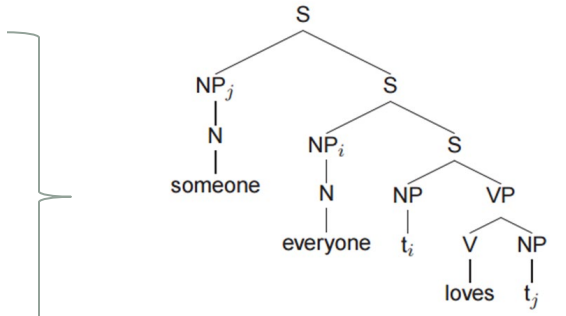
# PARSING SYNTAX AND SEMANTICS IN TANDEM

# (Morpho)syntax → semantics

- The interpretive approach
  - Morphology, syntax are prior
  - Map meaning *from*: morpho ➔ syn ➔ sem
  - Most current linguistic theories

Everybody loves someone.




$\exists y[\text{person}(y) \wedge \forall x[\text{person}(x) \rightarrow \text{love}(x, y)]]$

- 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

# Categorial Grammar

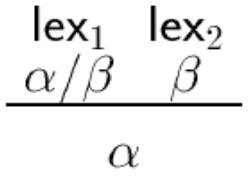
- Lexicalized grammatical formalism
  - The lexicon is primary and the driving force in determining structure.
- Since early 1940's, primarily in Europe
- Compositional syntax and semantics simultaneously
- Type-driven, combinatorial, categorial
- Particularly well suited for languages with complex morphosyntax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

# Syntactic categories in CG

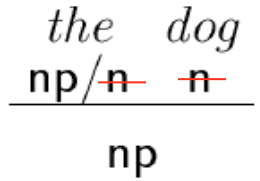
- Basic categories: np, n, and s
- Complex categories: composition of basic categories
  - α/β is a valid category if α and β are basic or complex categories
  - α\β is a valid category if α and β are basic or complex categories
- Example categories:
  - determiner: np/n
    - “A determiner can become a noun phrase if it can combine with a noun to the right (i.e. forwards).”
  - verb phrase: s\np
    - “A VP 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:

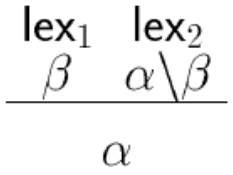



- Forward application example:

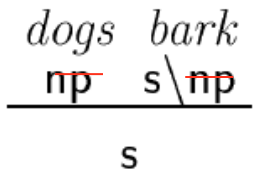



*the* can combine to the right with *dog* to create a noun phrase

- Backward application schema:




- Backward application example:



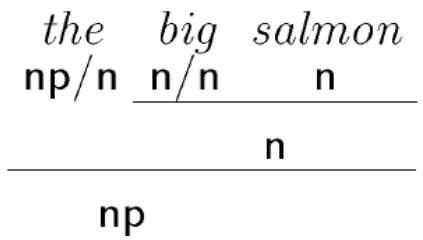

*bark* can combine to the left with *dogs* to create a sentence

# Example categories (syntax)

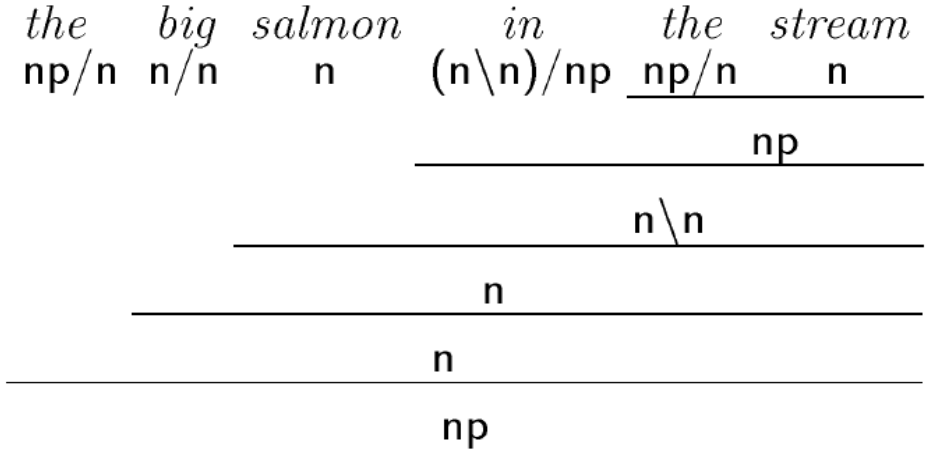
- determiner: np/n (*the*, *my*, *a*, *some*, *those*, etc.)
- predicate: s\np (*sneezed*, *ate tacos*, *is big*, etc.)
- adjective: n/n (*big*, *ugly*, *disappointed*, etc.)
- verb
  - intransitive: s\np (*yawns*, *somersaulted*, *died*, etc.)
  - transitive: (s\np)/np (*kicked*, *reads*,
  - ditransitive: (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

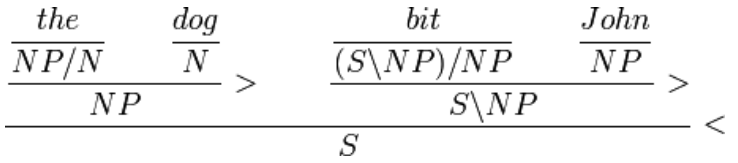
- Adjectival modification:

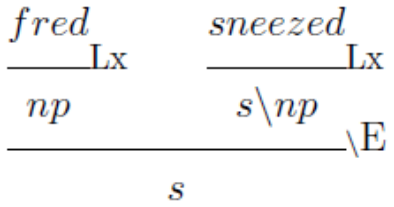


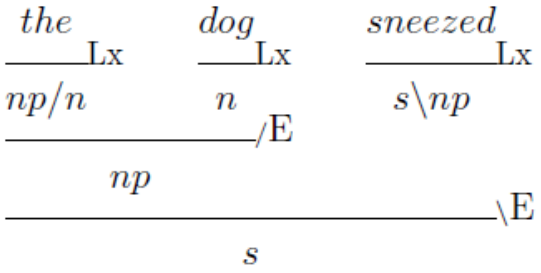

- Prepositional phrase modifier

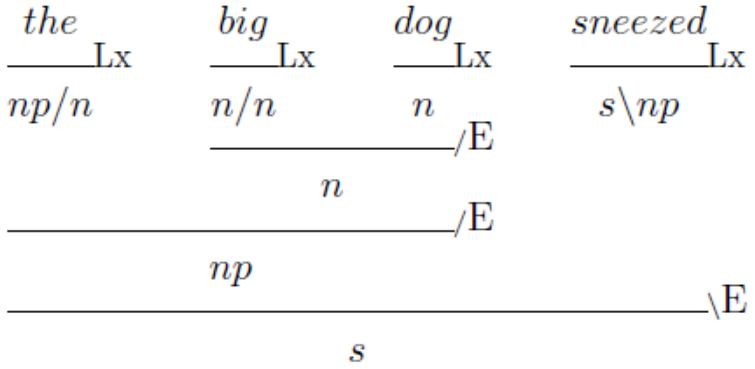


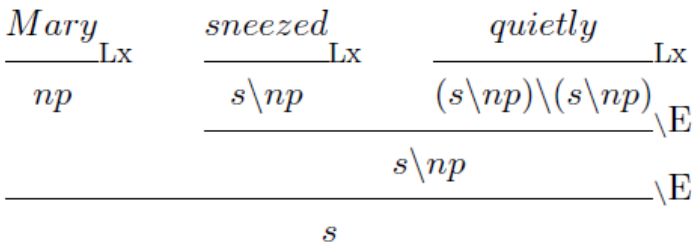
# Sentence parses

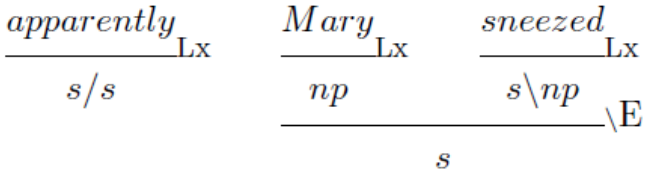


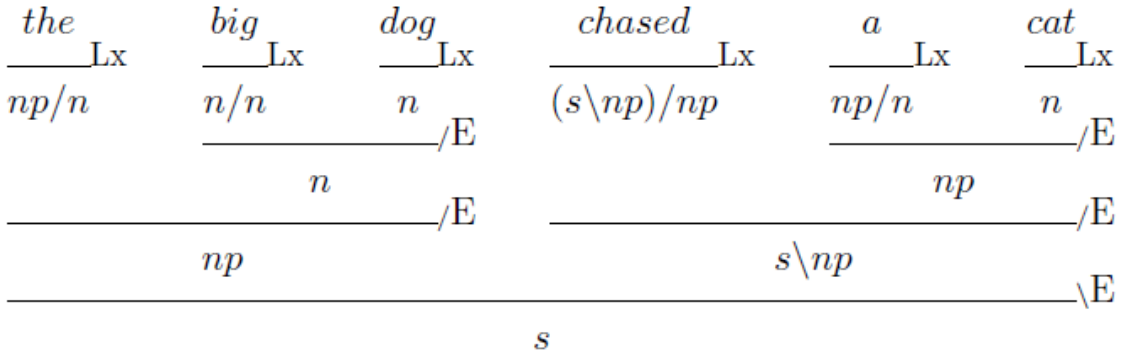



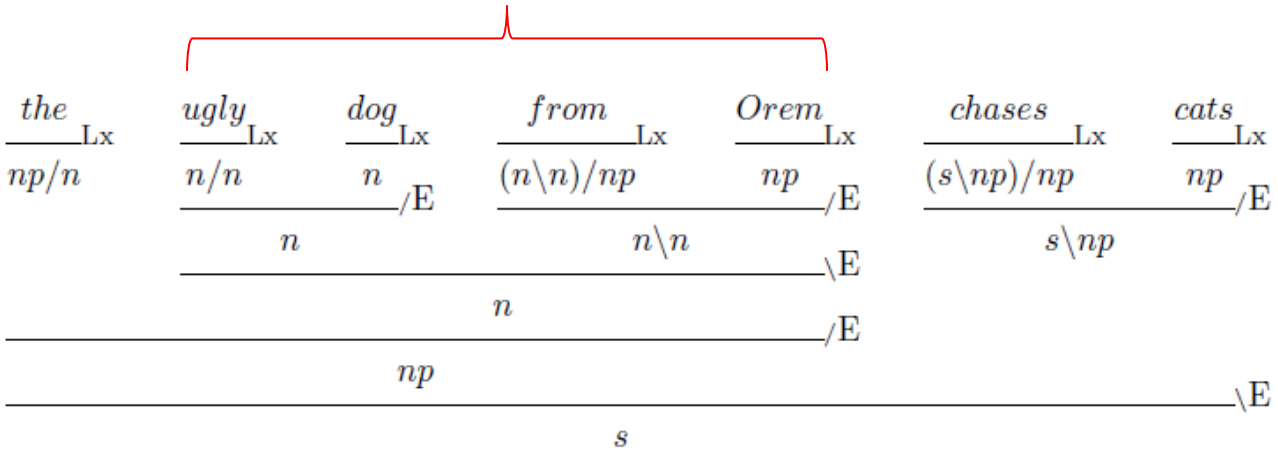



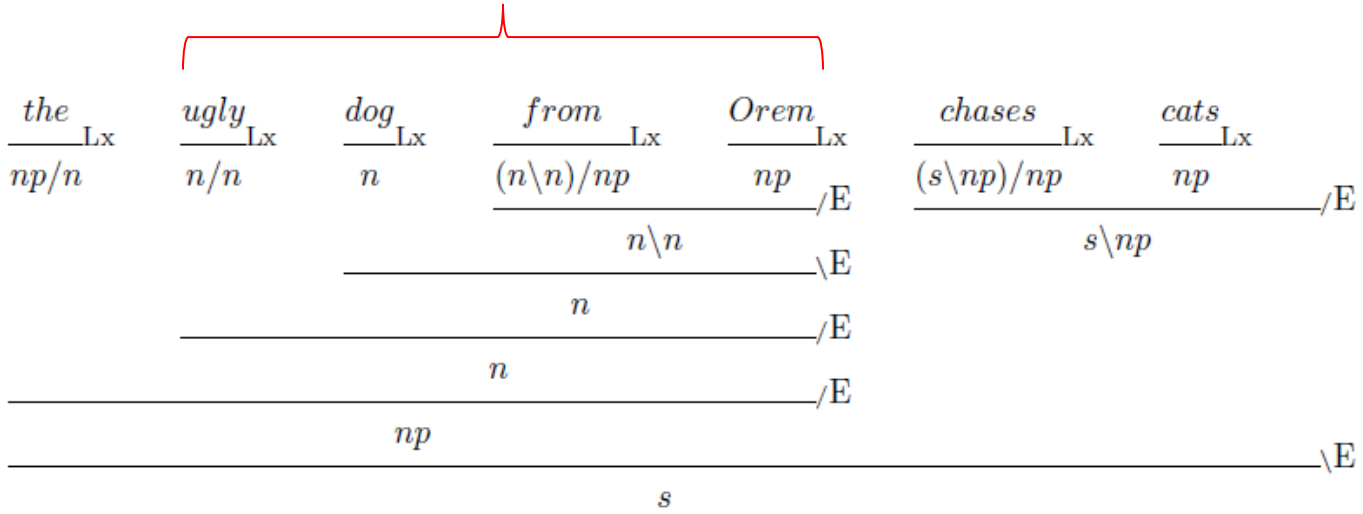



# Structural ambiguity

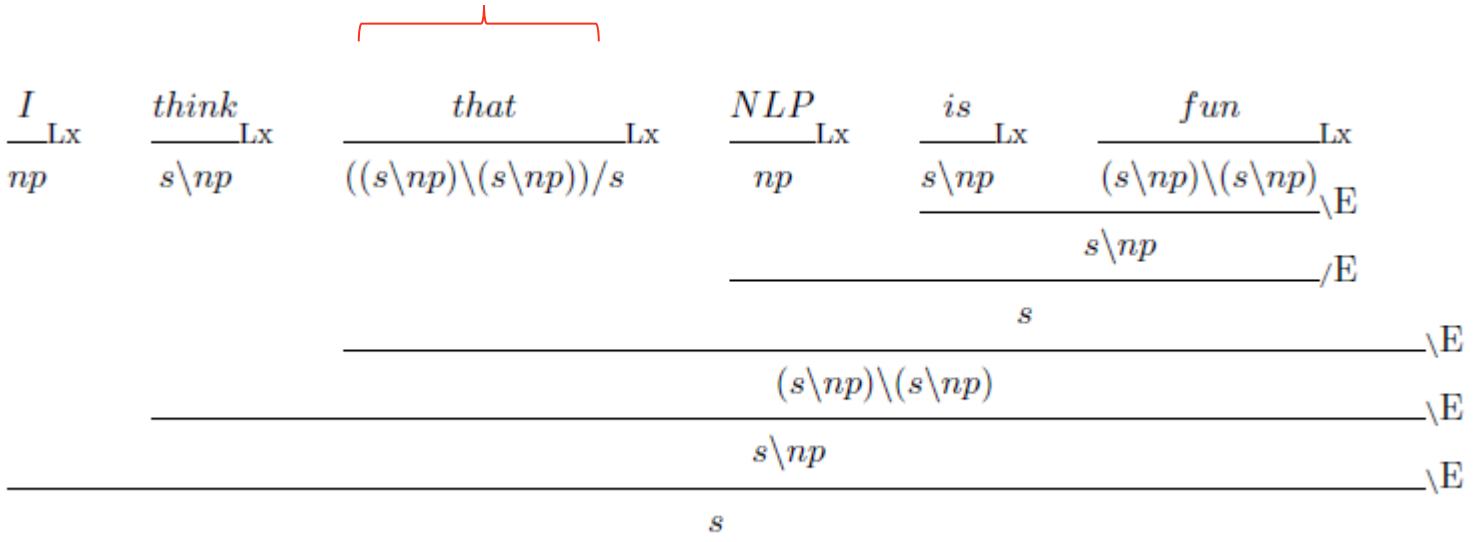
- In this case the ambiguity is spurious (no discernible meaning differences)

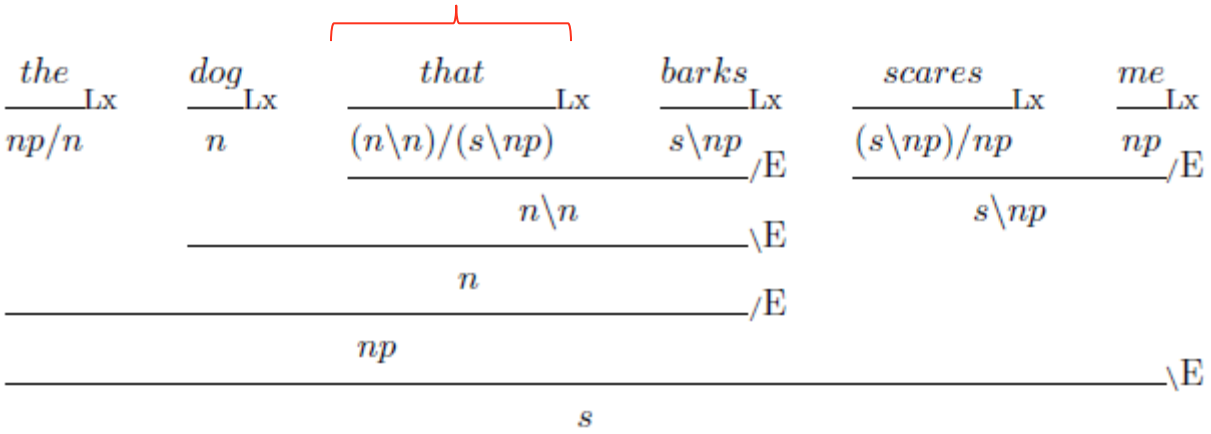


# More complex categories

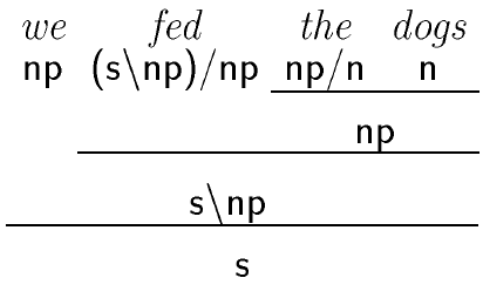
- complementizer: ((s\np)\(s\np))/s
- relative pronoun: (n\n)/(s\np)



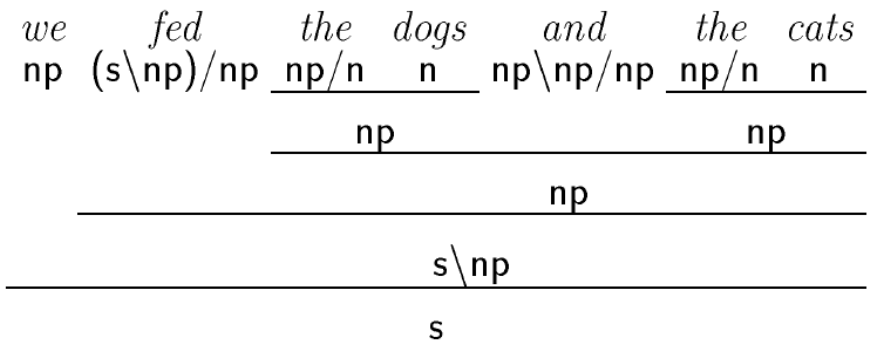



# Conjunction

- Simple




- Conjoined object



# Beyond classical categorial grammar

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

Forward function composition

$$\frac{X/Y \qquad Y/Z}{X/Z}\,B_{>}$$

Right node raising:
*John likes and Bill detests broccoli.*

Backward function composition

$$\frac{Y\backslash Z \qquad X\backslash Y}{X\backslash Z}\,B_{<}$$

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

(6) a woman whom Dexter thinks that Warren likes

$(N\backslash N)/(S/NP)$ $S/(S\backslash NP)$ $(S\backslash NP)/S'$ $S'/S$ $S/(S\backslash NP)$ $(S\backslash NP)/NP$

$>\mathbf{T}$ $>\mathbf{T}$

$>\mathbf{B}$ → $S/S'$

$>\mathbf{B}$ → $S/S$

$>\mathbf{B}$ → $S/(S\backslash NP)$

$>\mathbf{B}$ → $S/NP'$

$>$ → $N\backslash N$

# 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."

$$\frac{the}{NP/N}$$

$$\dfrac{\dfrac{\dfrac{the}{NP/N}\quad\dfrac{dog}{N}}{NP}>}{S/(S\backslash NP)}T_>$$

$$\dfrac{\dfrac{\dfrac{\dfrac{the}{NP/N}\quad\dfrac{dog}{N}}{NP}>}{S/(S\backslash NP)}T_>\quad\dfrac{bit}{(S\backslash NP)/NP}}{S/NP}B_>$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{the}{NP/N}\quad\dfrac{dog}{N}}{NP}>}{S/(S\backslash NP)}T_>\quad\dfrac{bit}{(S\backslash NP)/NP}}{S/NP}B_>\quad\dfrac{John}{NP}}{S}>$$

# 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

# Semantics

- Lambda calculus
- Lexical entries: as before, but with lambda expressions
- Semantic composition: function-argument structure

$$\frac{\frac{Fred}{np{:}fred}\,\mathrm{Lx} \qquad \frac{sneezed}{s\backslash np{:}\lambda x.sneezed(x)}\,\mathrm{Lx}}{\begin{array}{c}\mathrm{s}{:}\lambda x.sneezed(x)(fred) \\ \mathrm{s}{:}sneezed(fred)\end{array}}$$

λ-reduction

# Lambda reduction λx (plus extension to λP)

- Instantiating a variable with an individual

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

justification

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

- Instantiating a predicate variable with a predicate

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

# 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)

# Type-logical semantics

- Very tight coupling of morpho + syn + sem
  - All in tandem, no priority assumed
- Primitives expressed via predicate calculus
  - Variablized expressions with appropriate arity
- Lexical items feed semantic component
- Composition via lambda operators
- Draws heavily on formal logic principles (proof theory, lambda calculus)
- Multiple licensed parses: ambiguity

# Advantages

- Predicate logic formalism
  - Quantification, negation, constants, variables, etc.
  - Well-defined vocabulary and other machinery
  - Translatable to other semantic/pragmatic formalisms (DRS's, LCS's, feature structures)
- Full theorem-proving apparatus is supported
  - Ellipsis recovery, empty categories, discontinuous constituents, constituent raising
  - Pragmatic inferences, presuppositions, tracking common ground, truth-value computations, etc.
- Integrating higher-order logics is possible (intensionality, event semantics, possible worlds)

# Intransitive sentences

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

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

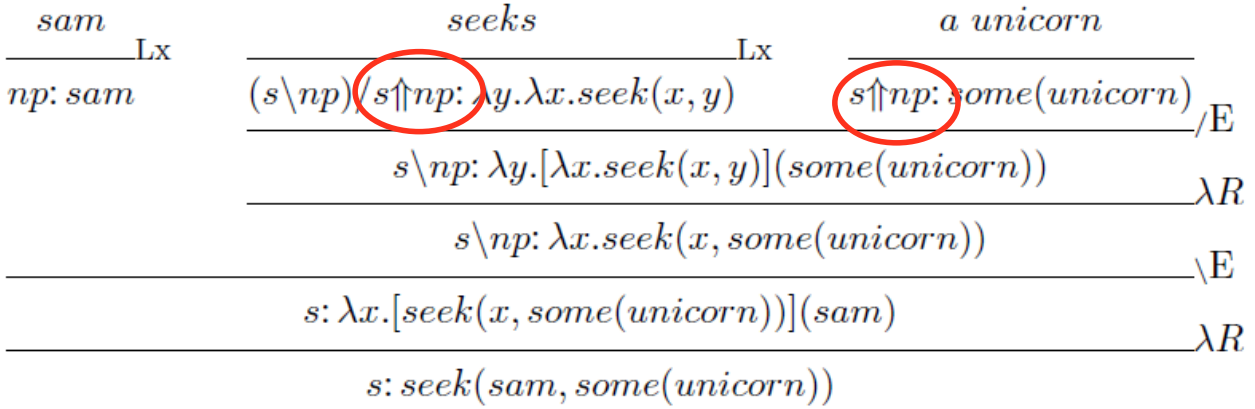
# Transitive sentences

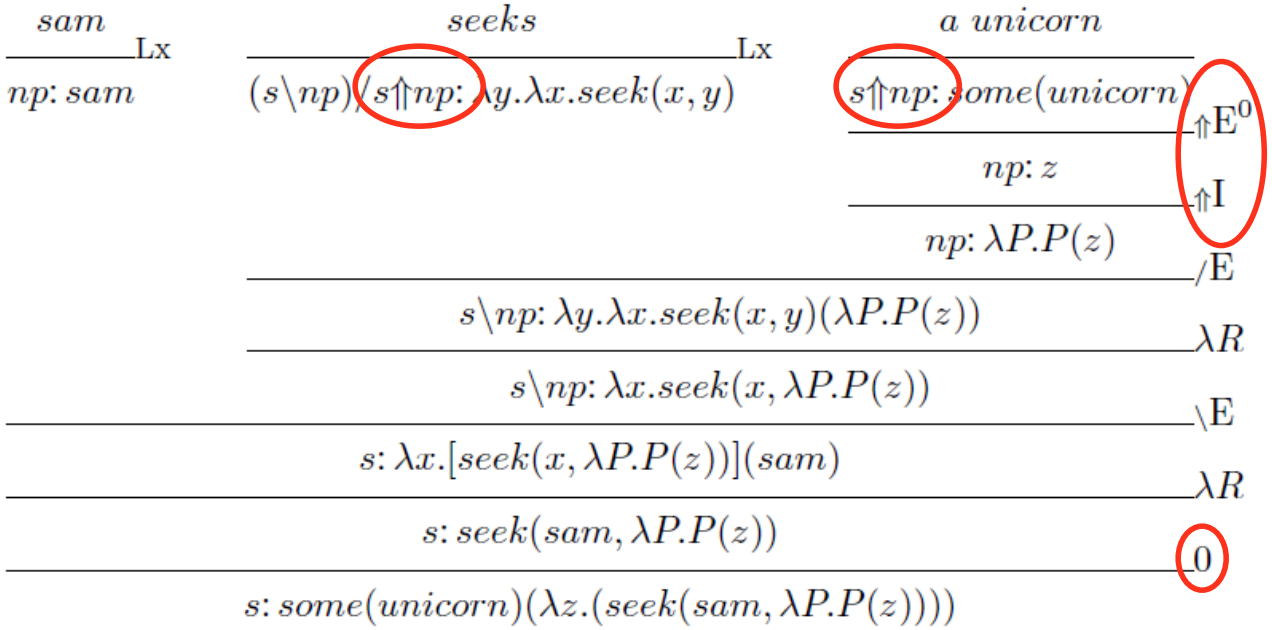
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{fido}{np\colon fido}\text{Lx} \quad \dfrac{\dfrac{chased}{(s\backslash np)/np\colon \lambda y.\lambda x.chased(x,y)}\text{Lx} \quad \dfrac{fred}{np\colon fred}\text{Lx}}{s\backslash np\colon \lambda y.\lambda x.[chased(x,y)](fred)}/\text{E}}{s\backslash np\colon \lambda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

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

# Intensionality and ambiguity

- Push/pop a quantifier
  - Use a memory unit (aka Cooper store)
- Assume generalized quantifiers (i.e. quantifiers as predicates)
- Two possible representations



# 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

# ASL parse

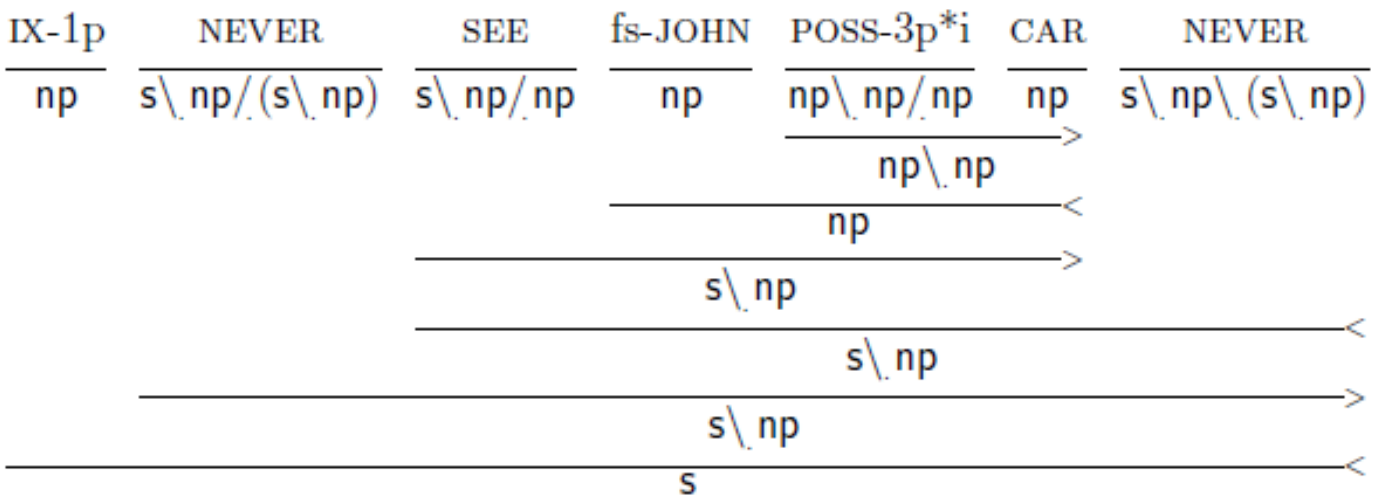



Figure 4.30: Sentence 794: Double negation

# 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

# 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)

$$\dfrac{\dfrac{\dfrac{uzun \quad kol}{n} \quad -lu}{n/n} \quad gömlek}{n}$$

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

(1b)

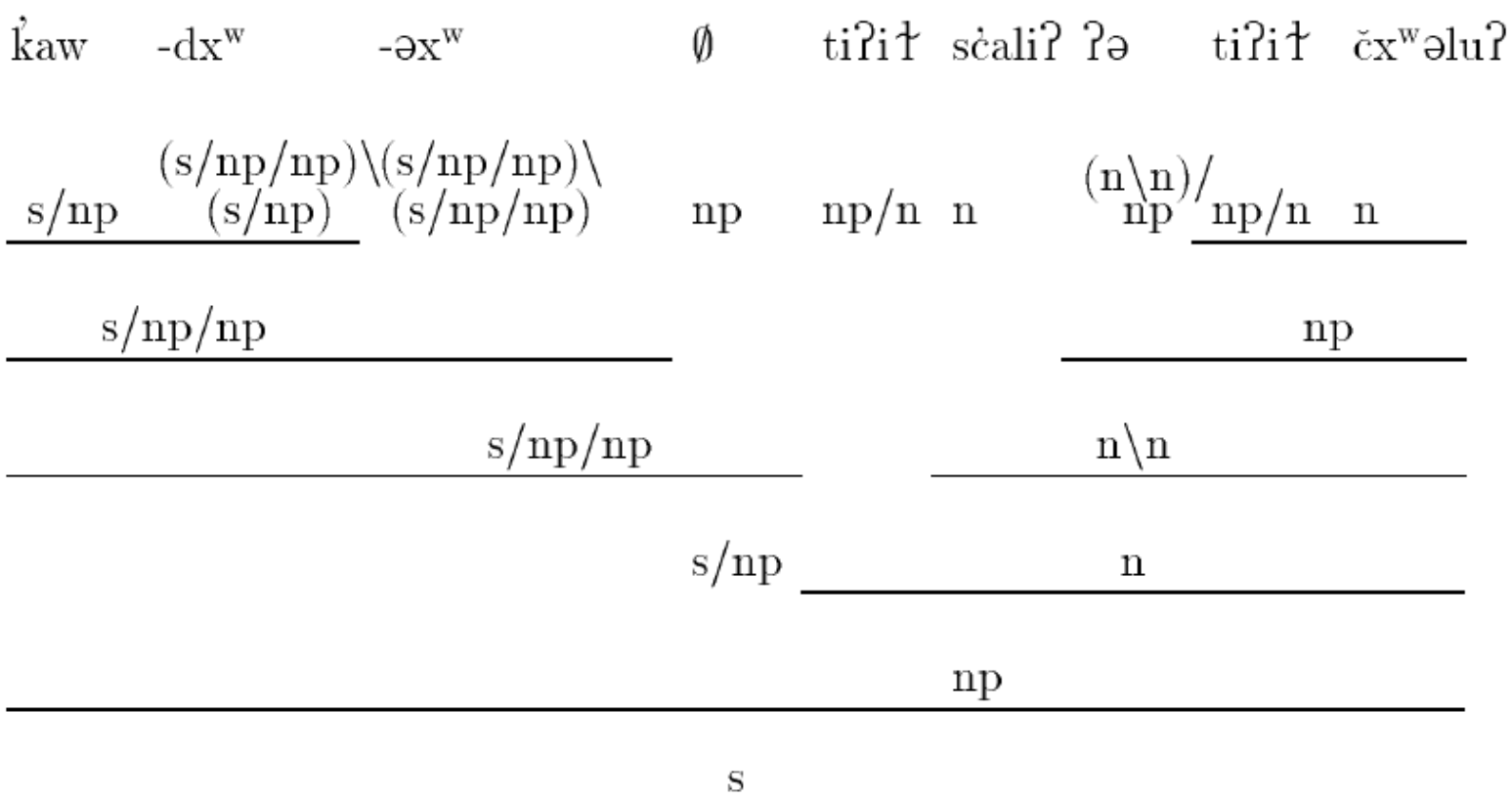
$$\dfrac{uzun \quad \dfrac{kol \quad -lu}{n/n} \quad gömlek}{n}$$

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

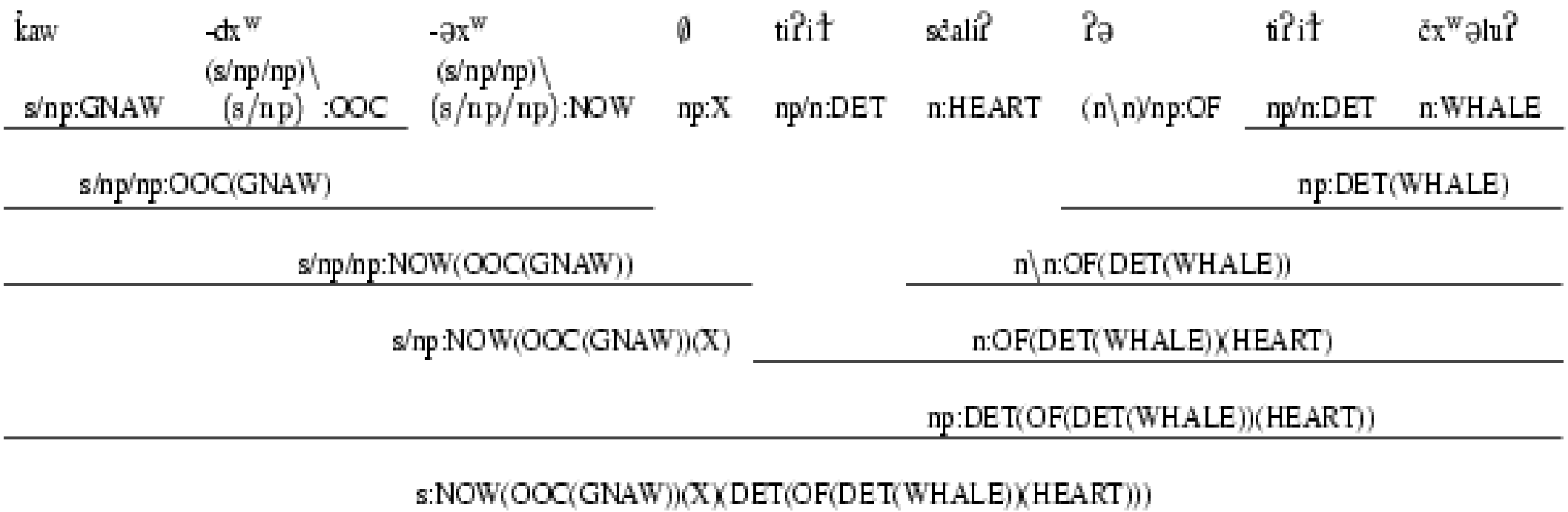
Figure 1: Scope ambiguity of a nominal bound morpheme

# Lushootseed syntactic derivation

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



# Lushootseed syn/sem parse



# Implementing a grammar

- Computational toolkits exist for instantiating computational CG grammars
  - Written in a programming language (Prolog, Java, C++)
  - Multiple-goal-directed problem solving
  - Pattern matching, backtracking
  - Some support 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