g(z) \wedge \mathrm{coord}(x,y,z)$

Logical Expression:

$$\lambda w.\exists xyz.\mathrm{Eminem}(x_a) \wedge \mathrm{coord}(w,y,z) \\ \wedge \mathrm{arg}_1(w_e,x_a) \wedge \mathrm{s\_to\_I}(y) \wedge \mathrm{d\_50}(z)$$

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Substitution:

conj-vp ⇒ $\lambda fgx.\exists yz.f(y) \wedge g(z) \wedge \mathrm{coord}(x,y,z)$

Logical Expression:

$$\lambda w.\exists xyz.\mathrm{Eminem}(x_a) \wedge \mathrm{coord}(w,y,z) \\ \wedge \mathrm{arg}_1(w_e,x_a) \wedge \mathrm{s\_to\_I}(y) \wedge \mathrm{d\_50}(z)$$

Post processing:

$$\lambda e.\exists xyz.\mathrm{Eminem}(x_a) \wedge \mathrm{arg}_1(y_e,x_a) \\ \wedge \mathrm{arg}_1(z_e,x_a) \wedge \mathrm{s\_to\_I}(y) \wedge \mathrm{d\_50}(z)$$

Reduced relatives: Object extraction in CCG

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \text{the} & \text{movie} & \text{Spielberg} & \text{directed} & \text{brilliantly} \\ \hline NP_x/N_x & N_2 & NP_3 & (S_4[dcl]\backslash NP_y)/NP_z & (S_e\backslash NP_y)\backslash(S_e\backslash NP_y) \end{array}$$

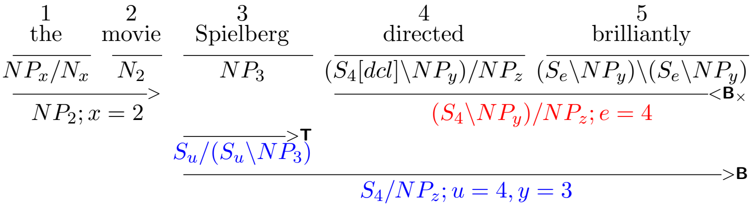
Reduced relatives: Object extraction in CCG

$$\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
\text{the} & \text{movie} & \text{Spielberg} & \text{directed} & \text{brilliantly} \\
\hline
NP_x/N_x & N_2 & NP_3 & (S_4[dcl]\backslash NP_y)/NP_z & (S_e\backslash NP_y)\backslash(S_e\backslash NP_y) \\
\cline{1-2}\cline{4-5}
\multicolumn{2}{c}{NP_2; x = 2 \quad >} & & \multicolumn{2}{c}{\color{red}{(S_4\backslash NP_y)/NP_z; e = 4} \quad <\mathbf{B}_\times}
\end{array}$$

Reduced relatives: Object extraction in CCG

$$\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5\\
\text{the} & \text{movie} & \text{Spielberg} & \text{directed} & \text{brilliantly}\\
\overline{NP_x/N_x} & \overline{N_2} & \overline{NP_3} & \overline{(S_4[dcl]\backslash NP_y)/NP_z} & \overline{(S_e\backslash NP_y)\backslash(S_e\backslash NP_y)}\\
\overline{\qquad NP_2; x = 2 \qquad}{>} & & & \overline{\qquad (S_4\backslash NP_y)/NP_z; e = 4 \qquad}{<\mathbf{B}_\times} & \\
 & & \overline{S_u/(S_u\backslash NP_3)}{>\mathbf{T}} & &
\end{array}$$

Reduced relatives: Object extraction in CCG



1 the NP_x/N_x

2 movie N_2

3 Spielberg NP_3

4 directed $(S_4[dcl]\backslash NP_y)/NP_z$

5 brilliantly $(S_e\backslash NP_y)\backslash(S_e\backslash NP_y)$

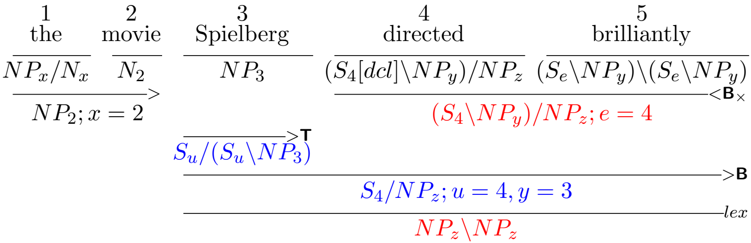
$>$ $NP_2; x = 2$

$<\mathbf{B}_\times$ $(S_4\backslash NP_y)/NP_z; e = 4$

$>\mathbf{T}$ $S_u/(S_u\backslash NP_3)$

$>\mathbf{B}$ $S_4/NP_z; u = 4, y = 3$

Reduced relatives: Object extraction in CCG



1 the NP_x/N_x

2 movie N_2

3 Spielberg NP_3

4 directed $(S_4[dcl]\backslash NP_y)/NP_z$

5 brilliantly $(S_e\backslash NP_y)\backslash(S_e\backslash NP_y)$

$>$ $NP_2; x = 2$

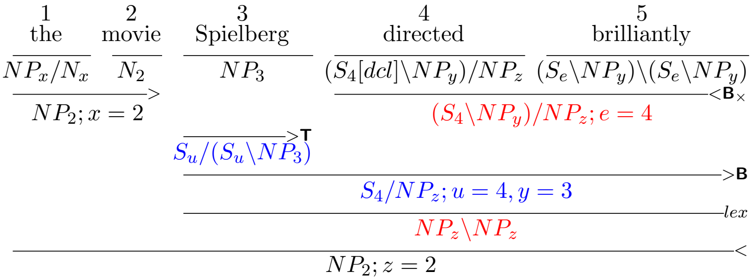
$<\mathbf{B}_\times$ $(S_4\backslash NP_y)/NP_z; e = 4$

$>\mathbf{T}$ $S_u/(S_u\backslash NP_3)$

$>\mathbf{B}$ $S_4/NP_z; u = 4, y = 3$

lex $NP_z\backslash NP_z$

Reduced relatives: Object extraction in CCG



1 the NP_x/N_x

2 movie N_2

3 Spielberg NP_3

4 directed $(S_4[dcl]\backslash NP_y)/NP_z$

5 brilliantly $(S_e\backslash NP_y)\backslash(S_e\backslash NP_y)$

$>$: $NP_2; x = 2$

$<\mathbf{B}_\times$: $(S_4\backslash NP_y)/NP_z; e = 4$

$>\mathbf{T}$: $S_u/(S_u\backslash NP_3)$

$>\mathbf{B}$: $S_4/NP_z; u = 4, y = 3$

lex: $NP_z\backslash NP_z$

$<$: $NP_2; z = 2$