COS 484

Natural Language Processing

L8: Constituency Parsing

Spring 2022

(Some slides adapted from Chris Manning, Mike Collins, Yoav Artzi)

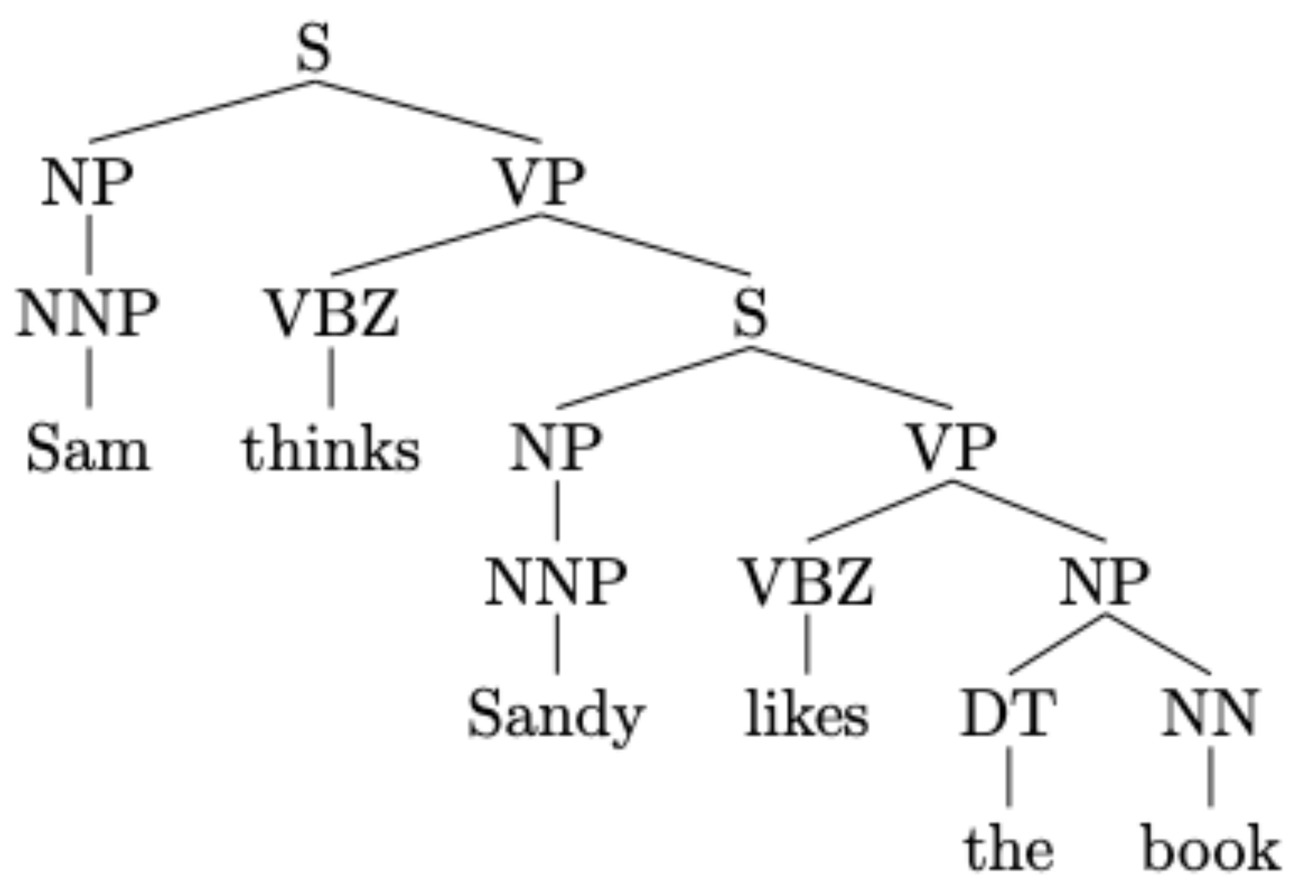
Syntactic structure: constituency and dependency

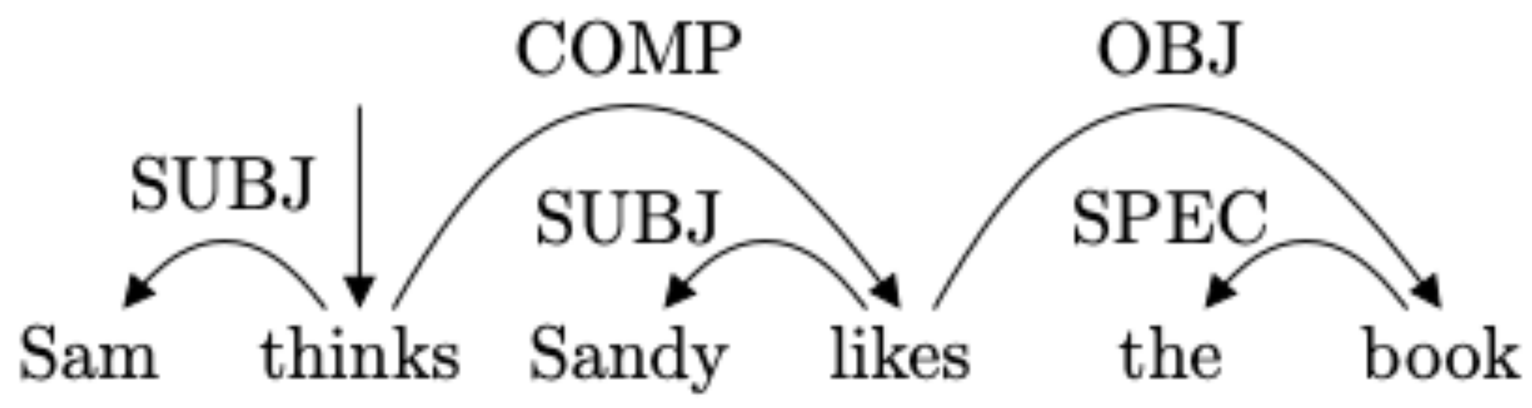
Theme: How do we represent the structure of **sentences** using (syntax) **trees**?

Two views of linguistic structure

- **Constituency** (today)
  - = phrase structure grammar
  - based on context-free grammars (CFGs)

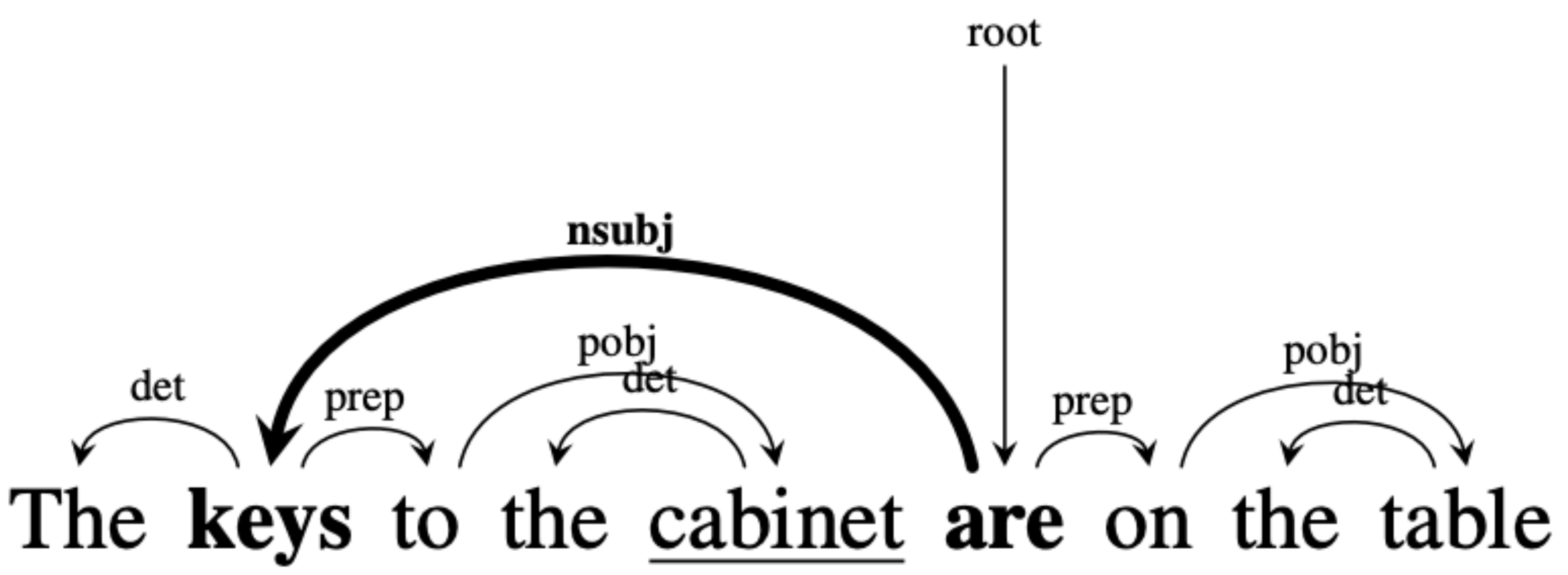
- **Dependency** (next class)

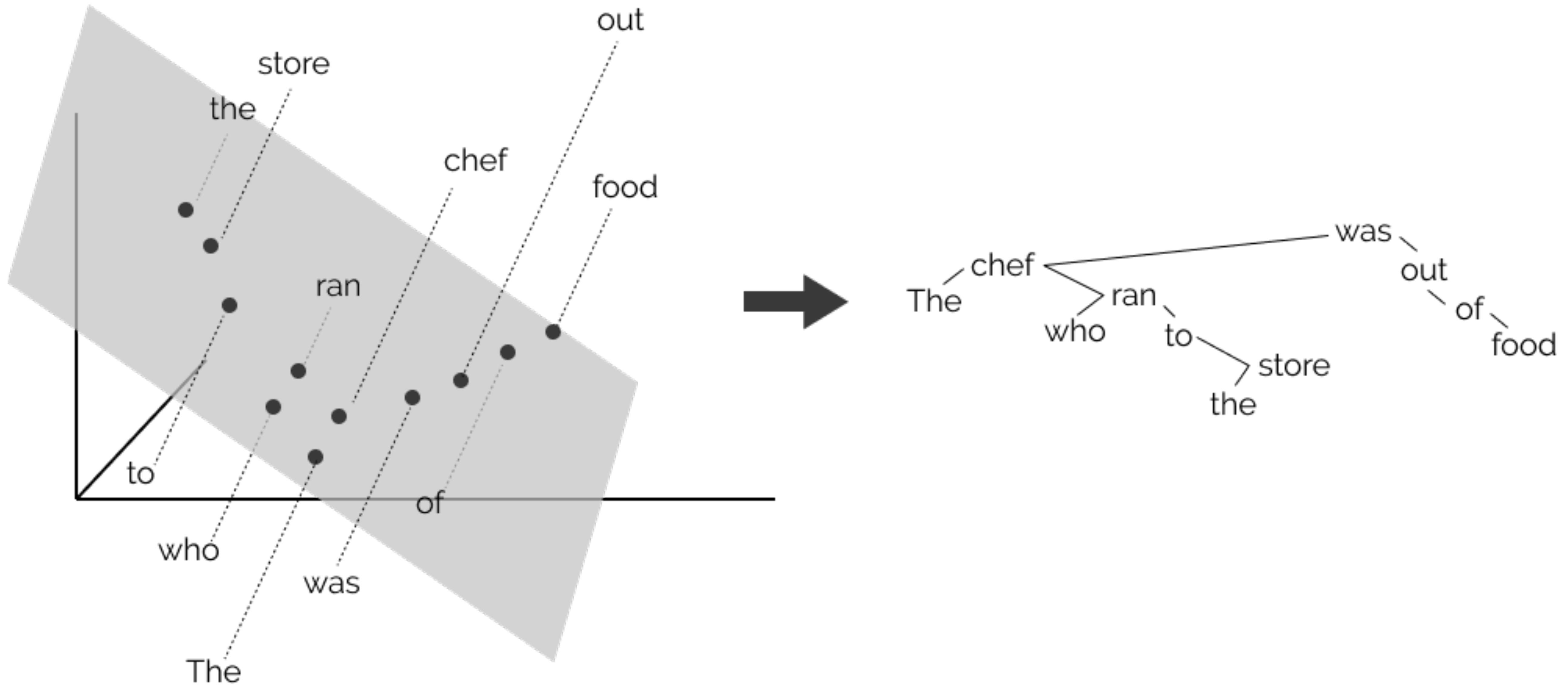




Tree structures in the deep learning era

The keys to the cabinet <u>is/are</u> on the table.





(Linzen et al., 2016): Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies

(Hewitt and Manning, 2019): A Structural Probe for Finding Syntax in Word Representations

This lecture

- Constituency structure
- Context-free grammar (CFG)
- Probabilistic context-free grammar (PCFG)
- Treebanks
- The CKY algorithm
- Evaluation
- Lexicalized PCFGs

Constituency structure

- **Phrase structure** organizes words into **nested constituents**
- Starting units: words are given a category: part-of-speech tags

  the, cuddly, cat, by, the, door

  Det, Adj, Noun, Prep, Det, Noun

- Words combine into phrases with categories

  the cuddly cat, by the door

  NP→Det Adj Noun   PP → Prep Det Noun

- Phrases can combine into bigger phrases recursively

  the cuddly cat by the door

  NP → NP PP

NP: noun phrase, PP: prepositional phrase

Syntactic parsing

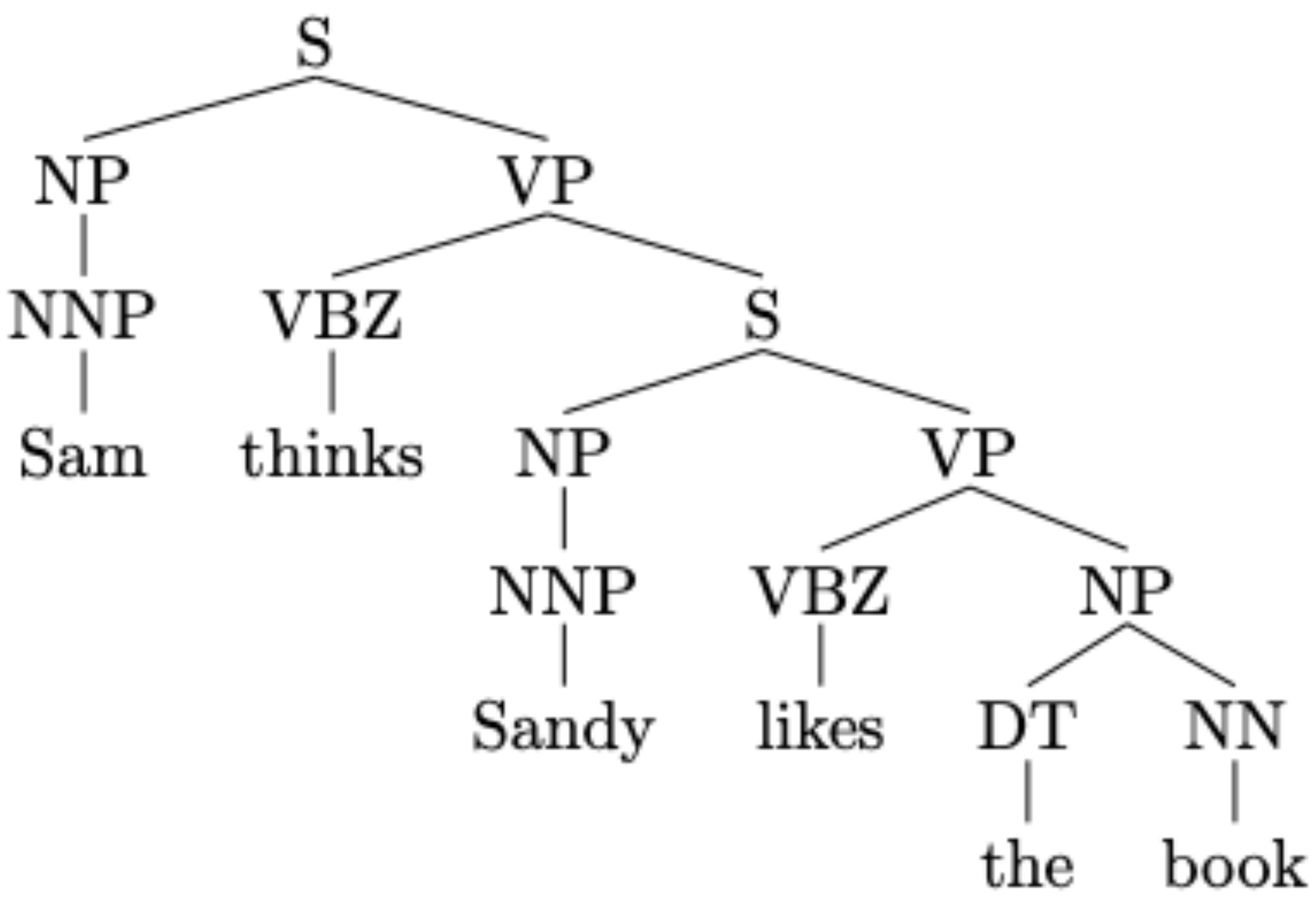
Syntactic parsing is the task of recognizing a sentence and assigning a structure to it.

(**Constituency** parsing is the task of recognizing a sentence and assigning a **constituency** structure to it.)

Input

Sam thinks Sandy likes the book

Output



Syntactic parsing: applications

- Grammar checking
  - If a sentence can't be parsed, it may have grammatical errors (or at least hard to read)

- Used as intermediate representations for downstream tasks
  - Machine translation (syntax-based statistical MT)
  - Information extraction
  - Question answering

Syntactic parsing: applications

Used as intermediate representation for downstream applications

English word order: subject — verb — object

Japanese word order: subject — object — verb

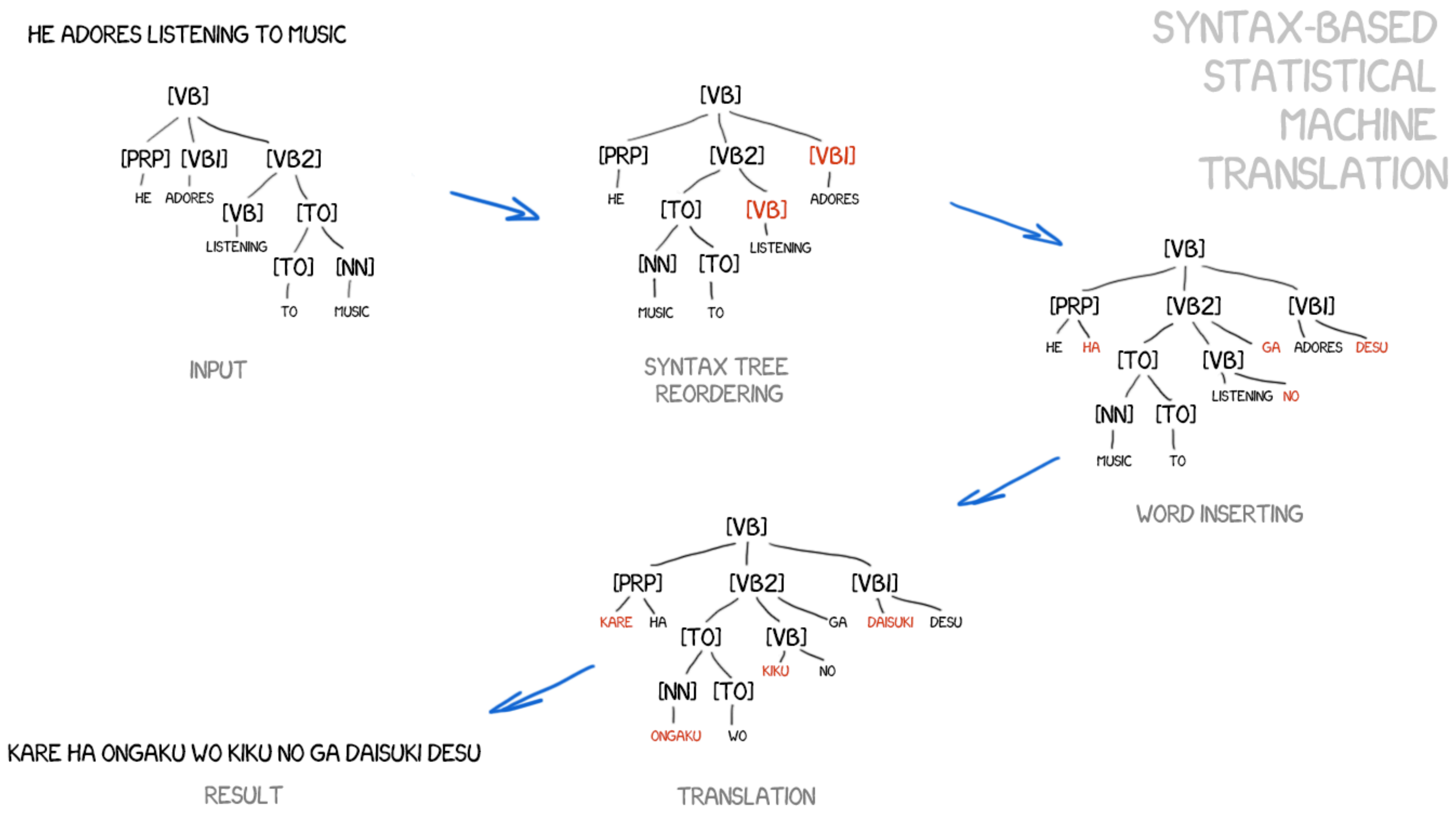


Image credit: http://vas3k.com/blog/machine_translation/

Syntactic parsing: applications

Used as intermediate representation for downstream applications

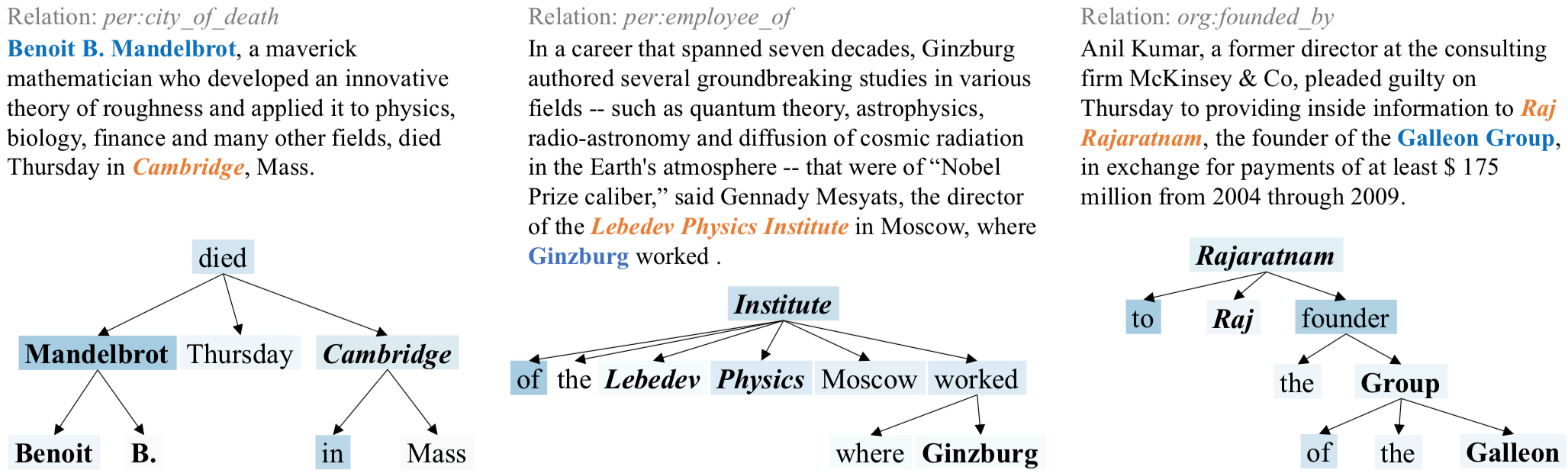


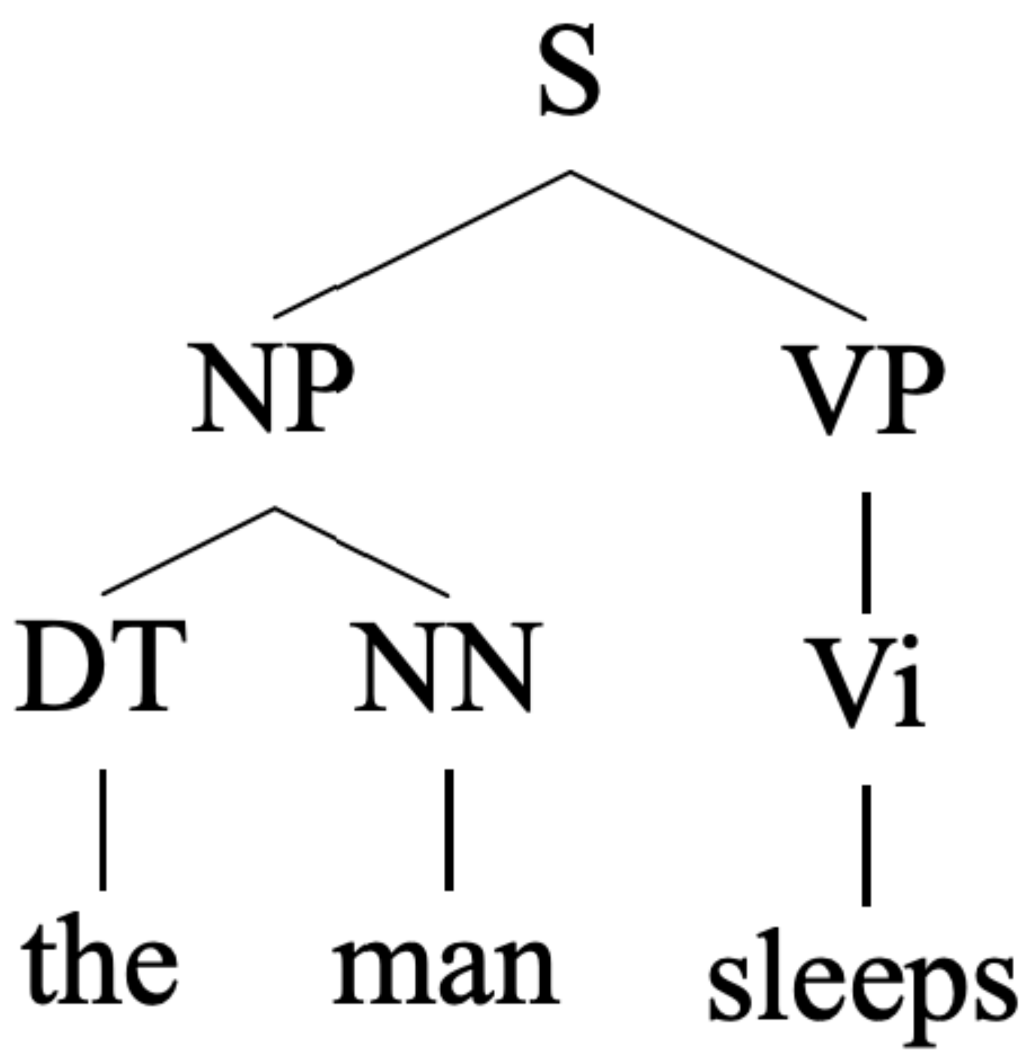
Image credit: (Zhang et al, 2018)

Context-free grammars (CFGs)

- The most widely used formal system for modeling constituency structure in English and other natural languages

- A context free grammar $G = (N, \Sigma, R, S)$ where
  - N is a set of non-terminal symbols
    - Phrasal categories: S, NP, VP, …
    - Parts-of-speech (pre-terminals): DT, NN, Vi, …
  - Σ is a set of terminal symbols: the, man, sleeps, ..
  - R is a set of rules of the form $X \rightarrow Y_1 Y_2 \ldots Y_n$ for $n \geq 0$, $X \in N, Y_i \in (N \cup \Sigma)$
    - Examples: S → NP VP, NP → DT NN, NN → man
  - $S \in N$ is a distinguished start symbol

Not always the sentence non-terminal S



S:sentence, VP:verb phrase, NP: noun phrase, DT:determiner, NN: noun, Vi: intransitive verb…

A context-free grammar for English

$N = \{\underline{\text{S, NP, VP, PP,}}\ \underline{\text{DT, Vi, Vt, NN, IN}}\}$
$S = \text{S}$
$\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$

$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Grammar

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

Lexicon

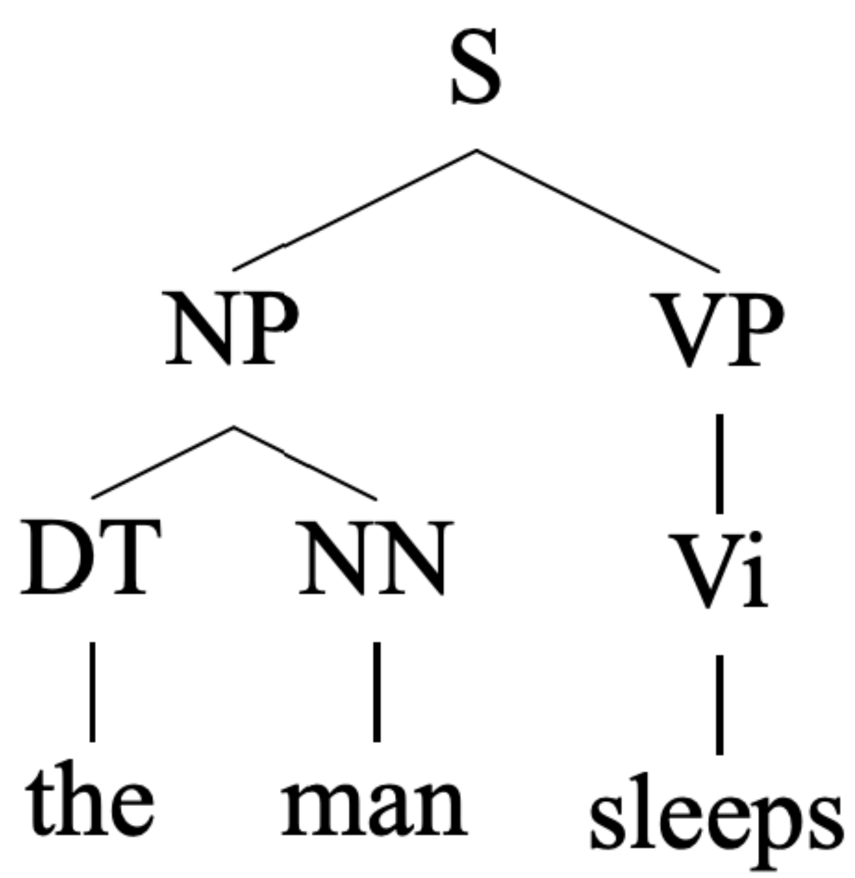
S:sentence, VP:verb phrase, NP: noun phrase, PP:prepositional phrase, DT:determiner, Vi:intransitive verb, Vt:transitive verb, NN: noun, IN:preposition

(Left-most) Derivations

- Given a CFG G, a left-most derivation is a sequence of strings $s_1, s_2, \ldots, s_n$, where
    - $s_1 = S$
    - $s_n \in \Sigma^*$: all possible strings made up of words from Σ
    - Each s_i for $i = 2, \ldots, n$ is derived from s_{i-1} by picking the left-most non-terminal X in s_{i-1} and replacing it by some β where $X \rightarrow \beta \in R$
- s_n: yield of the derivation

(Left-most) Derivations

- $s_1 = \text{S}$
- $s_2 = \text{NP VP}$
- $s_3 = \text{DT NN VP}$
- $s_4 = \text{the NN VP}$
- $s_5 = \text{the man VP}$
- $s_6 = \text{the man Vi}$
- $s_7 = \text{the man sleeps}$



A derivation can be represented as a parse tree!

$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

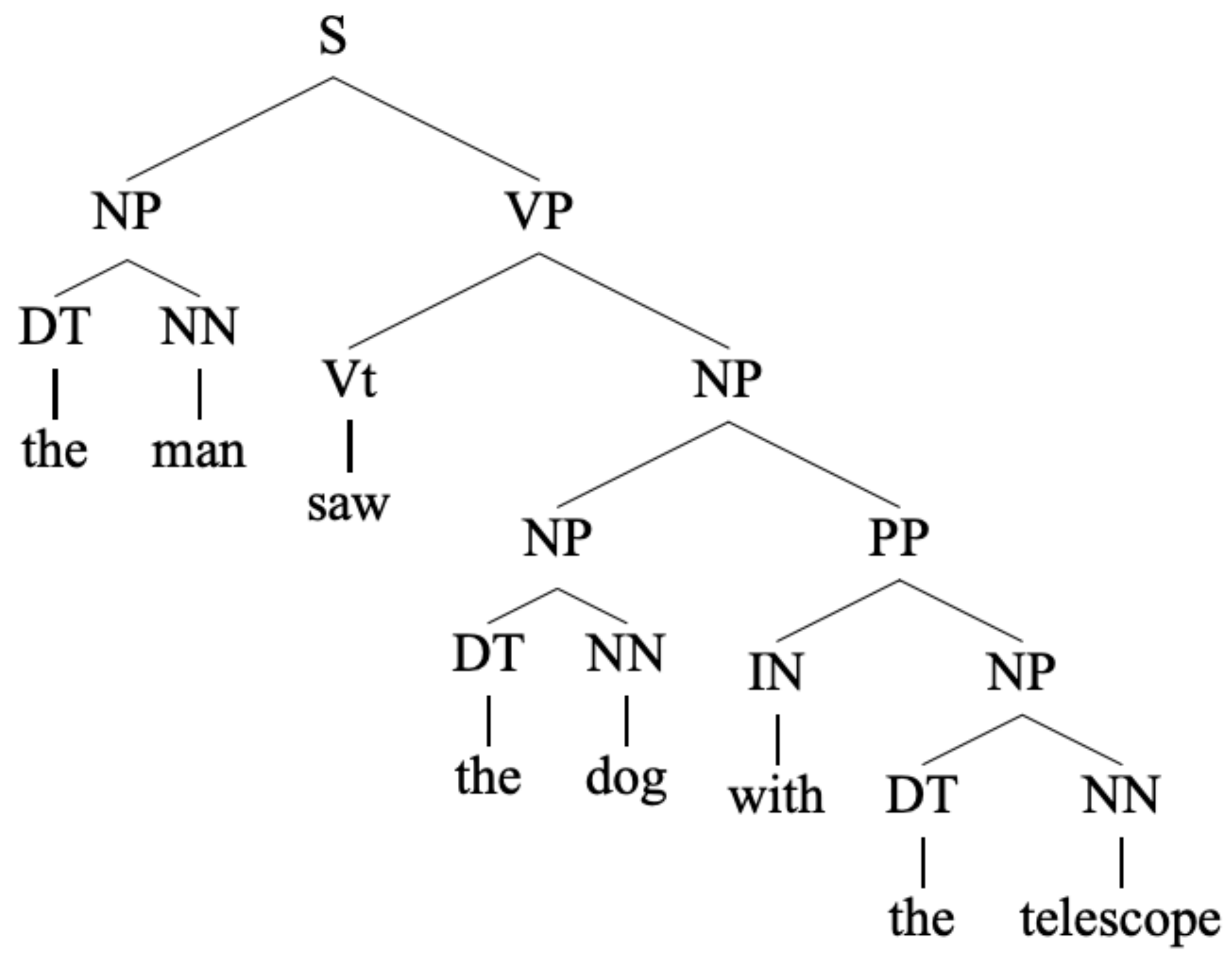
- A string $s \in \Sigma^*$ is in the language defined by the CFG if there is at least one derivation whose yield is s
- The set of possible derivations may be finite or infinite

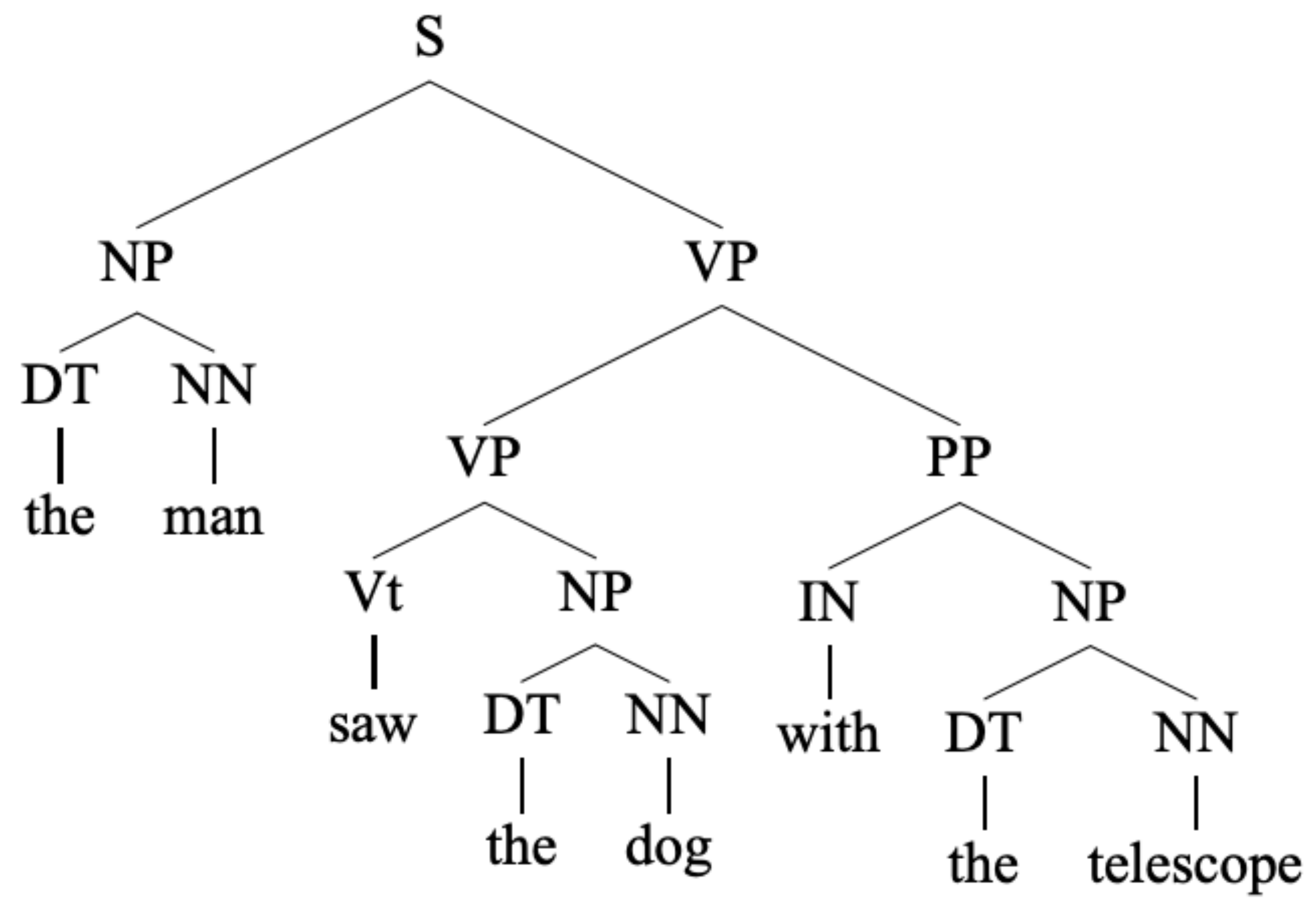
Q: Why do we want to replace the leftmost non-terminal every time?

Ambiguity

Some **sentences/phrases** may have more than one derivation (i.e. more than one parse tree!).

Attachment ambiguity (e.g., PP attachment)





Q: Which one is the correct parse tree?

Ambiguity

Some sentences/phrases may have more than one derivation (i.e. more than one parse tree!).

Coordination ambiguity

old men and women

old [men and women]   [old men] and women

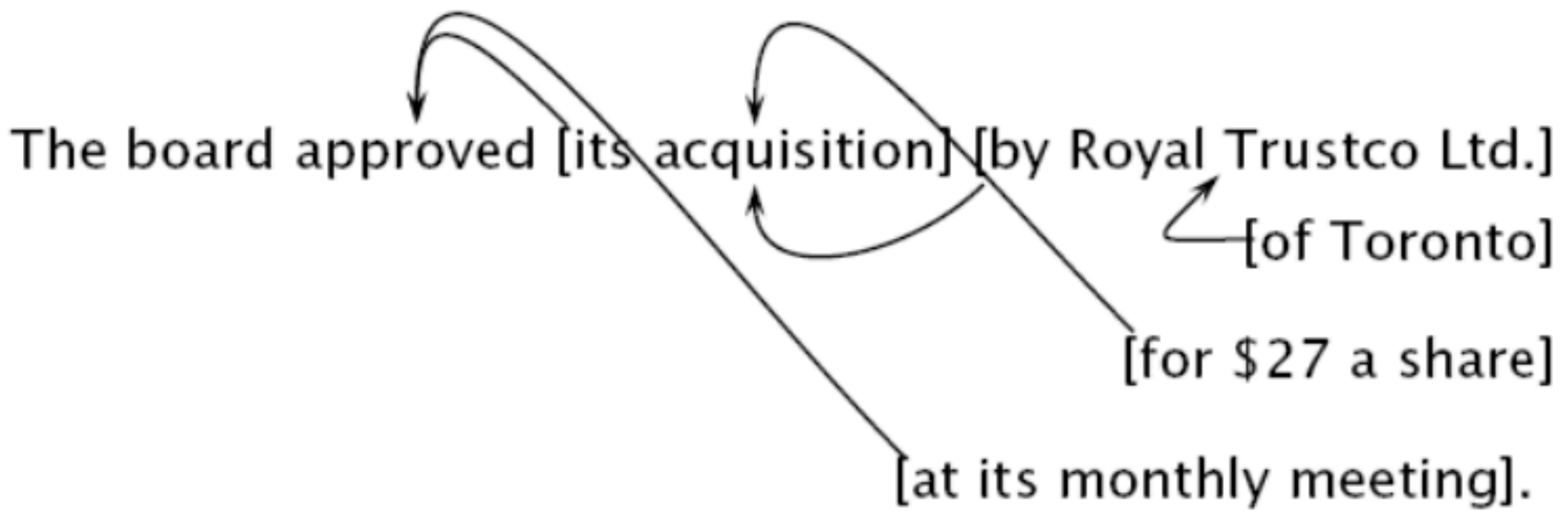
President Kennedy today pushed aside other White House business to devote all his time and attention to working on the Berlin crisis address he will deliver tomorrow night to the American people over nationwide television and radio.

Q: What ambiguities are there in this sentence?

Sentences can have a large number of parses

- In fact, sentences can have a very large number of possible parses

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



((ab)c)d   (a(bc))d   (ab)(cd)   a((bc)d)   a(b(cd))

Catalan number: $C_n = \frac{1}{n+1}\binom{2n}{n}$

- There is no way to choose the right parse!
- Constructing a grammar is difficult— a less constrained grammar can parse more sentences but result in more parses for even simple sentences

Probabilistic context-free grammars (PCFGs)

A probabilistic context-free grammar (PCFG) consists of:

- A context-free grammar: $G = (N, \Sigma, R, S)$
- For each rule $\alpha \to \beta \in R$, there is a parameter (probability) $q(\alpha \to \beta) \geq 0$.
  For any $X \in N$,
  $$\sum_{\alpha \to \beta : \alpha = X} q(\alpha \to \beta) = 1$$

$R, q =$

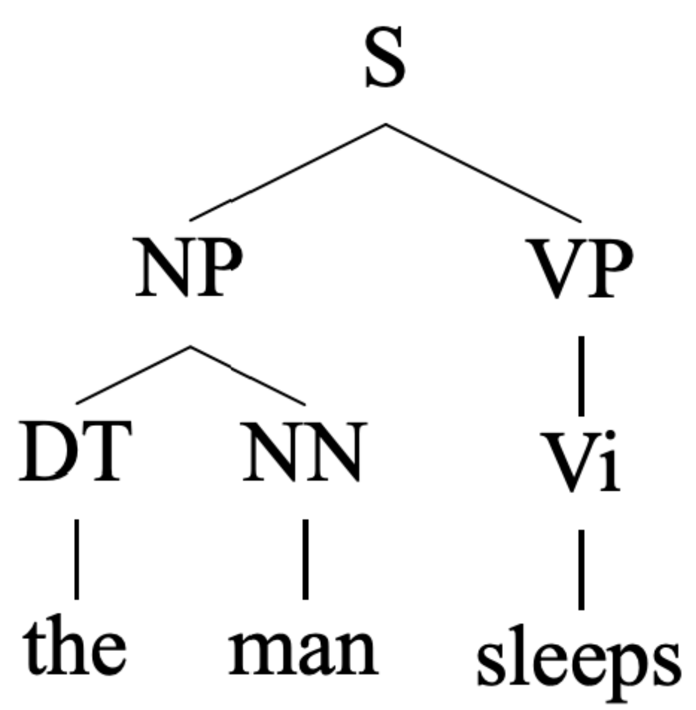
S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

Probabilistic context-free grammars (PCFGs)

For any derivation (parse tree) containing rules:
$\alpha_1 \to \beta_1, \alpha_2 \to \beta_2, \ldots, \alpha_l \to \beta_l$, the probability of the parse is:

$$\prod_{i=1}^{l} q(\alpha_i \to \beta_i)$$



$R, q =$

S	$\to$	NP	VP	1.0
VP	$\to$	Vi		0.3
VP	$\to$	Vt	NP	0.5
VP	$\to$	VP	PP	0.2
NP	$\to$	DT	NN	0.8
NP	$\to$	NP	PP	0.2
PP	$\to$	IN	NP	1.0

Vi	$\to$	sleeps	1.0
Vt	$\to$	saw	1.0
NN	$\to$	man	0.1
NN	$\to$	woman	0.1
NN	$\to$	telescope	0.3
NN	$\to$	dog	0.5
DT	$\to$	the	1.0
IN	$\to$	with	0.6
IN	$\to$	in	0.4

$$P(t) = q(\text{S} \to \text{NP VP}) \times q(\text{NP} \to \text{DT NN}) \times q(\text{DT} \to \text{the})$$
$$\times\, q(\text{NN} \to \text{man}) \times q(\text{VP} \to \text{Vi}) \times q(\text{Vi} \to \text{sleeps})$$
$$= 1.0 \times 0.8 \times 1.0 \times 0.1 \times 0.3 \times 1.0 = 0.024$$

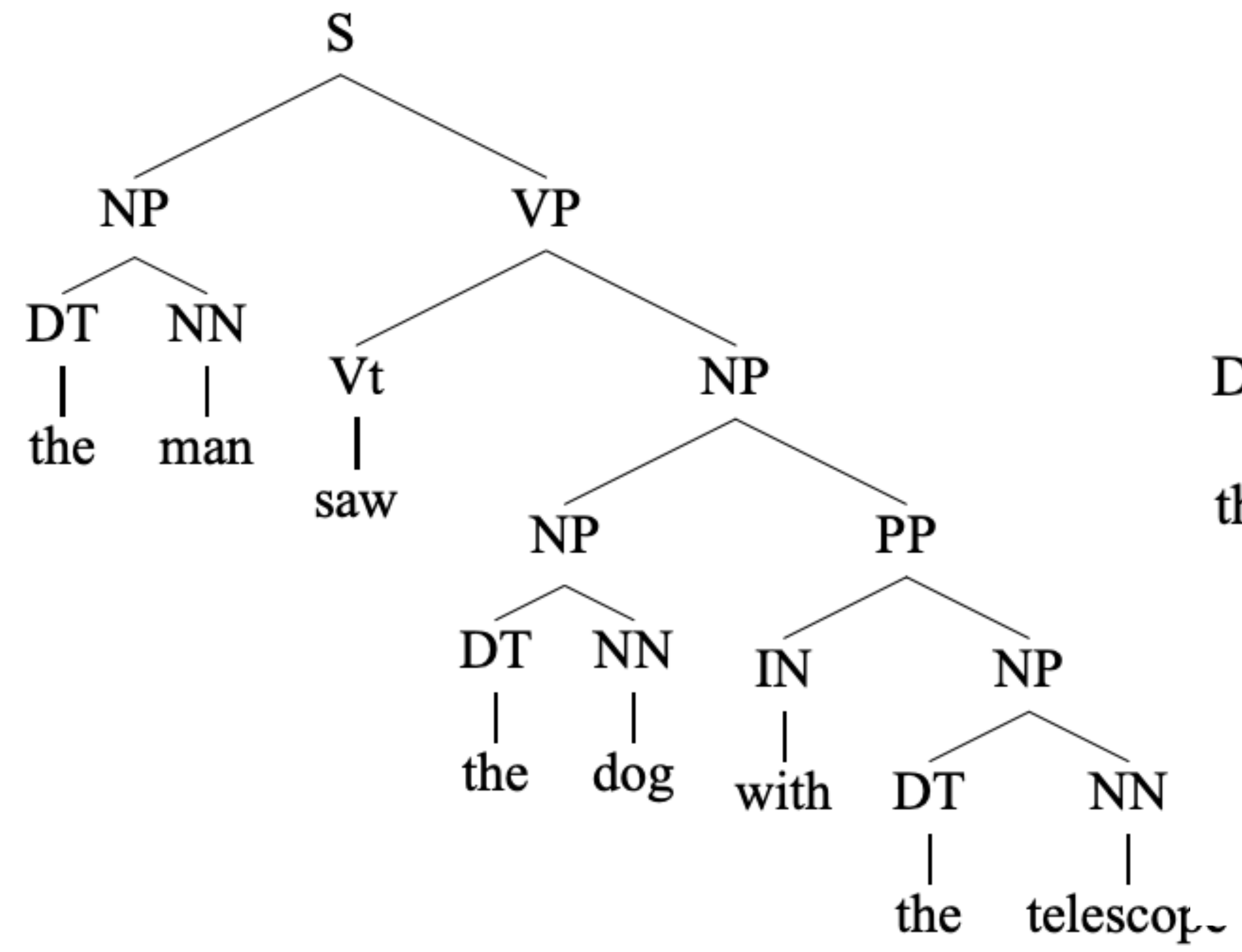
Q: Why do we want $\sum_{\alpha \to \beta : \alpha = X} q(\alpha \to \beta) = 1$?

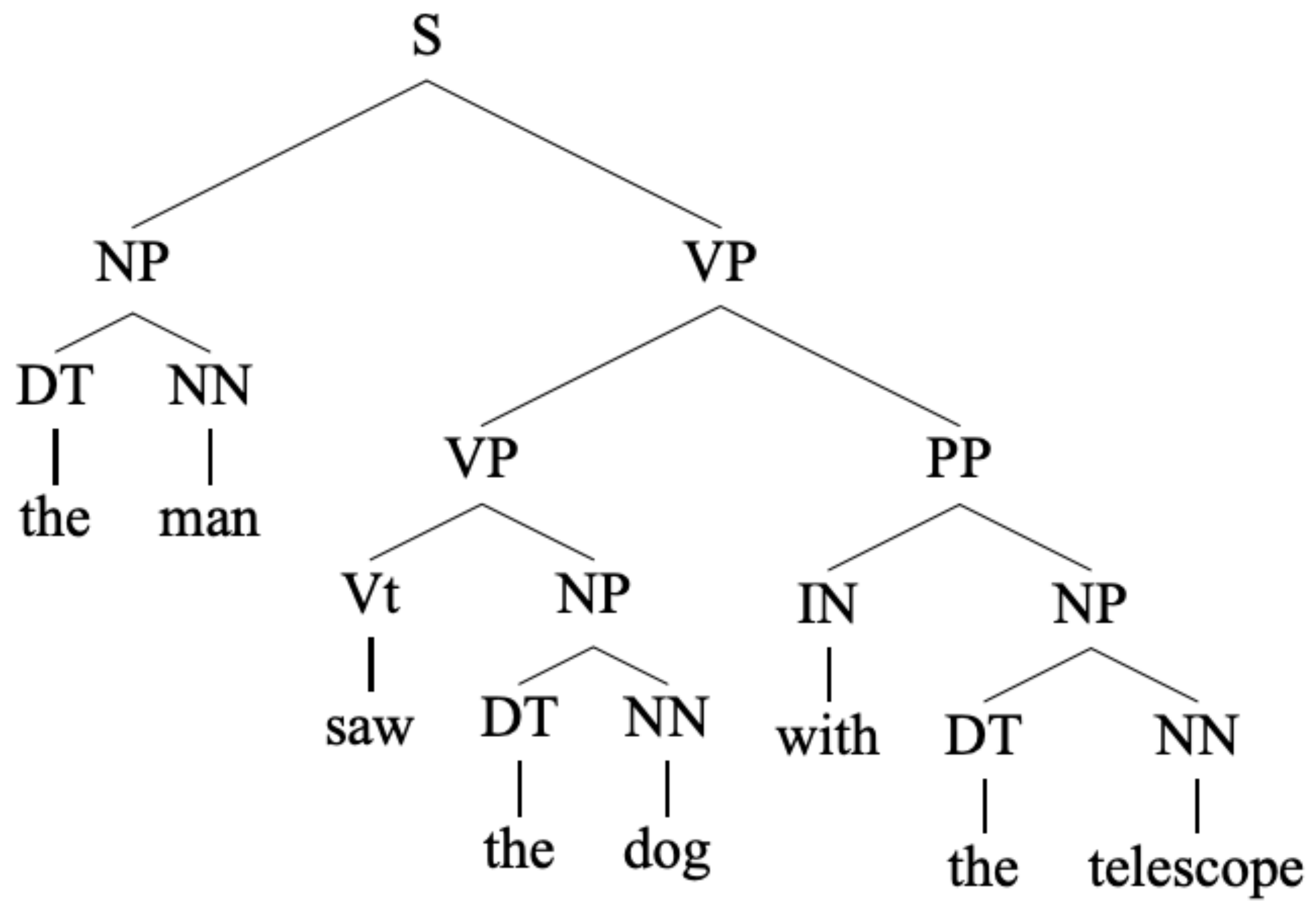
Which parse tree has a higher probability?

$R, q =$

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4





$$q(\text{VP} \rightarrow \text{Vt NP}) \times q(\text{NP} \rightarrow \text{NP PP}) = 0.5 \times 0.2 = 0.1$$

$$q(\text{VP} \rightarrow \text{VP PP}) \times q(\text{VP} \rightarrow \text{Vt NP}) = 0.2 \times 0.5 = 0.1$$

This PCFG can't identify the correct parse tree!

The rise of annotated data

- **Learning from data**: treebanks
- **Adding probabilities to the rules**: probabilistic CFGs

Treebanks: a collection of sentences paired with their annotated parse trees

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

The Penn Treebank Project (Marcus et al, 1993)

Penn Treebank

Standard setup

- 40,000 sentences for training
- 1,700 for development
- 2,400 for testing

Phrasal categories

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Subordinate clause
SBARQ	Direct question introduced by *wh*-element
SINV	Declarative sentence with subject-aux inversion
SQ	Yes/no questions and subconstituent of SBARQ excluding *wh*-element
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Constituent of unknown or uncertain category
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace of wh-Constituent

Penn Treebank

Part-of-speech tagset

CC	Coordinating conj.	TO	infinitival *to*
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present pple
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sg. present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sg. present
JJS	Adjective, superlative	WDT	Wh-determiner
LS	List item marker	WP	Wh-pronoun
MD	Modal	WP$	Possessive *wh*-pronoun
NN	Noun, singular or mass	WRB	Wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	”	Right close double quote

Zoom poll

Which of the following statements is incorrect?

(a) A treebank can provide us frequencies and distributional information

(b) A treebank provides us a way to evaluate systems

(c) The treebank data can be biased to the selection of sentences/documents

(d) It is easy to scale up a treebank to multiple domains and languages

The answer is (d).

Deriving a PCFG from a treebank

- Training data: a set of parse trees $t_1, t_2, \ldots, t_m$
- A PCFG (N, Σ, S, R, q):
  - N is the set of all non-terminals seen in the trees
  - Σ is the set of all words seen in the trees
  - S is taken to be S.
  - R is taken to be the set of all rules $\alpha \rightarrow \beta$ seen in the trees

Deriving a PCFG from a treebank

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Deriving a PCFG from a treebank

Grammar	Lexicon
$S \rightarrow NP\ VP\ .$	$PRP \rightarrow we \mid he$
$S \rightarrow NP\ VP$	$DT \rightarrow the \mid that \mid those$
$S \rightarrow$ " S " , $NP\ VP\ .$	$JJ \rightarrow cold \mid empty \mid full$
$S \rightarrow$ -NONE-	$NN \rightarrow sky \mid fire \mid light \mid flight \mid tomorrow$
$NP \rightarrow DT\ NN$	$NNS \rightarrow assets$
$NP \rightarrow DT\ NNS$	$CC \rightarrow and$
$NP \rightarrow NN\ CC\ NN$	$IN \rightarrow of \mid at \mid until \mid on$
$NP \rightarrow CD\ RB$	$CD \rightarrow eleven$
$NP \rightarrow DT\ JJ\ ,\ JJ\ NN$	$RB \rightarrow a.m.$
$NP \rightarrow PRP$	$VB \rightarrow arrive \mid have \mid wait$
$NP \rightarrow$ -NONE-	$VBD \rightarrow was \mid said$
$VP \rightarrow MD\ VP$	$VBP \rightarrow have$
$VP \rightarrow VBD\ ADJP$	$VBN \rightarrow collected$
$VP \rightarrow VBD\ S$	$MD \rightarrow should \mid would$
$VP \rightarrow VBN\ PP$	$TO \rightarrow to$
$VP \rightarrow VB\ S$	
$VP \rightarrow VB\ SBAR$	
$VP \rightarrow VBP\ VP$	
$VP \rightarrow VBN\ PP$	
$VP \rightarrow TO\ VP$	
$SBAR \rightarrow IN\ S$	
$ADJP \rightarrow JJ\ PP$	
$PP \rightarrow IN\ NP$	

A sample of the CFG grammar rules and lexical entries that would be extracted from the three treebank sentences

Deriving a PCFG from a treebank

- Training data: a set of parse trees $t_1, t_2, \ldots, t_m$
- A PCFG (N, Σ, S, R, q):
  - N is the set of all non-terminals seen in the trees
  - Σ is the set of all words seen in the trees
  - S is taken to be S.
  - R is taken to be the set of all rules $\alpha \rightarrow \beta$ seen in the trees
    - The maximum-likelihood parameter estimates are:

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

If we have seen the rule VP $\rightarrow$ Vt NP 105 times, and the the non-terminal VP 1000 times, $q(\text{VP} \rightarrow \text{Vt NP}) = 0.105$

Parsing with PCFGs

- Given a sentence s and a PCFG, how to find the highest scoring parse tree for s?

$$argmax_{t \in \mathcal{T}(s)} P(t)$$

- **The CKY algorithm:** applies to a restricted type of PCFG— a PCFG in Chomsky normal form (CNF)
  - CKY = the Cocke-Kasami-Younger algorithm

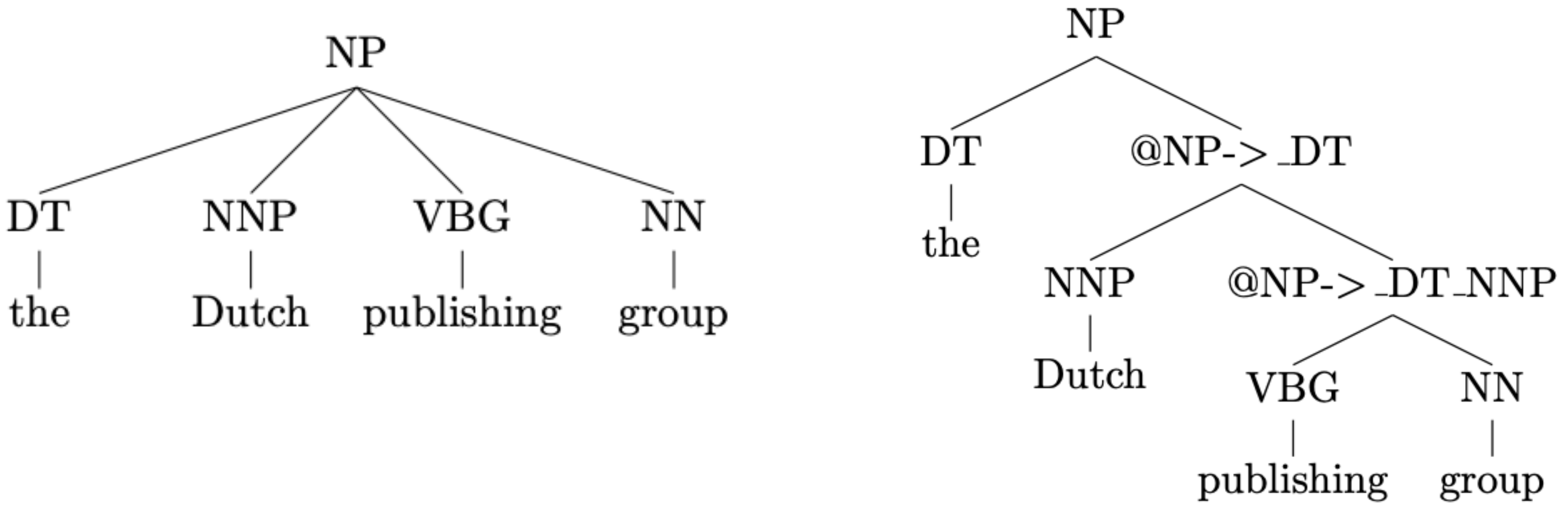
- **Chomsky Normal Form (CNF):** all the rules take one of the two following forms:

  - $X \rightarrow Y_1 Y_2$ where $X \in N, Y_1 \in N, Y_2 \in N$
  - $X \rightarrow Y$ where $X \in N, Y \in \Sigma$

- It is possible to convert any PCFG into an equivalent grammar in CNF!
  - However, the trees will look different; It is possible to do "reverse transformation"

Converting PCFGs into a CNF grammar

- n-ary rules ($n > 2$): NP $\rightarrow$ DT NNP VBG NN



- Unary rules: VP $\rightarrow$ Vi, Vi $\rightarrow$ sleeps
  - Eliminate all the unary rules recursively by adding VP $\rightarrow$ sleeps

The CKY algorithm

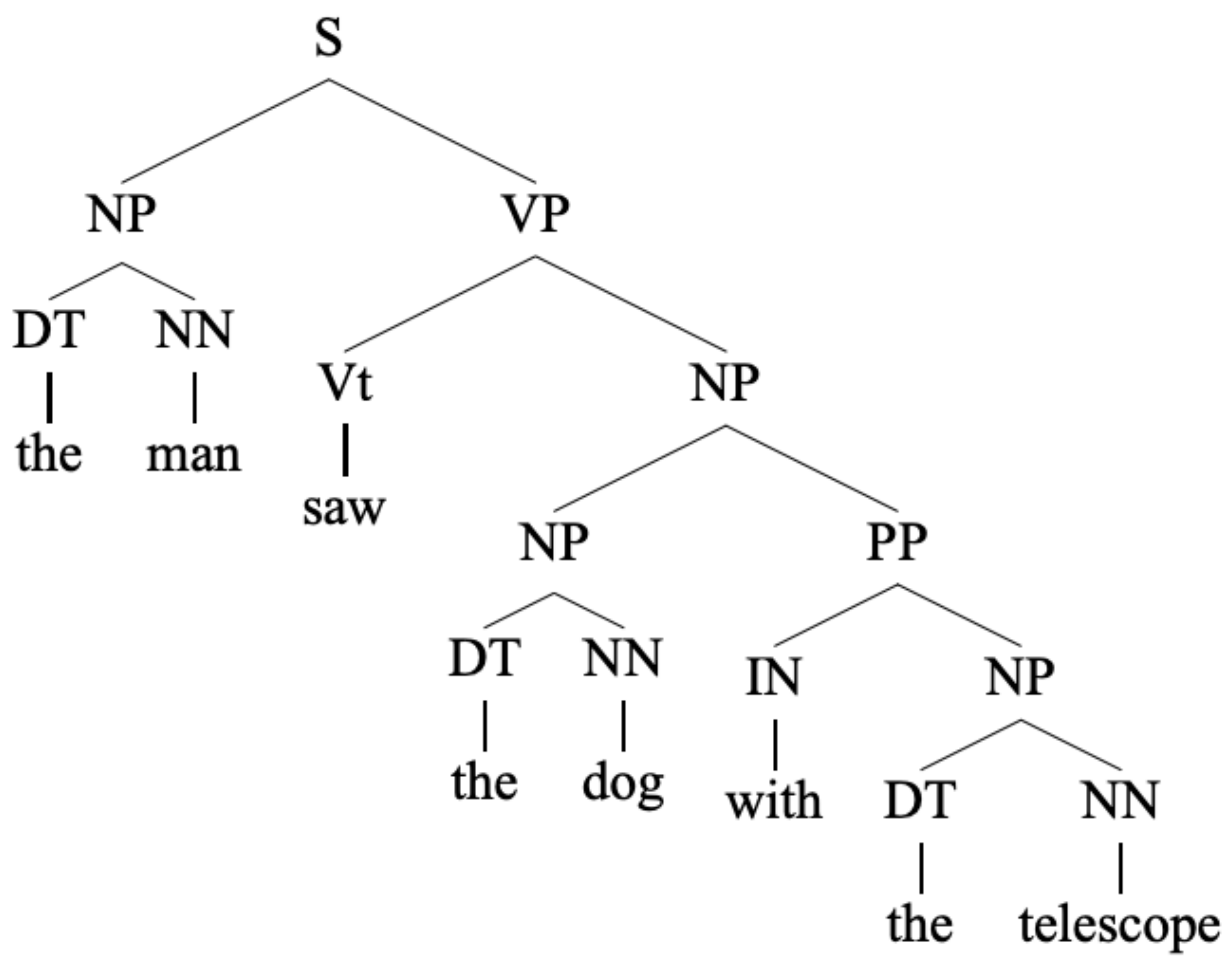
- Dynamic programming

- Given a sentence $x_1, x_2, \ldots, x_n$, denote $\pi(i, j, X)$ as the highest score for any parse tree that dominates words $x_i, \ldots, x_j$ and has non-terminal $X \in N$ as its root.

- Output: $\pi(1, n, S)$

- Initially, for $i = 1, 2, \ldots, n$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

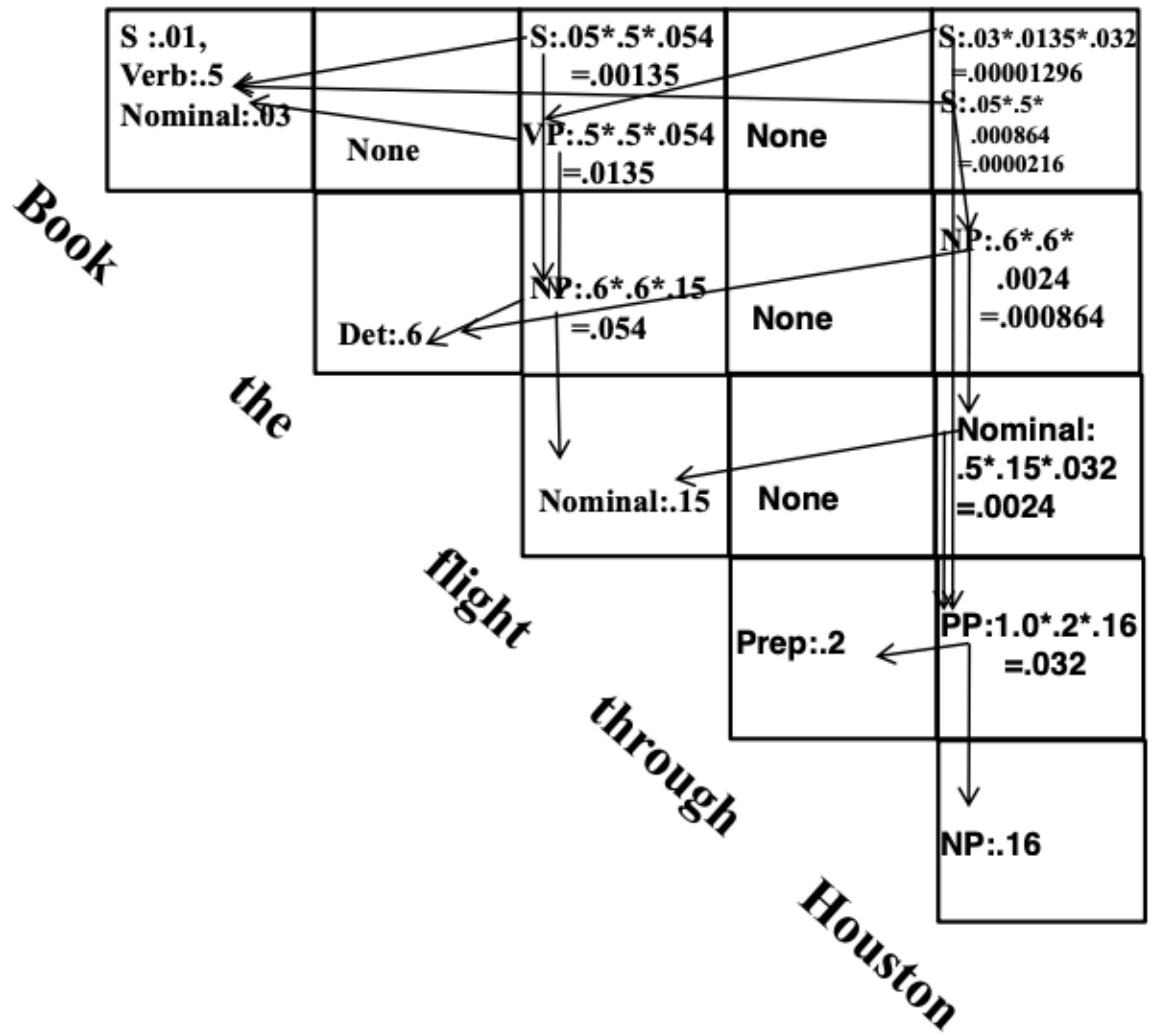


The CKY algorithm

- For all (i, j) such that $1 \leq i < j \leq n$ for all $X \in N$,

$$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, i \leq k < j} q(X \rightarrow YZ) \times \pi(i, k, Y) \times \pi(k+1, j, Z)$$

Also stores backpointers which allow us to recover the parse tree



The CKY algorithm

Input: a sentence $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.
Initialization:
For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate

$$\pi(i, j, X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z) \right)$$

and

$$bp(i, j, X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z) \right)$$

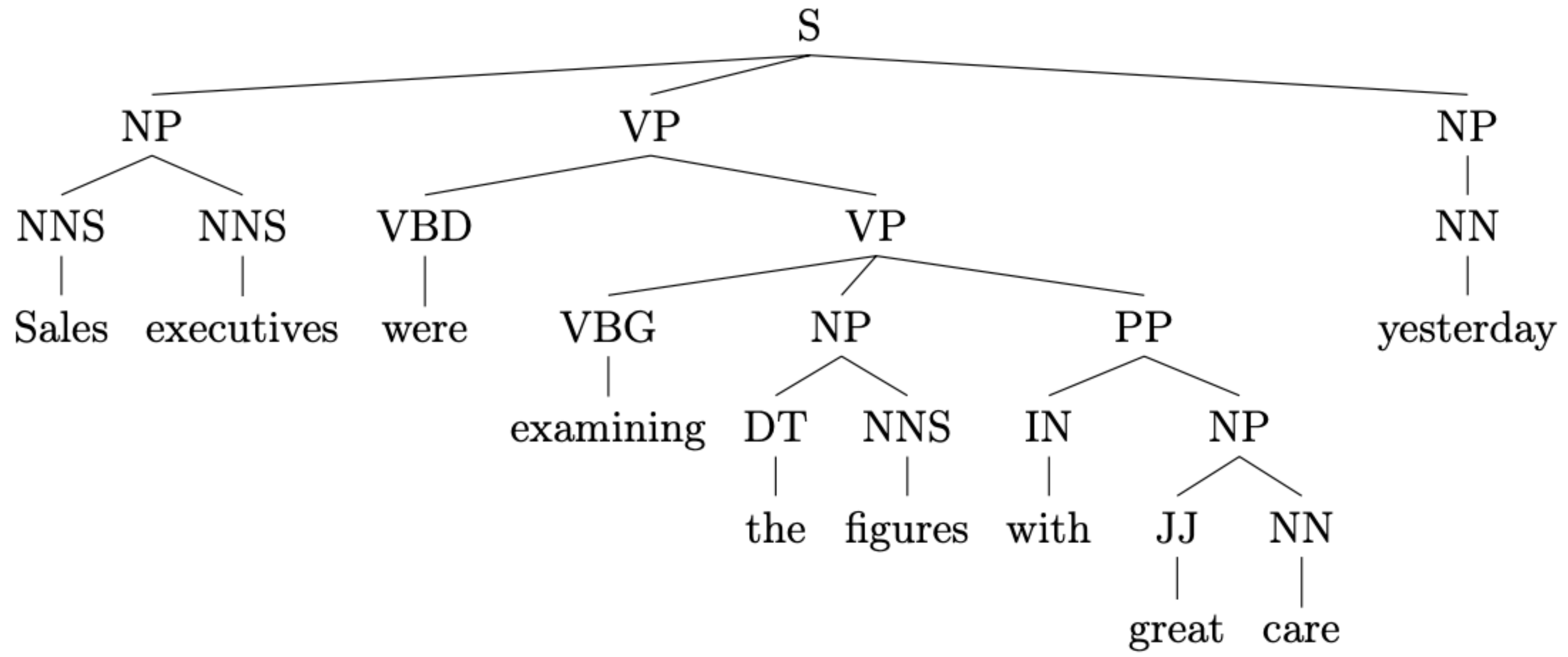
Output: Return $\pi(1, n, S) = \max_{t \in \mathcal{T}(s)} p(t)$, and backpointers bp which allow recovery of $\arg\max_{t \in \mathcal{T}(s)} p(t)$.

Q: Running time?

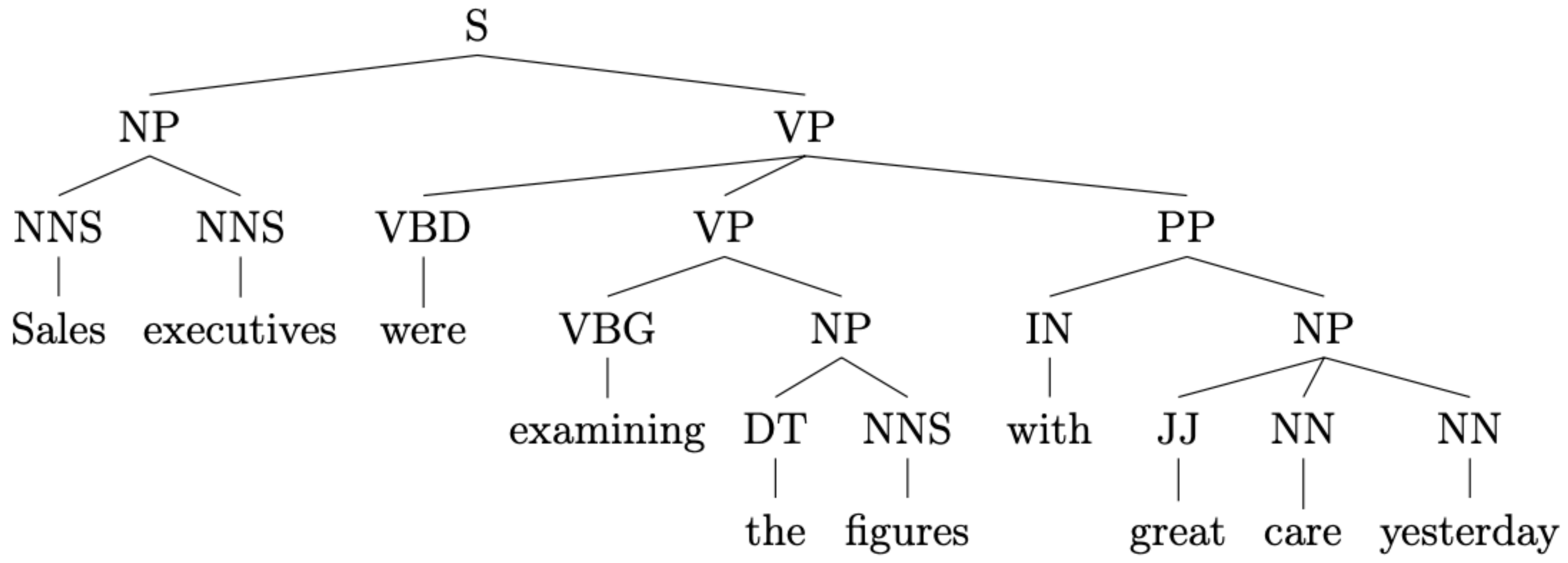
$O(n^3 |R|)$

Evaluating constituency parsing

Gold: **(1, 10, S), (1, 2, NP)**, (3, 9, VP), (4, 9, VP), **(5, 6, NP)**, (7, 9, PP), (8, 9, NP), (10, 10, NP)



Predicted: **(1, 10, S)**, **(1, 2, NP)**, (3, 10, VP), (4, 6, VP), **(5, 6, NP)**, (7, 10, PP), (8, 10, NP)



Evaluating constituency parsing

- Recall: (# correct constituents in candidate) / (# constituents in gold tree)
- Precision: (# correct constituents in candidate) / (# constituents in candidate)
- Labeled precision/recall require getting the non-terminal label correct
- F1 is the harmonic mean of precision and recall = (2 * precision * recall) / (precision + recall)
- Part-of-speech tagging accuracy is evaluated separately

Zoom poll

Gold: **(1, 10, S), (1, 2, NP)**, (3, 9, VP), (4, 9, VP), **(5, 6, NP)**, (7, 9, PP), (8, 9, NP), (10, 10, NP)

Predicted: **(1, 10, S), (1, 2, NP)**, (3, 10, VP), (4, 6, VP), **(5, 6, NP)**, (7, 10, PP), (8, 10, NP)

What are the **labeled** precision (P) / recall (R) in the above example?

(a) P = 3/8, R = 3/7

(b) P = 3/7, R = 3/8

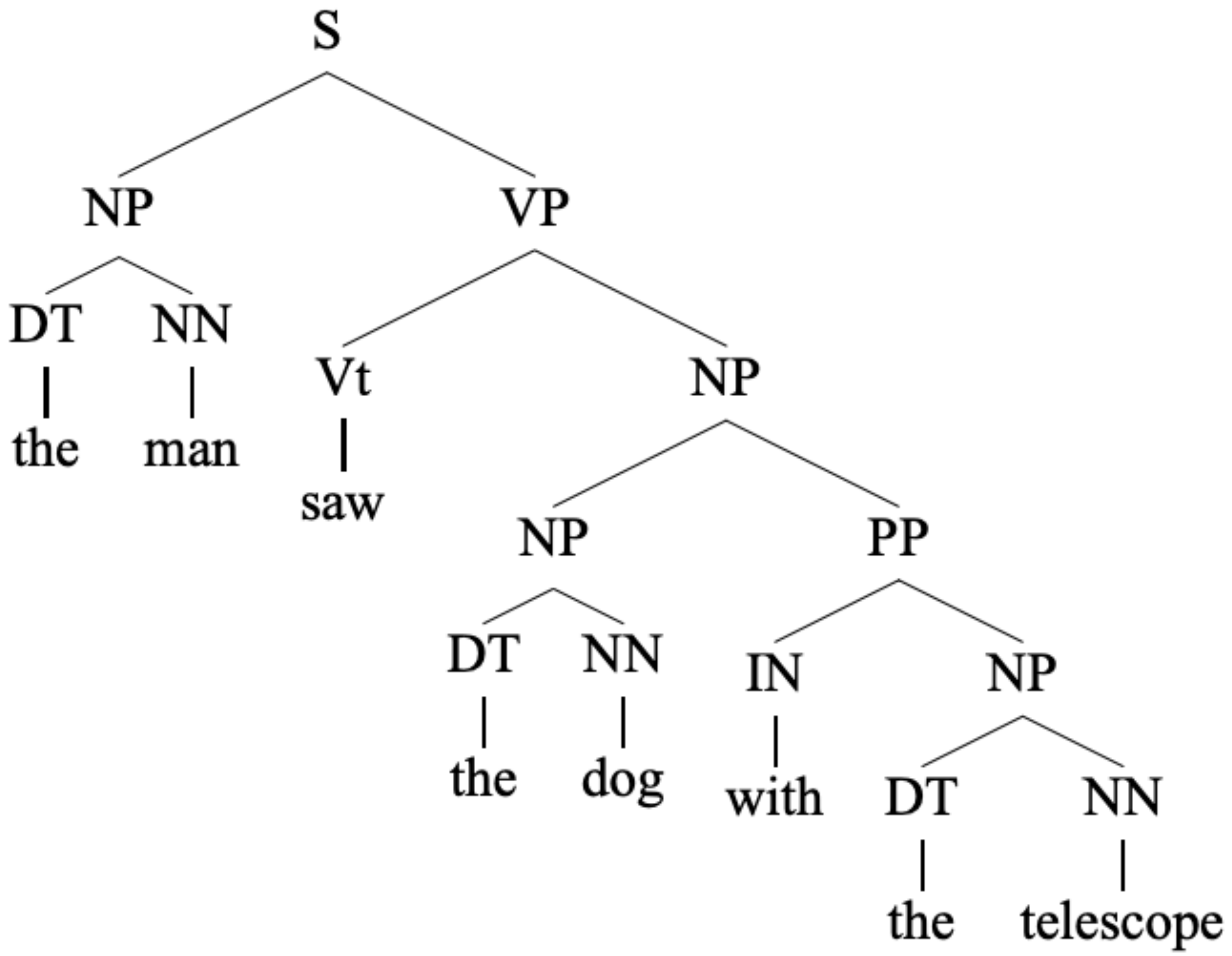
(c) P = 1/2, R = 1/2

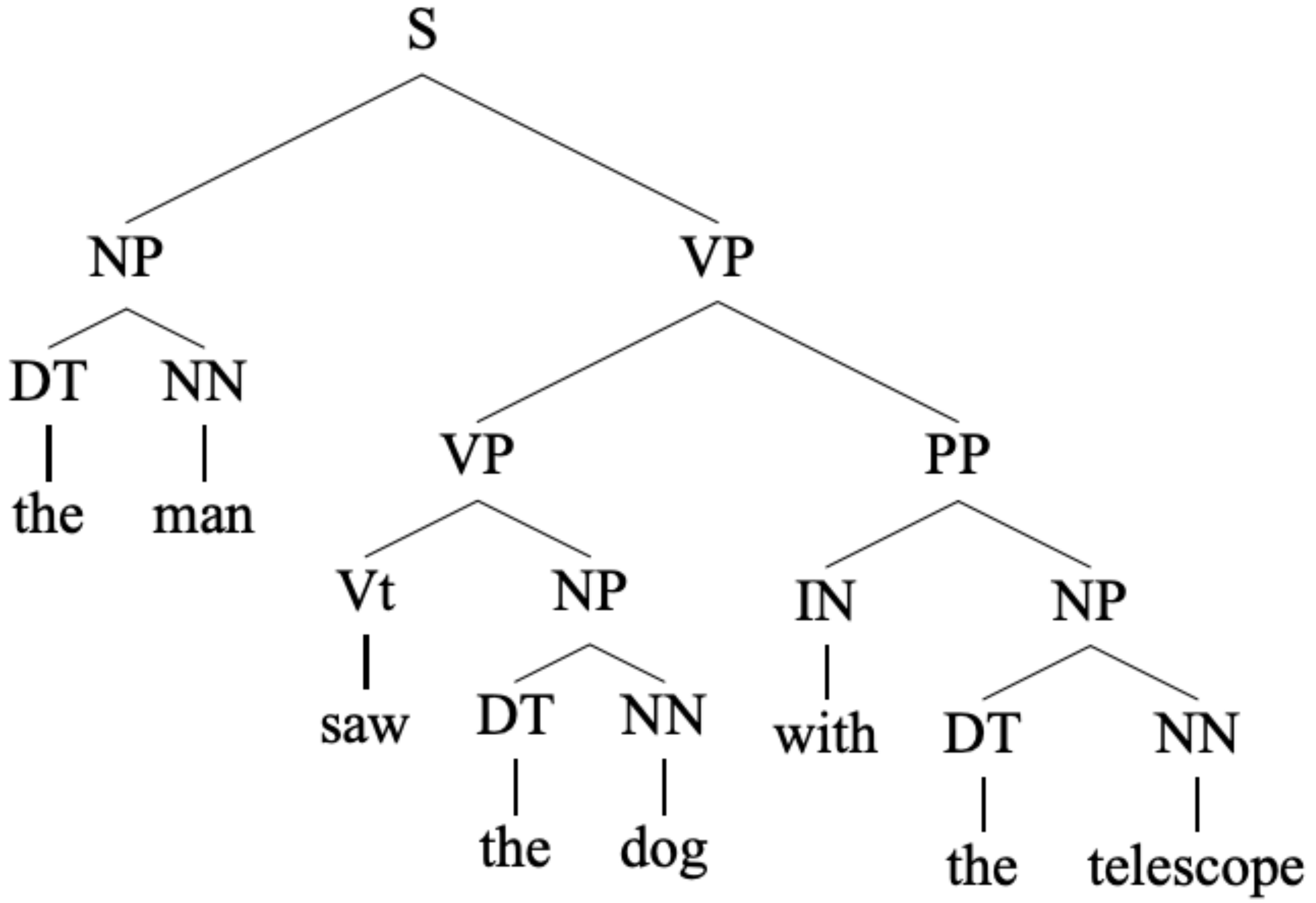
(d) P = 1, R = 1

The answer is (b). F1 = 40%, tagging accuracy = 100%

Weaknesses of PCFGs

Lack of sensitivity to lexical information (words)





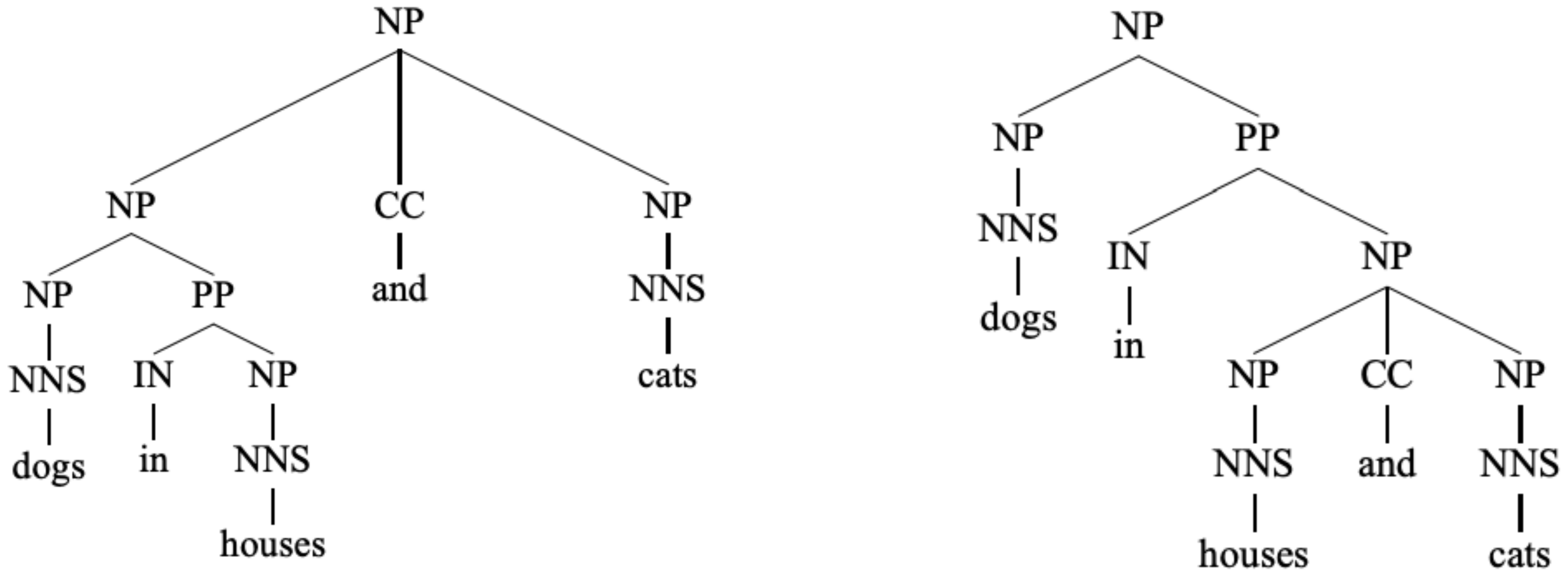
The only difference between these two parses:

$q(\text{VP} \rightarrow \text{VP PP})$ vs $q(\text{NP} \rightarrow \text{NP PP})$

… without looking at the words!

Weaknesses of PCFGs

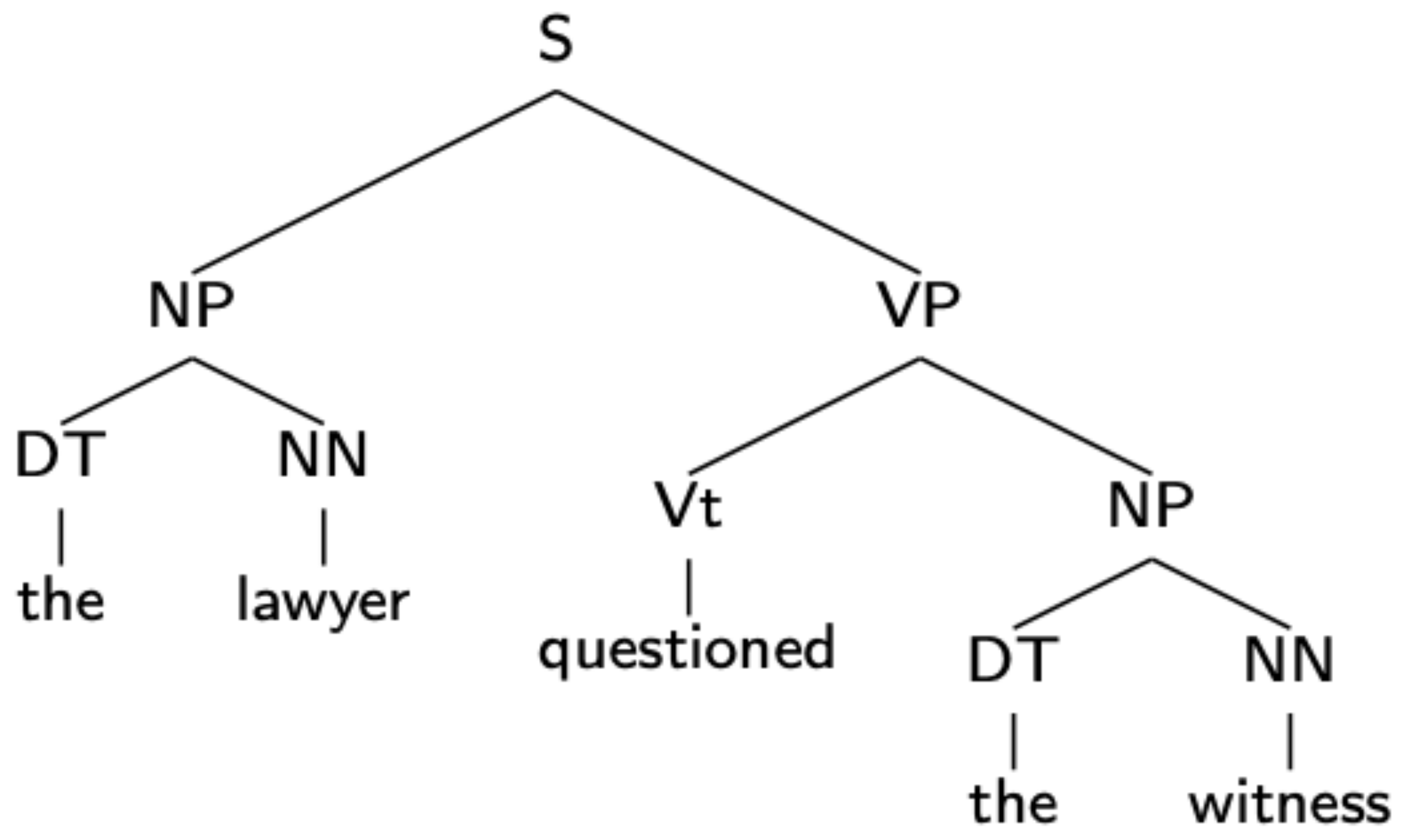
Lack of sensitivity to lexical information (words)

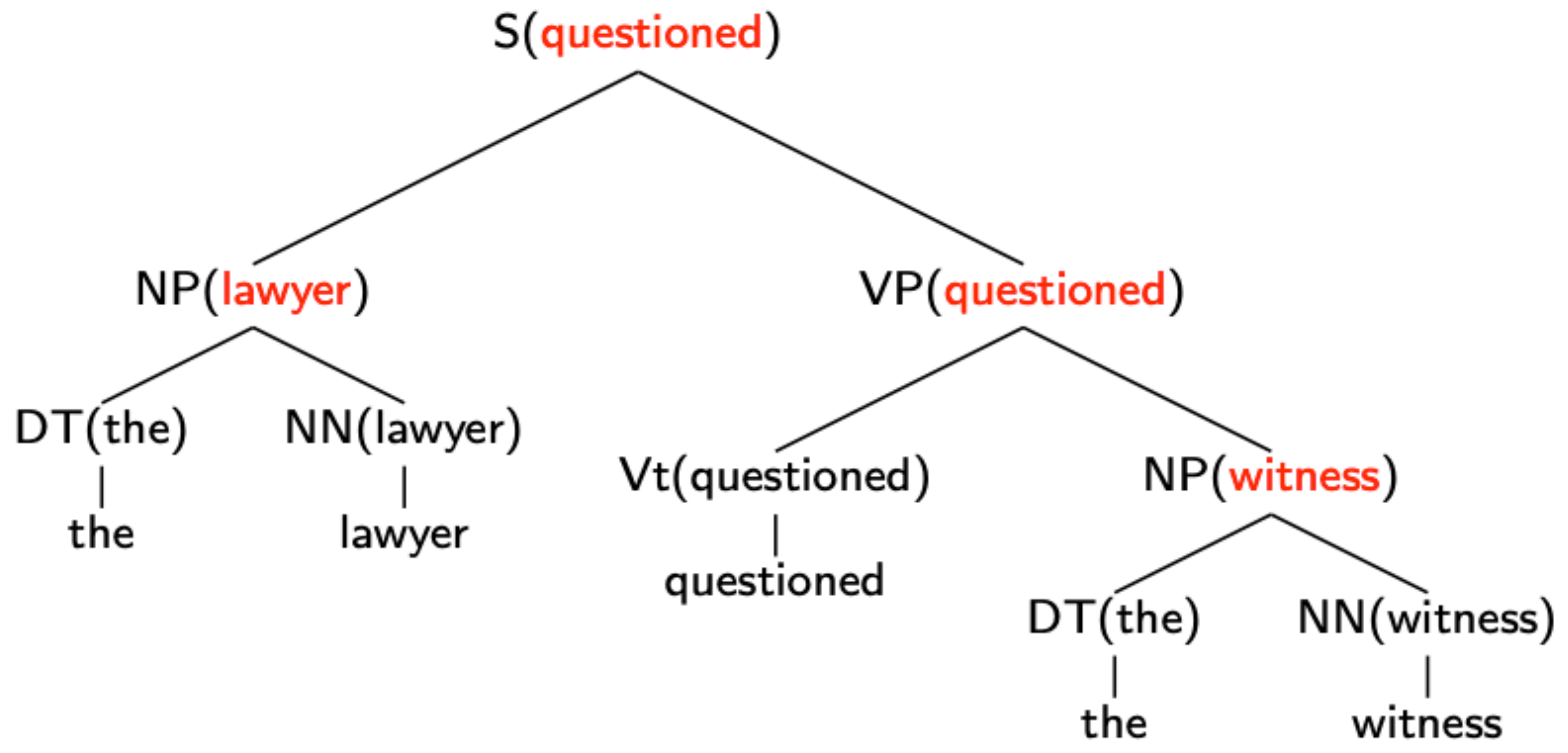


Exactly the same set of context-free rules!

Lexicalized PCFGs

- Key idea: add **headwords** to trees





- Each context-free rule has one special child that is the head of the rule (a core idea in syntax)

$$
\begin{array}{llllll}
\text{S} & \Rightarrow & \text{NP} & \text{VP} & & \text{(VP is the head)} \\
\text{VP} & \Rightarrow & \text{Vt} & \text{NP} & & \text{(Vt is the head)} \\
\text{NP} & \Rightarrow & \text{DT} & \text{NN} & \text{NN} & \text{(NN is the head)}
\end{array}
$$

Lexicalized PCFGs

The heads are decided by rules:

If the rule contains NN, NNS, or NNP:
Choose the rightmost NN, NNS, or NNP

Else If the rule contains an NP: Choose the leftmost NP

Else If the rule contains a JJ: Choose the rightmost JJ

Else If the rule contains a CD: Choose the rightmost CD

Else Choose the rightmost child

If the rule contains Vi or Vt: Choose the leftmost Vi or Vt

Else If the rule contains a VP: Choose the leftmost VP

Else Choose the leftmost child

Lexicalized PCFGs

S(saw)	$\rightarrow_2$	NP(man)	VP(saw)
VP(saw)	$\rightarrow_1$	Vt(saw)	NP(dog)
NP(man)	$\rightarrow_2$	DT(the)	NN(man)
NP(dog)	$\rightarrow_2$	DT(the)	NN(dog)
Vt(saw)	$\rightarrow$	saw	
DT(the)	$\rightarrow$	the	
NN(man)	$\rightarrow$	man	
NN(dog)	$\rightarrow$	dog	

- Further reading: *Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing.*
- Results for a PCFG: 70.6% recall, 74.8% precision
- Results for a lexicalized PCFG: 88.1% recall, 88.3% precision