# CMP462: Natural Language Processing

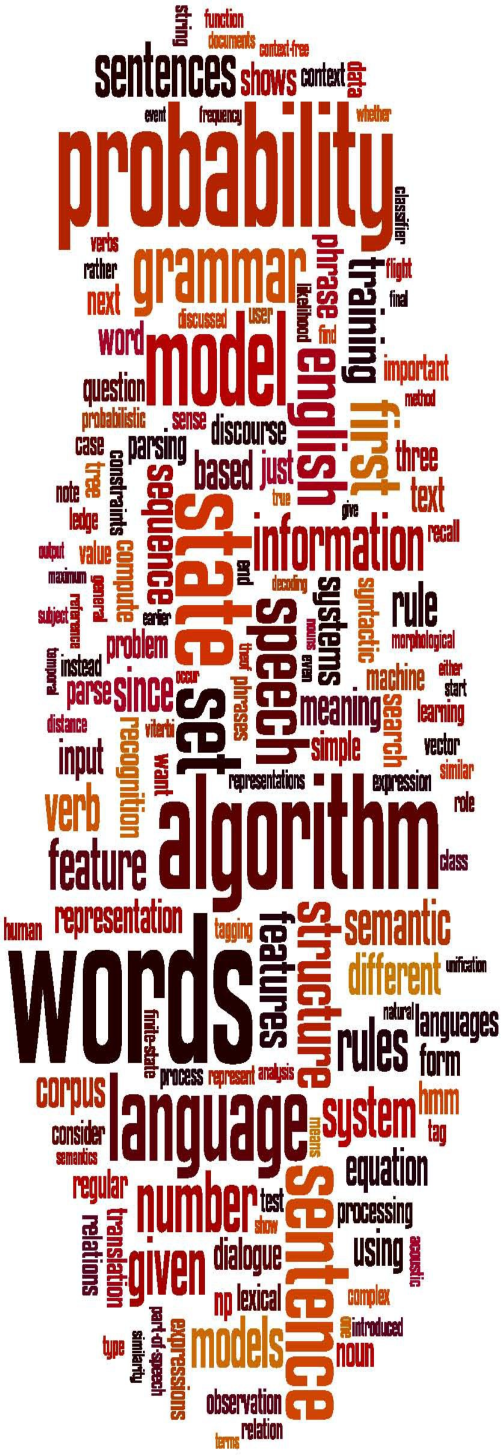
## Lecture 08: CFGs and PCFGs

Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Spring 2013

# Agenda

- Two views of syntactic structures
  - Constituency Parsing
  - Dependency Parsing
- Exponential number of trees
- Context Free Grammars (CFGs)
- Probabilistic Context Free Grammars (PCFGs)
- Chomsky Normal Form

**Acknowledgment:**
Most slides adapted from Chris Manning and Dan Jurafsky's NLP class on Coursera.

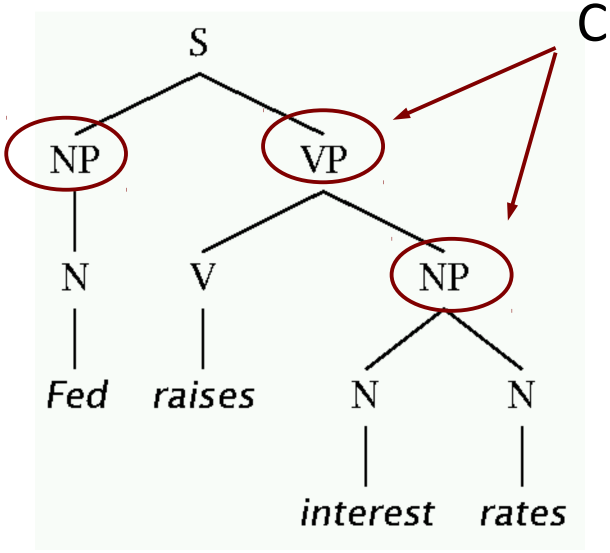


# Statistical Natural Language Parsing

Two views of syntactic structure

# Two views of linguistic structure: 1. Constituency (phrase structure)

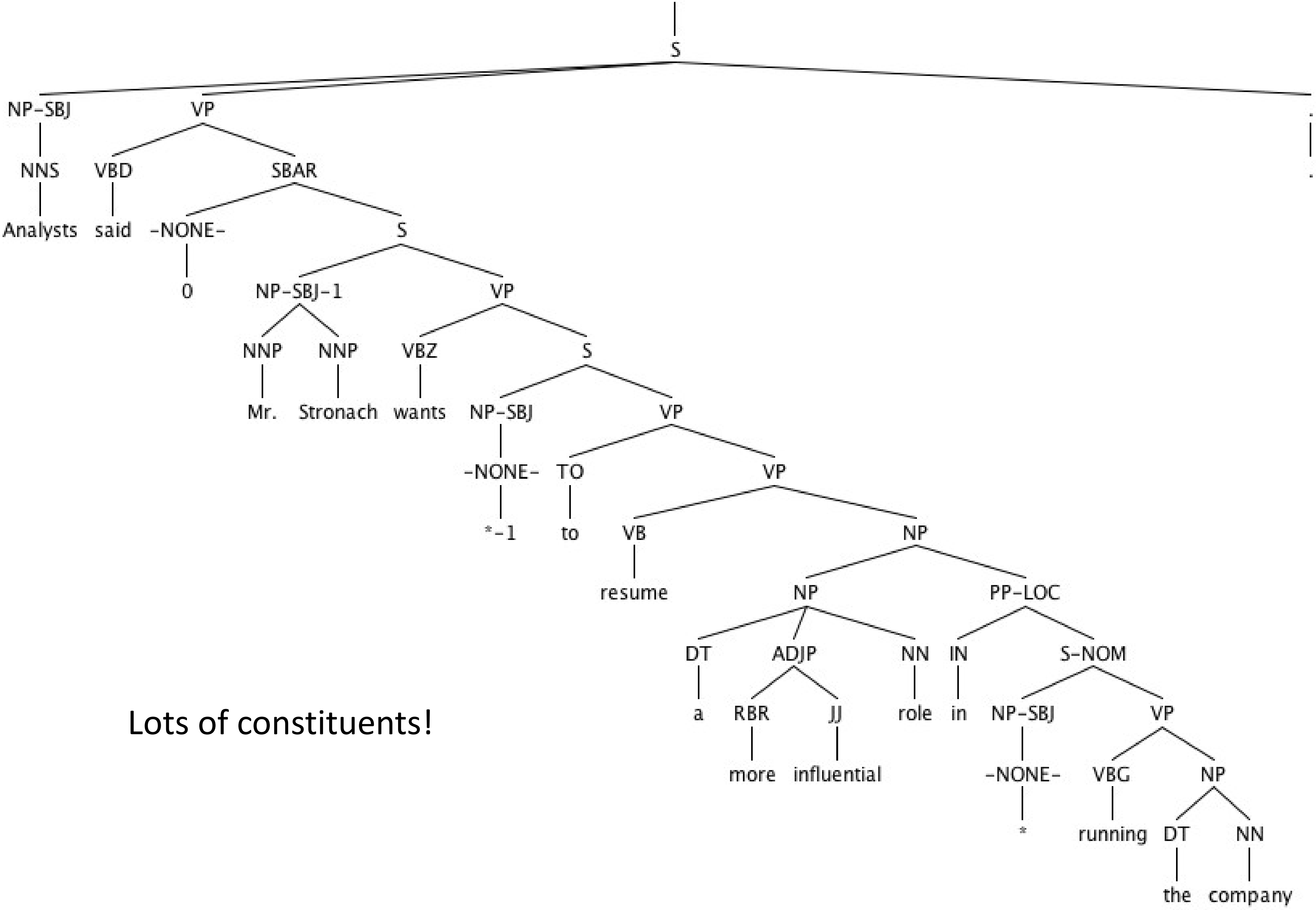
- Phrase structure organizes words into nested constituents.



Constituents

# Two views of linguistic structure: 1. Constituency (phrase structure)

- Phrase structure organizes words into nested constituents.
- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
  - Distribution: a constituent behaves as a unit that can appear in different places:
    - John talked [to the children] [about drugs].
    - John talked [about drugs] [to the children].
    - *John talked drugs to the children about (WRONG)
  - Substitution/expansion/pro-forms:
    - I sat [on the box/right on top of the box/there].
  - Coordination, regular internal structure, no intrusion, fragments, semantics, …



Lots of constituents!

# Headed phrase structure

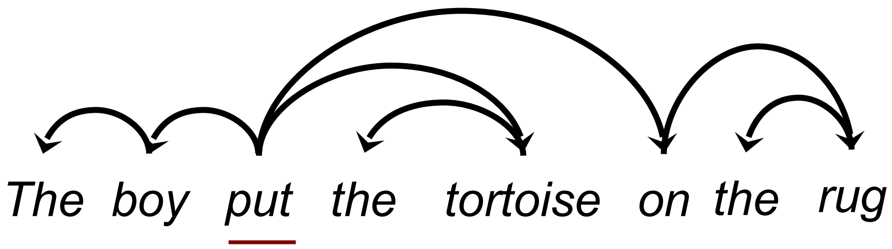
- Define constituents by *production rules* centered around *head* components

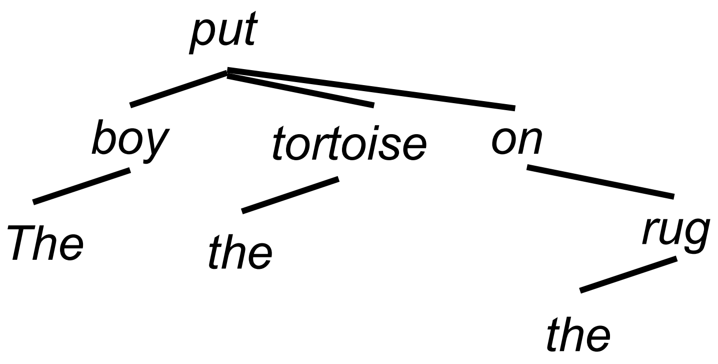
- VP → … VB* …
- NP → … NN* …
- ADJP → … JJ* …
- ADVP → … RB* …

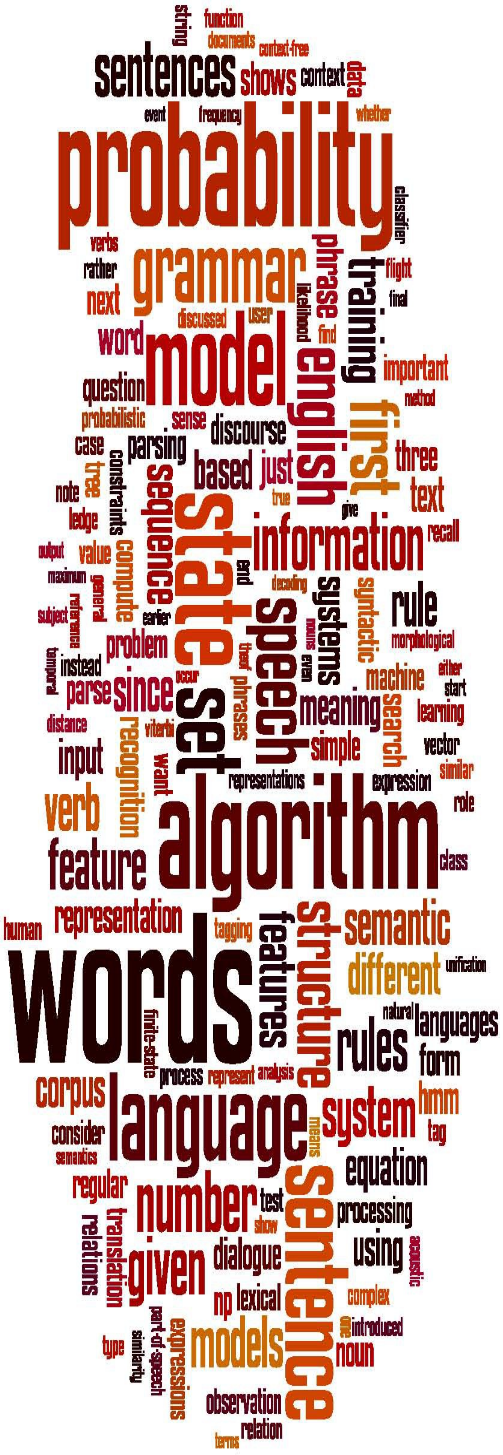
- Plus minor phrase types:
  - QP (quantifier phrase in NP), CONJP (multi word constructions: *as well as*), INTJ (interjections), etc.

# Two views of linguistic structure: 2. Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.







# Statistical Natural Language Parsing

Parsing: The rise of data and statistics

# Pre 1990 (“Classical”) NLP Parsing

- Wrote symbolic grammar e.g. Context Free Grammar (CFG) or often richer and lexicon

  S → NP VP<br>
  NP → (DT) NN<br>
  NP → NN NNS<br>
  NP → NNP<br>
  VP → V NP

  NN → *interest*<br>
  NNS → *rates*<br>
  NNS → *raises*<br>
  VBP → *interest*<br>
  VBZ → *rates*

  - Used grammar/proof systems to prove parses from words

- This scaled very badly and didn’t give coverage. For sentence:

  *Fed raises interest rates 0.5% in effort to control inflation*

| | |
|---|---|
| – Minimal grammar: | 36 parses |
| – Simple 10 rule grammar: | 592 parses |
| – Real-size broad-coverage grammar: | millions of parses |

# Classical NLP Parsing: The problem and its solution

- Categorical constraints can be added to grammars to limit unlikely/weird parses for sentences
  - But the attempt make the grammars not robust
    - In traditional systems, commonly 30% of sentences in even an edited text would have *no* parse.
- A less constrained grammar can parse more sentences
  - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
  - Statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but still quickly find the best parse(s)

# The rise of annotated data: The Penn Treebank

[Marcus et al. University of Pennsylvania 1993, *Computational Linguistics*]

A lot of sentences annotated with their structures

```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

# The rise of annotated data

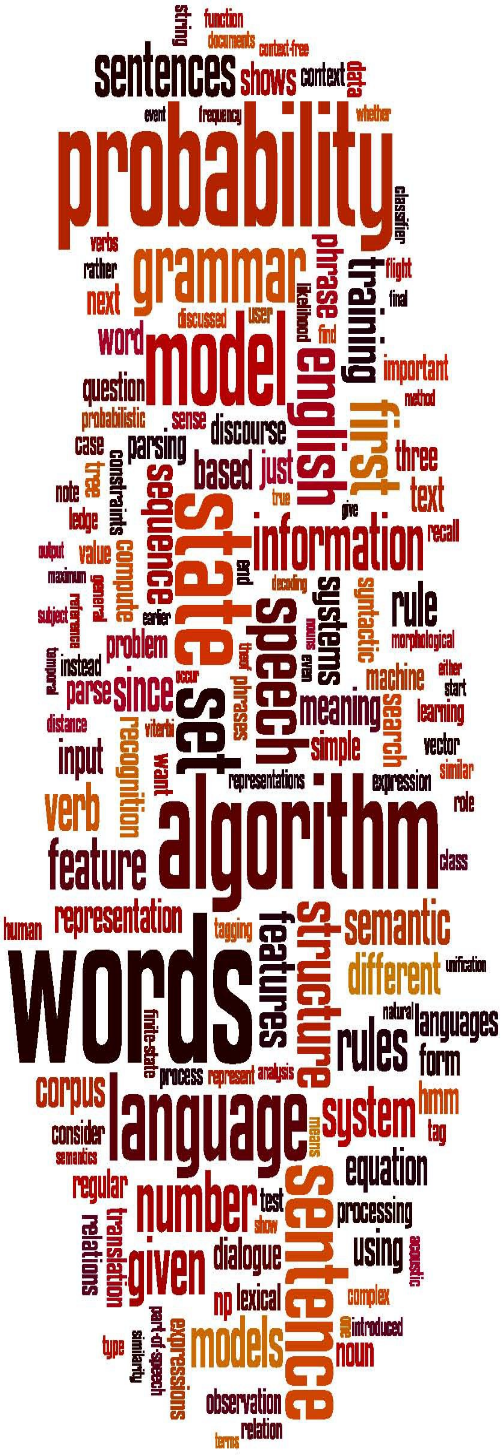
- Starting off, building a **treebank** seems a lot slower and less useful than building a **grammar**

- But a treebank gives us many things
  - Reusability of the labor
    - Many parsers, POS taggers, etc.
    - Valuable resource for linguistics
  - Broad coverage
  - Frequencies and distributional information
  - A way to evaluate systems

# Statistical parsing applications

Statistical parsers are now robust and widely used in larger NLP applications:

- High precision question answering [Pasca and Harabagiu SIGIR 2001]
- Improving biological named entity finding [Finkel et al. JNLPBA 2004]
- Syntactically based sentence compression [Lin and Wilbur 2007]
- Extracting opinions about products [Bloom et al. NAACL 2007]
- Improved interaction in computer games [Gorniak and Roy 2005]
- Helping linguists find data [Resnik et al. BLS 2005]
- Source sentence analysis for machine translation [Xu et al. 2009]
- Relation extraction systems [Fundel et al. *Bioinformatics* 2006]

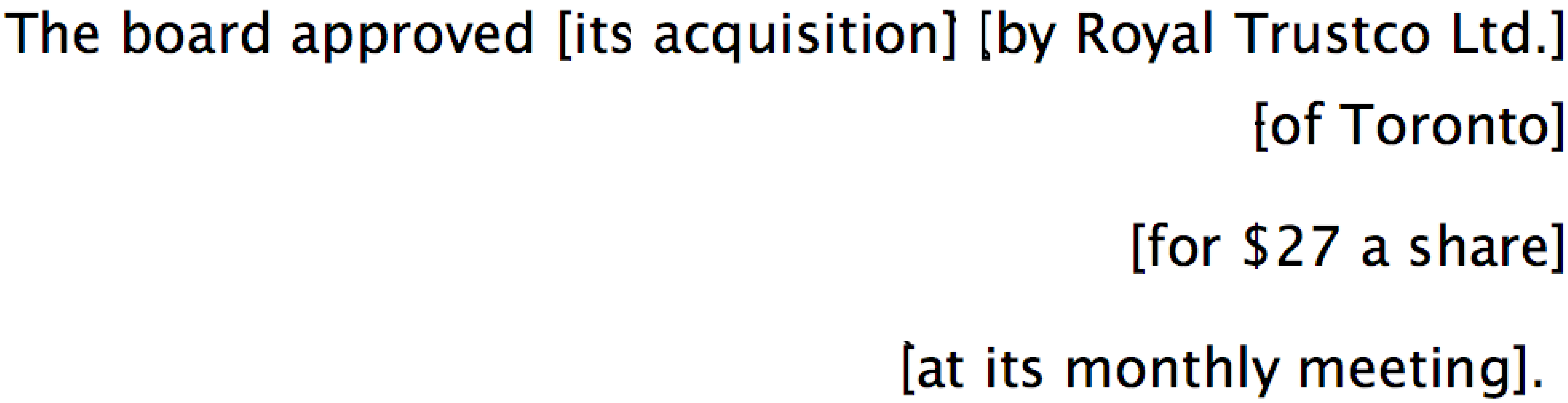


# Statistical Natural Language Parsing

An exponential number of attachments

# Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.



The board approved [its acquisition] [by Royal Trustco Ltd.]

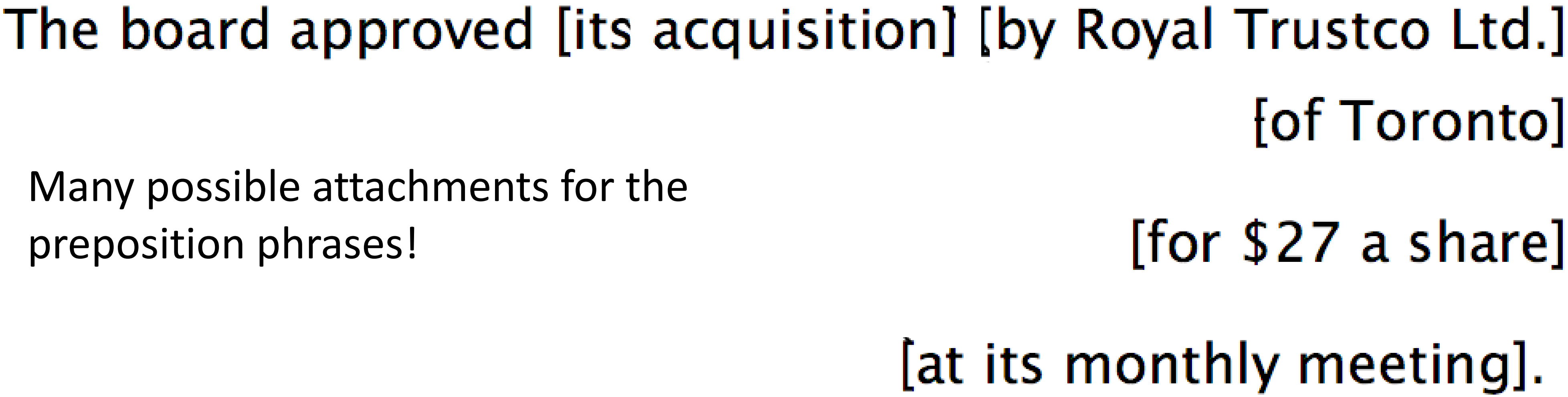
[of Toronto]

[for $27 a share]

[at its monthly meeting].

# Attachment ambiguities

- A key parsing decision is how we 'attach' various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.

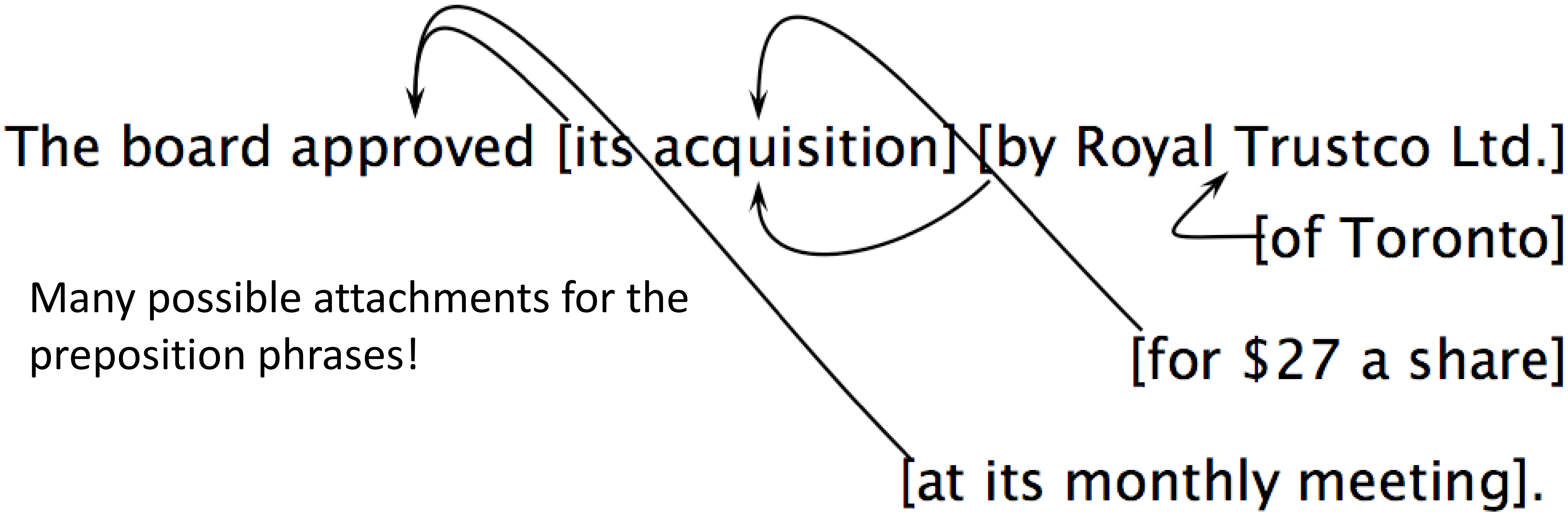


The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for $27 a share]
[at its monthly meeting].

Many possible attachments for the preposition phrases!

# Attachment ambiguities

- A key parsing decision is how we 'attach' various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.



The board approved [its acquisition] [by Royal Trustco Ltd.]

[of Toronto]

[for $27 a share]

[at its monthly meeting].

Many possible attachments for the preposition phrases!

Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$

- An exponentially growing series, which arises in many tree-like contexts e.g.
  - the number of possible triangulations of a polygon with $n$+2 sides

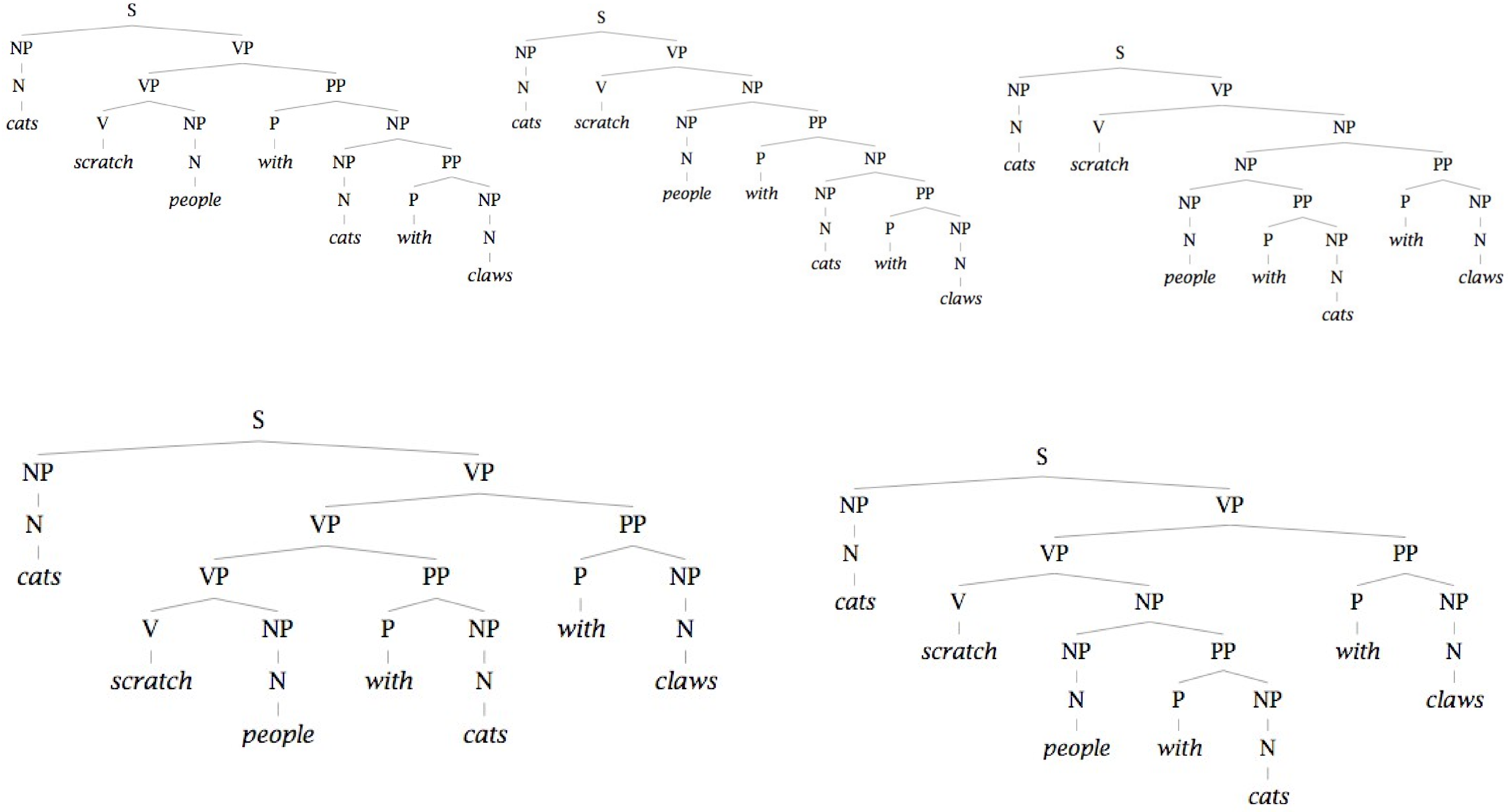
# Quiz Question!

- How many distinct parses does the following sentence have due to PP attachment ambiguities?
  - A PP can attach to any preceding V or N within the verb phrase, subject only to the parse still being a tree.
    - (This is equivalent to there being no crossing dependencies, where if *d*2 is a dependent of *d*1 and *d*3 is a dependent of *d*2, then the line *d*2–*d*3 begins at *d*2 under the line from *d*1 to *d*2.)

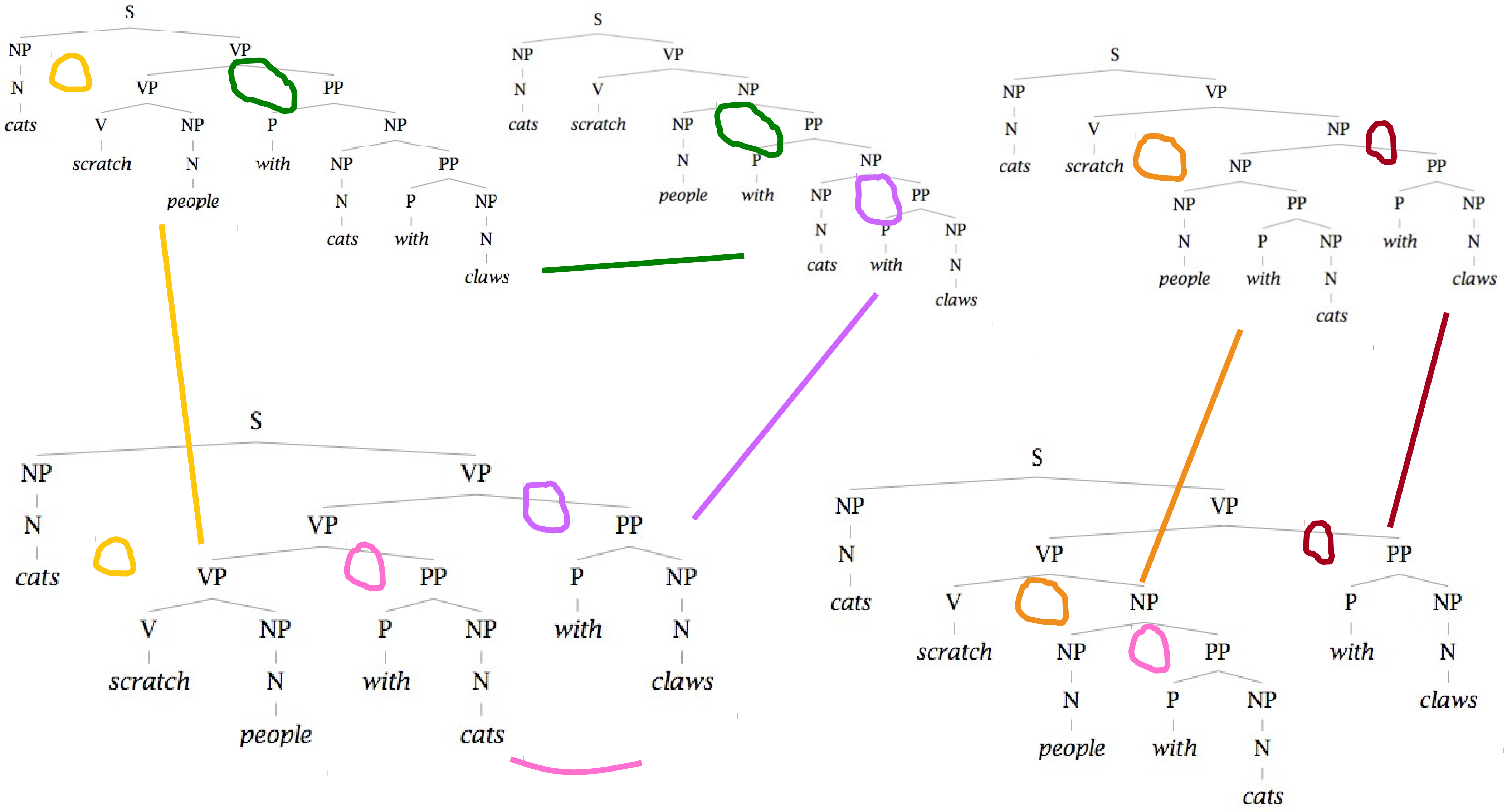
John wrote the book with a pen in the room.

5

# Two problems to solve: 1. Repeated work…



# Two problems to solve:
# 1. Repeated work…



# Two problems to solve:
# 2. Choosing the correct parse

- How do we work out the correct attachment:

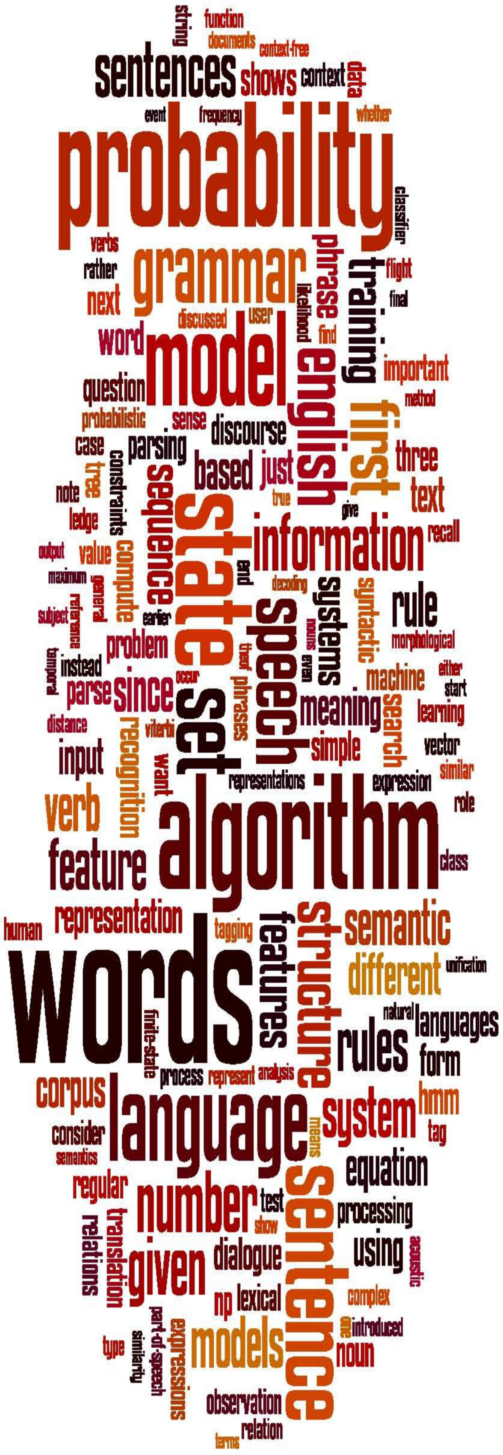
  She saw the man with a telescope

- Words are good predictors of attachment
  - Even absent full understanding

    – Moscow sent more than 100,000 soldiers into Afghanistan …

    – Sydney Water breached an agreement with NSW Health …

- Our statistical parsers will try to exploit such statistics.



# CFGs and PCFGs

(Probabilistic)
Context-Free
Grammars

# A phrase structure grammar

S → NP VP

VP → V NP

VP → V NP PP

NP → NP NP

NP → NP PP

NP → N

NP → *e*

PP → P NP

N → people

N → fish

N → tanks

N → rods

V → people

V → fish

V → tanks

P → with

*people fish tanks*

*people fish with rods*

# Phrase structure grammars = context-free grammars (CFGs)

- G = (T, N, S, R)
  - T is a set of terminal symbols e.g. fish, people
  - N is a set of nonterminal symbols e.g. NP, VP
  - S is the start symbol $(S \in N)$
  - R is a set of rules/productions of the form $X \rightarrow \gamma$ where $X \in N$ and $\gamma \in (N \cup T)^*$
    e.g. S → NP VP

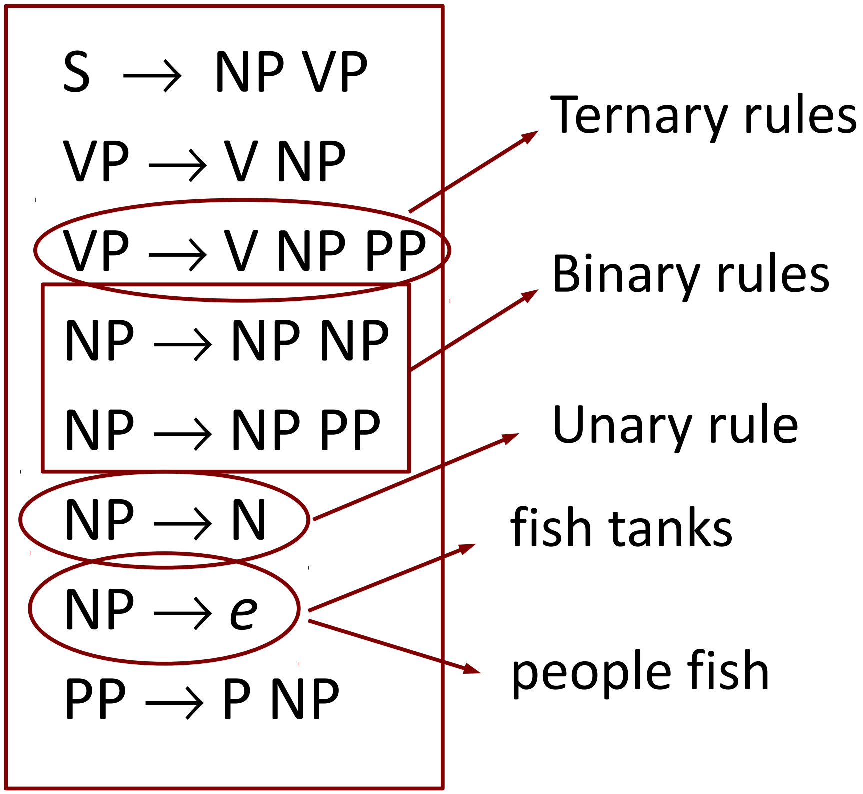
- A grammar G generates a language L.

# Phrase structure grammars in NLP

- G = (T, C, N, S, L, R)
  - T is a set of terminal symbols
  - C is a set of preterminal symbols e.g. DT, N
  - N is a set of nonterminal symbols
  - S is the start symbol ($S \in N$)
  - L is the lexicon, a set of items of the form $X \rightarrow x$
    - $X \in C$ and $x \in T$ e.g. DT → the, N → fish
  - R is the grammar, a set of items of the form $X \rightarrow \gamma$
    - $X \in N$ and $\gamma \in (N \cup C)^*$
- By usual convention, S is the start symbol, but in statistical NLP, we usually have an extra node at the top (ROOT, TOP), in cases when Treebanks contain non-sentence structures e.g. prepositional phrase
- We usually write *e* for an empty sequence, rather than nothing

# A phrase structure grammar

Grammar



S → NP VP

VP → V NP

VP → V NP PP

NP → NP NP

NP → NP PP

NP → N

NP → *e*

PP → P NP

Ternary rules

Binary rules

Unary rule

fish tanks

people fish

N → people

N → fish

N → tanks

N → rods

V → people

V → fish

V → tanks

P → with

*people fish tanks*

*people fish with rods*

Lexicon

# Probabilistic – or stochastic – context-free grammars (PCFGs)

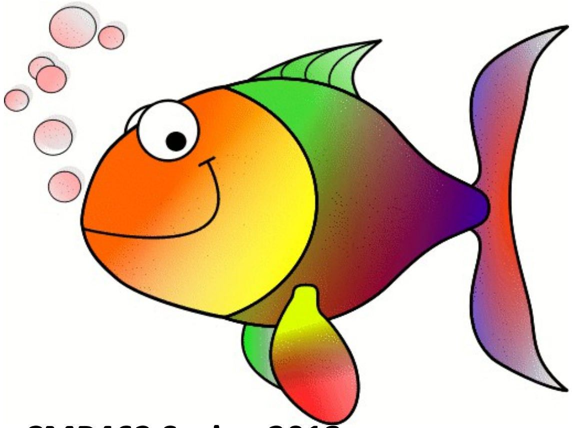
- G = (T, N, S, R, P)
  - T is a set of terminal symbols
  - N is a set of nonterminal symbols
  - S is the start symbol (S ∈ N)
  - R is a set of rules/productions of the form X → γ
  - P is a probability function
    - P: R → [0,1]
    - $\forall X \in N, \sum_{X \to \gamma \in R} P(X \to \gamma) = 1$
- A grammar G generates a language model L.

$$\sum_{\gamma \in T^*} P(\gamma) = 1$$

# A PCFG

| | | |
|---|---|---|
| S → NP VP | 1.0 | |
| VP → V NP | 0.6 | Sum to 1 |
| VP → V NP PP | 0.4 | |
| NP → NP NP | 0.1 | Sum to 1 |
| NP → NP PP | 0.2 | |
| NP → N | 0.7 | |
| PP → P NP | 1.0 | |

| | |
|---|---|
| N → *people* | 0.5 |
| N → *fish* | 0.2 |
| N → *tanks* | 0.2 |
| N → *rods* | 0.1 |
| V → *people* | 0.1 |
| V → *fish* | 0.6 |
| V → *tanks* | 0.3 |
| P → *with* | 1.0 |



[With empty NP removed so less ambiguous]

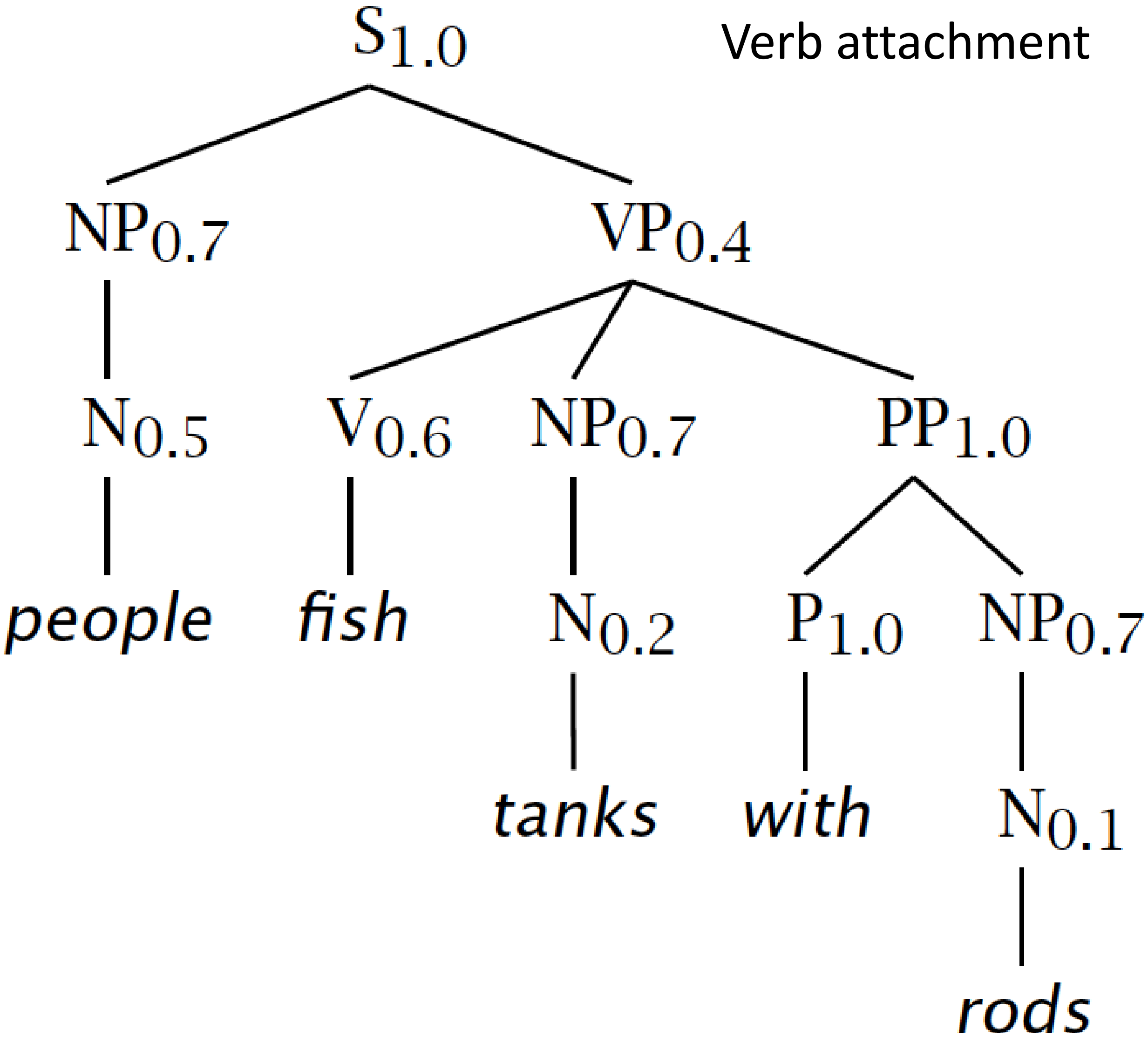
# The probability of trees and strings

- P(*t*) – The probability of a tree *t* is the product of the probabilities of the rules used to generate it.
- P(*s*) – The probability of the string *s* is the sum of the probabilities of the trees which have that string as their yield

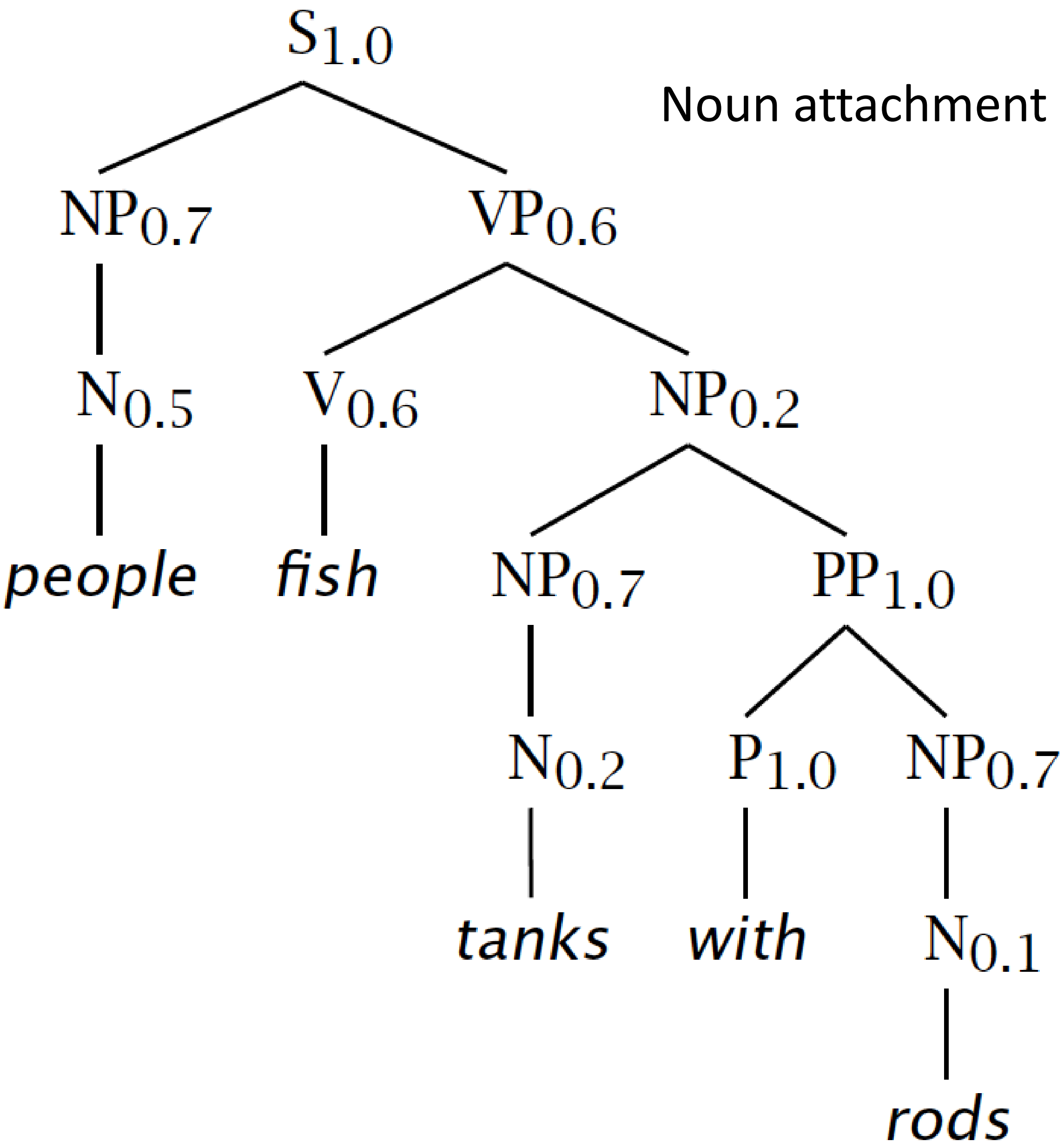
$$P(s) = \Sigma_j P(t) \text{ where } t \text{ is a parse of } s$$

$t_1$:

Verb attachment



$t_2$:



Noun attachment

# Tree and String Probabilities

- *s* = *people fish tanks with rods*
- P($t_1$) = 1.0 × 0.7 × 0.4 × 0.5 × 0.6 × 0.7 × 1.0 × 0.2 × 1.0 × 0.7 × 0.1
  = 0.0008232

Verb attach

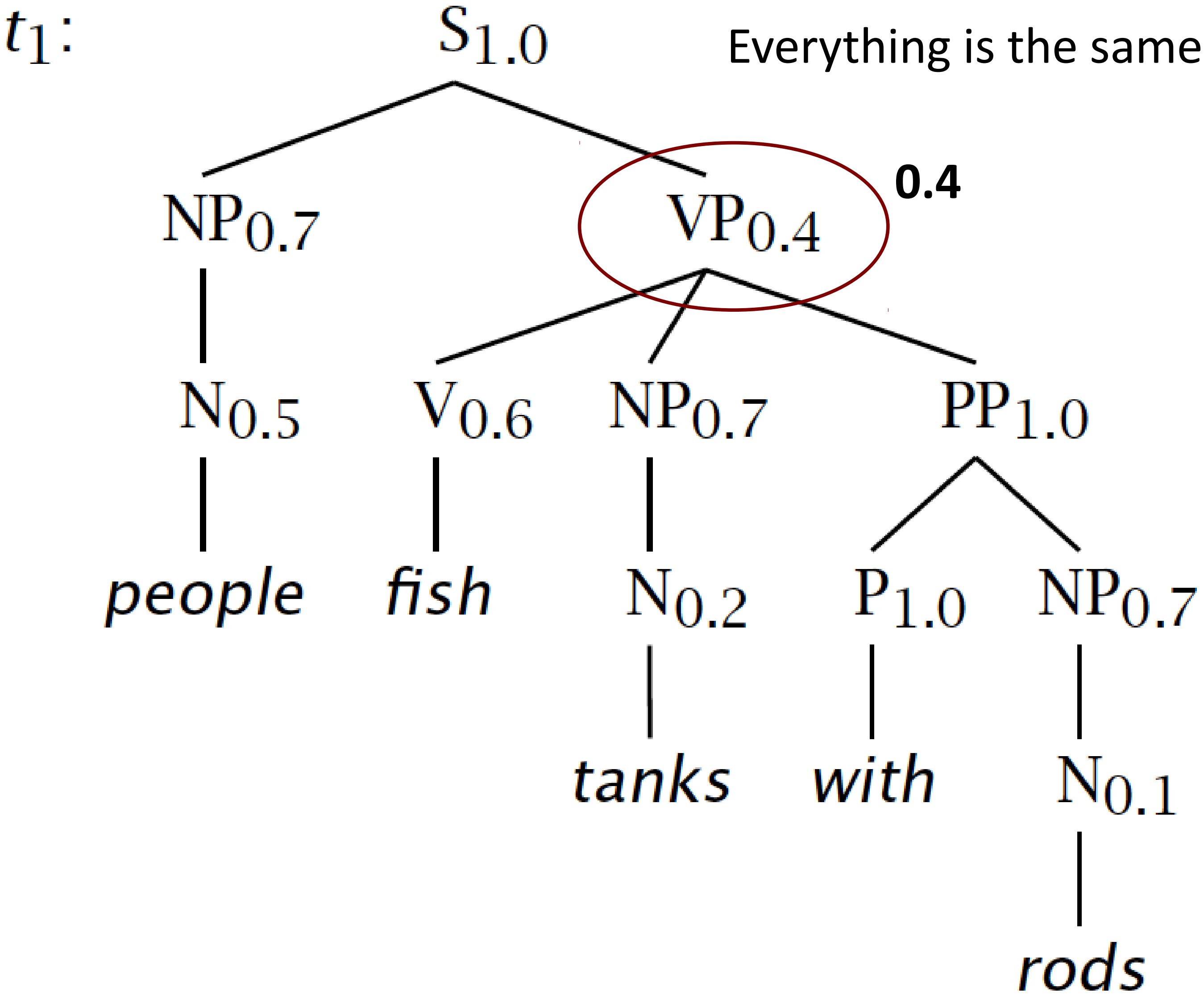
- P($t_2$) = 1.0 × 0.7 × 0.6 × 0.5 × 0.6 × 0.2 × 0.7 × 1.0 × 0.2 × 1.0 × 0.7 × 0.1
  = 0.00024696

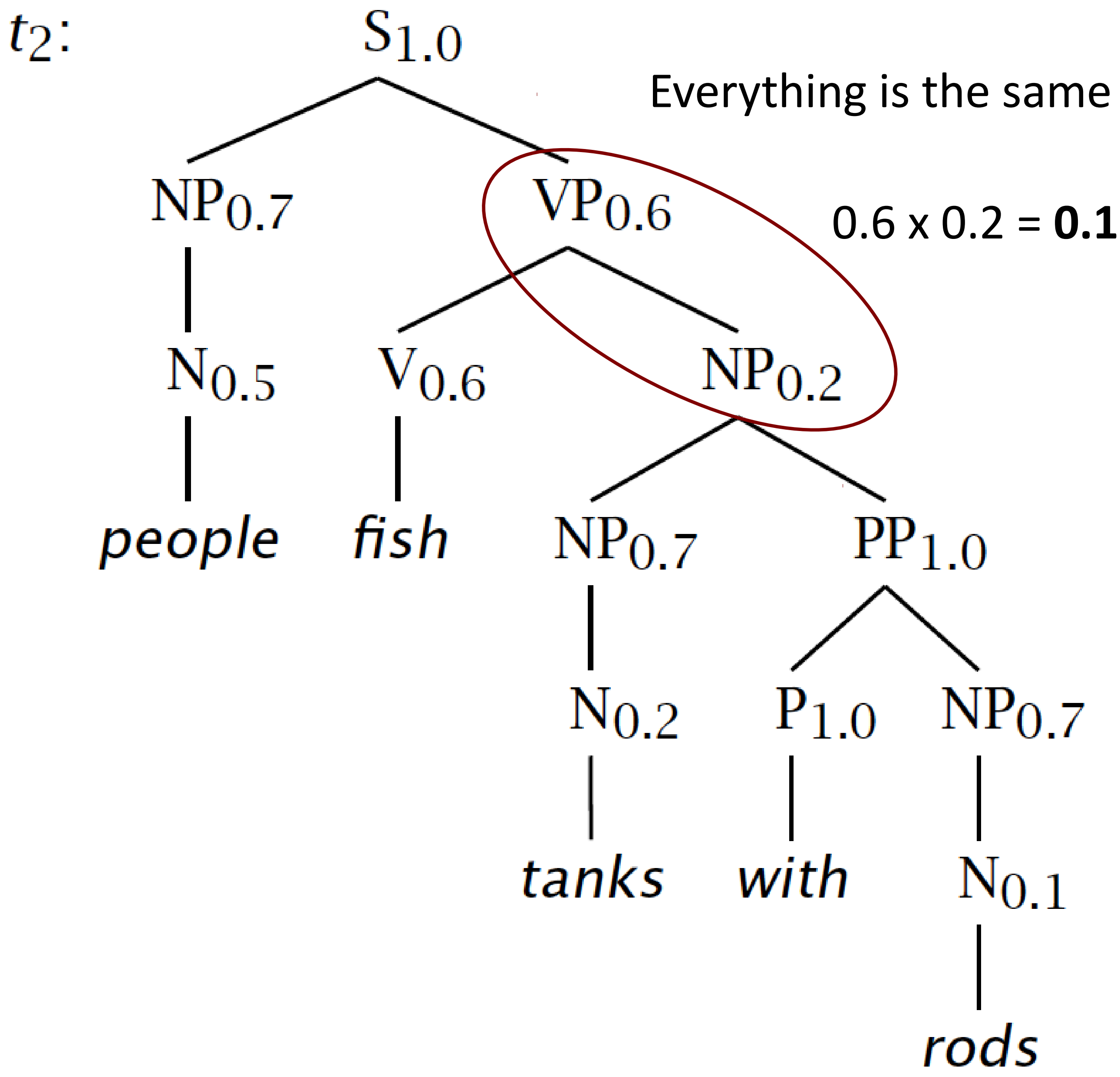
Noun attach

- P(*s*) = P(*t1*) + P(*t2*)
  = 0.0008232 + 0.00024696
  = 0.00107016

Which parse is more likely?
The first one

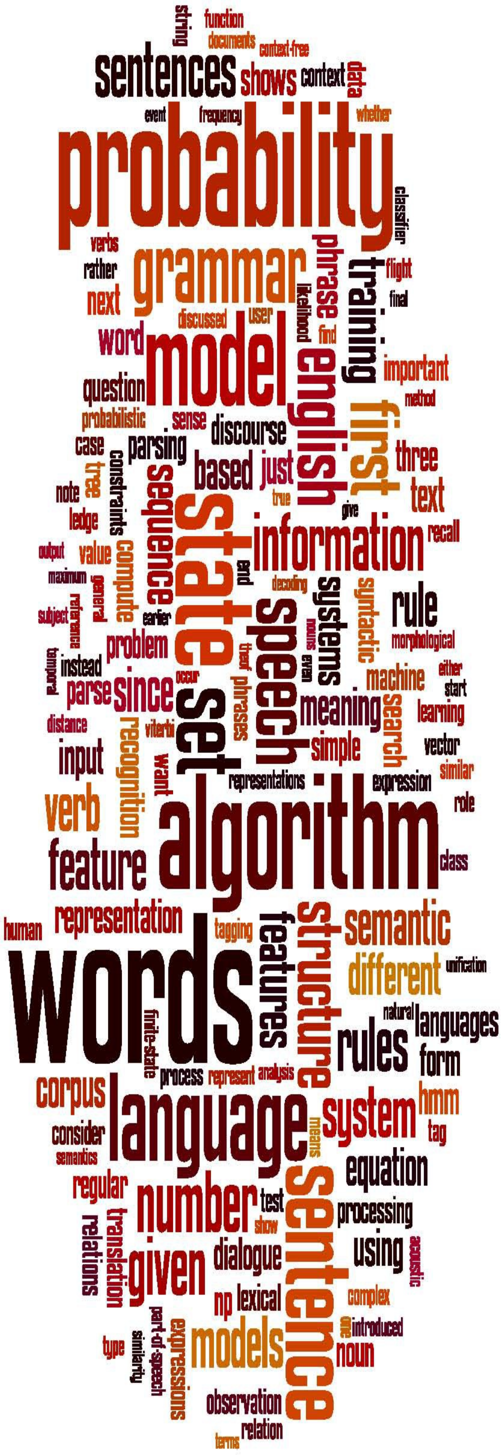
Everything is the same except:





Everything is the same except:

0.6 x 0.2 = **0.12**



# Grammar Transforms

Restricting the grammar form for efficient parsing

# Chomsky Normal Form

- All rules are of the form X → Y Z or X → w
  - X, Y, Z ∈ N and w ∈ T
- A transformation to this form doesn't change the weak generative capacity of a CFG
  - That is, it recognizes the same language
    - But maybe with different trees
- Transformations:
  - Empties and unaries are removed recursively
  - n-ary rules are divided by introducing new nonterminals (n > 2)

# A phrase structure grammar

S → NP VP

VP → V NP

VP → V NP PP

NP → NP NP

NP → NP PP

NP → N

NP → *e*

PP → P NP

N → people

N → fish

N → tanks

N → rods

V → people

V → fish

V → tanks

P → with

*people fish tanks*

*people fish with rods*

# Chomsky Normal Form steps (1)

S → NP VP

VP → V NP

VP → V NP PP

NP → NP NP

NP → NP PP

NP → N

NP → *e*

PP → P NP

1. Remove empty rules

2. Rewrite as:

S → NP VP

S → VP

3. Repeat

N → people

N → fish

N → tanks

N → rods

V → people

V → fish

V → tanks

P → with

*people fish tanks*

*people fish with rods*

# Chomsky Normal Form steps (2)

S → NP VP

S → VP

VP → V NP

VP → V

VP → V NP PP

VP → V PP

NP → NP NP

NP → NP

NP → NP PP

NP → PP

NP → N

PP → P NP

PP → P

1. Remove unary rule

S → V NP

S → V

S → V NP PP

S → V PP

2. Rewrite for rules where VP appears on the left

N → *people*

N → *fish*

N → *tanks*

N → *rods*

V → *people*

V → *fish*

V → *tanks*

P → *with*

# Chomsky Normal Form steps (3)

S → NP VP

VP → V NP

S → V NP

VP → V

S → V

VP → V NP PP

S → V NP PP

VP → V PP

S → V PP

NP → NP NP

NP → NP

NP → NP PP

NP → PP

NP → N

PP → P NP

PP → P

Keep removing *unaries*

N → *people*

N → *fish*

N → *tanks*

N → *rods*

V → *people*

V → *fish*

V → *tanks*

P → *with*

S → *people*

S → *fish*

S → *tanks*

# Chomsky Normal Form steps (4)

S → NP VP

VP → V NP

S → V NP

VP → V

VP → V NP PP

S → V NP PP

VP → V PP

S → V PP

NP → NP NP

NP → NP

NP → NP PP

NP → PP

NP → N

PP → P NP

PP → P

Keep removing *unaries*

N → *people*

N → *fish*

N → *tanks*

N → *rods*

V → *people*

S → *people*

V → *fish*

S → *fish*

V → *tanks*

S → *tanks*

P → *with*

⇒

VP → *people*

VP → *fish*

VP → *tanks*

# Chomsky Normal Form steps (5)

S → NP VP
VP → V NP
S → V NP
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

Keep removing *unaries*

N → *people*
N → *fish*
N → *tanks*
N → *rods*
V → *people*
S → *people*
VP → *people*
V → *fish*
S → *fish*
VP → *fish*
V → *tanks*
S → *tanks*
VP → *tanks*
P → *with*

⇒

NP → *people*
NP → *fish*
NP → *tanks*
NP → *rods*

# Chomsky Normal Form steps (6)

S → NP VP

VP → V NP

S → V NP

VP → V NP PP

S → V NP PP

VP → V PP

S → V PP

NP → NP NP

NP → NP PP

NP → P NP

PP → P NP

Done with unary rules

VP → V @VP_P

@VP_P → NP PP

Replace ternary rule with two binary rules by adding a new non-terminal symbol

NP → *people*

NP → *fish*

NP → *tanks*

NP → *rods*

V → *people*

S → *people*

VP → *people*

V → *fish*

S → *fish*

VP → *fish*

V → *tanks*

S → *tanks*

VP → *tanks*

P → *with*

PP → *with*

# Final Chomsky Normal Form

S → NP VP

VP → V NP

S → V NP

VP → V @VP_V

@VP_V → NP PP

S → V @S_V

@S_V → NP PP

VP → V PP

S → V PP

NP → NP NP

NP → NP PP

NP → P NP

PP → P NP

NP → *people*

NP → *fish*

NP → *tanks*

NP → *rods*

V → *people*

S → *people*

VP → *people*

V → *fish*

S → *fish*

VP → *fish*

V → *tanks*

S → *tanks*

VP → *tanks*

P → *with*

PP → *with*

# A phrase structure grammar

S → NP VP

VP → V NP

VP → V NP PP

NP → NP NP

NP → NP PP

NP → N

NP → *e*

PP → P NP

N → people

N → fish

N → tanks

N → rods

V → people

V → fish

V → tanks

P → with

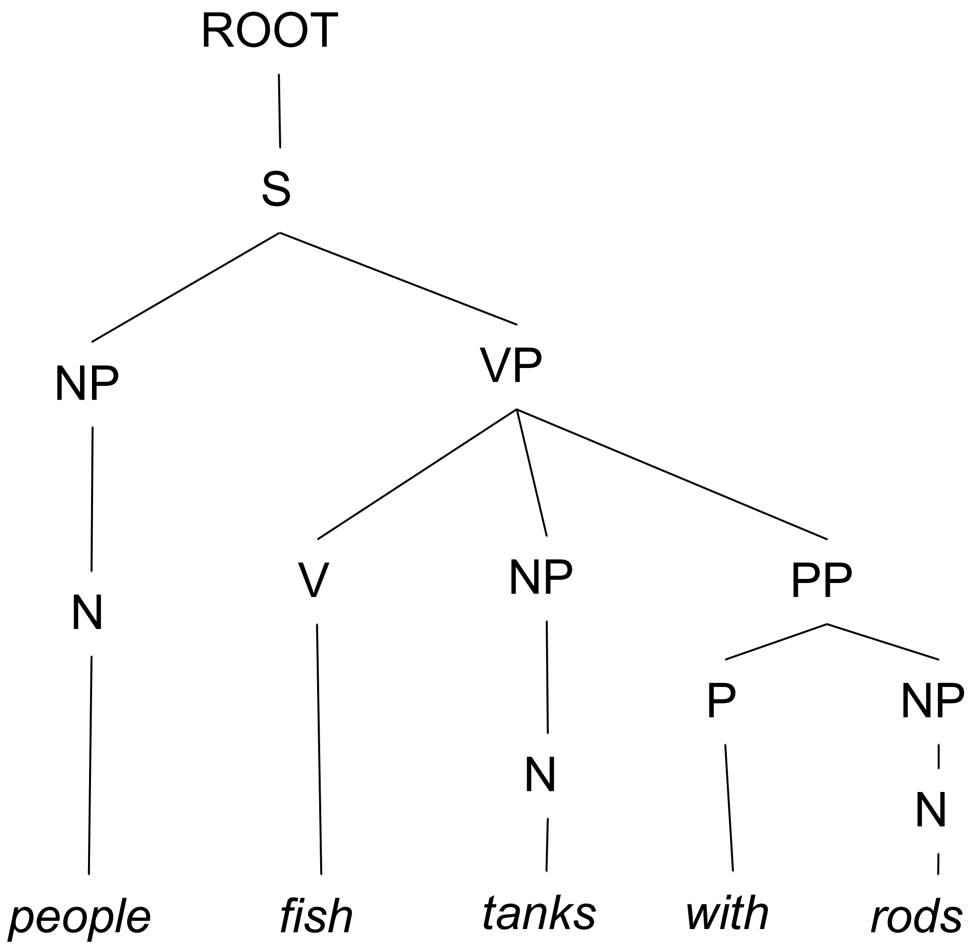
# Chomsky Normal Form

- You should think of this as a transformation for efficient parsing
- With some extra book-keeping in symbol names, you can even reconstruct the same trees with a detransform
- In practice full Chomsky Normal Form is a pain
  - Reconstructing n-aries is easy
  - Reconstructing unaries/empties is trickier

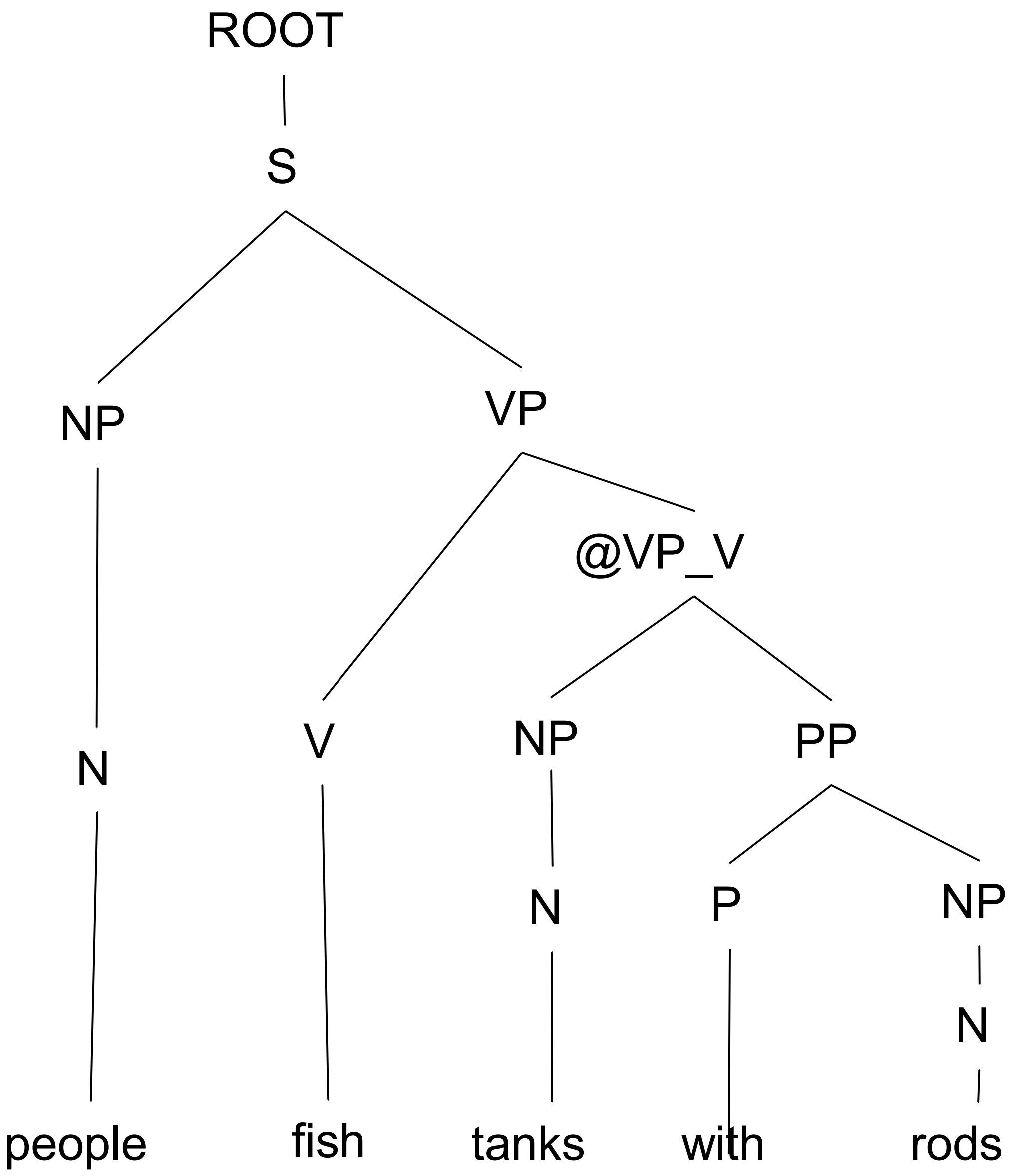
- *Binarization* is crucial for cubic time CFG parsing

- The rest isn’t necessary; it just makes the algorithms cleaner and a bit quicker

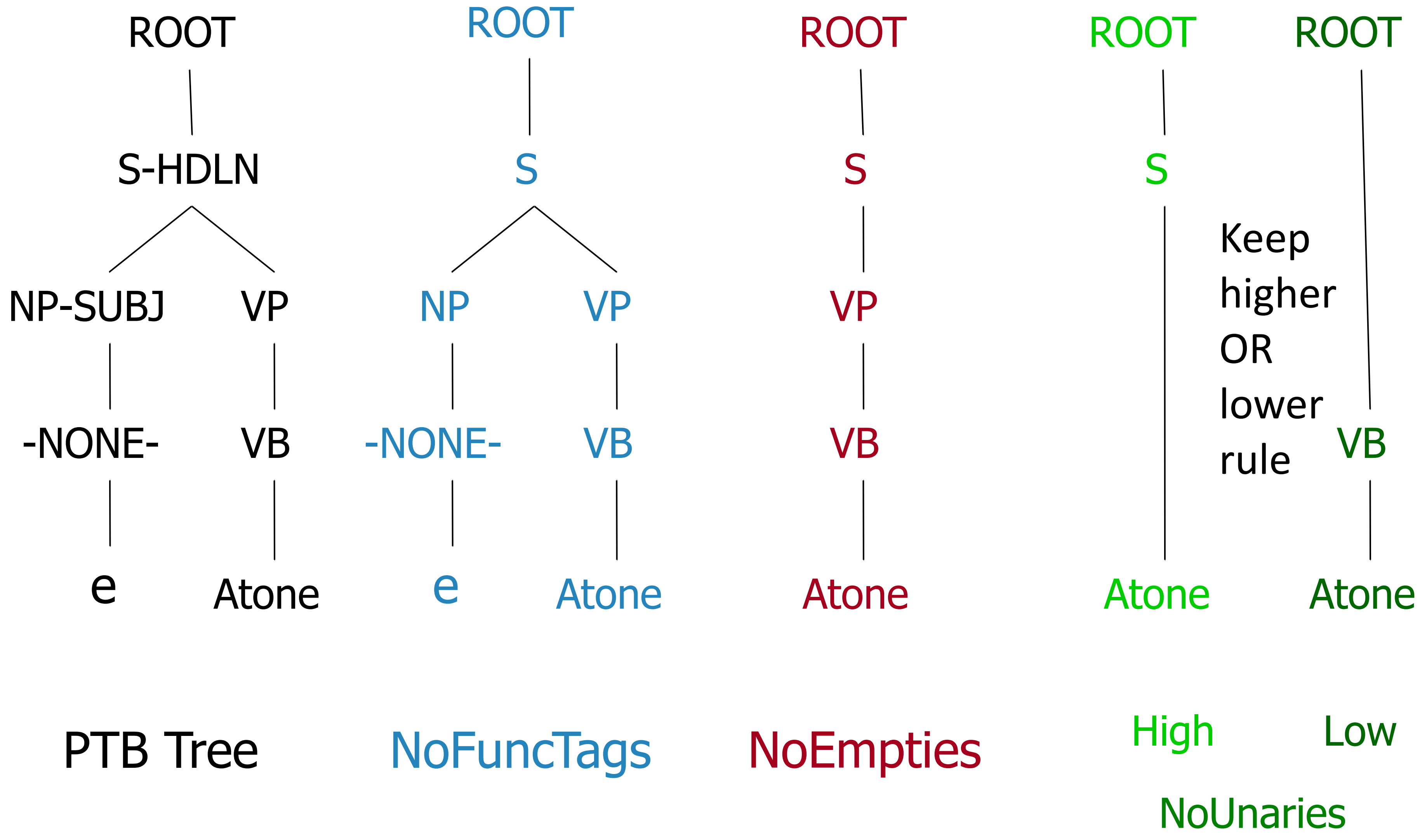
# An example: before binarization…



# After binarization…



# Treebank: empties and unaries



ROOT
S-HDLN
NP-SUBJ VP
-NONE- VB
e Atone

PTB Tree

ROOT
S
NP VP
-NONE- VB
e Atone

NoFuncTags

ROOT
S
VP
VB
Atone

NoEmpties

ROOT
S
Atone

ROOT
VB
Atone

Keep higher OR lower rule

High Low

NoUnaries

Usually just keep unary rules and work with NoEmpties version

# Recap

- Two views of syntactic structures
  - Constituency Parsing
  - Dependency Parsing
- Exponential number of trees
- Context Free Grammars (CFGs)
- Probabilistic Context Free Grammars (PCFGs)
- Chomsky Normal Form
- Next:
  - CKY Parsing Algorithm