Introduction to Syntactic Parsing

Syntax

Setting out together or arrangement

- Refers to the way words are arranged together

Syntax

- Why should you care?
- Grammars (and parsing) are key components in many applications
  - Grammar checkers
  - Dialogue management
  - Question answering
  - Information extraction
  - Machine translation

Sentence Types

- Declaratives: *A plane left.*

  S $\rightarrow$ NP VP

- Imperatives: *Leave!*

  S $\rightarrow$ VP

- Yes-No Questions: *Did the plane leave?*

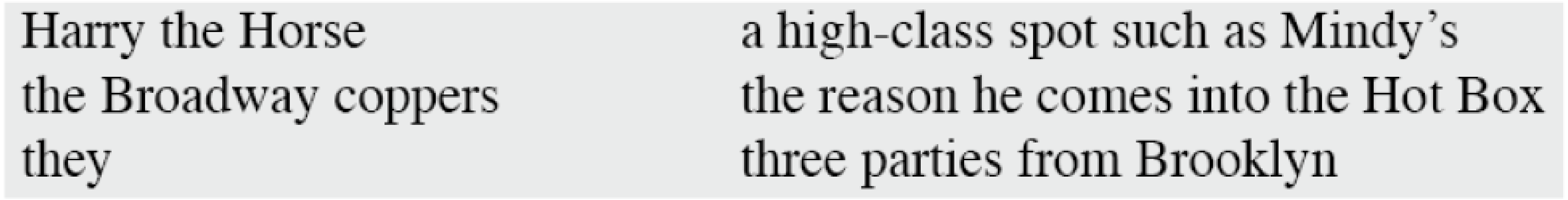
  S $\rightarrow$ Aux NP VP

- WH Questions: *When did the plane leave?*

  S $\rightarrow$ WH-NP Aux NP VP

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

- The basic idea here is that groups of words within utterances can be shown to act as single units
- For example, it makes sense to the say that the following are all ***noun phrases*** in English...

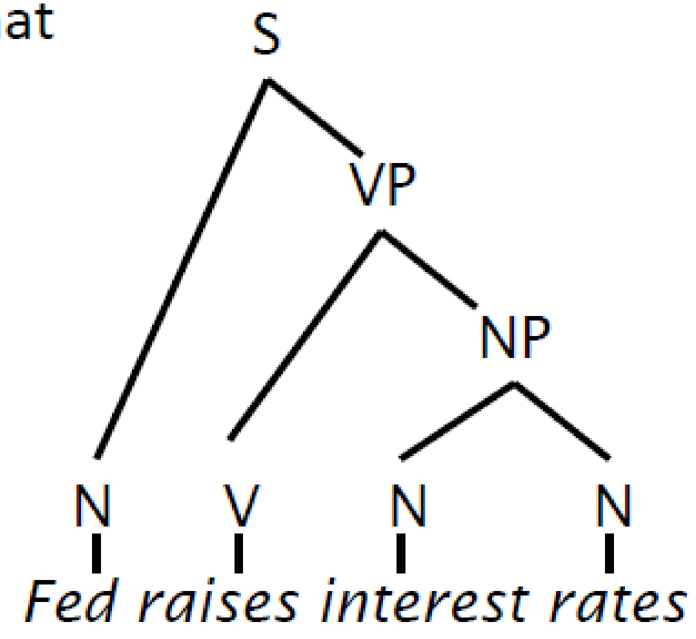


Harry the Horse	a high-class spot such as Mindy’s
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- Why? One piece of evidence is that they can all precede verbs.

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

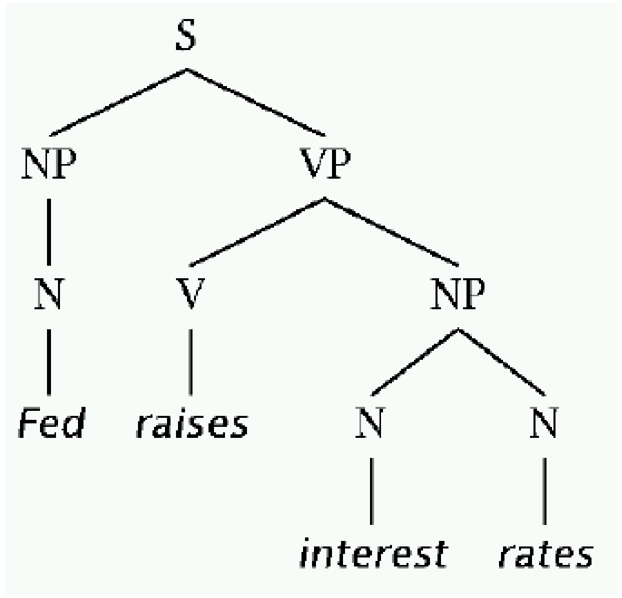
- Phrase structure organizes words into nested constituents.
- How do we know what is a constituent? (Not that linguists don’t argue about some cases.)
  - Distribution: a constituent behaves as a unit that can appear in different places:
    - John talked [to the children] [about drugs].
    - John talked [about drugs] [to the children].
    - *John talked drugs to the children about
  - Substitution/expansion/pro-forms:
    - I sat [on the box/right of the box/there].



Headed phrase structure

To model constituency structure:

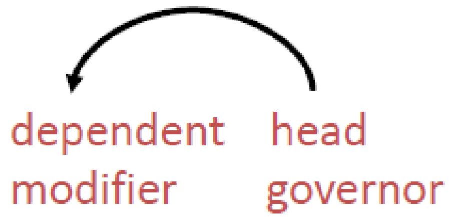
- VP → … VB* …
- NP → … NN* …
- ADJP → … JJ* …
- ADVP → … RB* …
- PP → … IN* …



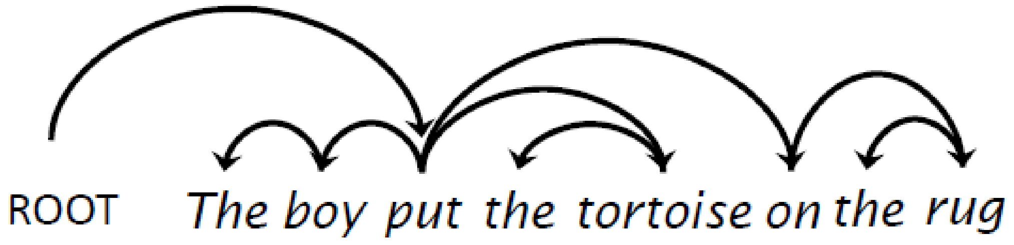
- Bracket notation of a tree (Lisp S-structure):
  (S (NP (N Fed)) (VP (V raises) (NP (N interest) (N rates)))

Two views of linguistic structure: 2. Dependency structure

- In CFG-style phrase-structure grammars the main focus is on *constituents.*
- But it turns out you can get a lot done with binary relations among the lexical items (words) in an utterance.
- In a dependency grammar framework, a parse is a tree where
  - the nodes stand for the words in an utterance
  - The links between the words represent dependency relations between pairs of words.
    - Relations may be typed (labeled), or not.

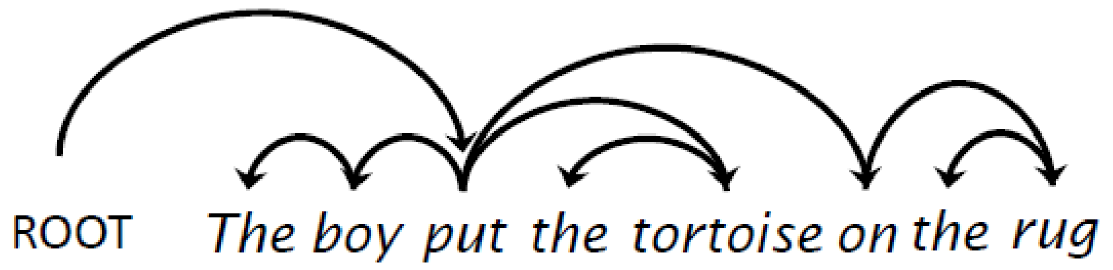


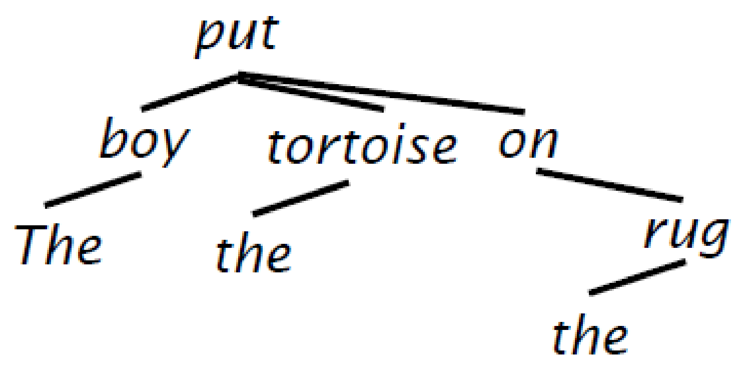
Sometimes arcs drawn in opposite direction



Two views of linguistic structure: 2. Dependency structure

Alternative notations (e.g. rooted tree):





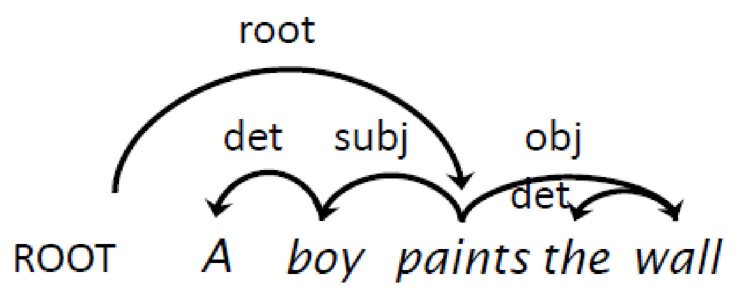
Dependency Labels

Argument dependencies:

- Subject (subj), object (obj), indirect object (iobj)…

Modifier dependencies:

- Determiner (det), noun modifier (nmod), verbal modifier (vmod), etc.



The rise of data and statistics: Pre 1990 (“Classical”) NLP Parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon

S $\rightarrow$ NP VP	NN $\rightarrow$ *interest*
NP $\rightarrow$ (DT) NN	NNS $\rightarrow$ *rates*
NP $\rightarrow$ NN NNS	NNS $\rightarrow$ *raises*
NP $\rightarrow$ NNP	VBP $\rightarrow$ *interest*
VP $\rightarrow$ V NP	VBZ $\rightarrow$ *rates*

- Used grammar/proof systems to prove parses from words
- This scaled very badly and didn’t give coverage.

Fed raises interest rate 0.5% in effort to control inflation.

Minimum grammar: 36 parses
Simple 10 rule grammar: 592 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)

Treebanks [Marcus et al, 1993]

- Treebanks are corpora in which each sentence has been paired with a parse tree (presumably the right one).
- These are generally created
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.

The rise of annotated data: The Penn Treebank

[Marcus et al. 1993, *Computational Linguistics*]

```
( (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)))))))
   (. .)))
```

Most well known part is the Wall Street Journal section of the Penn TreeBank.

1 M words from the 1987-1989 Wall Street Journal newspaper.

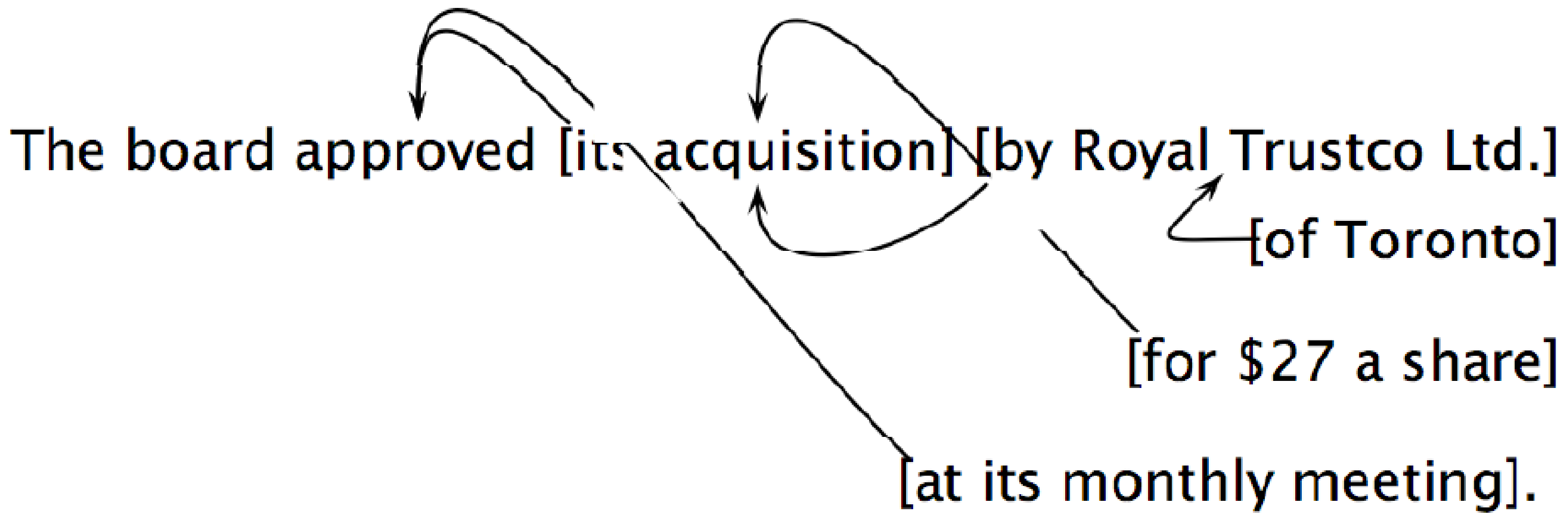
The rise of annotated data

- 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
  - Statistics to build parsers
  - A way to evaluate systems

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

- How many distinct parses does the following sentence have due to PP attachment ambiguities?

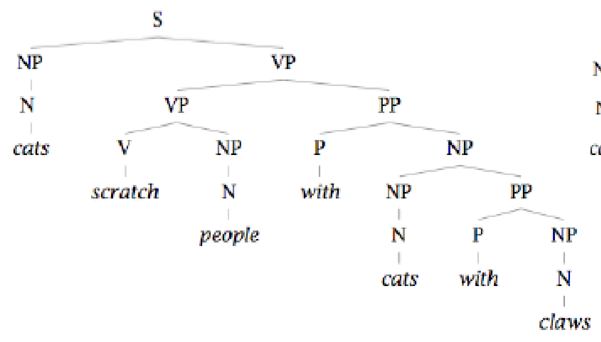
John wrote the book with a pen in the room.

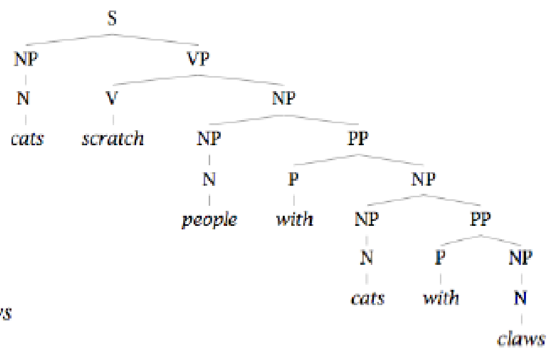
John wrote [the book] [with a pen] [in the room].
John wrote [[the book] [with a pen]] [in the room].
John wrote [the book] [[with a pen] [in the room]].
John wrote [[the book] [[with a pen] [in the room]]].
John wrote [[[the book] [with a pen]] [in the room]].

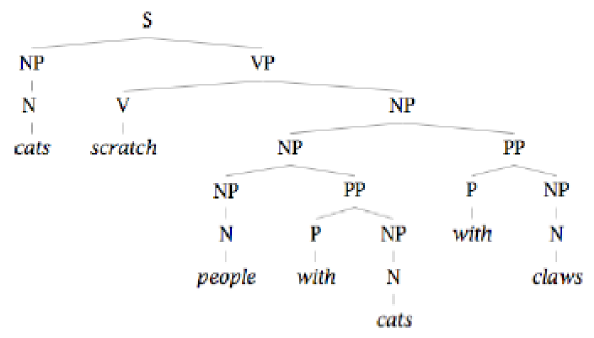
Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$ - an exponentially growing series

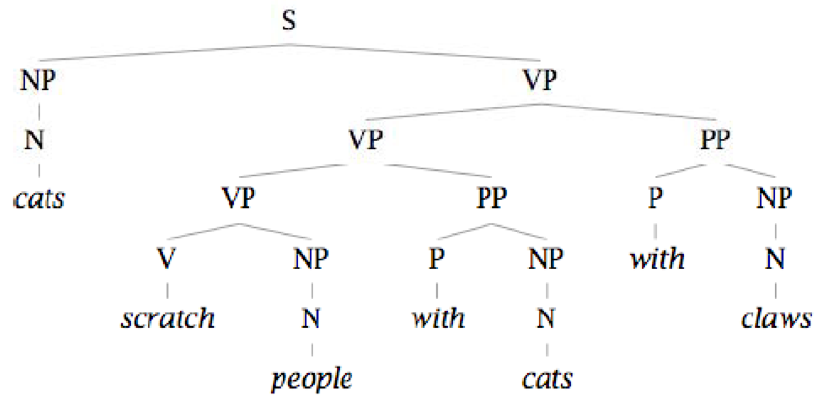
1	1
2	2
3	5
4	14
5	42
6	132
7	429
8	1430

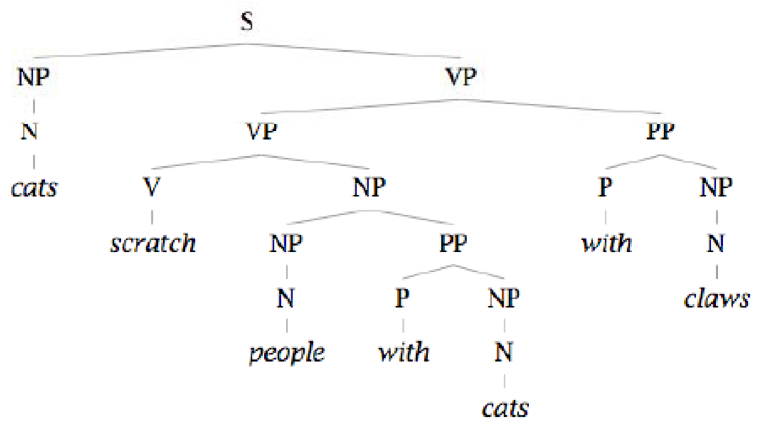
Two problems to solve:
1. Avoid repeated work…



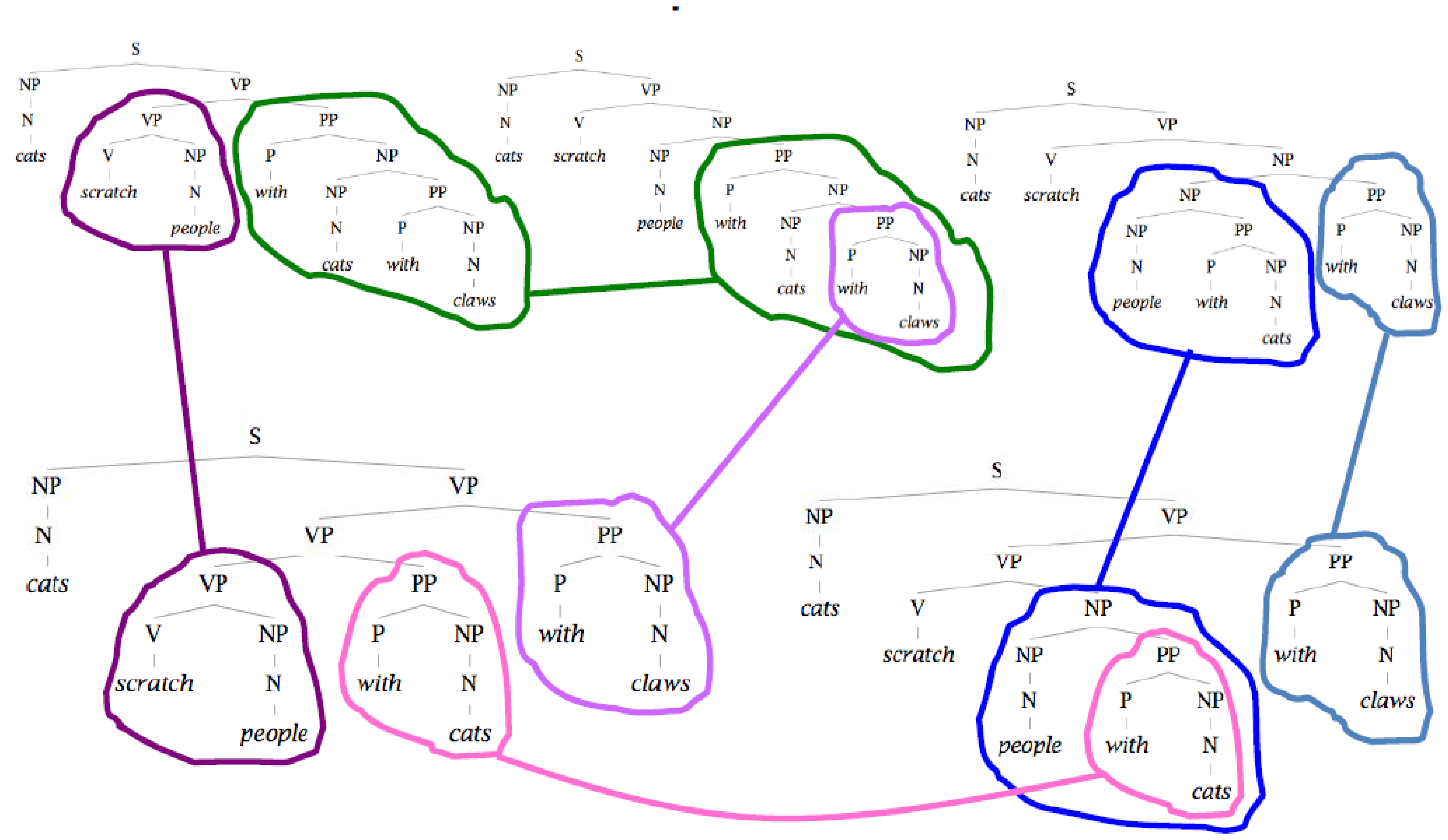






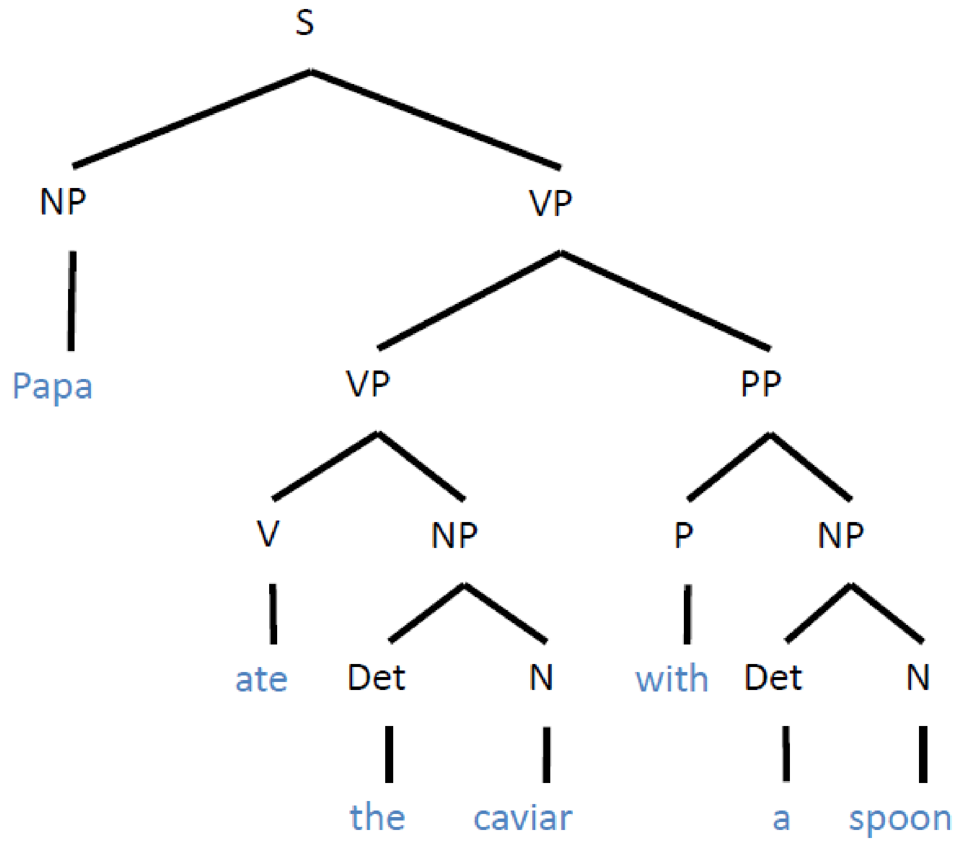


Two problems to solve:
1. Avoid repeated work…



Two problems to solve:
2. Ambiguity - Choosing the correct parse

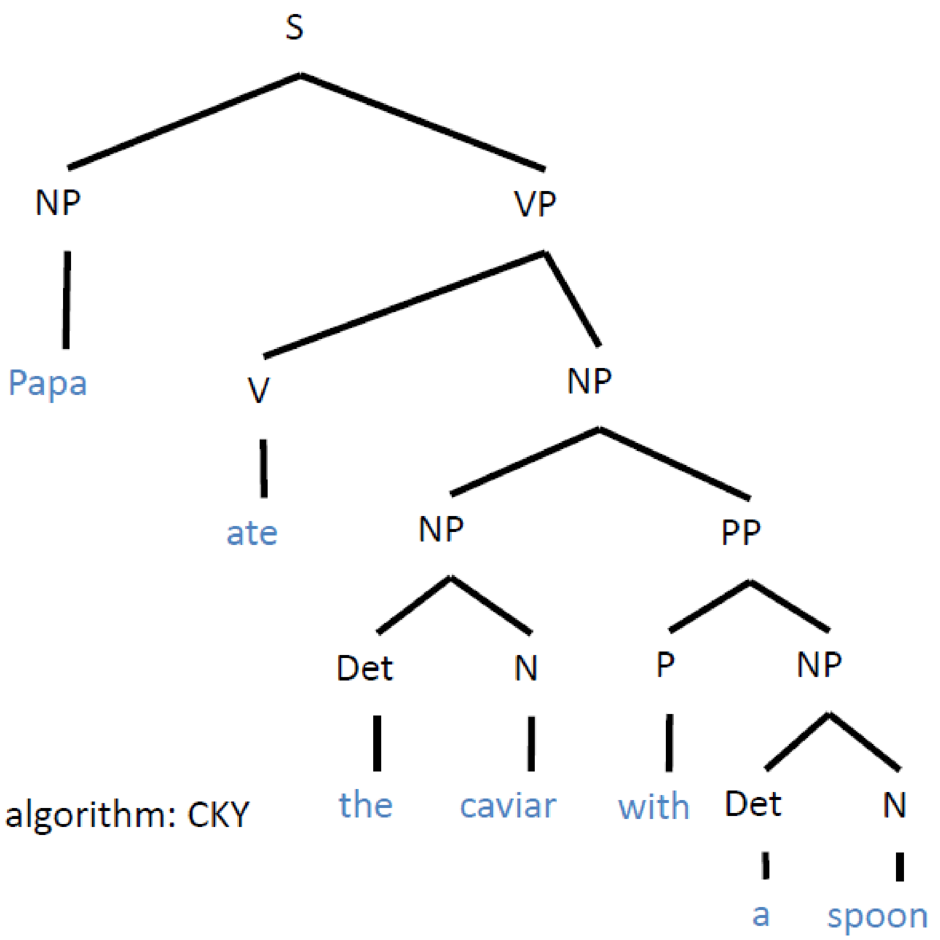
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

Two problems to solve:
2. Ambiguity - Choosing the correct parse

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



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

→ need an efficient algorithm: CKY