Cross-lingual Semantic Parsing

Part II: Combinatory Categorial Grammar

Kilian Evang

University of Düsseldorf

Combinatory Categorial Grammar (CCG)

- grammar formalism developed by Mark Steedman (Steedman, 2001a)
- based on earlier categorial grammars (Ajdukiewicz, 1935; Bar-Hillel, 1953)
- motivation
  - syntactic (grammaticality, non-canonical coordination...)
  - phonological (intonation phrases)
  - psychological (human sentence processing)
  - computational (efficient parsing)
  - semantic (syntax-semantics interface)

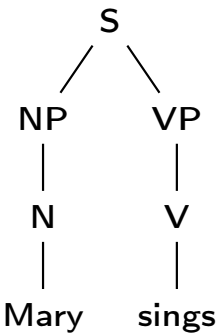
CCG and Syntactic Parsing

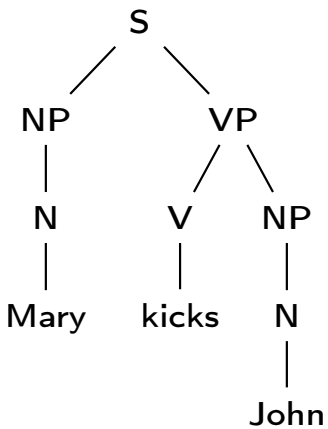
- Villavicencio (1997): first (?) implemented CCG parser
- CCGbank: Penn Treebank converted to CCG (Hockenmaier and Steedman, 2007)
- CCGrebank: CCGbank with semantically motivated annotation changes (Honnibal et al., 2010)
- C&C: First highly accurate statistical CCG parser (Clark and Curran, 2007)
- Active research on CCG parsing continues: Zhang and Clark (2011); Lewis and Steedman (2014); Lewis et al. (2016); Yoshikawa et al. (2017) and many others

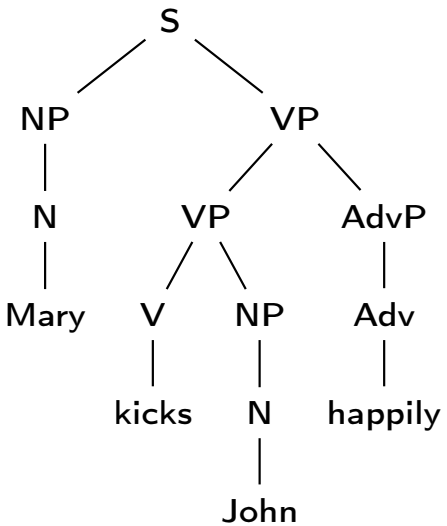
CCG and Semantic Parsing

- for narrow-domain database querying (Zettlemoyer and Collins, 2005a, 2007; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011) and many others
- for open-domain database querying (Kwiatkowski et al., 2013; Reddy et al., 2014) and many others
- for wide-coverage semantic interpretation
  - C&C+Boxer for DRT (Curran et al., 2007; Bos, 2015)
  - AMR (Artzi et al., 2015)

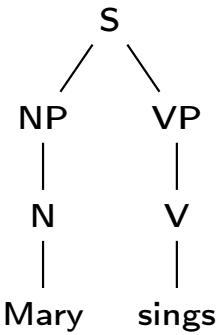
Plain Old Context-free Grammars

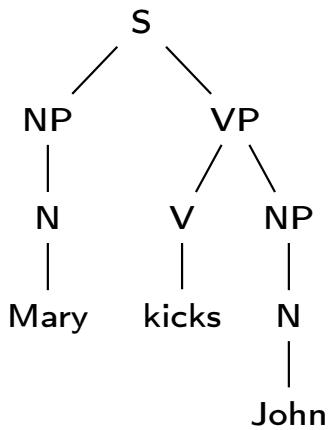


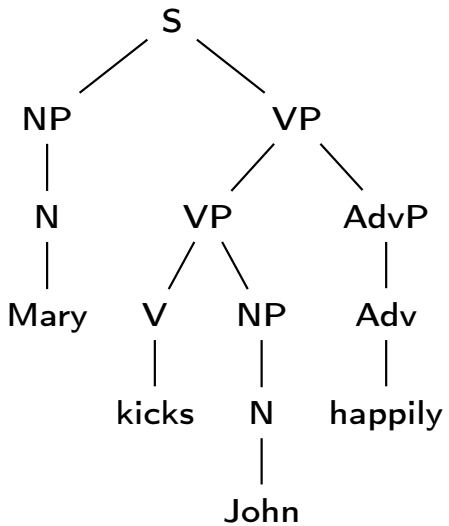




Plain Old Context-free Grammars



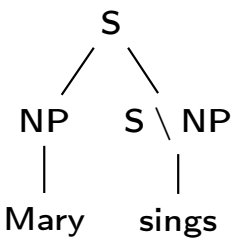


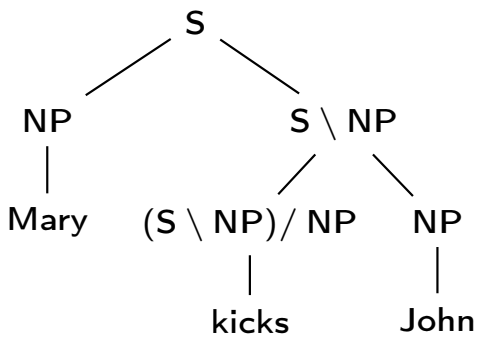


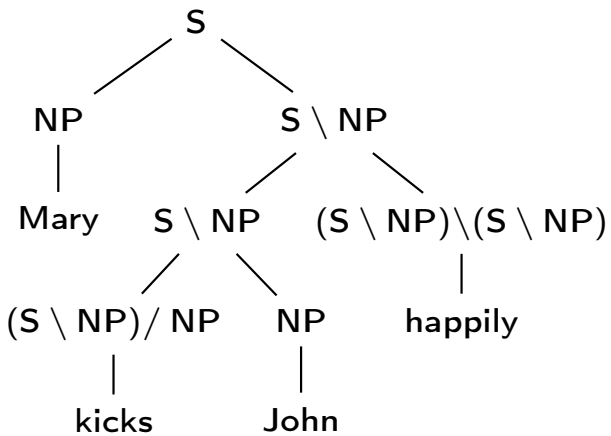
$S \rightarrow NP \quad VP$
$NP \rightarrow N$
$AdvP \rightarrow Adv$

$VP \rightarrow V$
$VP \rightarrow V \quad NP$
$VP \rightarrow VP \quad AdvP$

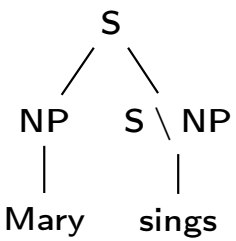
Categorial Grammars

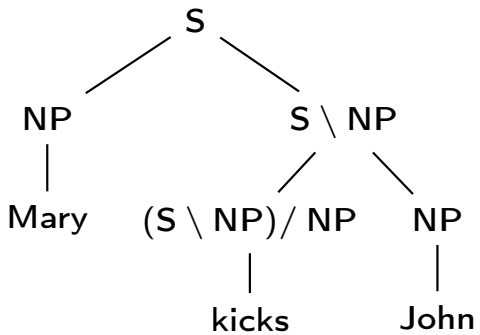


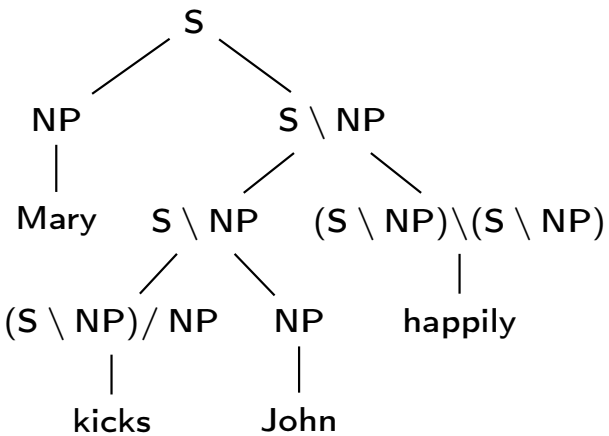




Categorial Grammars

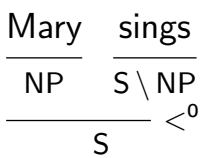


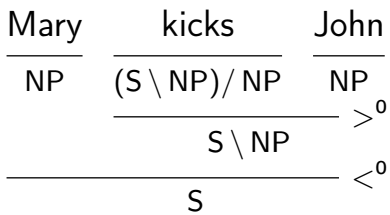


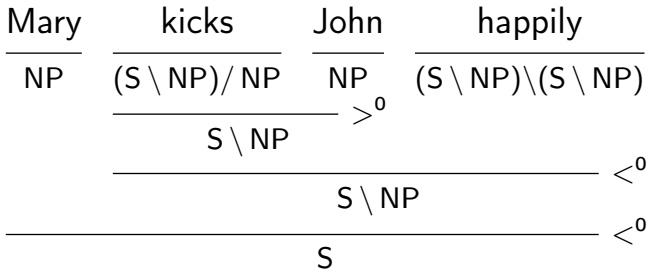


$$X \rightarrow X/Y \quad Y$$
$$X \rightarrow Y \quad X\backslash Y$$

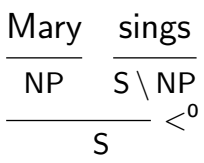
Notation

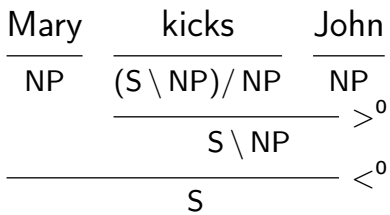

Mary
sings
NP
S \ NP
<0
S

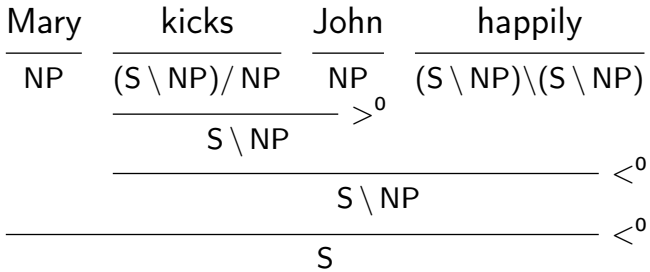

Mary
kicks
John
NP
(S \ NP)/ NP
NP
>0
S \ NP
<0
S


Mary
kicks
John
happily
NP
(S \ NP)/ NP
NP
(S \ NP)\(S \ NP)
>0
S \ NP
<0
S \ NP
<0
S

Notation







$$X/Y \quad Y \Rightarrow X \qquad (>^0) \quad \text{(forward application)}$$

$$Y \quad X\backslash Y \Rightarrow X \qquad (<^0) \quad \text{(backward application)}$$

Semantics

Mary	kicks	John
NP	$(S \backslash NP)/NP$	NP

Semantics

Mary	kicks	John
NP : *mary*	$(S \backslash NP)/NP$	NP

Semantics

Mary	kicks	John
NP : *mary*	$(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.kicks(y, x)$	NP

Semantics

$$\frac{\text{Mary}}{\text{NP} : \mathit{mary}} \qquad \frac{\text{kicks}}{(\text{S} \setminus \text{NP})/\text{NP} : \lambda x.\lambda y.\mathit{kicks}(y, x)} \qquad \frac{\text{John}}{\text{NP} : \mathit{john}}$$

Semantics

$$\frac{\text{Mary}}{\text{NP} : \textit{mary}} \qquad \frac{\text{kicks}}{(\text{S} \backslash \text{NP}) / \text{NP} : \lambda x.\lambda y.\textit{kicks}(y, x)} \qquad \frac{\text{John}}{\text{NP} : \textit{john}}$$

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \quad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X\backslash Y : f \Rightarrow X : f(g) \quad (<^0) \quad \text{(backward application)}$$

Semantics

$$\dfrac{\dfrac{\text{Mary}}{\text{NP} : \mathit{mary}} \quad \dfrac{\text{kicks}}{(\text{S} \setminus \text{NP})/\text{NP} : \lambda x.\lambda y.\mathit{kicks}(y,x)} \quad \dfrac{\text{John}}{\text{NP} : \mathit{john}}}{\dfrac{}{\text{S} \setminus \text{NP} : \lambda y.\mathit{kicks}(y,\mathit{john})}\ >^0}$$

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \quad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X \setminus Y : f \Rightarrow X : f(g) \quad (<^0) \quad \text{(backward application)}$$

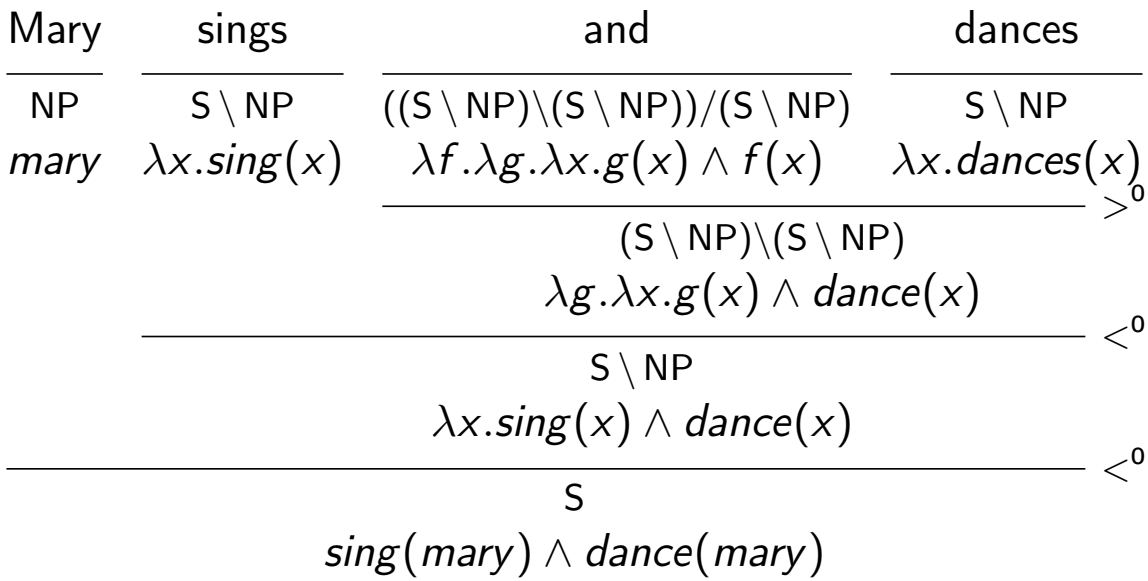
Semantics

$$\dfrac{\dfrac{\text{Mary}}{\text{NP} : \mathit{mary}} \quad \dfrac{\dfrac{\text{kicks}}{(\text{S} \setminus \text{NP})/\text{NP} : \lambda x.\lambda y.\mathit{kicks}(y,x)} \quad \dfrac{\text{John}}{\text{NP} : \mathit{john}}}{\text{S} \setminus \text{NP} : \lambda y.\mathit{kicks}(y,\mathit{john})}\ >^0}{\text{S} : \mathit{kicks}(\mathit{mary},\mathit{john})}\ <^0$$

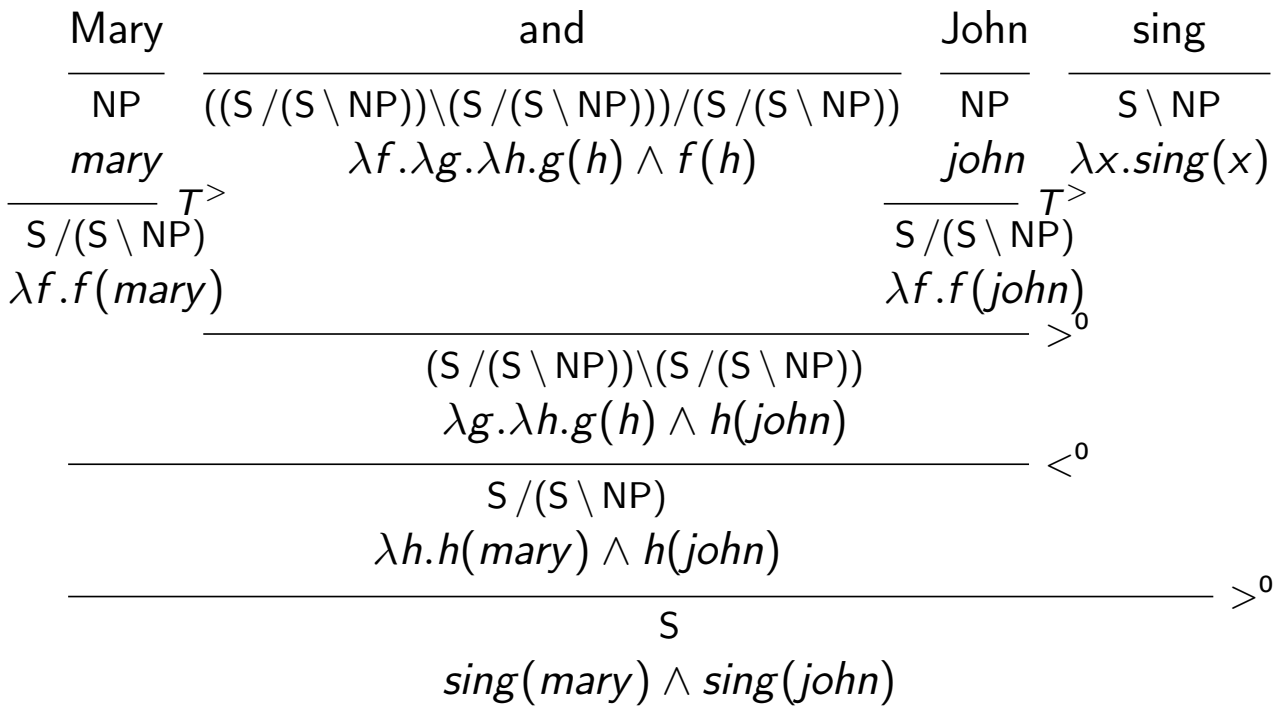
$$X/Y : f \quad Y : g \Rightarrow X : f(g) \quad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X\setminus Y : f \Rightarrow X : f(g) \quad (<^0) \quad \text{(backward application)}$$

Coordination of Functors



Coordination of Arguments



$$Y : g \Rightarrow T/(T\backslash Y) : \lambda f.f(g) \quad (T^{>}) \quad \text{(forward type raising)}$$

$$Y : g \Rightarrow T\backslash(T/Y) : \lambda f.f(g) \quad (T^{<}) \quad \text{(backward type raising)}$$

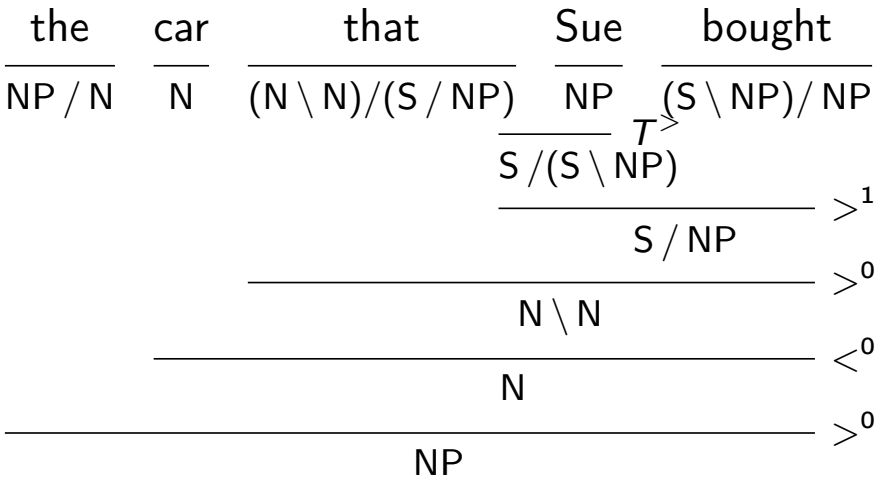
How CGs Differ from CFGs

1. informative labels (*categories*)
2. general rules (*combinators*)
3. transparent syntax-semantics interface
   (syntactic dependency $\cong$ function application)

Combinatory Categorial Grammar

- type of CG developed by Mark Steedman (Steedman, 2001b)
- additional combinators
- elegant treatment of "incomplete" constituents
  - object relative clauses
  - non-canonical coordination
  - internal functors
  - ...

Object Relative Clauses



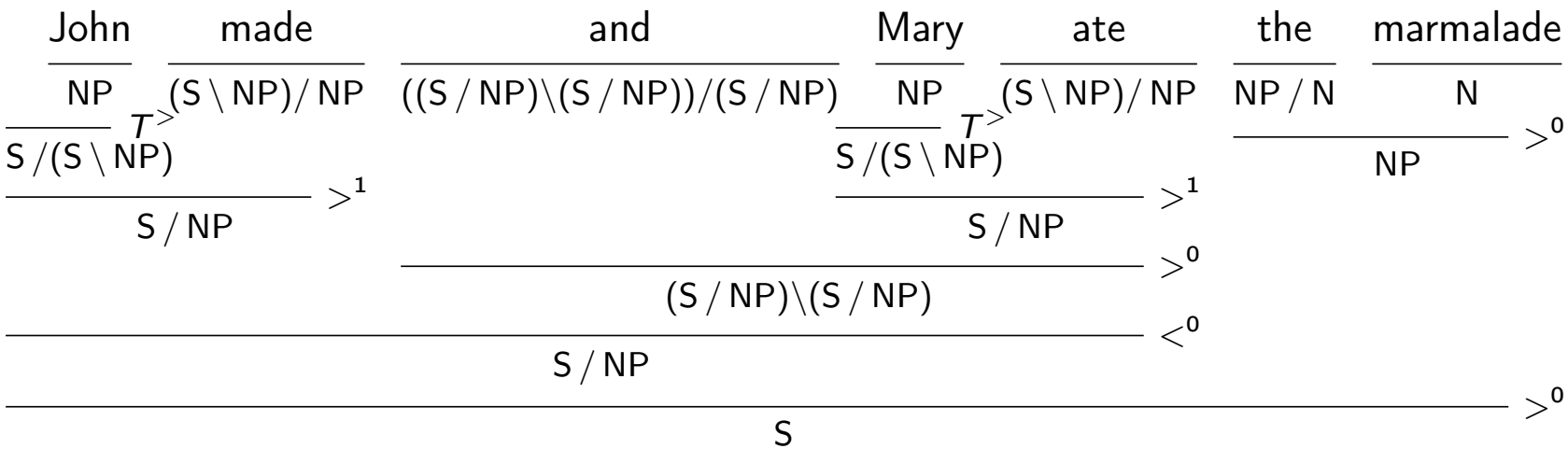
$X/Y \quad Y/Z \Rightarrow X/Z \quad (>^1)$ (forward harmonic composition)

$Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z \quad (<^1)$ (backward harmonic composition)

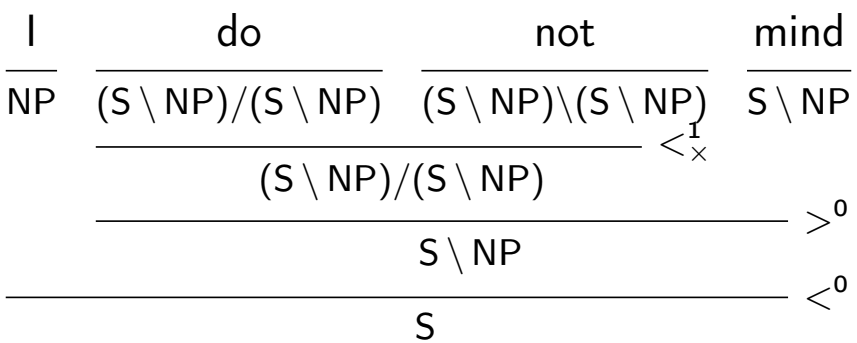
$X/Y \quad Y\backslash Z \Rightarrow X\backslash Z \quad (>^1_\times)$ (forward crossing composition)

$Y/Z \quad X\backslash Y \Rightarrow X/Z \quad (<^1_\times)$ (backward crossing composition)

Non-canonical Coordination



Internal Functors



CCG Annotation Exercise

https://texttheater.net/ccgweb

What Makes CCG Suitable for Semantic Parsing?

- small, fixed set of basic categories
- small, fixed set of rules
- flexible constituency: syntax adapts to semantics
- clear syntax-semantics interface

Semantic CCG Induction for GeoQuery

- (Zettlemoyer and Collins, 2005b): Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars
- First paper to apply CCG to semantic parser learning, many followed

Dimensions of ZC05's Parser I

NLU representation

list of words

MRL

GeoQuery and Jobs (in lambda format)

data

sentences (NLUs) + logical forms (MRs)

evaluation metric

exact match (modulo renaming variables)

lexicon representation

CCG categories with lambda terms

candidate lexicon generation

Dimensions of ZC05's Parser II

using templates (GENLEX)

parsing algorithm

CKY-style (chart)

features

only lexical features

model

log-linear

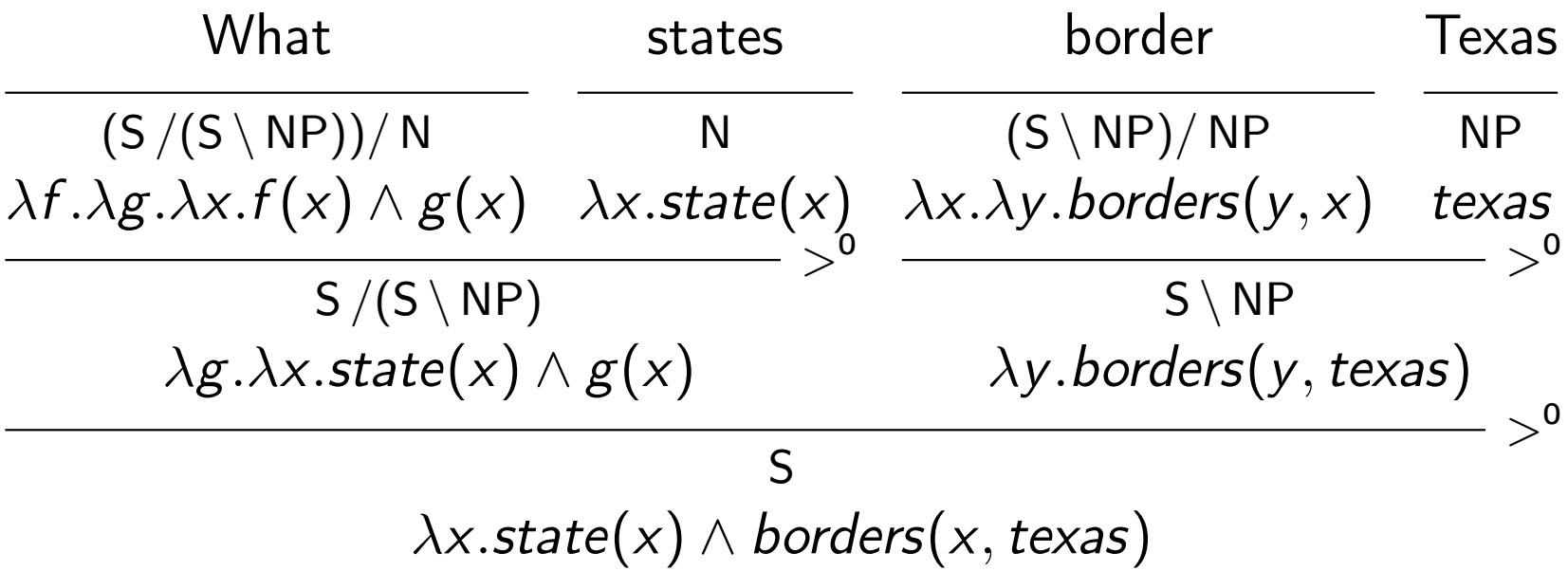
training algorithm

alternates between GENLEX, pruning, and parameter estimation

experimental setup

Geo880, Jobs640 datasets, standard splits

Example Parse



Learning Lexicon Entries

Input (a training example)

(1) a. What states border Texas
b. $\lambda x.state(x) \wedge borders(x, texas)$

Desired output

$$\text{What} := (\text{S}/(\text{S}\setminus\text{NP}))/\text{N} : \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$$
$$\text{states} := \text{N} : \lambda x.state(x)$$
$$\text{border} := (\text{S}\setminus\text{NP})/\text{NP} : \lambda x.\lambda y.borders(y, x)$$
$$\text{Texas} := \text{NP} : texas$$

Hard-coded Cross-domain Lexicon Entries

$$\text{What} := (\text{S}/(\text{S}\setminus\text{NP}))/\text{N} : \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$$

Candidate Lexicon Generation

$\text{GENLEX}(S, L) = \{x := y | x \in W(S), y \in C(L)\}$
where

- S is a sentence
- L is its logical form
- $W(S)$ is the set of all subsequences of words in S
- C maps L to a set of categories through rules (see next slide)

Candidate Lexicon Generation (cont.)

Training Example

(2) a. Utah borders Idaho
b. $next\_to(utah, idaho)$

Candidate Lexicon Generation (cont.)

Training Example

(2) a. Utah borders Idaho
b. $next\_to(utah, idaho)$

Triggered Rules

Input trigger	Output category
constant c	$\mathrm{NP} : c$
arity two predicate p_2	$(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.p(y, x)$
arity two predicate p_2	$(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.p(x, y)$

Candidate Lexicon Generation (cont.)

Generated Entries

Utah $\vdash$ NP : *utah*
borders $\vdash$ NP : *utah*
Idaho $\vdash$ NP : *utah*
Utah $\vdash$ NP : *idaho*
borders $\vdash$ NP : *idaho*
Idaho $\vdash$ NP : *idaho*
Utah $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(y, x)$
borders $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(y, x)$
Idaho $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(y, x)$
Utah $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(x, y)$
borders $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(x, y)$
Idaho $\vdash (S \setminus NP)/NP : \lambda x.\lambda y.next\_to(x, y)$

Full Set of Rules

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an $\arg\max$ / $\min$ with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

Probabilistic CCG Parsing

- mapping a sentence S to (T, L) where T is a parse tree (derivation) and L is a logical form
- even with a perfect lexicon, one S can have multiple (T, L) pairs (lexical ambiguity, attachment ambiguity, spurious ambiguity...)
- solution: define probability distribution $P(L, T|S)$

Probabilistic CCG Parsing (cont.)

- here: features = lexicon entries used in a parse
- feature function $\overline{f}$ maps parses (L, T, S) to a feature vector that indicates for each lexicon entry how often it is used in T
- assume a parameter vector $\overline{\theta}$ that scores individual lexical features such that $P(L, T|S; \overline{\theta}) = \frac{e^{\overline{f}(L,T,S)\cdot\overline{\theta}}}{\sum_{(L,T)} e^{\overline{f}(L,T,S)\cdot\overline{\theta}}}$ (log-linear model)
- Given S, return the parse (L, T) with the highest $P(L, T|S; \theta)$
- problem: how to find a good lexicon and a good $\overline{\theta}$?

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0 :=$ vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
    for $i = 1 \ldots n$ **do**

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π
  end for

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π
  end for
  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▷ update the lexicon

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π
  end for
  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▷ update the lexicon
  $\overline{\theta}^t := \text{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▷ parameter estimation

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π
  end for
  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▷ update the lexicon
  $\overline{\theta}^t := \text{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▷ parameter estimation
end for

Bird's-eye View of the Learning Algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
  for $i = 1 \ldots n$ **do**
    $\lambda := \Lambda_{t-1} \cup \text{GENLEX}(S_i, L_i)$ ▷ lexical generation
    $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    $\lambda_i :=$ the set of lexicon entries in π
  end for
  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▷ update the lexicon
  $\overline{\theta}^t := \text{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▷ parameter estimation
end for
Output: lexicon Λ_T, parameters $\overline{\theta}^T$

Results

	Geo880		Jobs640	
	Precision	Recall	Precision	Recall
Zettlemoyer and Collins (2005b)	96.25	79.29	97.36	79.29
Tang and Mooney (2001)	89.92	79.40	93.25	79.84

Some Learned Lexicon Entries

$$\begin{aligned}
\text{states} &:= \mathrm{N} : \lambda x.\mathit{state}(x) \\
\text{major} &:= \mathrm{N} / \mathrm{N} : \lambda f.\lambda x.\mathit{major}(x) \wedge f(x) \\
\text{population} &:= \mathrm{N} : \lambda x.\mathit{population}(x) \\
\text{cities} &:= \mathrm{N} : \lambda x.\mathit{river}(x) \\
\text{rivers} &:= \mathrm{N} : \lambda x.\mathit{river}(x) \\
\text{run through} &:= (\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.\mathit{traverse}(y, x) \\
\text{the largest} &:= \mathrm{NP} / \mathrm{N} : \lambda f.\arg\max(f, \lambda x.\mathit{size}(x)) \\
\text{river} &:= \mathrm{N} : \lambda x.\mathit{river}(x) \\
\text{the highest} &:= \mathrm{NP} / \mathrm{N} : \lambda f.\arg\max(f, \lambda x.\mathit{elev}(x)) \\
\text{the longest} &:= \mathrm{NP} / \mathrm{N} : \lambda f.\arg\max(f, \lambda x.\mathit{len}(x))
\end{aligned}$$

Bibliography I

Ajdukiewicz, K. (1935). Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27.

Artzi, Y., Lee, K., and Zettlemoyer, L. (2015). Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710. Association for Computational Linguistics.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Bos, J. (2015). Open-domain semantic parsing with boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304.

Bibliography II

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics, Volume 33, Number 4, December 2007*.

Curran, J., Clark, S., and Bos, J. (2007). Linguistically motivated large-scale nlp with c
&c and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36. Association for Computational Linguistics.

Hockenmaier, J. and Steedman, M. (2007). Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics, Volume 33, Number 3, September 2007*.

Bibliography III

Honnibal, M., Curran, J. R., and Bos, J. (2010). Rebanking ccgbank for improved np interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Kwiatkowksi, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556. Association for Computational Linguistics.

Bibliography IV

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523. Association for Computational Linguistics.

Lewis, M., Lee, K., and Zettlemoyer, L. (2016). Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231. Association for Computational Linguistics.

Lewis, M. and Steedman, M. (2014). A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar.

Bibliography V

Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Steedman, M. (2001a). *The Syntactic Process*. The MIT Press.

Steedman, M. (2001b). *The Syntactic Process*. The MIT Press.

Tang, L. R. and Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477.

Villavicencio, A. (1997). Building a wide-coverage combinatory categorial grammar. Master's thesis, University of Cambridge.

Bibliography VI

Yoshikawa, M., Noji, H., and Matsumoto, Y. (2017). A* ccg parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287. Association for Computational Linguistics.

Zettlemoyer, L. and Collins, M. (2005a). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–666. AUAI Press.

Zettlemoyer, L. and Collins, M. (2005b). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–666. AUAI Press.

Bibliography VII

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Zhang, Y. and Clark, S. (2011). Shift-reduce ccg parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 683–692. Association for Computational Linguistics.