DAY 4: SYNTAX AND PARSING

Mark Granroth-Wilding

COURSE OUTLINE

	Day	Topic
Week 1	1	Introduction to NLP
	2	NLU pipeline and toolkits
	3	Finite state methods; statistical NLP
	4	Syntax and parsing
	5	Evaluation
Week 2	6	NLG and dialogue
	7	Vector space models and lexical semantics
	8	Information extraction; advanced stat. NLP
	9	*Ascension: no lectures*
	10	Semantics and pragmatics; the future

STRUCTURE IN LANGUAGE

- NL sentences have structure
- Hidden: expressed as a sequence

Agreement in number

Alice walk**s** quickly
Athletes **walk** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

STRUCTURE IN LANGUAGE

- NL sentences have structure
- Hidden: expressed as a sequence

Alice, with her big boots on, walk**s** quickly
Athletes, after much training, **walk** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

STRUCTURE IN LANGUAGE

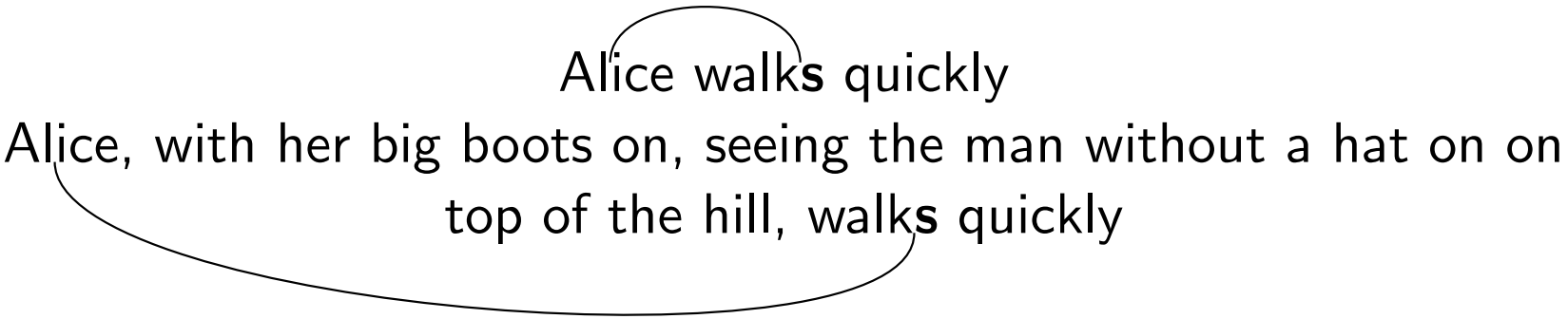
- NL sentences have structure
- Hidden: expressed as a sequence

Alice, with her big boots on, walk**s** quickly

Alice, with her big boots on, seeing the man without a hat on on top of the hill, walk**s** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

LONG-RANGE DEPENDENCIES



- Need models that capture these to understand language
- **Long-range** dependencies (and others)
- Much language processing depends on this
- Linguistic sequence $\Rightarrow$ meaning:
  infer hidden connections & relationships
- **Syntax** models *structures* underlying this *process*

SYNTAX: SUBSTITUTABILTY

- POS tags capture type of **substitutabilty** of words:

Alice walks quickly
John walks quickly
Alice walks occasionally

- **Syntactic categories** capture substitutabilty at **phrasal level**

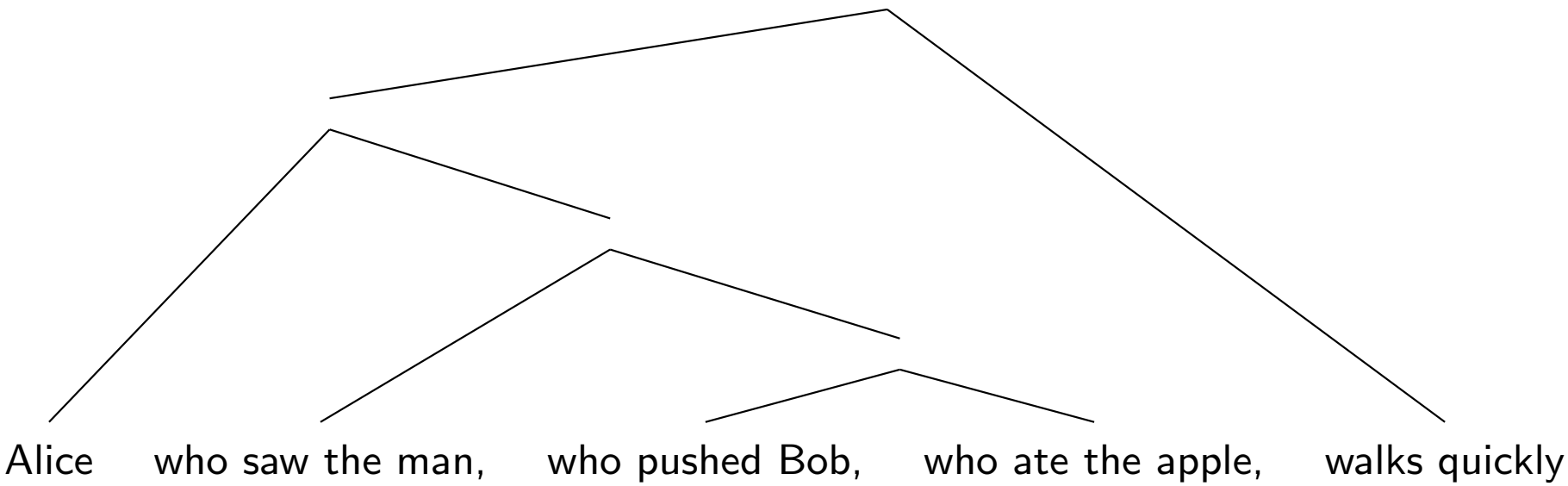
Alice, with her boots on, walks quickly
Alice, without a second thought, walks quickly
Alice, who longs to get home, walks quickly

SYNTAX: RECURSION

- (Probably) all human languages exhibit **recursion**

  Alice, who saw the man, who pushed Bob,
  who ate the apple, walks quickly

- Natural to analyse as a **tree structure**:



FORMAL SYNTAX

- Theory of syntax: characterizes **well-formed** sentences

Alice walks quickly
* Alice walks Alice

- Early formal syntax focussed on this
- **Well-formed**: conforms to rule of language's grammar
- Rules: produce **semantic interpretation**
- But can follow *syntax* rules with being meaningful:

Colorless green ideas sleep furiously

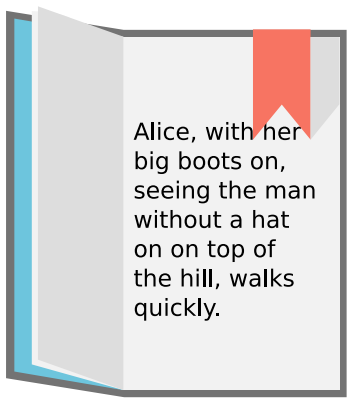
Noam Chomsky, *Syntactic Structures* (1957)

FORMAL SYNTAX

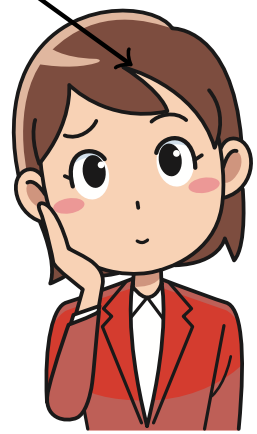
Rules: produce **semantic interpretation** by **composition** of parts

Syntax describes *process* mapping
surface form to *meaning*
via analysis of *hidden structure*

Parsing

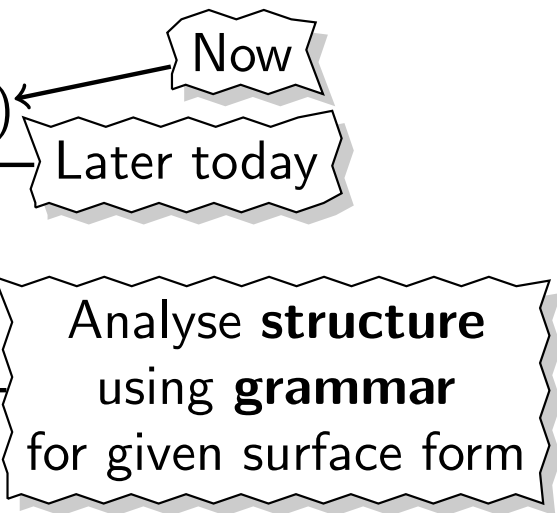


Alice, with her
big boots on,
seeing the man
without a hat
on on top of
the hill, walks
quickly.



GRAMMAR FORMALISMS

- *Formal language* to capture hidden structure of NL
- There are **many** formalisms
- Two widely used ones:
  1. Context-free grammar (CFG)
  2. Dependency grammar
- Need:
  - Grammar for language
  - Algorithm to **parse**
  - Statistical model / data



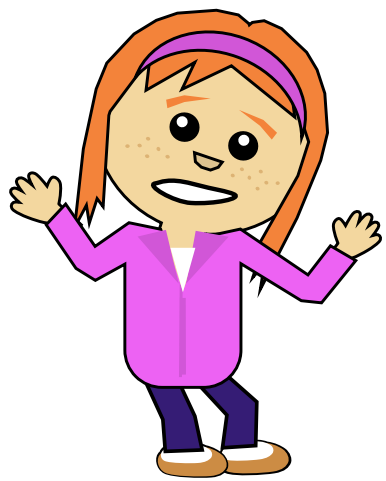
AMBIGUITY

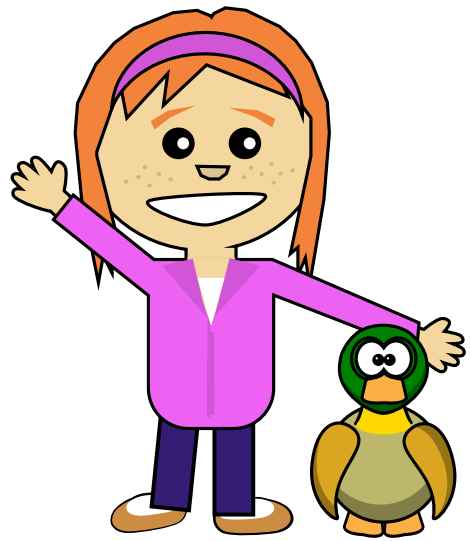
- Syntactic structure is ambiguous
- One sentence can have many structures
- Inferring structure can disambiguate meaning

parsing

POS ambiguity

I saw her duck

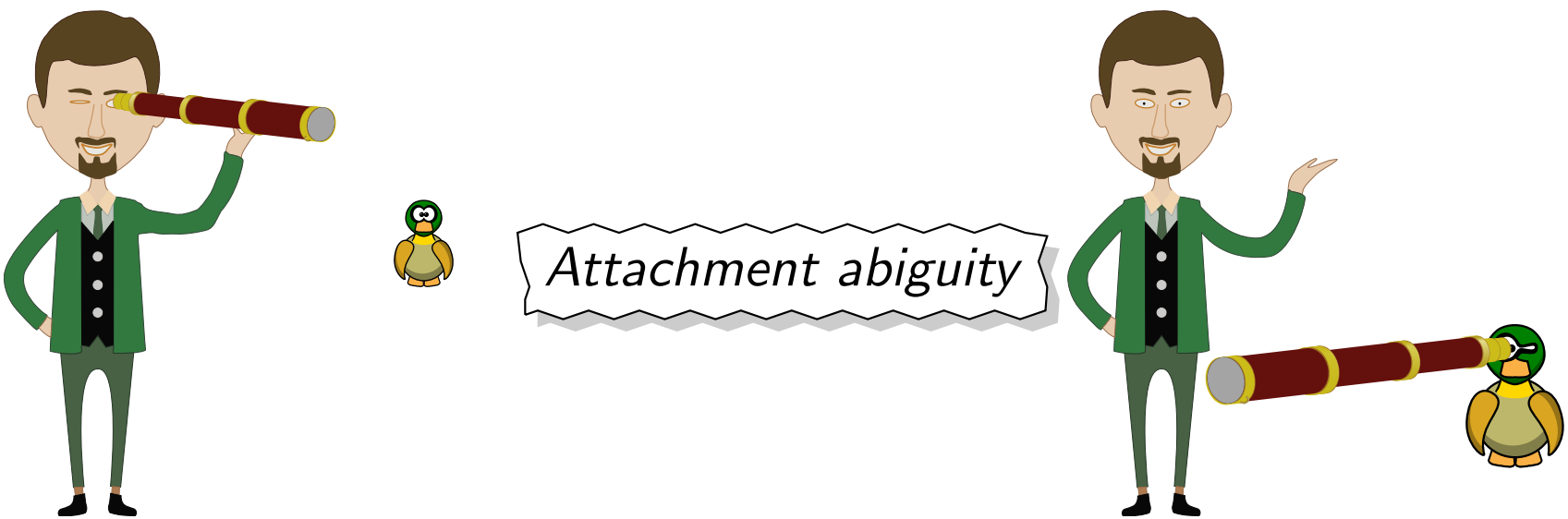




DISAMBIGUATION: ATTACHMENT

- Prepositional phrase: *with the telescope*
- What does it **attach** to?

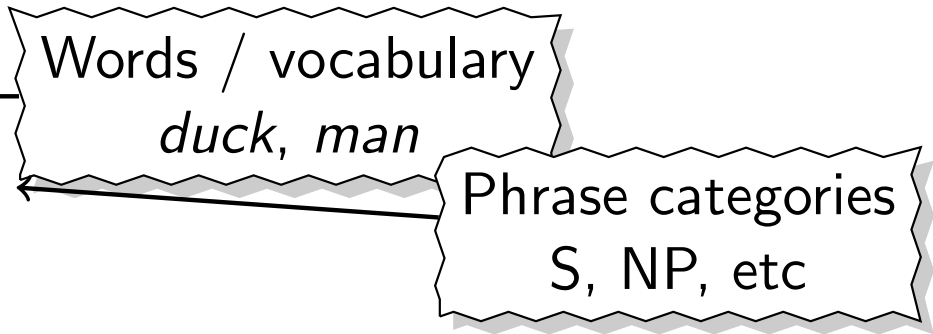
I saw the duck with the telescope

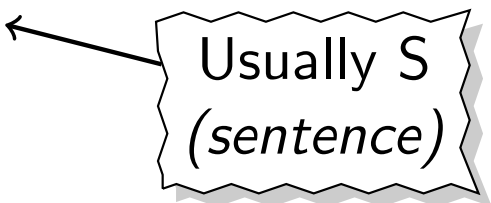


CONTEXT-FREE GRAMMARS: REMINDER

Grammar consists of:

1. Set of **terminal** symbols
2. Set of **non-terminal** symbols
3. Set of **rules** (productions)
   single NT $\rightarrow$ *sequence of NTs / terminals*
4. Start symbol





Generative grammar:
words of sentence *generated* from *start symbol* via *productions*

A SMALL CFG

Rules

S $\rightarrow$ NP VP

NP $\rightarrow$ Det N

NP $\rightarrow$ PropN

Det $\rightarrow$ PosPro

Det $\rightarrow$ Art

VP $\rightarrow$ Vt NP

Lexicon

Art $\rightarrow$ the | a

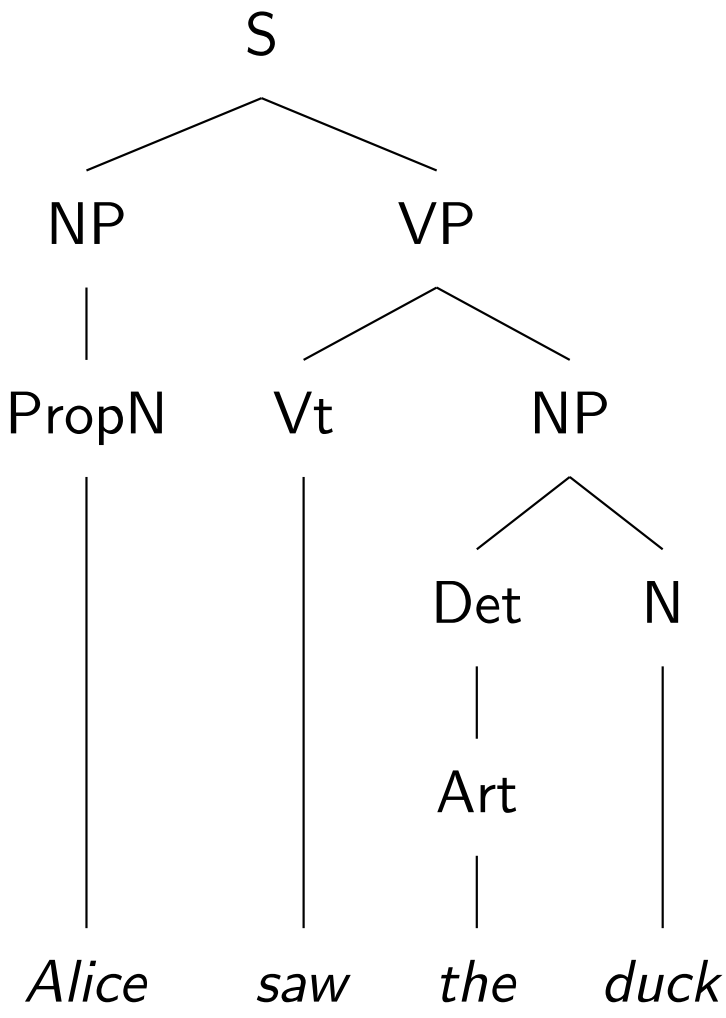
PropN $\rightarrow$ Alice

N $\rightarrow$ duck | telescope | park

Vt $\rightarrow$ saw

PosPro $\rightarrow$ my | her

EXAMPLE SENTENCE DERIVATION



Rules

S → NP VP
NP → Det N
NP → PropN
Det → PosPro
Det → Art
VP → Vt NP

Lexicon

Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her

PARSING

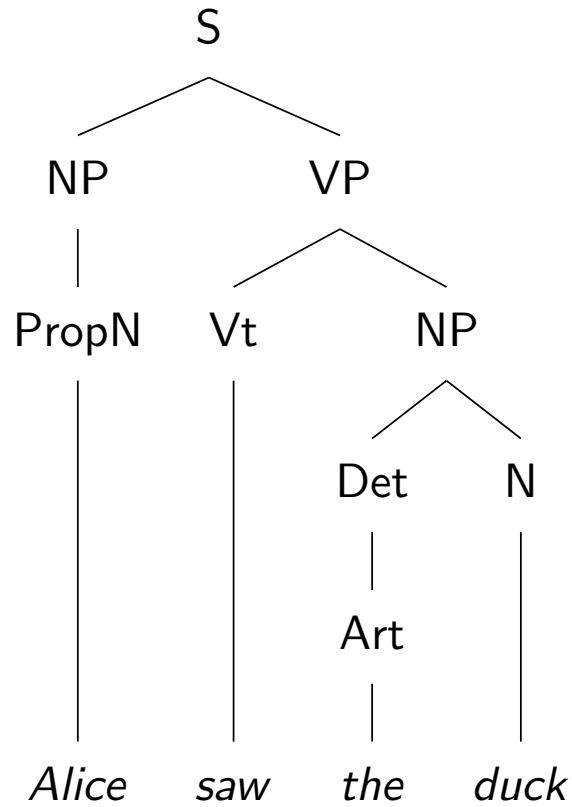
- **Generated** words of sentence from S
- Other way: **analyse** possible underlying structures – **parse**

Alice saw the duck

+

Rules
S → NP VP
NP → Det N
NP → PropN
Det → PosPro
Det → Art
VP → Vt NP
Lexicon
Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her

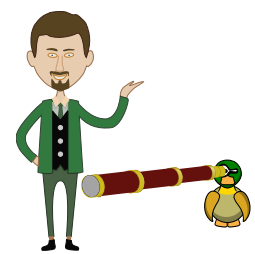
→



SYNTACTIC AMBIGUITY

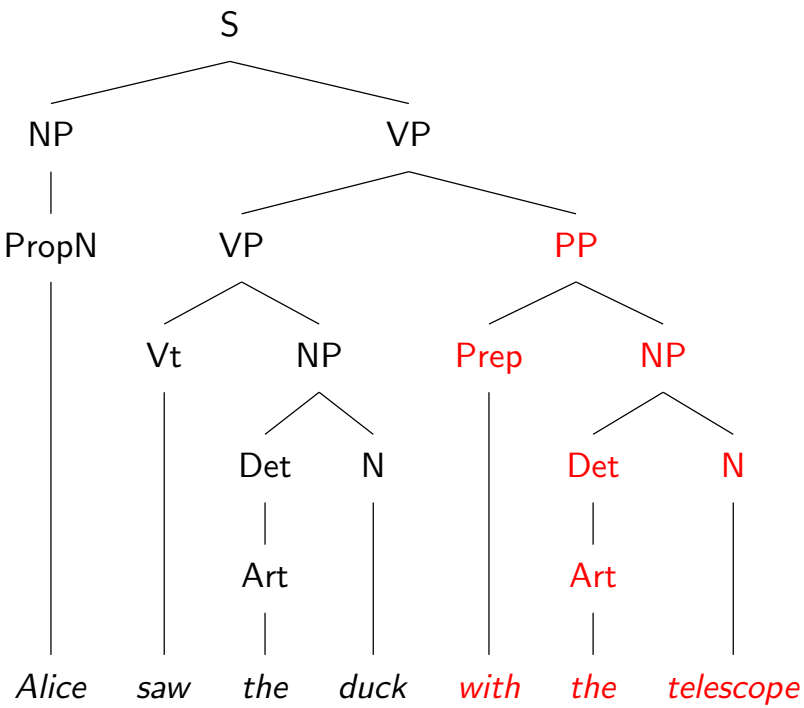
- Can be many underlying structures for one sentence
- NL parsers are **non-deterministic**
- Unlike programming languages:
  - Unambiguous structure
  - Deterministic parser

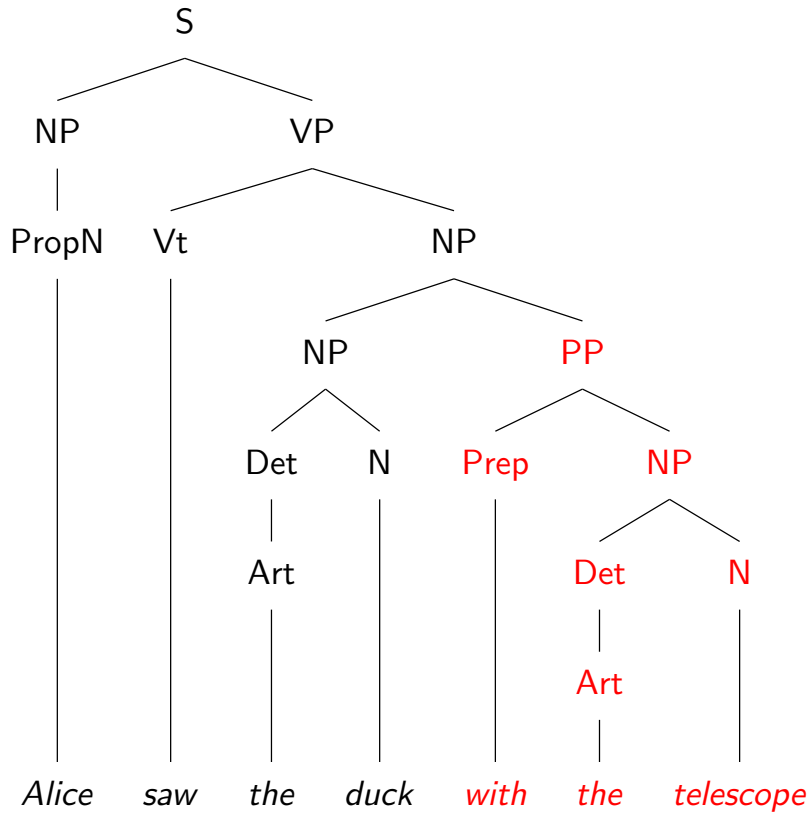
I saw the duck with the telescope



- *Some* 'real' ambiguities – perceived by humans
- Others: *overgeneration* by the grammar

AMBIGUOUS STRUCTURES





PARSING

String + Grammar $\rightarrow$ Syntactic structures (trees)

- Potentially many possible trees
- *All possible derivations* using grammar for given string
- Capture structures important to **semantics**
- Coming up:
  algorithm to find *all* derivations: **exhaustive parsing**

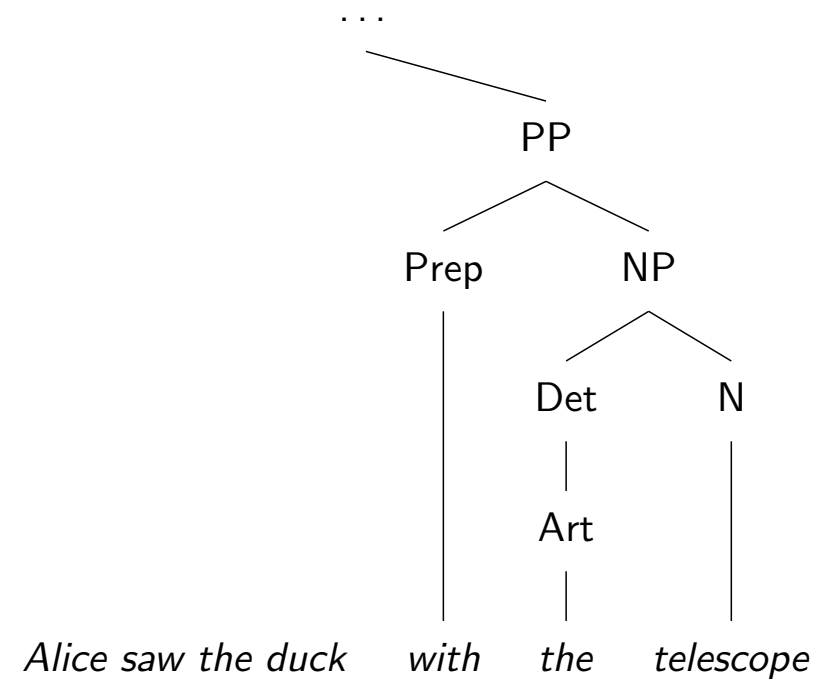
BOTTOM-UP PARSING

- Grammar **top-down**: S $\Rightarrow$ words
- Parsing **bottom-up**: words $\Rightarrow$ tree (S)
- Bottom-up strategy:
  - Consider all *NTs* that could have generated a *word*
  - Consider all NTs that could have generated *that*
  - . . . until S covering whole sentence
- May be multiple full **derivations**

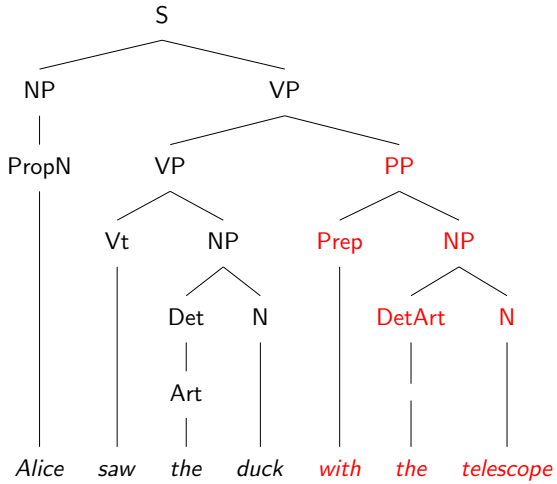
BOTTOM-UP PARSING

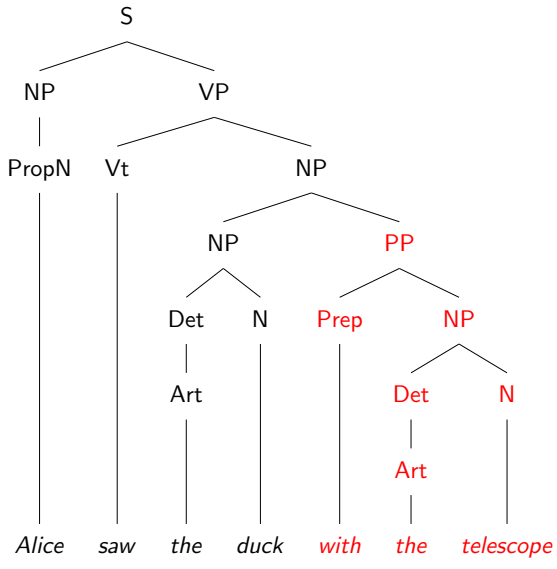
Dynamic programming approach:

- Rule is **context free**
- Doesn't depend on **subtree** or **context outside**
- Same NT for same span: treated same further up
- Remember all derivations to retrieve at end



CFG WITH PREPOSITIONAL PHRASES





Rules

S → NP VP

NP → Det N | PropN

Det → PosPro | Art

VP → Vt NP

VP → VP PP

NP → NP PP

PP → Prep NP

Lexicon

Art → the | a

PropN → Alice

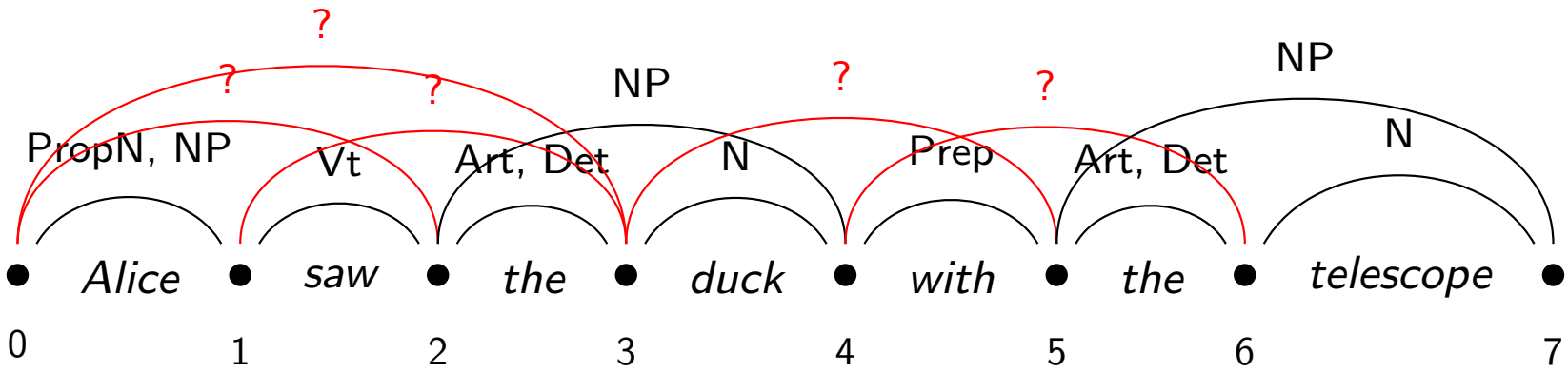
N → duck | telescope | park

Vt → saw

PosPro → my | her

Prep → with

CHART PARSING



THE CKY ALGORITHM

(AKA: CYK algorithm)

```
for len = 1 to n:                    # Num words in span
    for i = 0 to n-len:              # Start of span
        j = i+len                    # End of span
        # Apply any unary rules
        foreach A->B where B in c[i,j]:
            Add A to c[i,j]
            for k = i+1 to j-1:  # Middle position
            # Process binary rules covering sub-spans
            foreach A->B C where B in c[i,k] and C in c[k,j]:
                Add A to c[i,j]
```

- Goal: S covering the whole sentence (`c[0,n+1]`)
- Store traces to retrieve all trees

CKY

- This version strictly bottom-up (depth-first)
- Can change ordering to make more **incremental** ⟵ Left-to-right
  See version in *J&M*
- Grammar must be in **Chomsky normal form**
  - Can convert any CFG to CNF

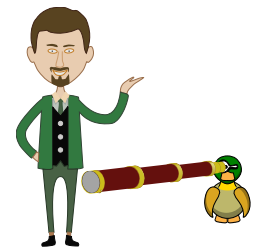
EFFICIENCY OF CKY

- Dynamic programming:
  avoid *recomputing* unnecessary structures
- Still computes many unused structures
- Other algorithms exist, motivated by
  - Cognitive modelling
  - Average time efficiency
  - Space efficiency, ...
- Another approach: **statistical parsing**
- Later today...

AMBIGUITY

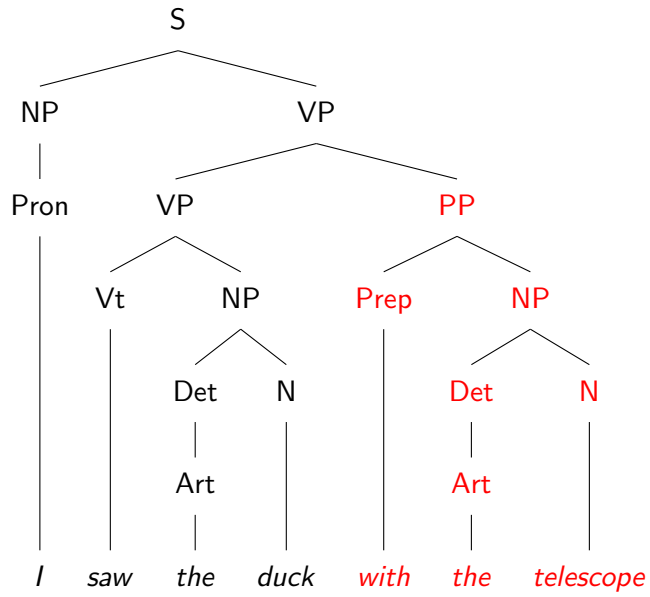
- Parsing ambiguity a big problem
- **Correctness**: finding correct structure from *many* possible
- **Efficiency**: many structures to consider $\rightarrow$ parsing is slow

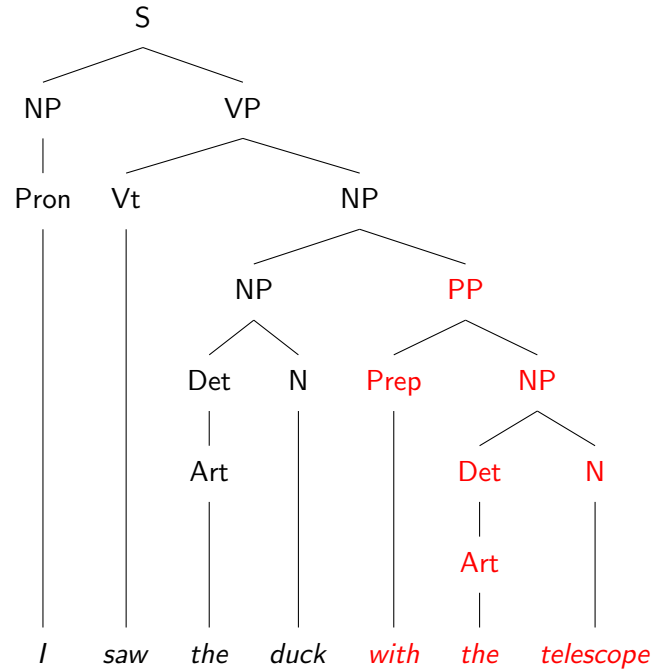
I saw the duck with the telescope



- Real ambiguities (perceived by humans) – want these
- *Overgeneration* by grammar – as few as possible

PARSING FOR SEMANTICS





- *Some* **parsing ambiguities** $\Rightarrow$ ambiguities of **meaning**
- Parsing helps resolve ambiguity
- Choosing single parse tree, some may remain

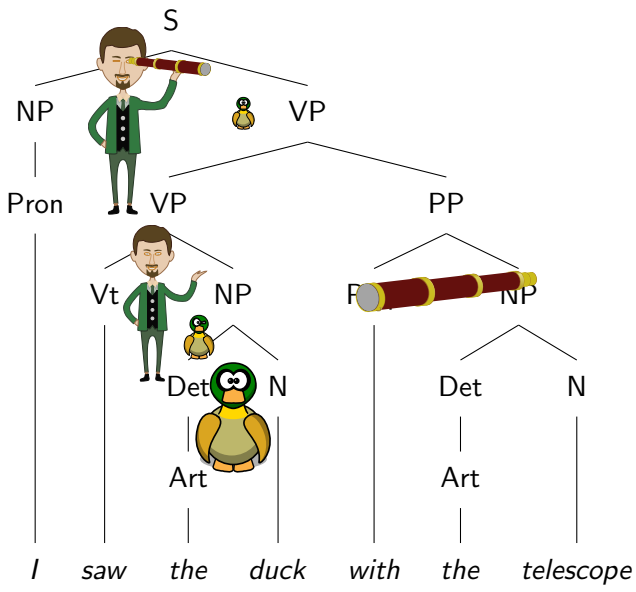
PARSING FOR SEMANTICS

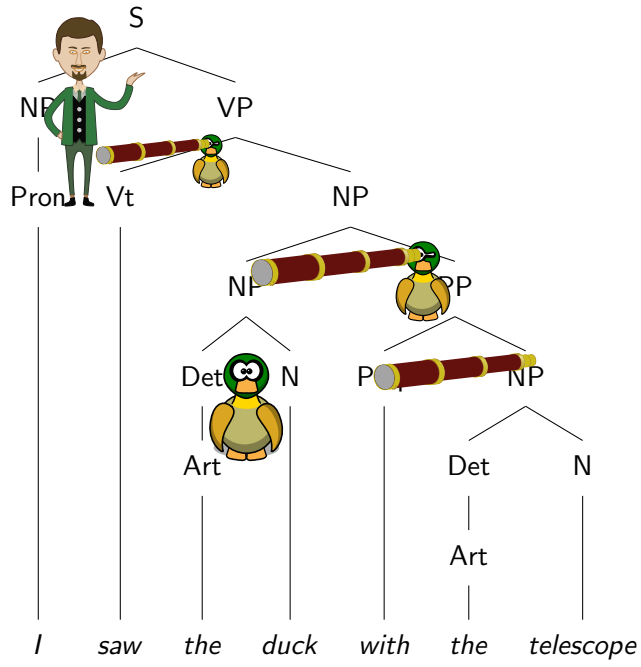
- Representing **meaning** of sentences: **semantics**
- Putting together meanings of *parts of sentence* to produce whole meaning:

  compositional semantics

  - Sometimes meaning is **non-compositional**
  - *Kick the bucket* $\neq$ *kick* + *bucket*
- Syntax: how to combine parts
- Ambiguities lead to different combinations

PARSING FOR SEMANTICS





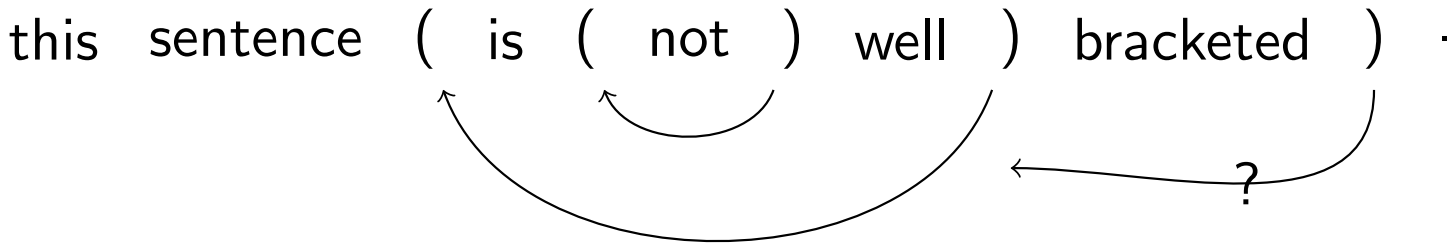
- Represent meaning using **formal logic** (more on day 10)
- Bits of sentence: bits of logic
  - Often with unfilled ‘holes’
- Follow parse tree to combine bits

PARSING FOR SEMANTICS

- Process of **composition** main reason for parsing
- No time for details now
- Not all semantic representations are *logical*
- More on day 7...

FSTs FOR SYNTAX?

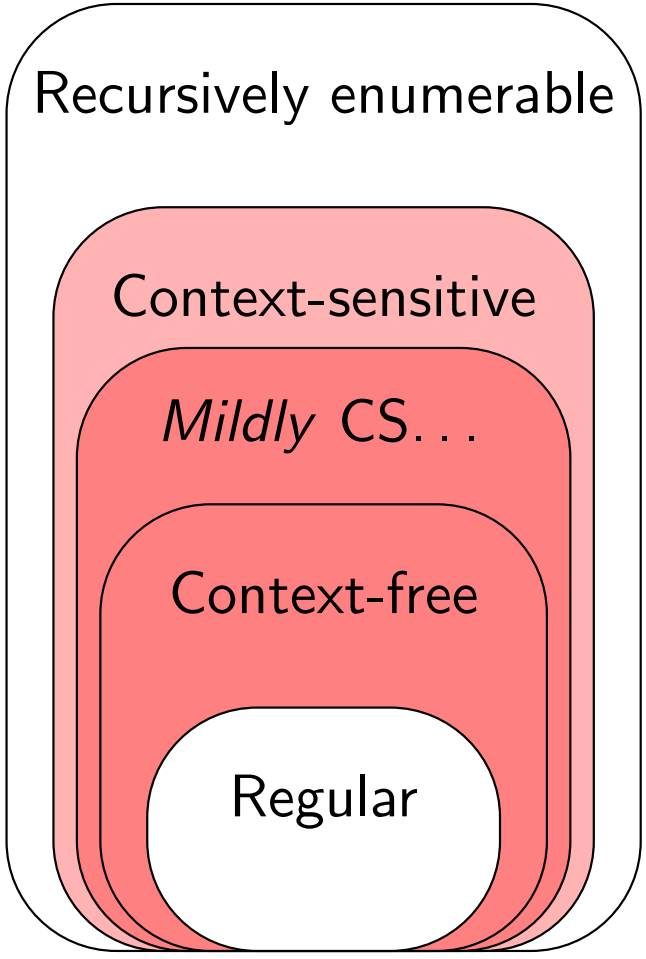
Question: why cant we use FSTs?



Alice, with her big boots on, seeing the man without a hat on on top of the hill, walk**s** quickly

THE CHOMSKY HIERARCHY: SYNTAX

- Expressivity to capture recursion, hierarchical structure
- *At least* CF power
- Enough for *most* language use
- A *few* phenomena in many languages exceed CF
- Other formalisms capture this *CCG, TAG, HG, LIG*, ...



STRONG VS WEAK EQUIVALENCE

Two grammars *equivalent* if they generate same languages

- **Weakly equivalent**: same set of strings
- **Strongly equivalent**: same strings *and* same underlying structures: bracketing/dependencies

STATISTICAL PARSING

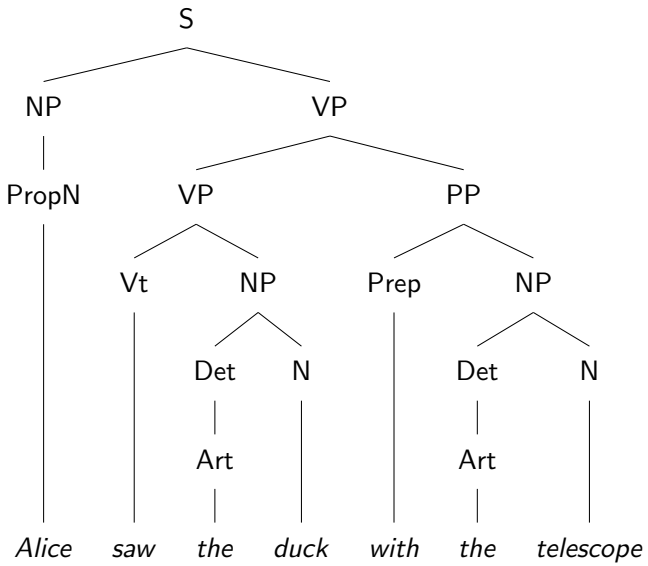
- CKY parses **exhaustively**
- Finds **all** possible trees
- But how do we know which is correct?
- Use **statistics** from corpus of parse trees
    - Compute scores for whole trees
    - Ignore unlikely-looking trees
- Another use: non-exhaustive parsing
    - Compute scores for partial trees
    - Throw away unlikely-looking *subtrees*
    - Keep high-scoring subtrees

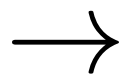
STATISTICS FROM WHERE?

- Need a **corpus** of correct parse trees
  ‘**treebank**’
- Get linguists to **annotate** sentences
- Compute **probabilities** for subtrees from counts in corpus
- Combine to estimate probabilities of whole tree

TREEBANK GRAMMARS

- Can also derive **grammar** from treebank
- Don't write grammar: just annotate treebank
- Grammar $\equiv$ expansions used in treebank





Rules

S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
NP $\rightarrow$ PropN
Det $\rightarrow$ Art
VP $\rightarrow$ Vt NP

Lexicon

Art $\rightarrow$ the
PropN $\rightarrow$ Alice
N $\rightarrow$ duck | telescope
Vt $\rightarrow$ saw

THE PENN TREEBANK

- First treebank: 1989
- Annotated syntactic trees (and other things)
- *Wall Street Journal* articles
- Other domains added later
- $\sim$40k sentences, $\sim$1M words
- Similar treebanks built for other languages, domains, annotation types

PROBABILISTIC CFGs (PCFGs)

- CFG with probability distributions associated with expansion
- **Generative** prob dist follows generative grammar rules

For rule $A \rightarrow B$ estimate conditional probability $p(B|A)$

- Finite possible Bs

$$\sum_X p(X|A) = 1$$

- Maximum likelihood estimate from treebank

$$p(B|A) = \frac{count(A \rightarrow B)}{\sum_X count(A \rightarrow X)}$$

PCFGs

- Problem: many rules, lots of low counts
- Use standard **smoothing** techniques with MLE

- Probability of tree is product of all expansions:

$$p(t) = \prod_{A \to B \in t} p(B|A)$$

- Exact extension of **Markov assumption** to CFGs

PCFG EXAMPLE

Rules

S $\rightarrow$ NP VP	1.0
NP $\rightarrow$ Det N	0.5
NP $\rightarrow$ NP PP	0.3
NP $\rightarrow$ PropN	0.2
Det $\rightarrow$ PosPro	0.3
Det $\rightarrow$ Art	0.7
VP $\rightarrow$ Vt NP	0.65
VP $\rightarrow$ VP PP	0.35
PP $\rightarrow$ Prep NP	1.0

Lexicon

Art $\rightarrow$ the	0.5
Art $\rightarrow$ a	0.5
PropN $\rightarrow$ Alice	1.0
N $\rightarrow$ duck	0.55
N $\rightarrow$ telescope	0.45
Vt $\rightarrow$ saw	1.0
PosPro $\rightarrow$ my	0.45
PosPro $\rightarrow$ her	0.55
Prep $\rightarrow$ with	0.6
Prep $\rightarrow$ on	0.4

PCFG EXAMPLE

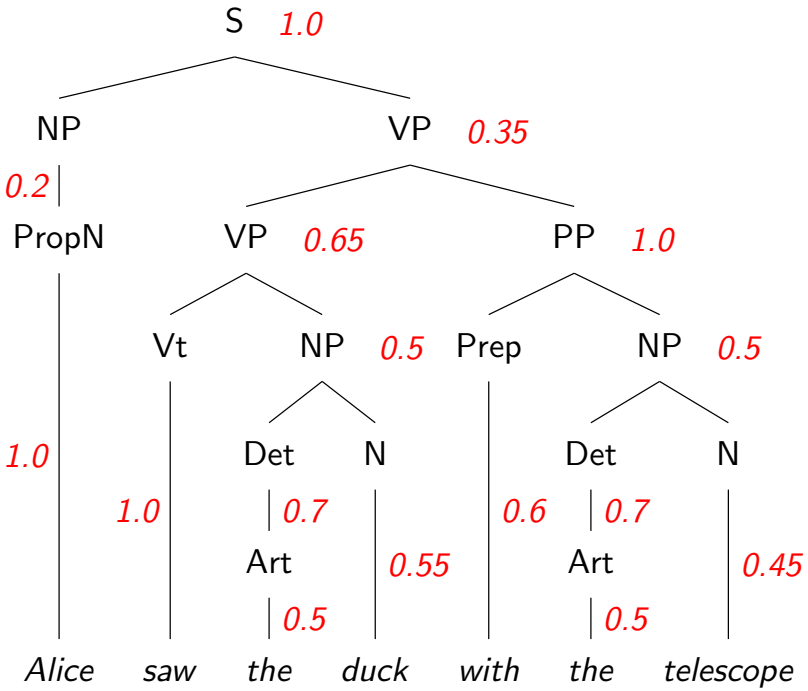
Rules

S $\rightarrow$ NP VP	1.0
NP $\rightarrow$ Det N	0.5
NP $\rightarrow$ NP PP	0.3
NP $\rightarrow$ PropN	0.2
Det $\rightarrow$ PosPro	0.3
Det $\rightarrow$ Art	0.7
VP $\rightarrow$ Vt NP	0.65
VP $\rightarrow$ VP PP	0.35
PP $\rightarrow$ Prep NP	1.0

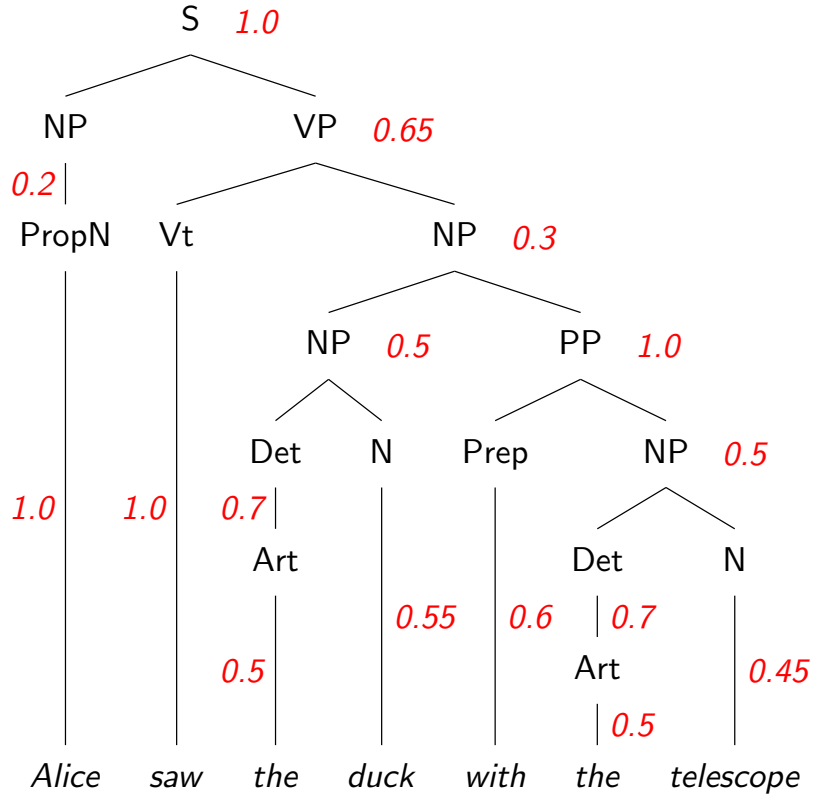
Lexicon

Art $\rightarrow$ the	0.5
Art $\rightarrow$ a	0.5
PropN $\rightarrow$ Alice	1.0
N $\rightarrow$ duck	0.55
N $\rightarrow$ telescope	0.45
Vt $\rightarrow$ saw	1.0
PosPro $\rightarrow$ my	0.45
PosPro $\rightarrow$ her	0.55
Prep $\rightarrow$ with	0.6
Prep $\rightarrow$ on	0.4

PCFG DISAMBIGUATION



$$p(t) = 2.07 \times 10^{-4}$$



$$p(t) = 2.96 \times 10^{-4}$$

SYNTACTIC LANGUAGE MODEL

- Derivation probability actually

$$p(t, W|S)$$

  W: word sequence (sentence)
- Compute $p(W)$ by summing over all possible trees

$$\sum_{t \in T} p(t, W|S)$$

- **Language model**
- Takes account of syntactic structures to compute probability

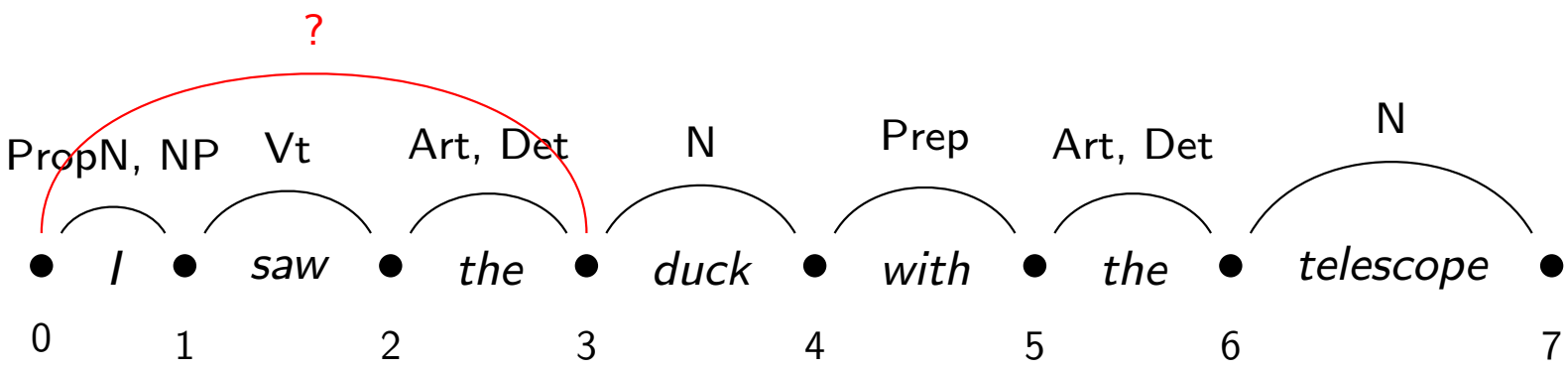
USING PROBABILITIES TO PARSE

- Treebank grammars are *huge*!
- Exhaustive parsing of *shorter* PTB sentences:
  average 1M items in chart![1]
- Most full parses: low probability
- Can we avoid generating many parses that will never be used?
  - Compute scores for partial trees
  - Throw away unlikely-looking *subtrees*
  - Keep high-scoring subtrees

[1] Charniak, Goldwater & Johnson (1998)

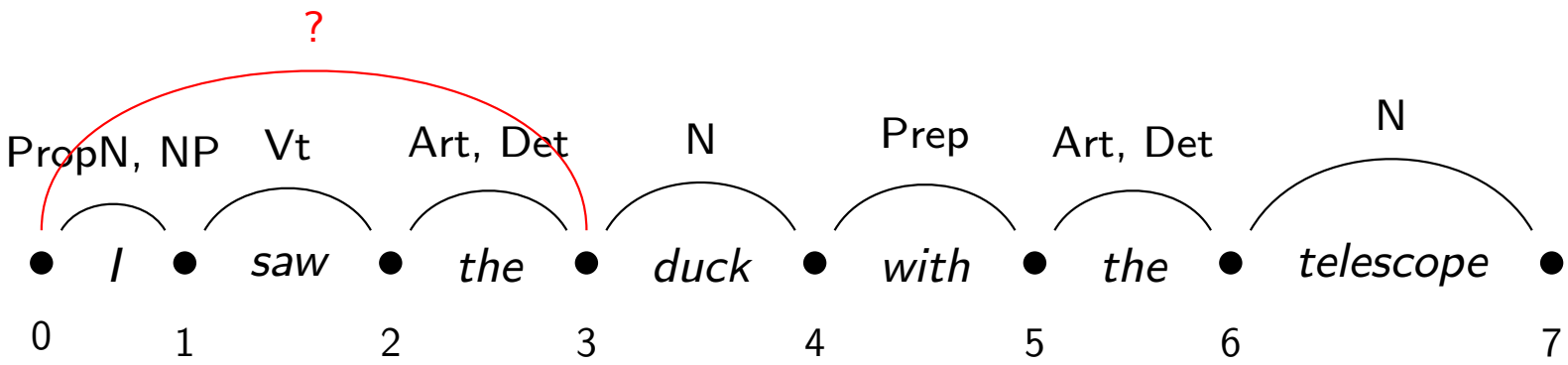
BEAM SEARCH

- Keep just most probable constituent for each span
  $\rightarrow$ might not result in full parse
- Others (less probable) might have led to S
- Throw away *very* improbable constituents
  1. Don't look promising
  2. If produce full parse, likely to have low score



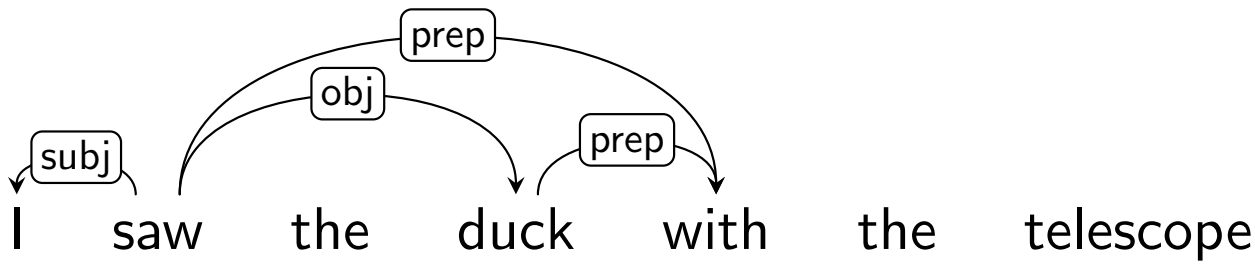
BEAM SEARCH

- Throw away *very* improbable constituents
- On each edge, store top K subtrees
- Higher K:
  - less likely to miss things
  - slower parsing



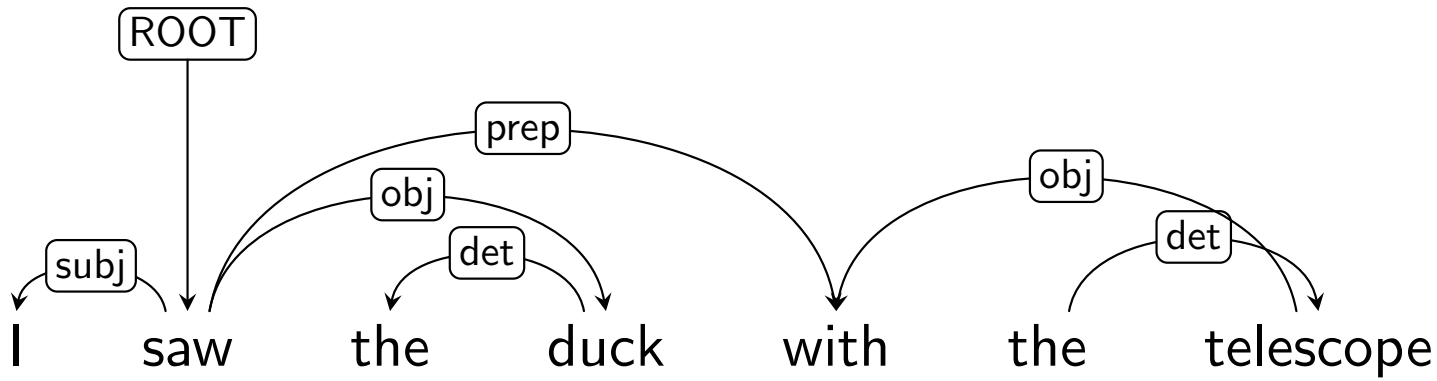
DEPENDENCIES

- **Dependencies**: syntactic *connections* between words
- Parse trees capture
  - syntactic dependencies
  - constituency (phrase categories)
- Can draw dependencies explicitly:

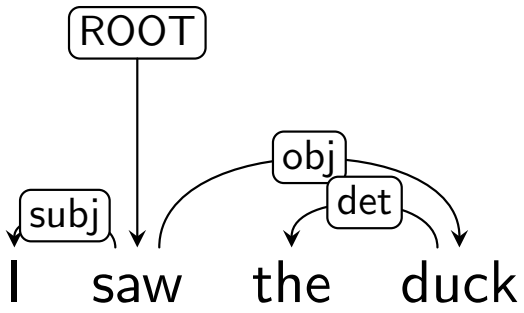


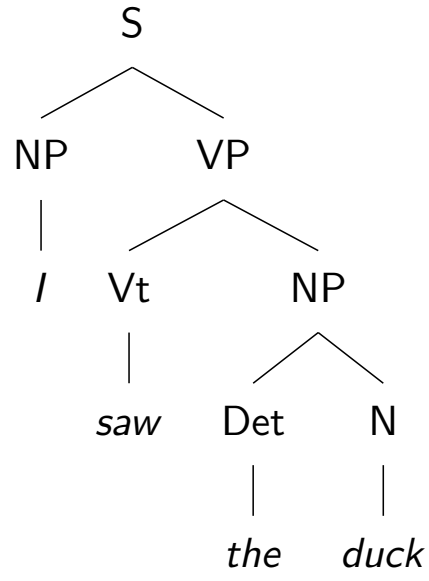
DEPENDENCIES

- Different way to represent syntax
- Complete **tree** of dependency relations
- Covers whole sentence



DEPENDENCY STRUCTURES AND TREES

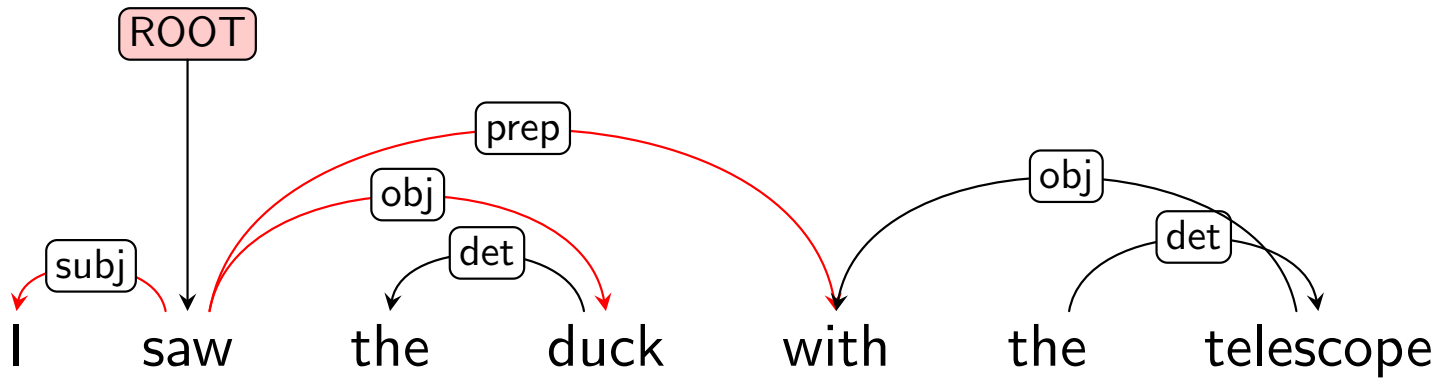




- Deps: no notion of **constituency**
- In some sense, closer to **meaning** / **semantics**
- More natural way to handle **word order** variation
    - Easy to adjust formalism for **free word-order** languages

DEPENDENCY GRAPHS

- Every word **dependent** on exactly one other
- One word dependent on unique **root**
- A word may have multiple dependent words



DEPENDENCY PARSING

- Various algorithms
- Here: **transition-based** parsing
- Incremental: word by word, left to right
- Same algorithm *can* be applied to CFG parsing
    - But we didn't do that
    - *Chart parsing*

TRANSITION-BASED PARSING

- Single pass through words, left to right
- Maintain a **stack**
- In each step, either
  - Shift new word onto stack; or
  - link top two items, using LeftArc or RightArc
- Just keep head word on stack after linking
- Continue until
  - No words left
  - Just ROOT on stack
- **Oracle** component choose actions:

  Shift, LeftArc, RightArc

DEPENDENCY PARSING EXAMPLE

Input: *I saw the duck*

Stack	**Words**	Action
ROOT	I, saw, the, duck	SHIFT
ROOT, I	saw, the, duck	SHIFT
ROOT, I, saw	the, duck	LEFTARC: I $\leftarrow$ saw
ROOT, saw	the, duck	SHIFT
ROOT, saw, the	duck	SHIFT
ROOT, saw, the, duck		LEFTARC: the $\leftarrow$ duck
ROOT, saw, duck		RIGHTARC: saw $\rightarrow$ duck
ROOT, saw		RIGHTARC: ROOT $\rightarrow$ saw
ROOT		*Done*

ORACLE

- **Oracle**: chooses action at each step
- Type of **classification**
- Input: stack, word list
- Classes: SHIFT, LEFTARC, RIGHTARC
- **Supervised** ML: train on dependency *treebank*
- Any classifier: MaxEnt, SVM, neural network, ...
- Many possible features: words seen, previous actions, POSs, morphology, ...

DEPENDENCY PARSING WITH LABELS

- Same algorithm
- Oracle must now choose **label** as well as action
- Still a classifier: more classes

$$\textsc{LeftArc} \Rightarrow \text{LA}\textsc{Subj}, \text{LA}\textsc{Obj}, \text{LA}\textsc{Prep}, \ldots$$

etc. . .

DEPENDENCY PARSING

- Can be extremely fast
- Transition-based algorithm is **greedy**
- If it makes a mistake, can't backtrack
- No grammar!
  - All actions are allowed
  - Effectively learn grammar from *treebank*
  - Soft grammar: learned by classifier
- Extremely flexible in what it captures
  - Just add **features** to classifier
  - Agreement, case, ...

INCREMENTALITY

- Transition-based algorithm is **incremental**
- Processes **left to right**
- Desirable for parsing: *it's what humans do!* (mostly)
- Practical: partial syntax available halfway through
- Other formalisms permit incrementality
  - E.g. CKY for CFGs
  - Adjust **grammar** to make more incremental
  - Other **algorithms** more incremental
- Some formalisms more focussed on this:
  e.g. *Combinatory Categorial Grammar* (CCG)

SUMMARY

- **Structure** in language
- **Long-range dependencies**
- **Formal syntax**
- Syntactic disambiguation
- Syntax with **CFG**s
- **Parsing**, **CKY**
- **Statistical parsing**
- **Dependency** structures
- Transition-based **dependency parsing**

READING MATERIAL

- Syntax, CFGs, treebanks:
  J&M3 10.1-4, *Eisenstein ch 9*
- Parsing, CKY:
  J&M3 ch 11, *Eisenstein 10.1-2*
- Statistical parsing, PCFGs:
  J&M3 ch 12, *Eisenstein 10.3-4*
- Dependency parsing:
  J&M3 ch 13, *Eisenstein ch 11*

NEXT UP

After lunch: practical assignments in **BK107**

9:15 – 12:00	Lectures
12:00 – 13:15	Lunch
13:15 – ∼13:30	Introduction
13:30 – 16:00	Practical assignments

- Writing a CFG
- Parsing
- Building a *treebank* grammar
- Estimating PCFG probabilities
- Out-of-the-box parsers