PARSING
SYNTAX AND SEMANTICS
IN TANDEM

(Morpho)syntax → semantics

- The interpretive approach
  - Morphology, syntax are prior
  - Map meaning *from*: morpho ➔ syn ➔ sem
  - Most current linguistic theories
- The tandem approach
  - Semantics emerges in tandem with morpho, syn
  - Map meaning *while*: morpho + syn + sem
  - Less widely practiced here in North America
  - Popular in AI, NLP, cognitive science

SYNTAX

Categorial Grammar

- Lexicalized grammatical formalism
  - The lexicon is primary and the driving force in determining structure.
- Since early 1940’s, primarily in Europe
- Compositional syntax and semantics simultaneously
- Type-driven, combinatorial, categorial
- Particularly well suited for languages with complex morphosyntax
- Computational implementations

Syntactic categories in CG

- Basic categories: np, n, and s
- Complex categories: composition of basic categories
  - α/β is a valid category if α and β are basic or complex categories
  - α\β is a valid category if α and β are basic or complex categories
- Example categories:
  - determiner: np/n
    - "A determiner can become a noun phrase if it can combine with a noun to the right (i.e. forwards)."
  - verb phrase: s\np
    - "A verb phrase can become a sentence if it can combine with a noun phrase to the left (i.e. backwards)."
- Delimiting slash (\ or /) specifies the directionality of combination

Application schemas

- Forward application schema: $$\frac{\overset{\text{lex}_1}{\alpha/\beta} \quad \overset{\text{lex}_2}{\beta}}{\alpha}$$
- Forward application example: $$\frac{\overset{\textit{the}}{\text{np}/\text{n}} \quad \overset{\textit{dog}}{\text{n}}}{\text{np}}$$ ← *the* can combine to the right with *dog* to create a noun phrase
- Backward application schema: $$\frac{\overset{\text{lex}_1}{\beta} \quad \overset{\text{lex}_2}{\alpha\backslash\beta}}{\alpha}$$
- Backward application example: $$\frac{\overset{\textit{dogs}}{\text{np}} \quad \overset{\textit{bark}}{\text{s}\backslash\text{np}}}{\text{s}}$$ ← *bark* can combine to the left with *dogs* to create a sentence

Example categories (syntax)

- determiner: np/n (*the*, *my*, *a*, *some*, *those*, etc.)
- predicate: s\np (*sneezed*, *ate tacos*, *is big*, etc.)
- adjective: n/n (*big*, *ugly*, *disappointed*, etc.)
- verb
  - intransitive: s\np (*yawns*, *somersaulted*, *died*, etc.)
  - transitive: (s\np)/np (*kicked*, *reads*,
  - ditransitive: (s\np)/np/np (*gave, sent, told,* etc.)
- preposition:
  - (n\n)/np (*with a moustache*, *from Paris*, etc.)
  - ((s\np)\(s\np))/np
- pronoun, bare plural noun, proper noun: np (*you*, *dogs*, *Fred*, etc.)
- adverb
  - (s\np)\(s\np) or s/s or s\s (*quickly*, *naturally*, *yesterday*, etc.)
  - (n/n)/(n/n) (*very, quite, rather,* etc.)

Sample noun phrase parses

- Adjectival modification:

the	*big*	*salmon*
np/n	n/n	n

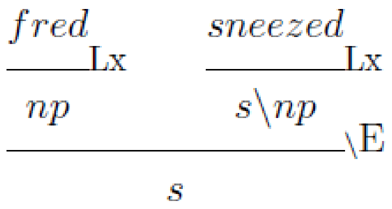
$$\dfrac{\text{np/n} \quad \dfrac{\text{n/n} \quad \text{n}}{\text{n}}}{\text{np}}$$

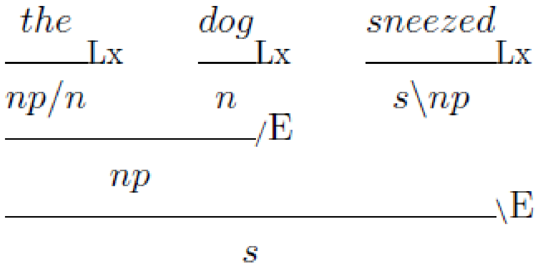
- Prepositional phrase modifier

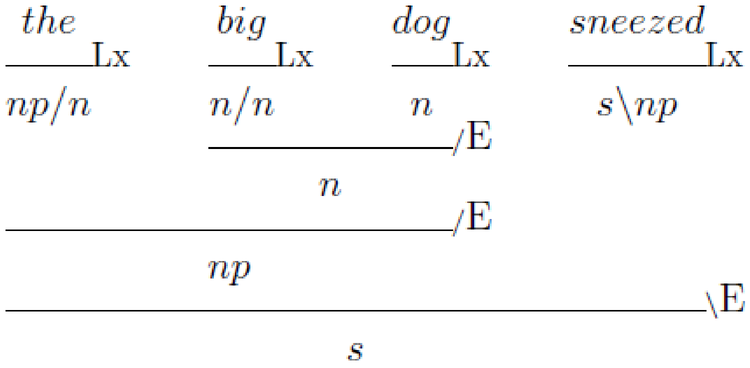
the	*big*	*salmon*	*in*	*the*	*stream*
np/n	n/n	n	(n\n)/np	np/n	n

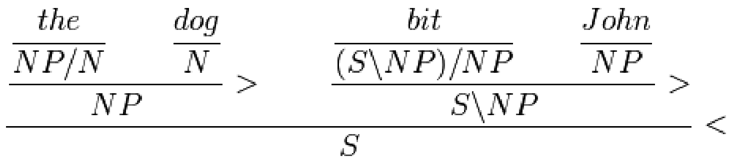
$$\dfrac{\text{np/n} \quad \dfrac{\text{n/n} \quad \dfrac{\text{n} \quad \dfrac{\text{(n\textbackslash n)/np} \quad \dfrac{\text{np/n} \quad \text{n}}{\text{np}}}{\text{n\textbackslash n}}}{\text{n}}}{\text{n}}}{\text{np}}$$

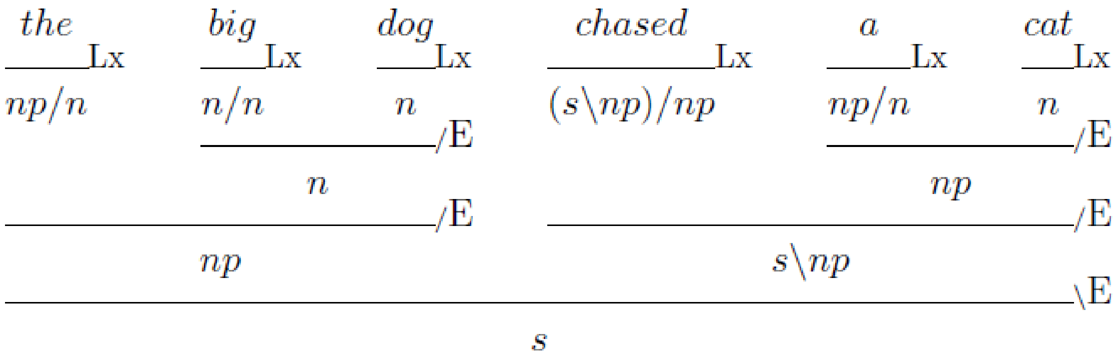
Sentence parses





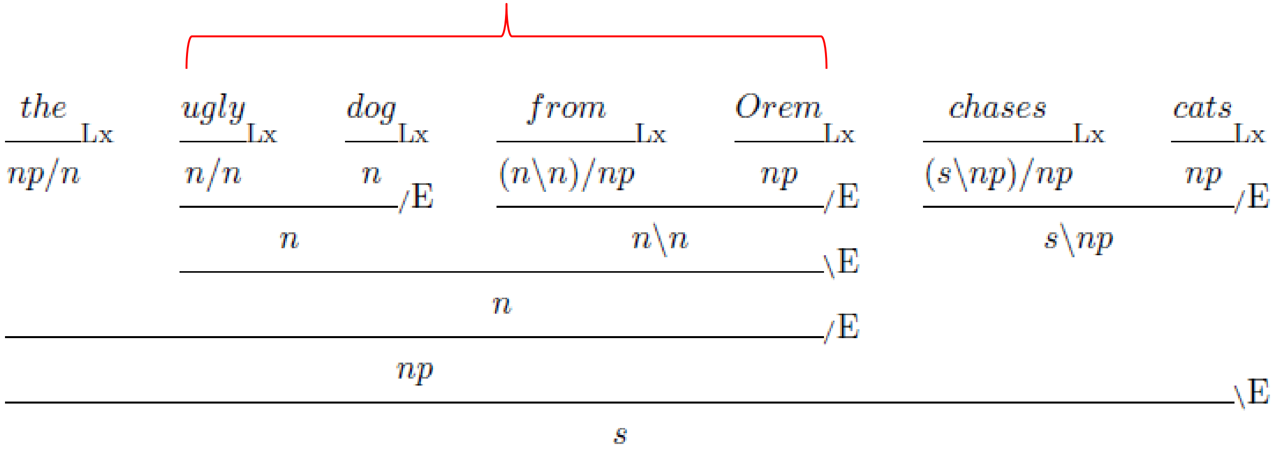


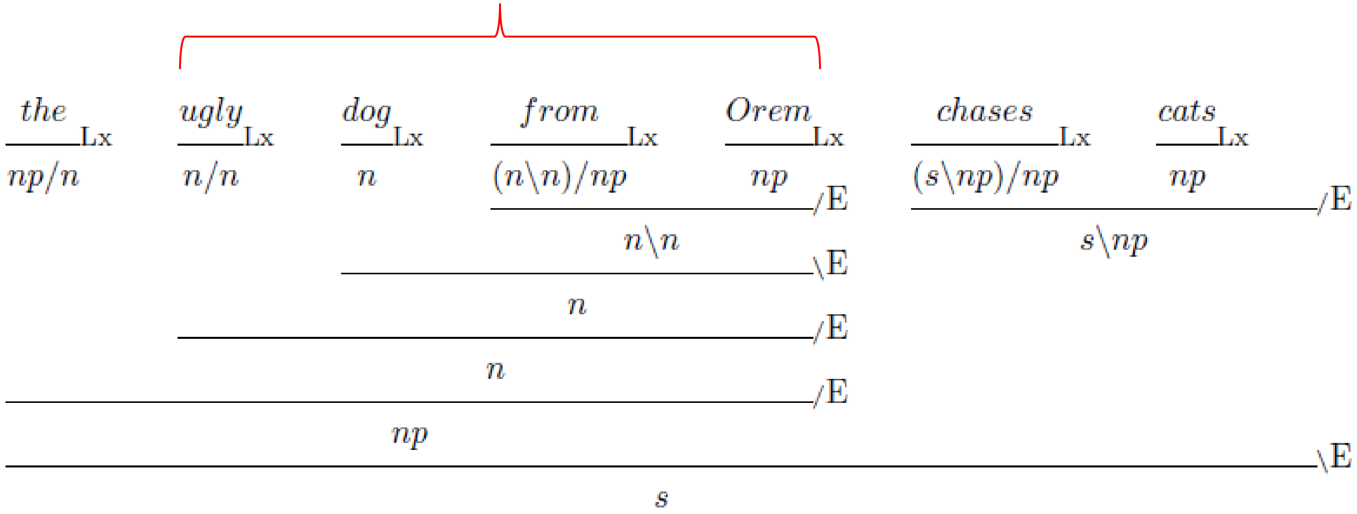




Structural ambiguity

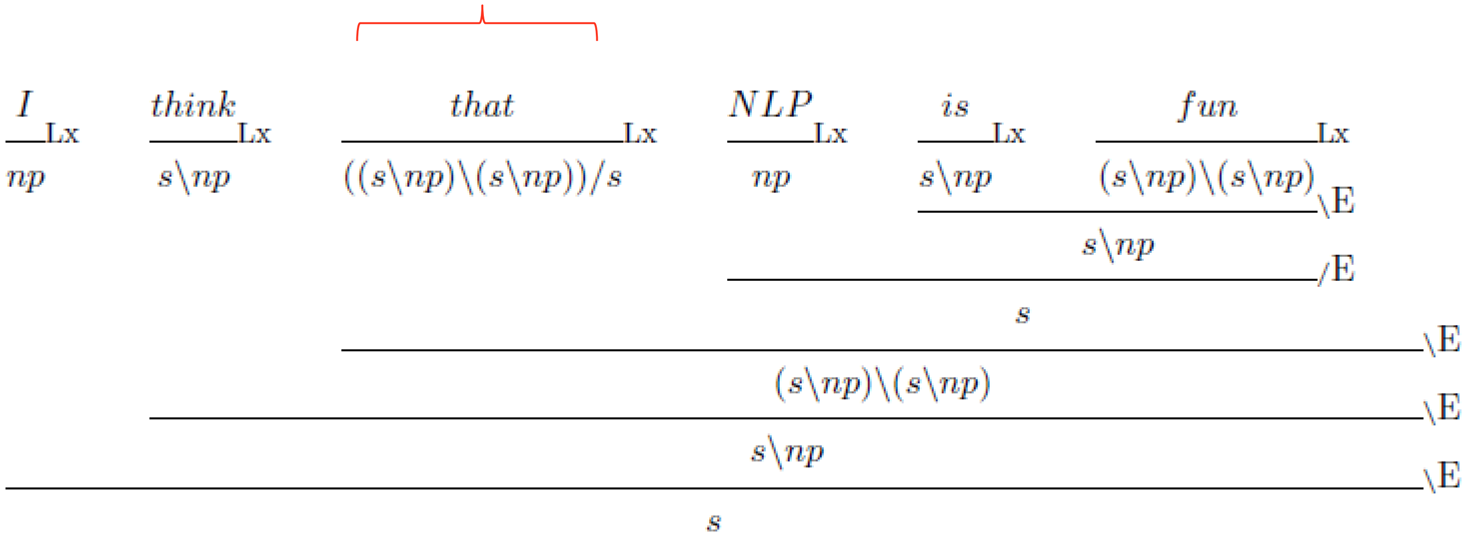
- In this case the ambiguity is spurious (no discernible meaning differences)

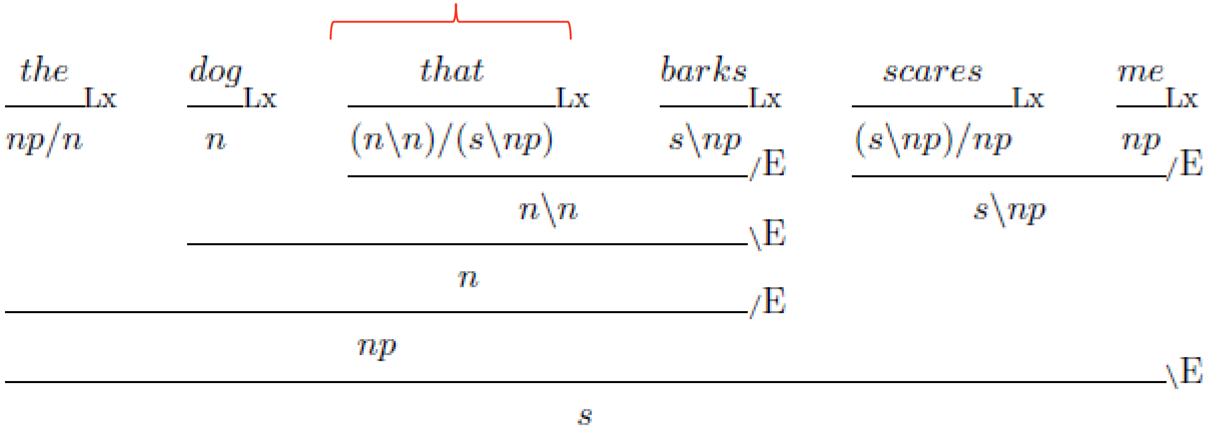




More complex categories

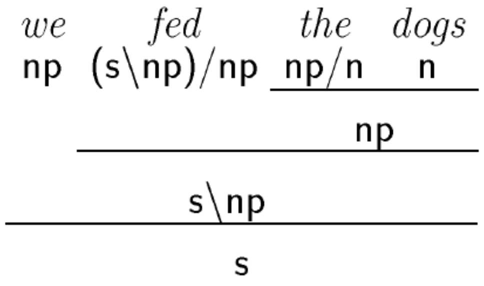
- complementizer: ((s\np)\(s\np))/s
- relative pronoun: (n\n)/(s\np)



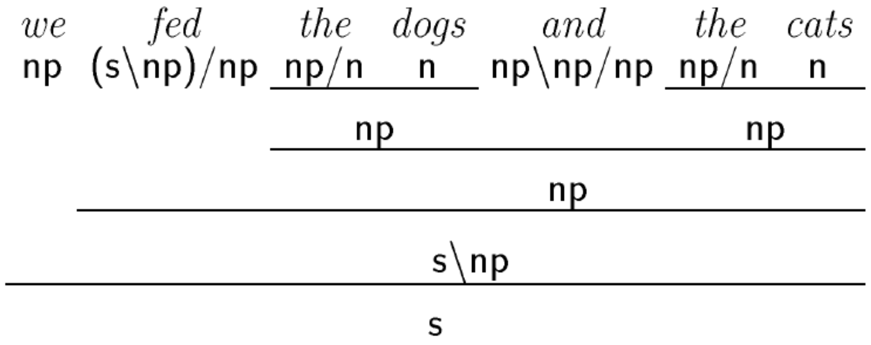


Conjunction

- Simple



- Conjoined object



Beyond classical categorial grammar

- Combinatorial Categorial Grammar
- More functions for combining (beyond just the elimination schemas)

Forward function composition

$$\frac{\alpha : X/Y \qquad \beta : Y/Z}{\alpha\beta : X/Z} B_{>}$$

Right node raising:
John likes and Bill detests broccoli.

Backward function composition

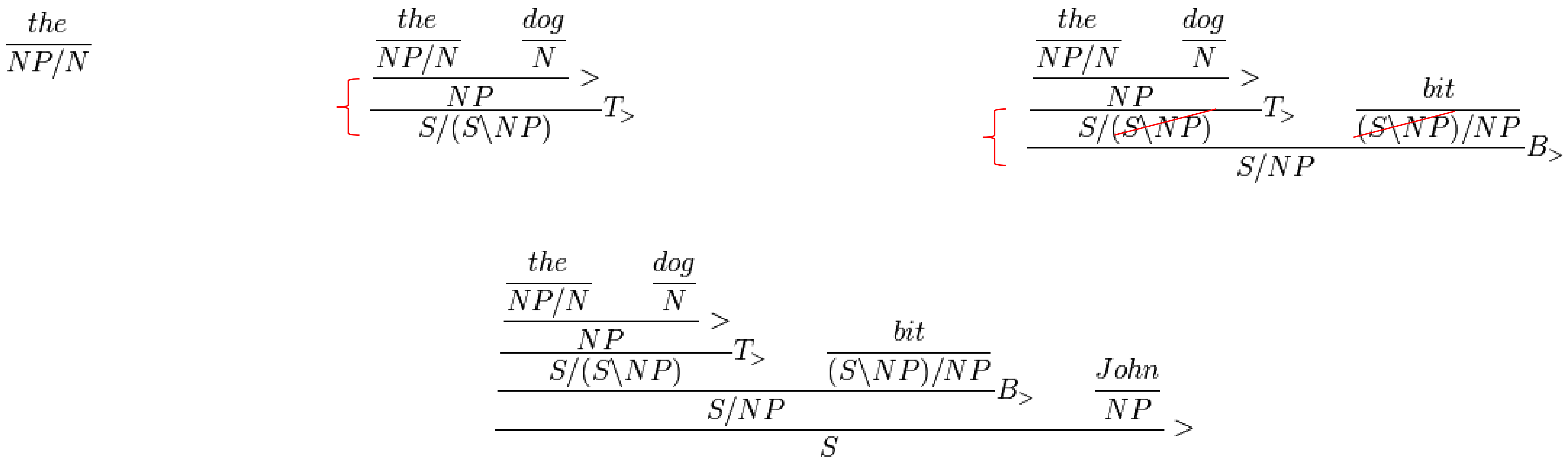
$$\frac{\beta : Y\backslash Z \qquad \alpha : X\backslash Y}{\beta\alpha : X\backslash Z} B_{<}$$

Left node raising:
John introduced Bill to Sue and Harry to Sally.

- Other functions for more complex syntactic constructions

Type raising and incremental parsing

- Combination of type raising and forward composition
- Tom:np is equivalent to: Tom:s/(s\np)
  - "*Tom* can be a sentence if it can combine with a predicate to its right."



SEMANTICS

How is this a tandem approach?

- It isn't yet: so far we only have syntax.
- Semantic representation
  - Predicate logic with lambda operations
  - When "head" combines with "dependent" in syntax, use semantics of "head" as a predicate/function, and the semantics of the "dependent" as its argument.
- Lexical category: now has both a syntactic category (as before) and a semantic category (lambda expression)
- Combination can be done based on either syntax, semantics, or both
  - Syntax: CG schema application
  - Semantics: λ reduction

Review: lambda reduction (λR)

- Instantiating a variable with an individual

$$\frac{\lambda x.[dog(x)\ \&\ furry(x)](fido)}{dog(fido)\ \&\ furry(fido)}\lambda R$$

justification

$$\frac{\dfrac{\lambda y.[\lambda x.[sees(x,y)](fred)](fido)}{\lambda x.[sees(x, fido)](fred)}\lambda R}{sees(fred, fido)}\lambda R$$

- Instantiating a predicate variable with a predicate

$$\frac{\dfrac{\lambda Q.[\lambda P.[Q\ \&\ P\ \&\ (tail(y)\ \&\ wags(x,y))](barks(x))](dog(x))}{\lambda P.[dog(x)\ \&\ P\ \&\ (tail(y)\ \&\ wags(x,y))](barks(x))}\lambda R}{dog(x)\ \&\ barks(x)\ \&\ tail(y)\ \&\ wags(x,y)}\lambda R$$

Example semantic categories

- Nominal predicates: dog(x)
- Intransitive: λP[P→sneezed(x)] or λP[P & sneezed(x)]
  - Ignore the distinction for now
- Adjective: λP[P & big(x)]
- Two-place predicates: λQλP[P & Q & likes(x,y)], λQλP[P & Q & in(x,y)]

- The iota operator: semantics for “the” (definite descriptor): ι
  - Shorthand for: ∃x[(P(x) & ∀y(P(y) ↔ y=x)) & Q(x)]

- Ignore semantics: use λP.P for the semantic category: λP.[P](happy(fred)) = happy(fred)

Intransitive sentences

$$\dfrac{the}{np/n\colon \iota}\text{Lx} \quad \dfrac{dog}{n\colon dog(x)}\text{Lx} \quad \dfrac{sneezed}{s\backslash np\colon \lambda P.[P\ \&\ sneeze(x)]}\text{Lx}$$

$$\dfrac{np/n\colon \iota \quad n\colon dog(x)}{np\colon \iota(dog(x))}/\text{E}$$

$$\dfrac{np\colon \iota(dog(x)) \quad s\backslash np\colon \lambda P.[P\ \&\ sneeze(x)]}{s\colon \lambda P.[P\ \&\ sneeze(x)](\iota(dog(x)))}\backslash\text{E}$$

$$\dfrac{s\colon \lambda P.[P\ \&\ sneeze(x)](\iota(dog(x)))}{s\colon \iota(dog(x))\ \&\ sneeze(x)}\lambda R$$

$$\dfrac{the}{np/n\colon \iota}\text{Lx} \quad \dfrac{big}{n/n\colon \lambda P.[P\ \&\ big(x)]}\text{Lx} \quad \dfrac{dog}{n\colon dog(x)}\text{Lx} \quad \dfrac{bites}{s\backslash np\colon \lambda Q.[Q\ \&\ bites(x)]}\text{Lx}$$

$$\dfrac{n/n\colon \lambda P.[P\ \&\ big(x)] \quad n\colon dog(x)}{n\colon \lambda P.[P\ \&\ big(x)](dog(x))}/\text{E}$$

$$\dfrac{n\colon \lambda P.[P\ \&\ big(x)](dog(x))}{n\colon dog(x)\ \&\ big(x)}\lambda R$$

$$\dfrac{np/n\colon \iota \quad n\colon dog(x)\ \&\ big(x)}{np\colon \iota(dog(x)\ \&\ big(x))}/\text{E}$$

$$\dfrac{np\colon \iota(dog(x)\ \&\ big(x)) \quad s\backslash np\colon \lambda Q.[Q\ \&\ bites(x)]}{s\colon \lambda Q.[Q\ \&\ bites(x)](\iota(dog(x)\ \&\ big(x)))}\backslash\text{E}$$

$$\dfrac{s\colon \lambda Q.[Q\ \&\ bites(x)](\iota(dog(x)\ \&\ big(x)))}{s\colon \iota(dog(x)\ \&\ big(x))\ \&\ bites(x)}\lambda R$$

Transitive sentences

$$\dfrac{fido}{np\colon fido}\,\text{Lx} \qquad \dfrac{chased}{(s\backslash np)/np\colon \lambda y.\lambda x.chased(x,y)}\,\text{Lx} \qquad \dfrac{fred}{np\colon fred}\,\text{Lx}$$

$$\dfrac{(s\backslash np)/np\colon \lambda y.\lambda x.chased(x,y) \quad np\colon fred}{s\backslash np\colon \lambda y.\lambda x.[chased(x,y)](fred)}\,/\text{E}$$

$$\dfrac{s\backslash np\colon \lambda y.\lambda x.[chased(x,y)](fred)}{s\backslash np\colon \lambda x.chased(x,fred)}\,\lambda R$$

$$\dfrac{np\colon fido \quad s\backslash np\colon \lambda x.chased(x,fred)}{s\colon \lambda x.[chased(x,fred)](fido)}\,\backslash\text{E}$$

$$\dfrac{s\colon \lambda x.[chased(x,fred)](fido)}{s\colon chased(fido,fred)}\,\lambda R$$

$$\dfrac{the}{np/n\colon \iota}\,\text{Lx} \qquad \dfrac{dog}{n\colon dog(x)}\,\text{Lx} \qquad \dfrac{sees}{(s\backslash np)/np\colon \lambda R.\lambda Q.[Q \;\&\; R \;\&\; sees(x,y)]}\,\text{Lx} \qquad \dfrac{a}{np/n\colon \lambda P.P}\,\text{Lx} \qquad \dfrac{cat}{n\colon cat(y)}\,\text{Lx}$$

$$\dfrac{np/n\colon \iota \quad n\colon dog(x)}{np\colon \iota(dog(x))}\,/\text{E}$$

$$\dfrac{np/n\colon \lambda P.P \quad n\colon cat(y)}{np\colon \lambda P.P[cat(y)]}\,/\text{E}$$

$$\dfrac{np\colon \lambda P.P[cat(y)]}{np\colon cat(y)}\,\lambda R$$

$$\dfrac{(s\backslash np)/np\colon \lambda R.\lambda Q.[Q \;\&\; R \;\&\; sees(x,y)] \quad np\colon cat(y)}{s\backslash np\colon \lambda R.\lambda Q.[Q \;\&\; R \;\&\; sees(x,y)](cat(y))}\,/\text{E}$$

$$\dfrac{s\backslash np\colon \lambda R.\lambda Q.[Q \;\&\; R \;\&\; sees(x,y)](cat(y))}{s\backslash np\colon \lambda Q.[Q \;\&\; cat(y) \;\&\; sees(x,y)]}\,\lambda R$$

$$\dfrac{np\colon \iota(dog(x)) \quad s\backslash np\colon \lambda Q.[Q \;\&\; cat(y) \;\&\; sees(x,y)]}{s\colon \lambda Q.[Q \;\&\; cat(y) \;\&\; sees(x,y)](\iota(dog(x)))}\,\backslash\text{E}$$

$$\dfrac{s\colon \lambda Q.[Q \;\&\; cat(y) \;\&\; sees(x,y)](\iota(dog(x)))}{s\colon \iota(dog(x)) \;\&\; cat(y) \;\&\; sees(x,y)}\,\lambda R$$

Intensionality and ambiguity

- Push/pop a quantifier
  - Use a Cooper store
- Assume generalized quantifiers
- Two possible representations

$$\frac{sam}{np\colon sam}\,\mathrm{Lx} \qquad \frac{seeks}{(s\backslash np)/s{\Uparrow}np\colon \lambda y.\lambda x.seek(x,y)}\,\mathrm{Lx} \qquad \frac{a\ unicorn}{s{\Uparrow}np\colon some(unicorn)}$$

$$\frac{(s\backslash np)/s{\Uparrow}np\colon \lambda y.\lambda x.seek(x,y) \quad s{\Uparrow}np\colon some(unicorn)}{s\backslash np\colon \lambda y.[\lambda x.seek(x,y)](some(unicorn))}\,/\mathrm{E}$$

$$\frac{s\backslash np\colon \lambda y.[\lambda x.seek(x,y)](some(unicorn))}{s\backslash np\colon \lambda x.seek(x, some(unicorn))}\,\lambda R$$

$$\frac{np\colon sam \quad s\backslash np\colon \lambda x.seek(x, some(unicorn))}{s\colon \lambda x.[seek(x, some(unicorn))](sam)}\,\backslash\mathrm{E}$$

$$\frac{s\colon \lambda x.[seek(x, some(unicorn))](sam)}{s\colon seek(sam, some(unicorn))}\,\lambda R$$

$$\frac{sam}{np\colon sam}\,\mathrm{Lx} \qquad \frac{seeks}{(s\backslash np)/s{\Uparrow}np\colon \lambda y.\lambda x.seek(x,y)}\,\mathrm{Lx} \qquad \frac{a\ unicorn}{s{\Uparrow}np\colon some(unicorn)}$$

$$\frac{s{\Uparrow}np\colon some(unicorn)}{np\colon z}\,{\Uparrow}\mathrm{E}^0 \qquad \frac{np\colon z}{np\colon \lambda P.P(z)}\,{\Uparrow}\mathrm{I}$$

$$\frac{(s\backslash np)/s{\Uparrow}np\colon \lambda y.\lambda x.seek(x,y) \quad np\colon \lambda P.P(z)}{s\backslash np\colon \lambda y.\lambda x.seek(x,y)(\lambda P.P(z))}\,/\mathrm{E}$$

$$\frac{s\backslash np\colon \lambda y.\lambda x.seek(x,y)(\lambda P.P(z))}{s\backslash np\colon \lambda x.seek(x, \lambda P.P(z))}\,\lambda R$$

$$\frac{np\colon sam \quad s\backslash np\colon \lambda x.seek(x, \lambda P.P(z))}{s\colon \lambda x.[seek(x, \lambda P.P(z))](sam)}\,\backslash\mathrm{E}$$

$$\frac{s\colon \lambda x.[seek(x, \lambda P.P(z))](sam)}{s\colon seek(sam, \lambda P.P(z))}\,\lambda R$$

$$\frac{s\colon seek(sam, \lambda P.P(z))}{s\colon some(unicorn)(\lambda z.(seek(sam, \lambda P.P(z))))}\,0$$

Summary

- (Combinatory) Categorial Grammar for syntax
- Lambda calculus for semantics
- Done in tandem
  - Either could initiate combinations
- Lots of flexibility
- Captures ambiguity
- Multilingual application
- Morphological syn/sem
- Tools exist

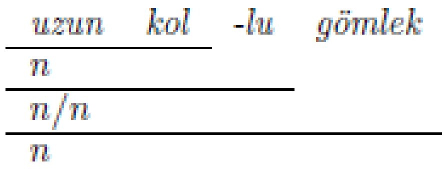
OTHER LANGUAGES

Working with other languages

- Free word order languages: use | instead of \ and /
- Morphologically complex languages: assign categories to morphemes
- Agreement/concord: assign features to slashes
  - Subject-verb agreement
  - NP-internal concord
  - Verb subcategorization

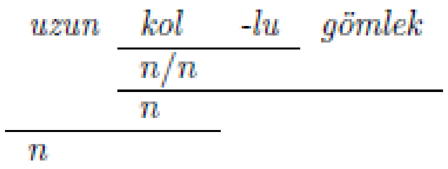
lexical entry	syntactic category
uzun	n/n
kol	n
-lu	$(n/n) \backslash n$
gömlek	n

(1a)



a shirt with long sleeves

(1b)



a long shirt with sleeves

ASL parse

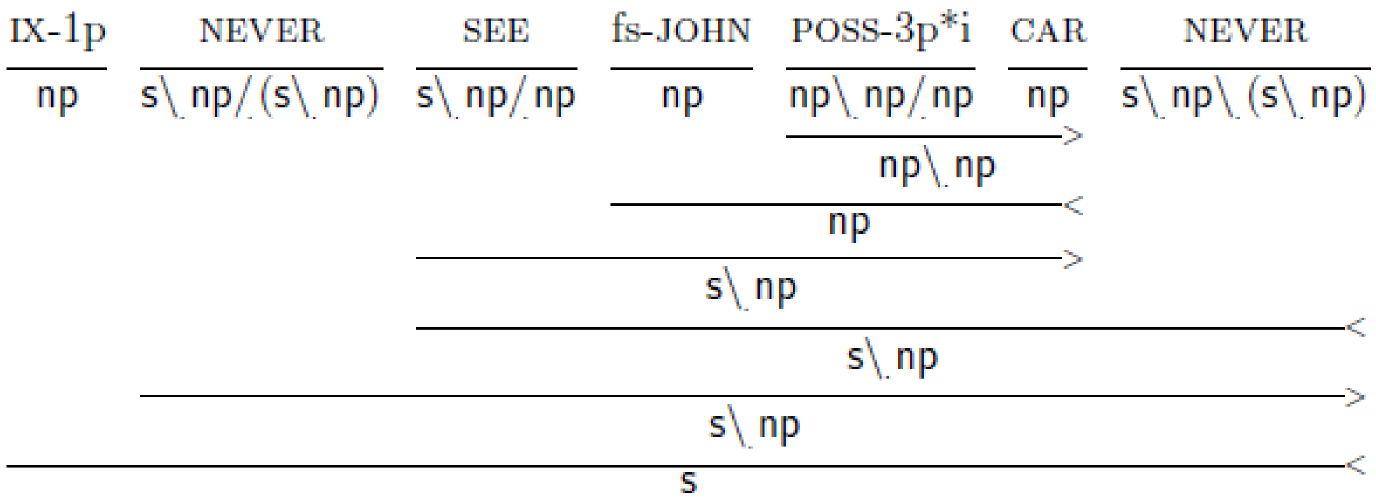
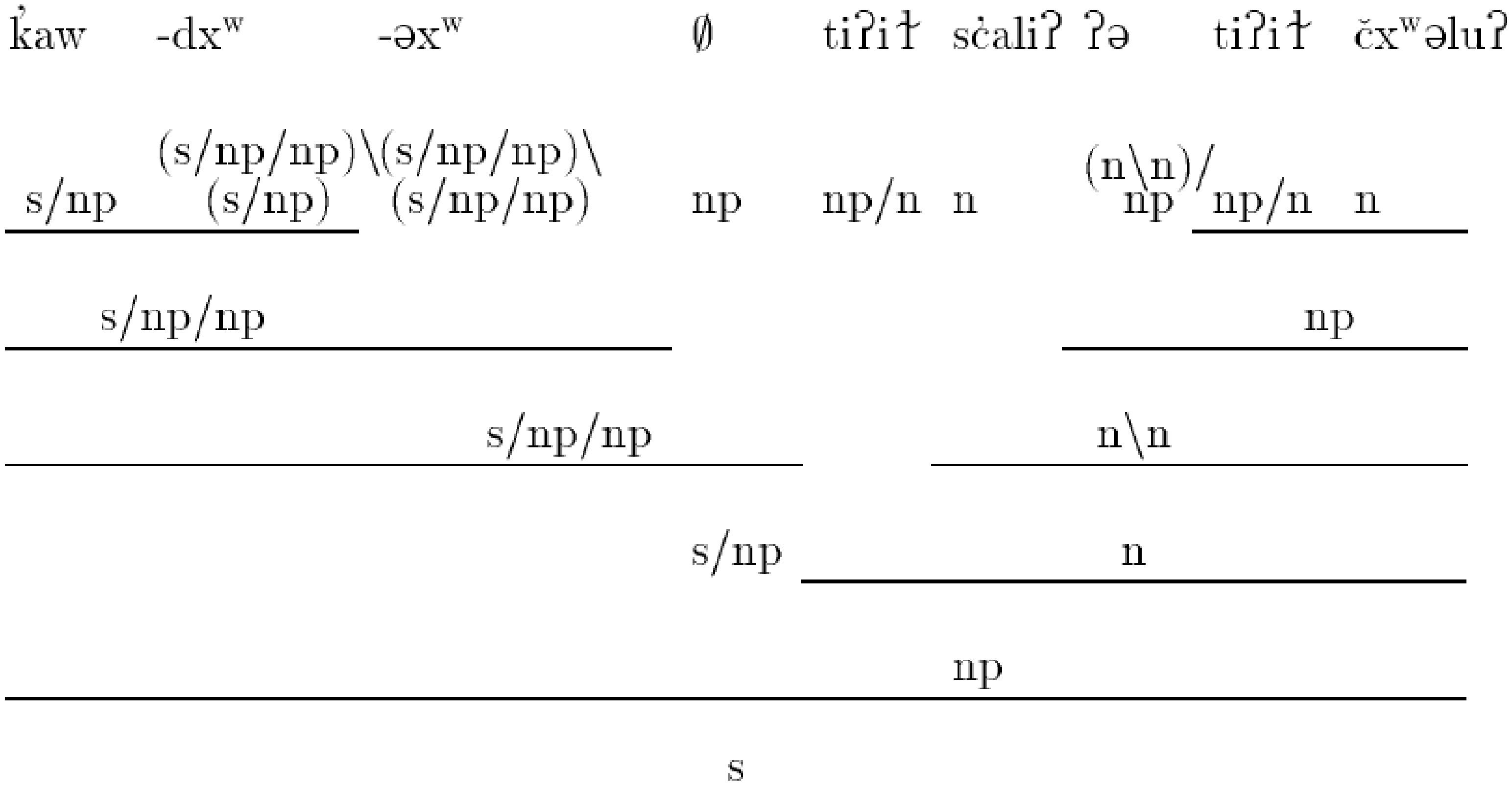


Figure 4.30: Sentence 794: Double negation

Lushootseed syntactic derivation

(Eng: *He chews on the heart of the whale.*)



Lushootseed syn/sem parse

k̓aw	-dxʷ	-əxʷ	∅	tiʔiɬ	sc̓aliʔ	ʔə	tiʔiɬ	čxʷəluʔ
s/np:GNAW	(s/np/np)\ (s/np) :OOC	(s/np/np)\ (s/np/np):NOW	np:X	np/n:DET	n:HEART	(n\n)/np:OF	np/n:DET	n:WHALE

s/np/np:OOC(GNAW)

np:DET(WHALE)

s/np/np:NOW(OOC(GNAW))

n\n:OF(DET(WHALE))

s/np:NOW(OOC(GNAW))(X)

n:OF(DET(WHALE))(HEART)

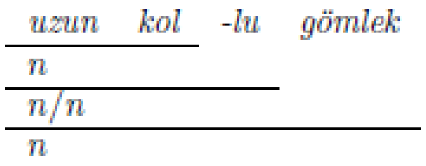
np:DET(OF(DET(WHALE))(HEART))

s:NOW(OOC(GNAW))(X)(DET(OF(DET(WHALE))(HEART)))

Turkish example (syntax and semantics)

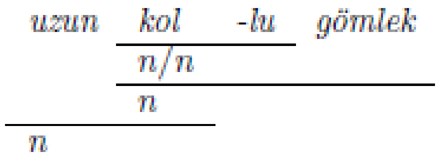
lexical entry	syntactic category	semantic category
uzun	n/n	$\lambda p.long(p(z))$
kol	n	$\lambda x.sleeve(x)$
-lu	$(n/n) \setminus n$	$\lambda q.\lambda r.r(y, has(q))$
gömlek	n	$\lambda w.shirt(w)$

(1a)



$shirt(y, has(long(sleeve(z))))$ = 'a shirt with long sleeves'

(1b)



$long(shirt(y, has(sleeve(z))))$ = 'a long shirt with sleeves'

Figure 1: Scope ambiguity of a nominal bound morpheme

TOOLS AND APPLICATIONS

Implementing a grammar

- Computational toolkit for instantiating computational grammars
  - Written in a programming language (Prolog, Java, C++)
  - Multiple-goal-directed problem solving
  - Pattern matching, backtracking
  - Some supports other parsing formalisms (HPSG, LFG)
- Define lexical items, syn/sem categories
- Select which application schemas, functions you want to invoke
- System takes input, parses and generates sentences

Sample rule and parse

```
tv(Rel) macro
  synsem:(forward,
    arg:(syn:np,
         sem:Y),
    res:(forward,
         arg:(syn:np,
              sem:X),
         res:(syn:s,
              sem:(Rel,
                   agent:Y,
                   theme:X))))),
  @ quantifier_free.
```

```
rec[7ugWECEd,CEL,ti7E7,hikW,spa7c].
QSTORE e_list
SYNSEM basic
   SEM DEF
       RESTR and
             CONJ1 BEAR
                   ARG1 [0] indiv.
             CONJ2 BIG
                   ARG1 [0]
       SCOPE SEEK
             AGENT pro1p
             THEME [0]
       VAR [0]
   SYN s
```

Applications, engines, resources, and tutorials

- Parsing languages (especially morphosyntactically complex ones)
- Modeling human incremental processing
- Information extraction
- Question answering
- Recognizing Textual Entailment
- Inference
- NL query for DB
- Text processing
- etc.

- OpenCCG
- Attribute Logic Engine
- Semantic parsing tutorials
- CCG bank: translation of Penn Treebank parses into CCG
  - At the comprehensive CCG Site

Finding out more…

- A nice overview is here

- Parsers:
  - http://yoavartzi.com/tutorial/
  - http://openccg.sourceforge.net/
  - http://www.cs.toronto.edu/~gpenn/ale.htm