

Natural Language Processing with Deep Learning

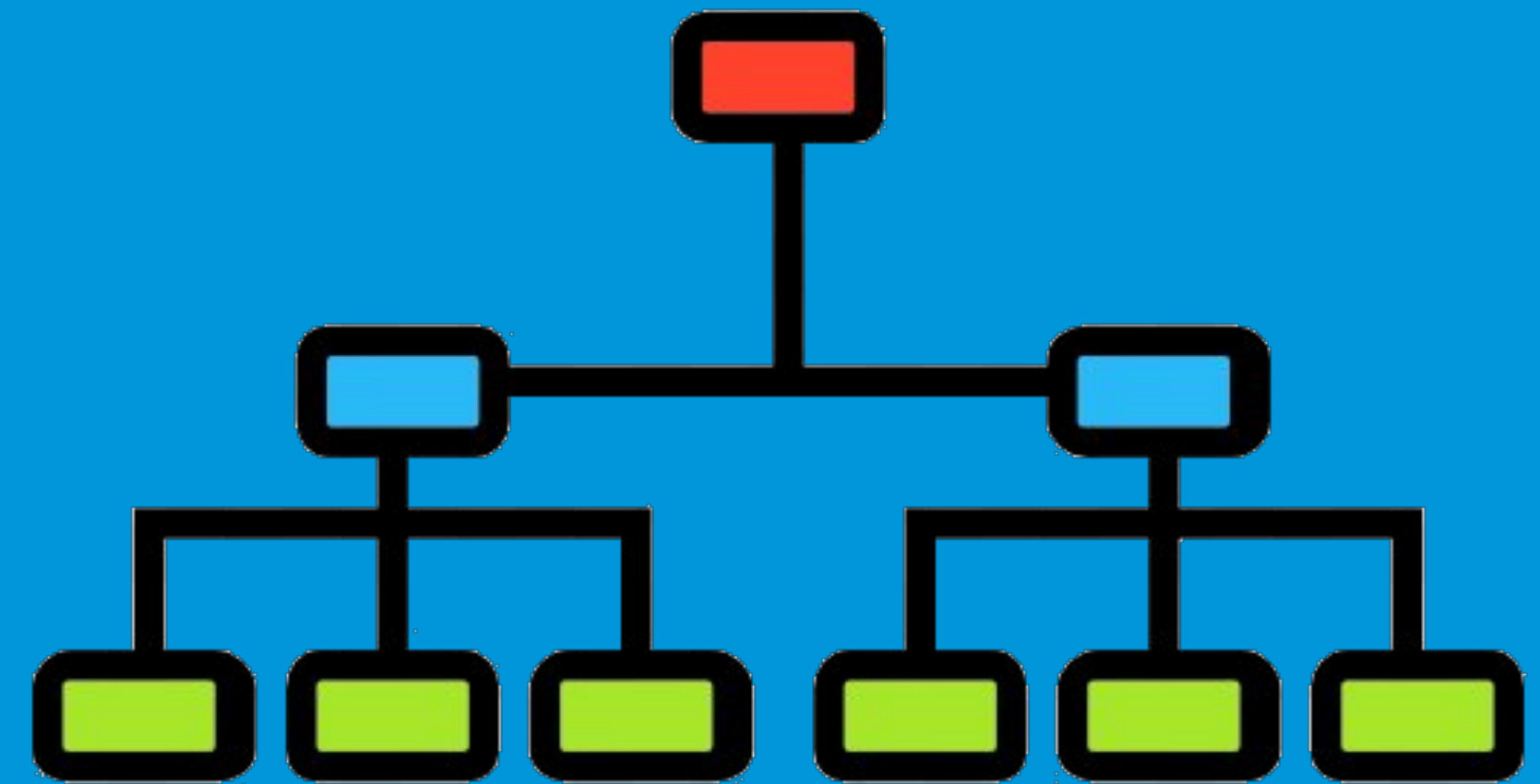
IFT6289, Winter 2022

Lecture 15: Constituency Parsing
Bang Liu

2 Lecture outline

1. Principle of Compositionality
2. Context Free Grammar
3. CKY Parsing
4. Parsing by TreeRNNs
5. Parsing by Pre-trained LM

Compositionality



Compositionality

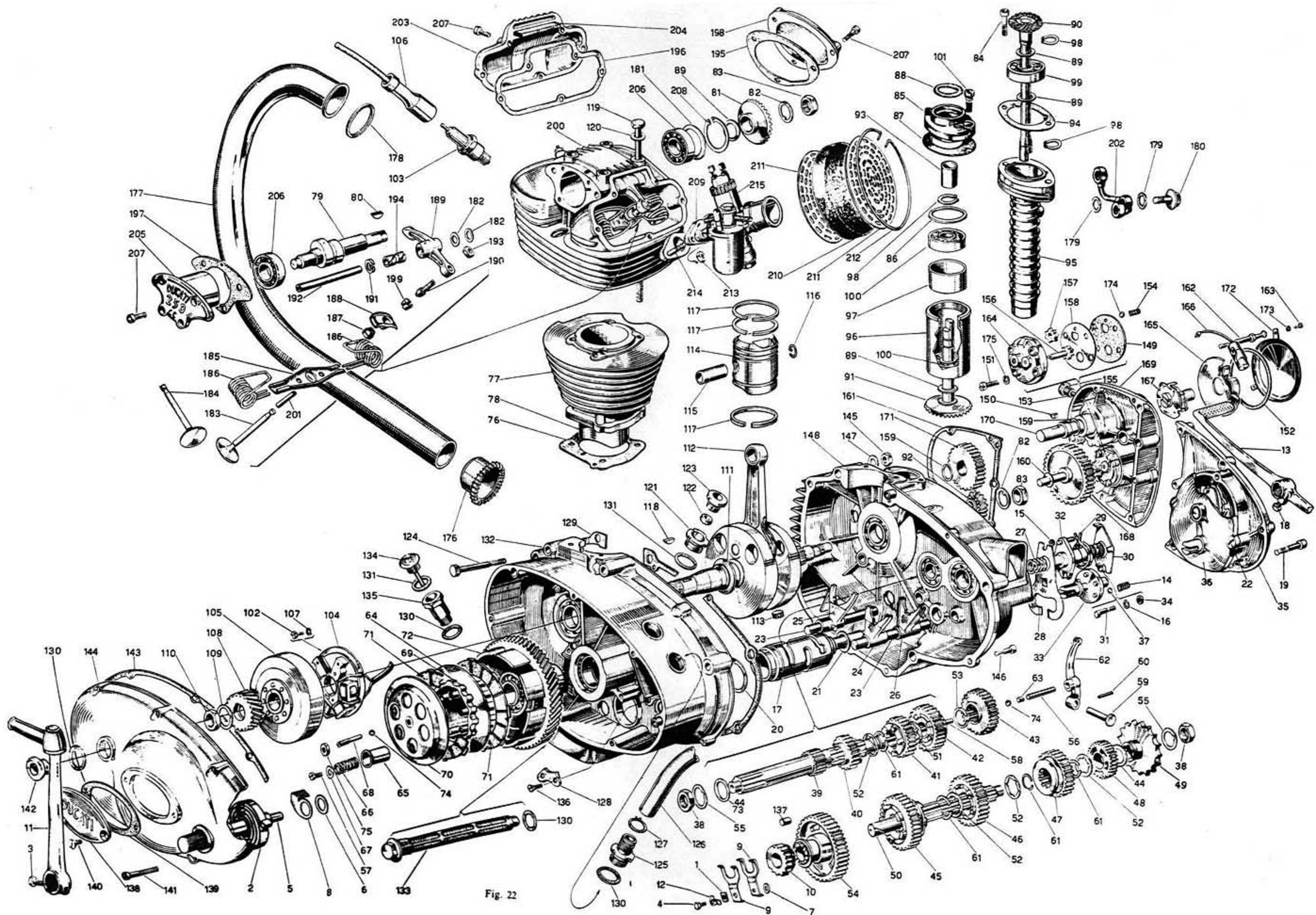
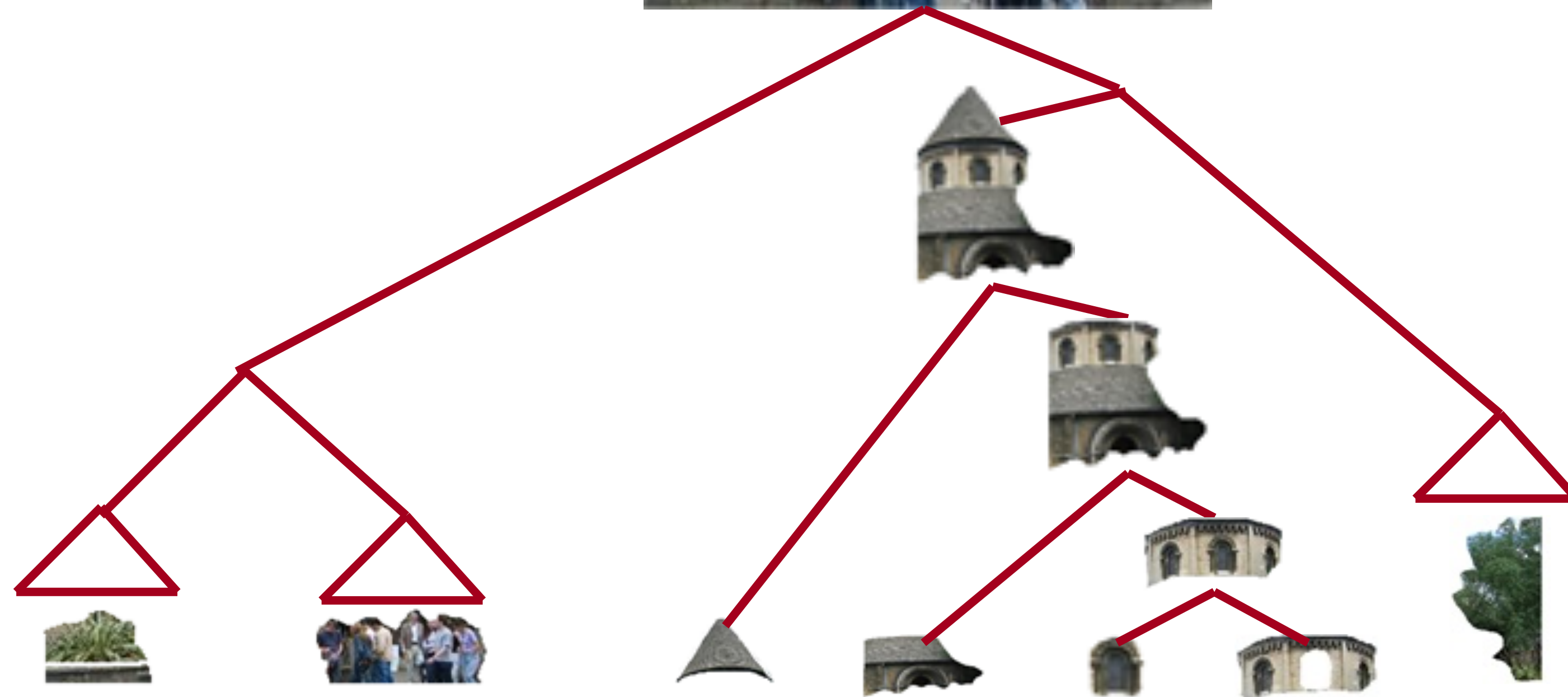


Fig. 22



6 Semantic Interpretation of Language

- How can we interpret the meaning of larger phrases?

The **snowboarder** is leaping over a mogul

A **person on a snowboard** jumps into the air

- People interpret the meaning of larger text units – entities, descriptive terms, facts, arguments, stories – by **semantic composition** of smaller elements

Language understanding -
& Artificial Intelligence - requires
being able to understand bigger
things from knowing about smaller
parts

8 The Nature of Language

- Natural language is **flexible, compositional, hierarchical**

Sentence A:

The brown mouse is being chased by the blue cat.

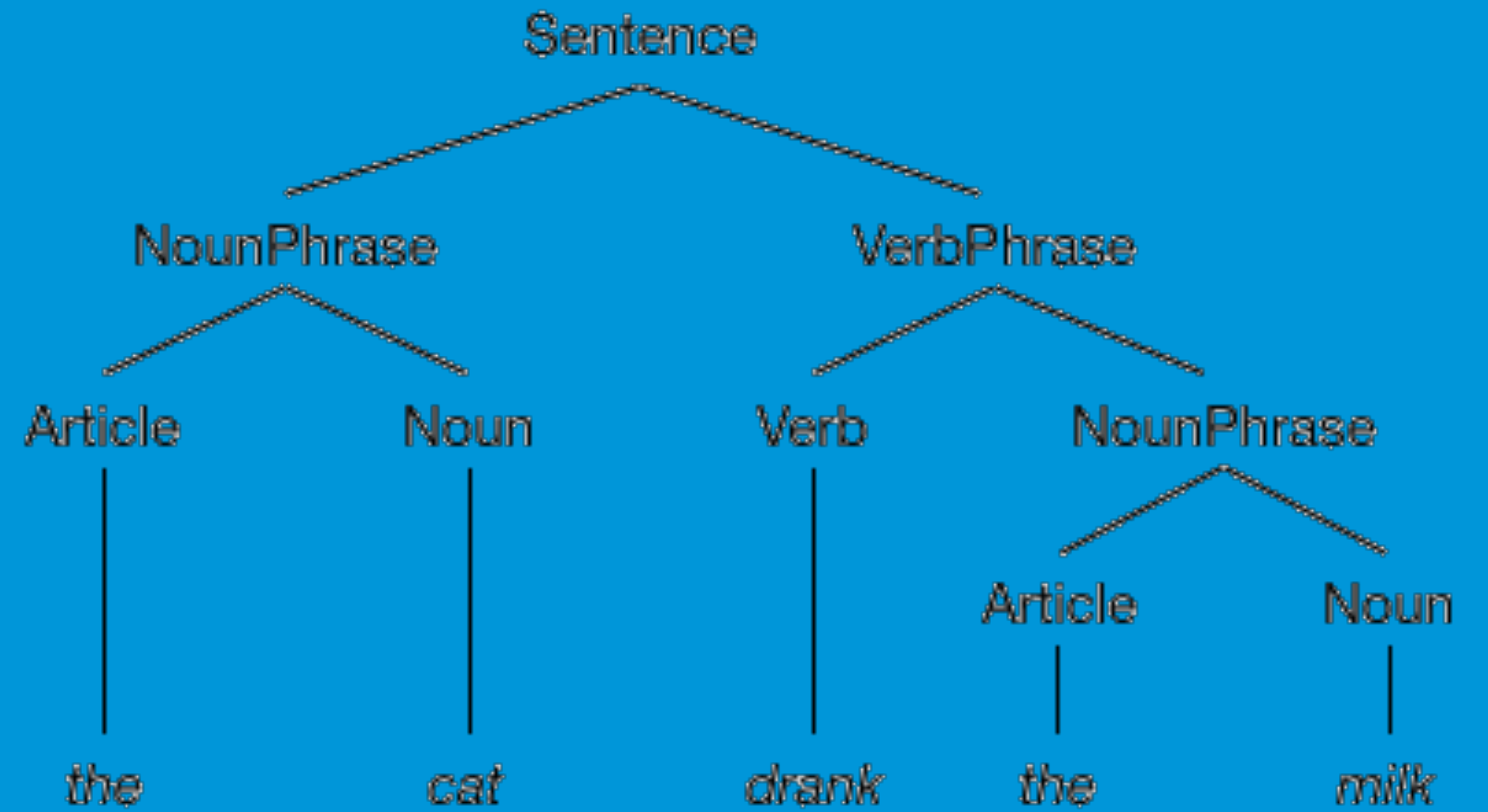
Sentence B:

The blue cat is chasing the brown mouse.

The blue cat is chasing the brown mouse.

The blue cat is chasing the brown mouse.

Context Free Grammar



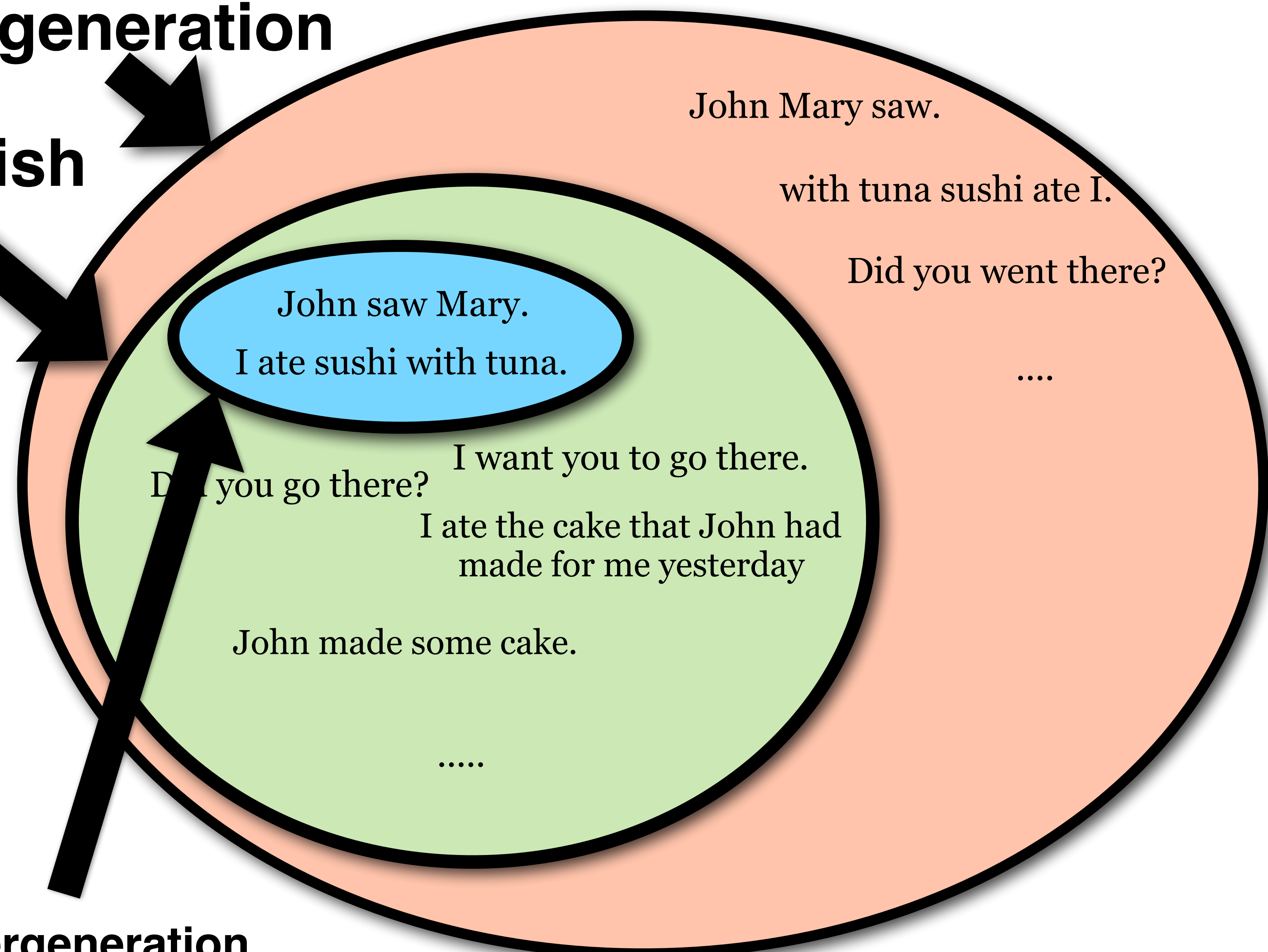
Can we define a program that generates all English sentences?

The number of sentences is infinite.
But we need our program to be finite.

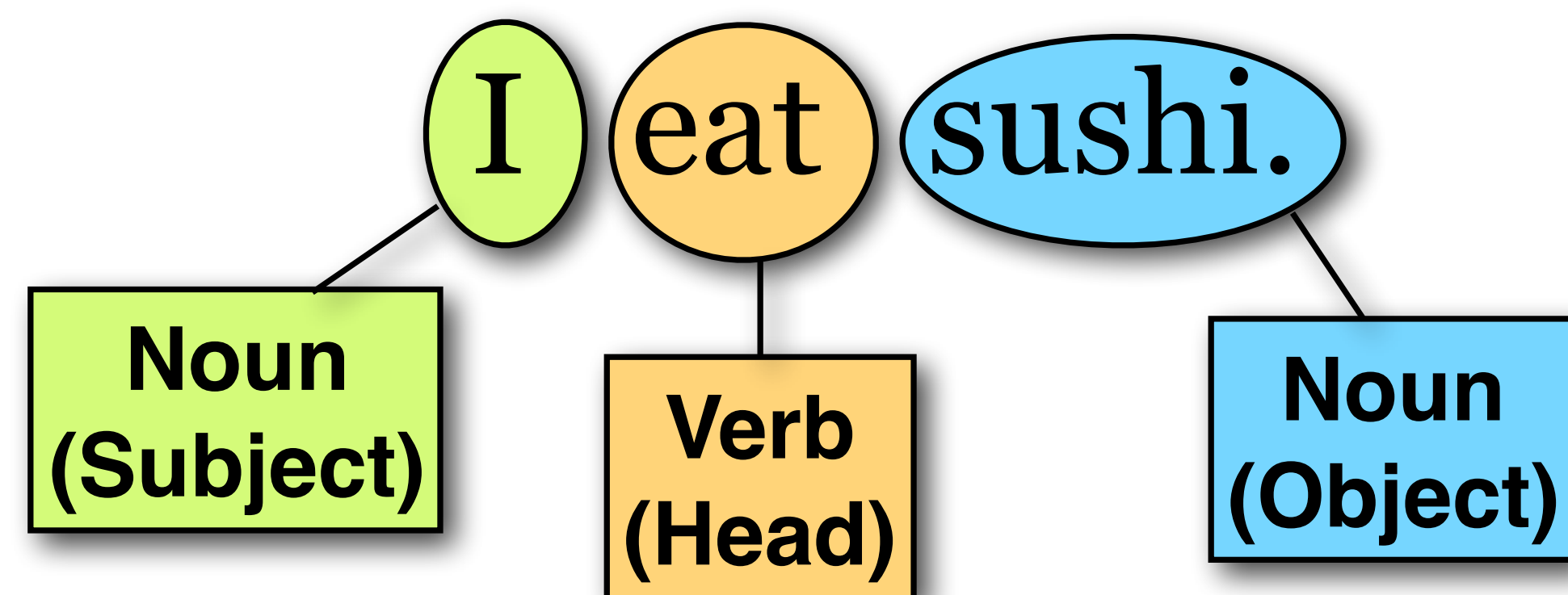
Overgeneration

English

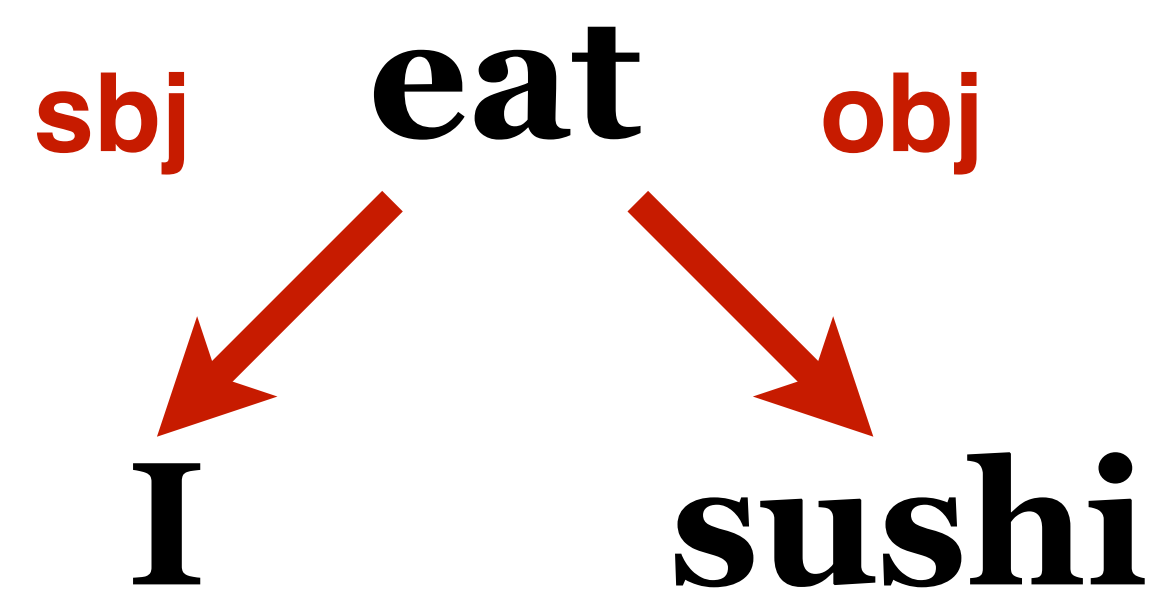
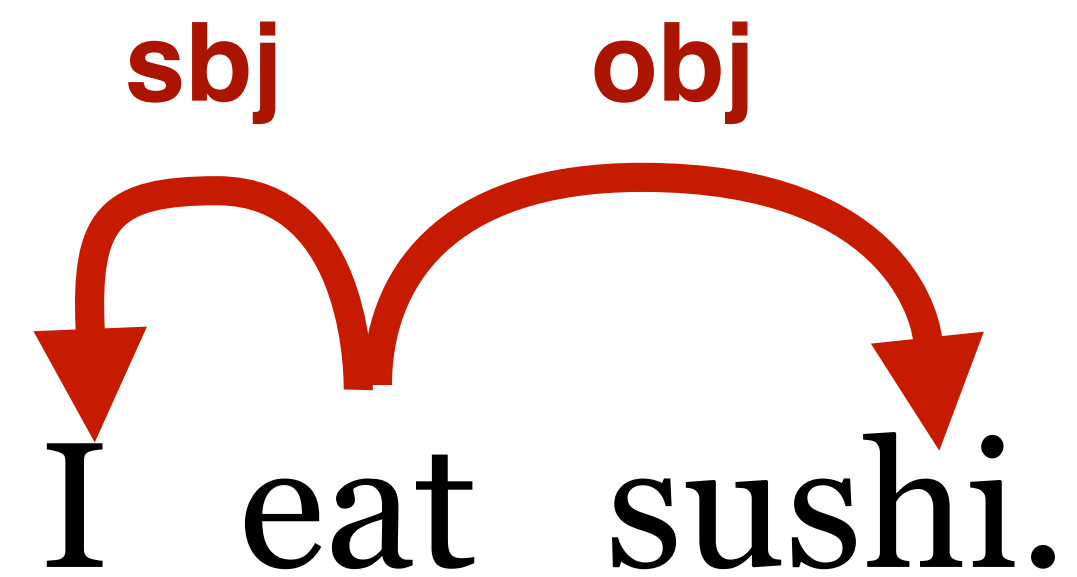
Undergeneration



12 Basic sentence structure



13 This is a dependency graph



14 Language is recursive

the ball
*the **big** ball*
*the **big, red** ball*
*the **big, red, heavy** ball*
....

Adjectives can **modify** nouns.

The **number of modifiers (aka adjuncts)**

a word can have is (in theory) **unlimited**.

15 Recursion can be more complex

the ball

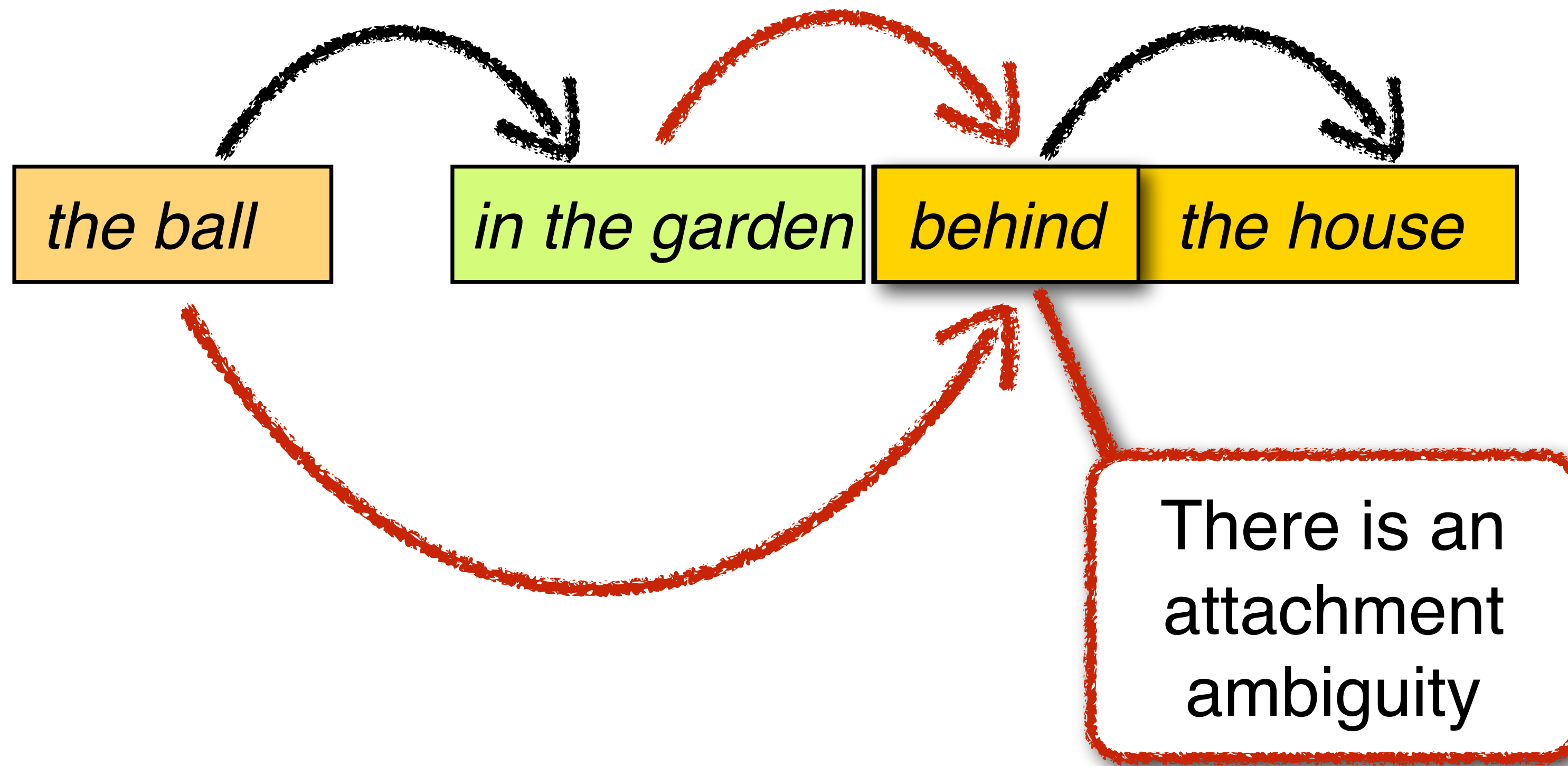
the ball in the garden

the ball in the garden behind the house

the ball in the garden behind the house next to the school

...

What does this mean?

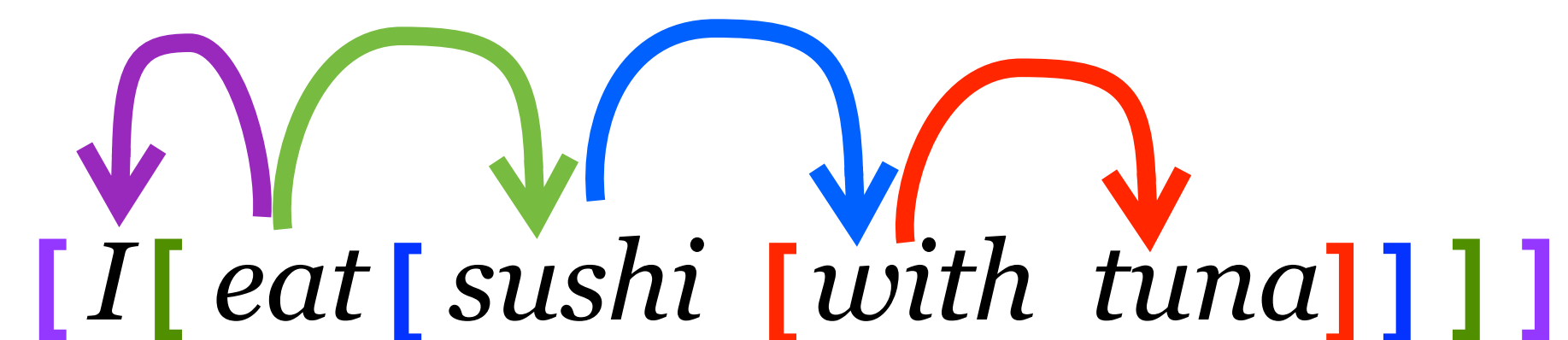


17 What is the structure of a sentence?

Sentence structure is **hierarchical**:

A sentence consists of **words** (I, eat, sushi, with, tuna)
..which form phrases or **constituents**: “sushi with tuna”

Sentence structure defines **dependencies**
between words or phrases:



18 Context-free grammars (CFGs) capture recursion

Language has complex constituents
(“the garden behind the house”)

Syntactically, these constituents behave
just like simple ones.

(“behind the house” can always be omitted)

CFGs define nonterminal categories
to capture equivalent constituents.

Context-free grammars

A CFG is a 4-tuple $\langle \mathbf{N}, \Sigma, \mathbf{R}, S \rangle$ consisting of:

A set of nonterminals \mathbf{N}

(e.g. $\mathbf{N} = \{S, NP, VP, PP, Noun, Verb, \dots\}$)

A set of terminals Σ

(e.g. $\Sigma = \{I, you, he, eat, drink, sushi, ball, \}$)

A set of rules \mathbf{R}

$\mathbf{R} \subseteq \{A \rightarrow \beta \text{ with left-hand-side (LHS) } A \in \mathbf{N}$

and right-hand-side (RHS) $\beta \in (\mathbf{N} \cup \Sigma)^* \}$

A start symbol $S \in \mathbf{N}$

An example

DT \rightarrow {the, a}

N \rightarrow {ball, garden, house, sushi }

P \rightarrow {in, behind, with}

NP \rightarrow DT N

NP \rightarrow NP PP

PP \rightarrow P NP

N: noun

P: preposition

NP: “noun phrase”

PP: “prepositional phrase”

An example

lexicon

DT \rightarrow {the, a} **terminal symbols**
N \rightarrow {ball, garden, house, sushi }
P \rightarrow {in, behind, with}

**other
rules**

NP \rightarrow DT N
NP \rightarrow NP PP
PP \rightarrow P NP

**definitions
of
non-terminals**

N: noun
P: preposition
NP: “noun phrase”
PP: “prepositional phrase”
DT: Determiner

22 Context-free or not?

$S \rightarrow BBB$

$B \rightarrow \emptyset$

$B \rightarrow 1$

?

$S \rightarrow AB$

$AB \rightarrow 1$

$A \rightarrow AA$

$B \rightarrow \emptyset$

?

$$S \rightarrow BBB$$
$$B \rightarrow \emptyset$$
$$B \rightarrow 1$$

Context-Free

$$S \rightarrow AB$$
$$AB \rightarrow 1$$
$$A \rightarrow AA$$
$$B \rightarrow \emptyset$$

Not Context-Free

CFGs define parse trees

$N \rightarrow \{\text{sushi, tuna}\}$

$P \rightarrow \{\text{with}\}$

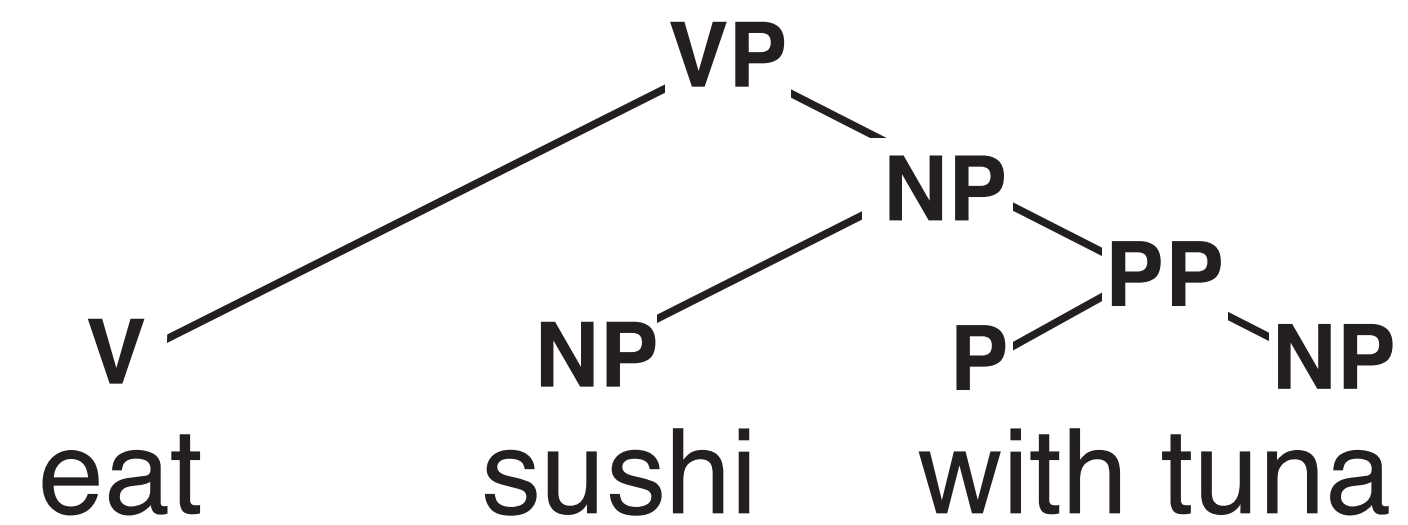
$V \rightarrow \{\text{eat}\}$

$NP \rightarrow N$

$NP \rightarrow NP PP$

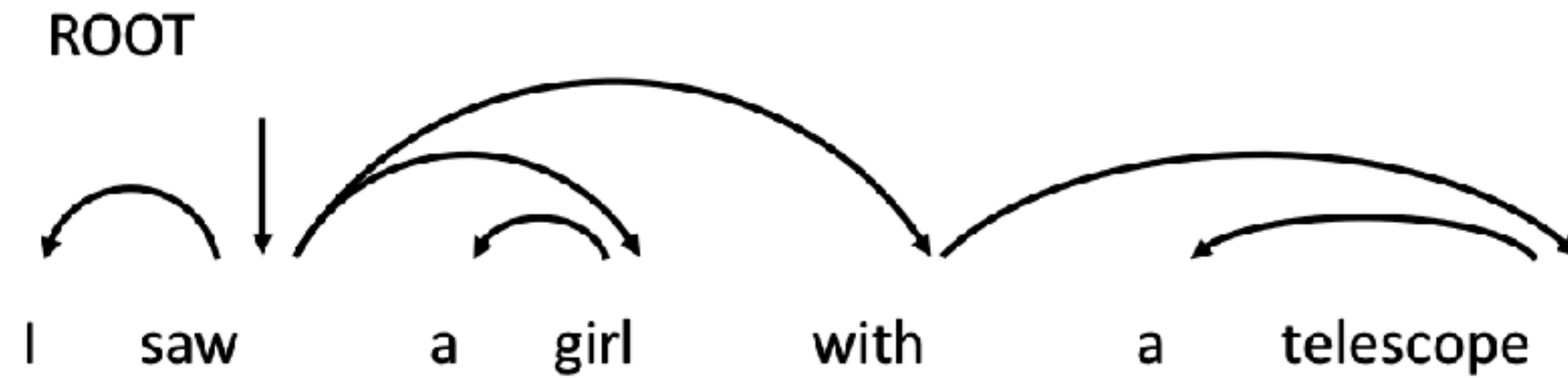
$PP \rightarrow P NP$

$VP \rightarrow V NP$

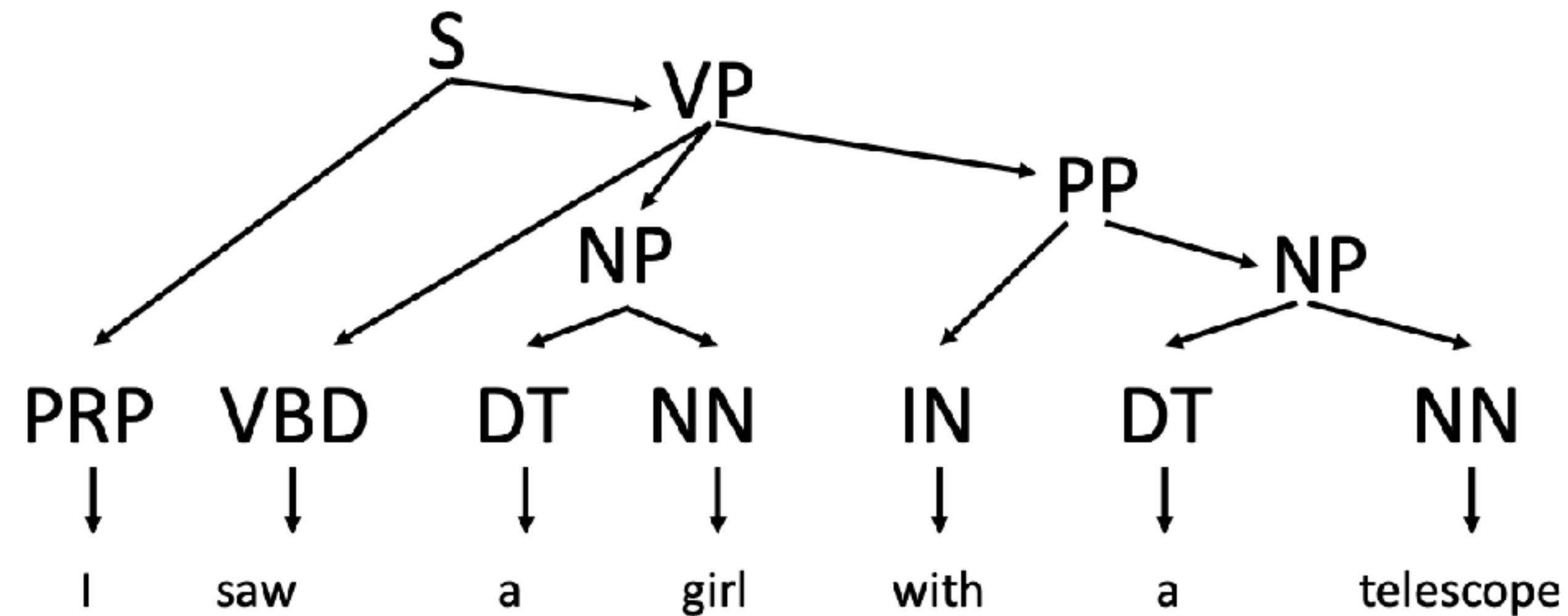


25 Two Most Common of Linguistic Tree Structures

Dependency Trees focus on relations between words



Phrase Structure models the structure of a sentence



Dependency grammar

DGs describe the structure of sentences as a directed acyclic graph.

The **nodes** of the graph are the **words**

The **edges** of the graph are the **dependencies**.

Typically, the graph is assumed to be a **tree**.

Note: the relationship between DG and CFGs:

If a CFG phrase structure tree is translated into DG, the resulting dependency graph has no crossing edges.

27 Constituents: heads and dependents

There are different kinds of constituents:

Noun phrases: the man, a girl with glasses, Illinois

Prepositional phrases: with glasses, in the garden

Verb phrases: eat sushi, sleep, sleep soundly

Every phrase has a **head**:

Noun phrases: the man, a girl with glasses, Illinois

Prepositional phrases: with glasses, in the garden

Verb phrases: eat sushi, sleep, sleep soundly

The other parts are its **dependents**.

Dependents are either **arguments** or **adjuncts**

Is string α a constituent?

He talks [in class].

Substitution test:

Can α be replaced by a single word?

He talks [there].

Movement test:

Can α be moved around in the sentence?

[In class], he talks.

Answer test:

Can α be the answer to a question?

Where does he talk? - [In class].

Arguments are obligatory

Words subcategorize for specific sets of arguments:

Transitive verbs (sbj + obj): [John] likes [Mary]

All arguments have to be present:

*[John] likes. *likes [Mary].

No argument can be occupied multiple times:

*[John] [Peter] likes [Ann] [Mary].

Words can have multiple subcat frames:

Transitive eat (sbj + obj): [John] eats [sushi].

Intransitive eat (sbj): [John] eats.

Adjuncts are optional

Adverbs, PPs and adjectives can be adjuncts:

Adverbs: John runs [fast].

a [very] heavy book.

PPs: John runs [in the gym].

the book [on the table]

Adjectives: a [heavy] book

There can be an arbitrary number of adjuncts:

John saw Mary.

John saw Mary [yesterday].

John saw Mary [yesterday] [in town]

John saw Mary [yesterday] [in town] [during lunch]

[Perhaps] John saw Mary [yesterday] [in town] [during lunch]

A context-free grammar
for a fragment of
English

Noun phrases (NPs)

Simple NPs:

[He] sleeps. (pronoun)

[John] sleeps. (proper name)

[A student] sleeps. (determiner + noun)

Complex NPs:

[A tall student] sleeps. (det + adj + noun)

[The student in the back] sleeps. (NP + PP)

[The student who likes MTV] sleeps. (NP + Relative Clause)

The NP fragment

NP → Pronoun

NP → ProperName

NP → Det Noun

Det → {a, the, every}

Pronoun → {he, she,...}

ProperName → {John, Mary,...}

Noun → AdjP Noun

Noun → N

NP → NP PP

NP → NP RelClause

34 Adjective phrases (AdjP) and prepositional phrases (PP)

AdjP \rightarrow Adj

AdjP \rightarrow Adv AdjP

Adj \rightarrow {big, small, red,...}

Adv \rightarrow {very, really,...}

PP \rightarrow P NP

P \rightarrow {with, in, above,...}

The verb phrase (VP)

He [eats].

He [eats sushi].

He [gives John sushi].

He [eats sushi with chopsticks].

VP \rightarrow V

VP \rightarrow V NP

VP \rightarrow V NP PP

VP \rightarrow VP PP

V \rightarrow {eats, sleeps gives,...}

[He eats sushi].

[Sometimes, he eats sushi].

[In Japan, he eats sushi].

$S \rightarrow NP VP$

$S \rightarrow AdvP S$

$S \rightarrow PP S$

He says [he eats sushi].

$VP \rightarrow Vcomp S$

$Vcomp \rightarrow \{\text{says, think, believes}\}$

37 Coordination

[He eats sushi] and [she drinks tea]
[John] and [Mary] eat sushi.
He [eats sushi] and [drinks tea]

$S \rightarrow S \text{ conj } S$

$NP \rightarrow NP \text{ conj } NP$

$VP \rightarrow VP \text{ conj } VP$

He says [he eats sushi].

$VP \rightarrow V_{\text{comp}} S$

$V_{\text{comp}} \rightarrow \{\text{says, think, believes}\}$

Relative clauses

Relative clauses modify a noun phrase:

the girl [that eats sushi]

There are subject and object relative clauses:

subject: 'the girl that eats sushi'

object: 'the sushi that the girl eats'

Yes/No questions

Yes/no questions consist of an auxiliary, a subject and an (untensed) verb phrase:

does she eat sushi?

have you eaten sushi?

YesNoQ → Aux NP VP_{inf}

YesNoQ → Aux NP VP_{pastPart}

40 Wh-questions

Subject wh-questions consist of an wh-word, an auxiliary and an (untensed) verb phrase:

Who has eaten the sushi?

Object wh-questions consist of an wh-word, an auxiliary, an NP and an (untensed) verb phrase:

What does Mary eat?

41 More Details

Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2020. All rights reserved. Draft of December 30, 2020.

CHAPTER

12

Constituency Grammars

<https://web.stanford.edu/~jurafsky/slp3/12.pdf>

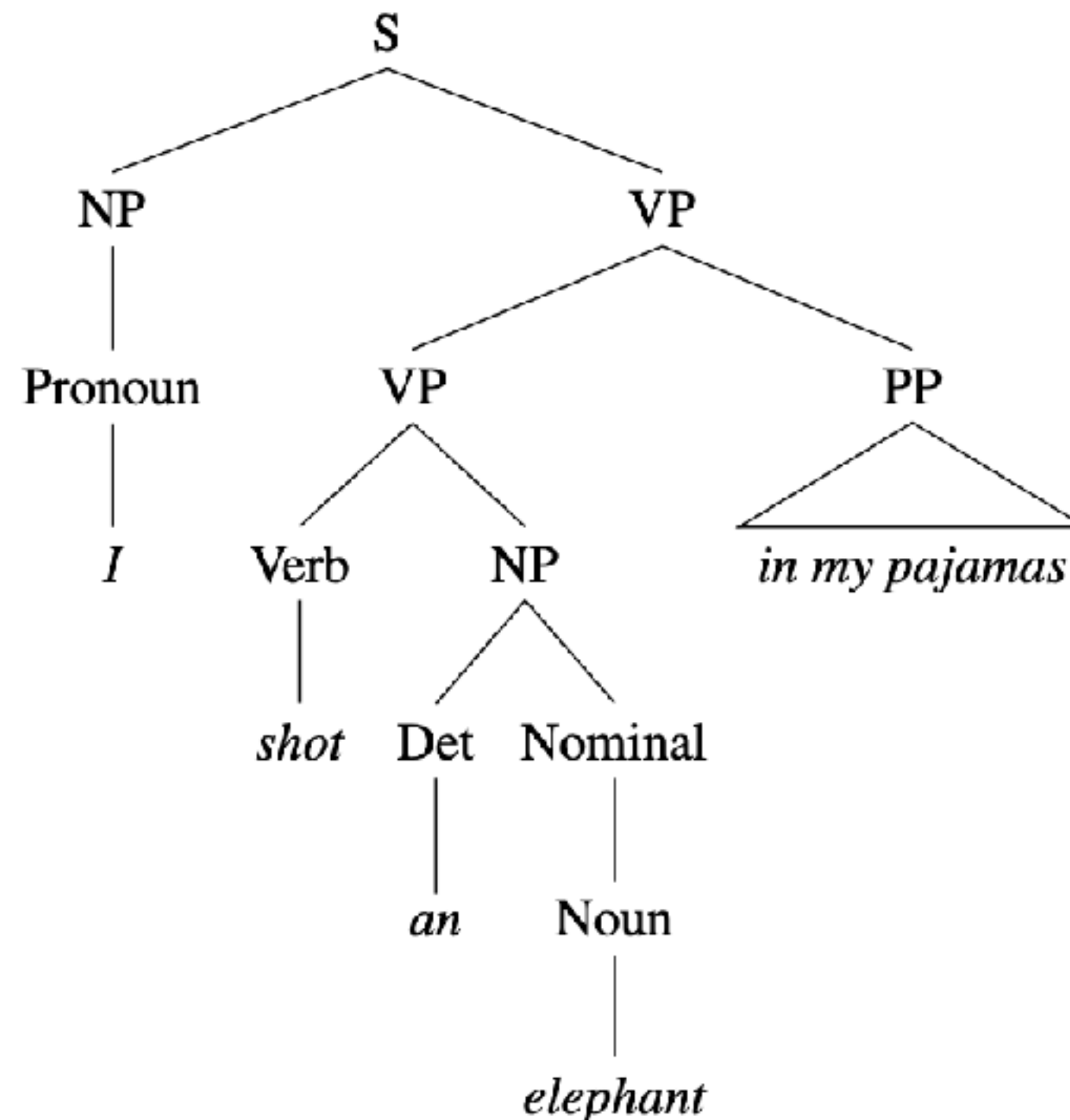
CKY Parsing Algorithm

	1	2	3	4
0	Det	NP		S
1		N Nom NP		S
2			TV	VP
3				N IV Nom VP NP
	the	frogs	ate	fish

Constituency Parsing

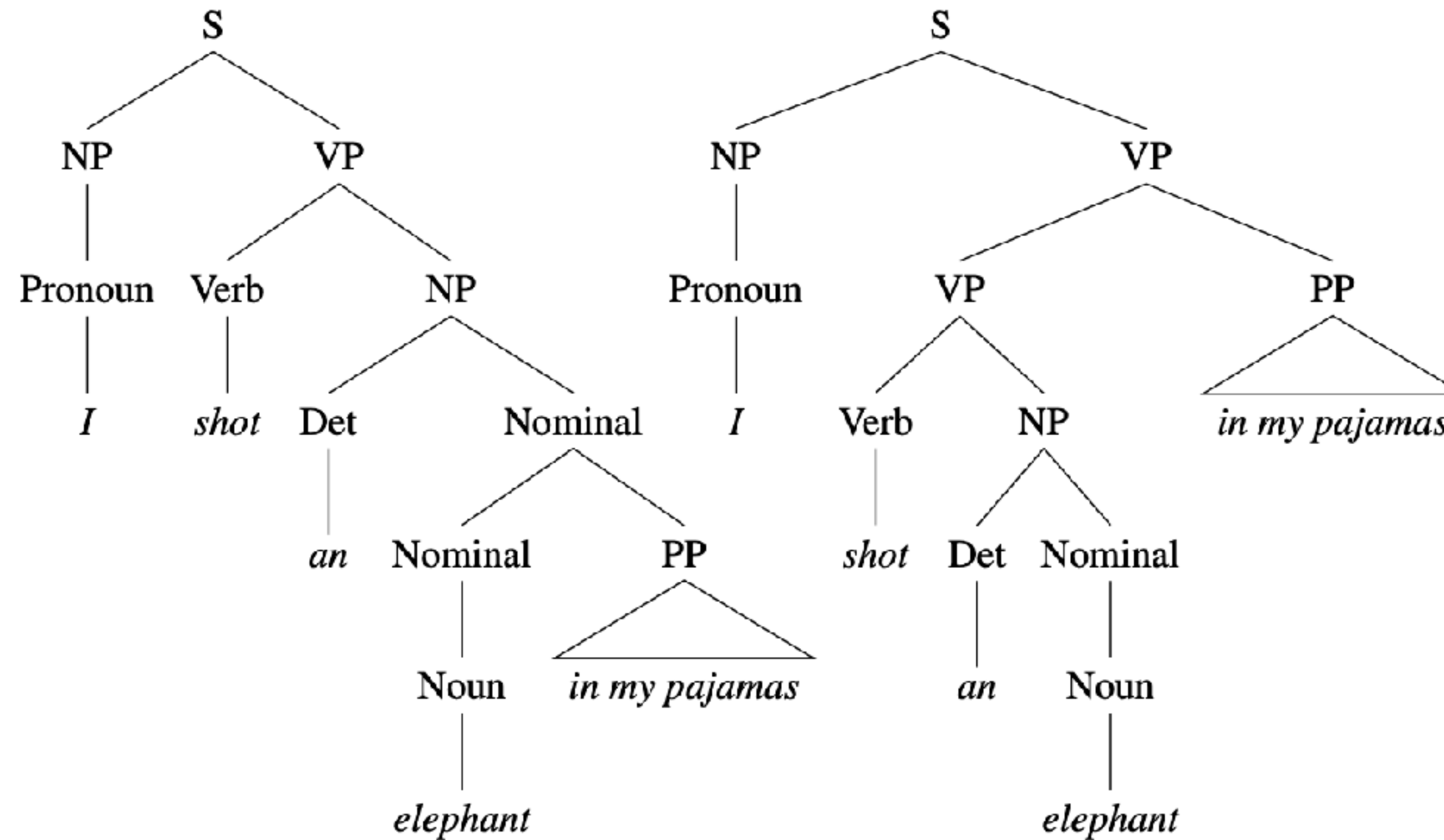
- Syntactic parsing is the task of assigning a syntactic structure to a sentence.
- Constituency parsing assigns constituency structures, those assigned by context-free grammars.
- Parse trees can be used in applications such as grammar checking, semantic analysis, applications like question answering, etc.

I shot an elephant in my pajamas



44 Ambiguity

- Ambiguity is the most serious problem faced by syntactic parsers.
- **Structural ambiguity** occurs when the grammar can assign more than one parse to a sentence.



45 Cocke-Kasami-Younger (CKY) Parsing

Bottom-up parsing:

start with the words

Dynamic programming:

save the results in a table/chart

re-use these results in finding larger constituents

Complexity: $O(n^3|G|)$

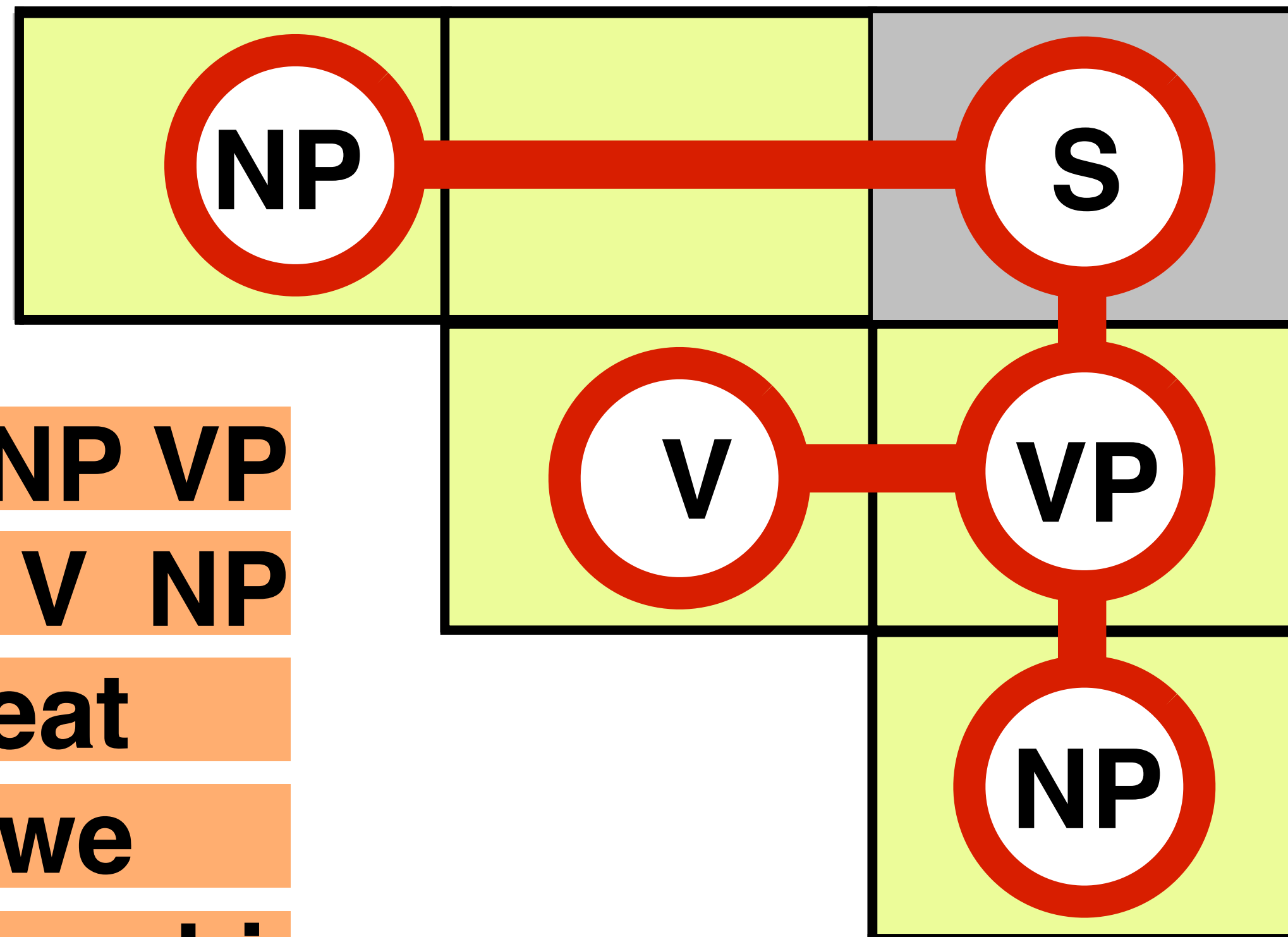
n : length of string, $|G|$: size of grammar)

Presumes a CFG in **Chomsky Normal Form**:

Rules are all either $A \rightarrow BC$ or $A \rightarrow a$

(with A, B, C nonterminals and a a terminal)

The CKY parsing algorithm



To recover the parse tree, each entry needs **pairs** of backpointers.

- S → NP VP
- VP → V NP
- V → eat
- NP → we
- NP → sushi

We eat sushi

1. Create the chart

(an $n \times n$ upper triangular matrix for an sentence with n words)

– Each cell $\text{chart}[i][j]$ corresponds to the substring $w^{(i)} \dots w^{(j)}$

2. Initialize the chart (fill the diagonal cells $\text{chart}[i][i]$):

For all rules $X \rightarrow w^{(i)}$, add an entry X to $\text{chart}[i][i]$

3. Fill in the chart:

Fill in all cells $\text{chart}[i][i+1]$, then $\text{chart}[i][i+2]$, ...,

until you reach $\text{chart}[1][n]$ (the top right corner of the chart)

– To fill $\text{chart}[i][j]$, consider all binary splits $w^{(i)} \dots w^{(k)} | w^{(k+1)} \dots w^{(j)}$

– If the grammar has a rule $X \rightarrow YZ$, $\text{chart}[i][k]$ contains a Y and $\text{chart}[k+1][j]$ contains a Z , add an X to $\text{chart}[i][j]$ with two backpointers to the Y in $\text{chart}[i][k]$ and the Z in $\text{chart}[k+1][j]$

4. Extract the parse trees from the S in $\text{chart}[1][n]$.

CKY: filling the chart

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

w	w _i	...	w	
							w
							...
							..
							w _i
							...
							w

49 Converting a Generic CFG into CNF

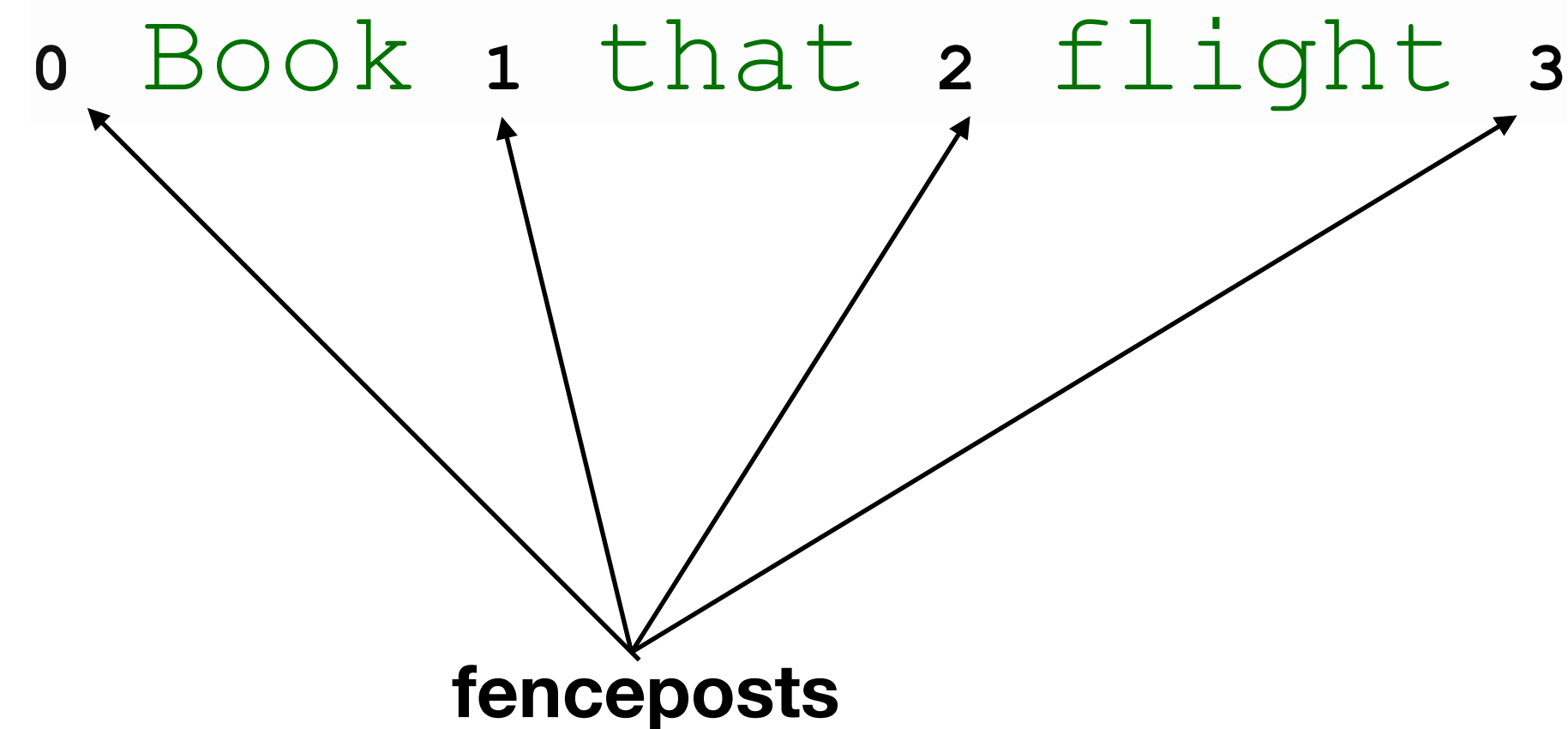
- The CKY algorithm requires grammars to first be in Chomsky Normal Form (CNF).

\mathcal{L}_1 Grammar	\mathcal{L}_1 in CNF
$S \rightarrow NP VP$	$S \rightarrow NP VP$
$S \rightarrow Aux NP VP$	$S \rightarrow XI VP$
	$XI \rightarrow Aux NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb NP$
	$S \rightarrow X2 PP$
	$S \rightarrow Verb PP$
	$S \rightarrow VP PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper-Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det Nominal$	$NP \rightarrow Det Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal Noun$	$Nominal \rightarrow Nominal Noun$
$Nominal \rightarrow Nominal PP$	$Nominal \rightarrow Nominal PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb NP$	$VP \rightarrow Verb NP$
$VP \rightarrow Verb NP PP$	$VP \rightarrow X2 PP$
	$X2 \rightarrow Verb NP$
$VP \rightarrow Verb PP$	$VP \rightarrow Verb PP$
$VP \rightarrow VP PP$	$VP \rightarrow VP PP$
$PP \rightarrow Preposition NP$	$PP \rightarrow Preposition NP$

Figure 13.3 \mathcal{L}_1 Grammar and its conversion to CNF. Note that although they aren't shown here, all the original lexical entries from \mathcal{L}_1 carry over unchanged as well.

50 Indexing Scheme

- With grammar in CNF, each non-terminal node have exactly two daughters
- Use a $(n+1) * (n+1)$ matrix to encode the structure of a tree
- $[i, j]$ represents a constituent between fencepost i and j



51 A Completed Parse Table

- Proceed in a bottom-up fashion
- This scheme guarantees that at each point in time we have all the information we need (to the left, since all the columns to the left have already been filled, and below since we're filling bottom to top).

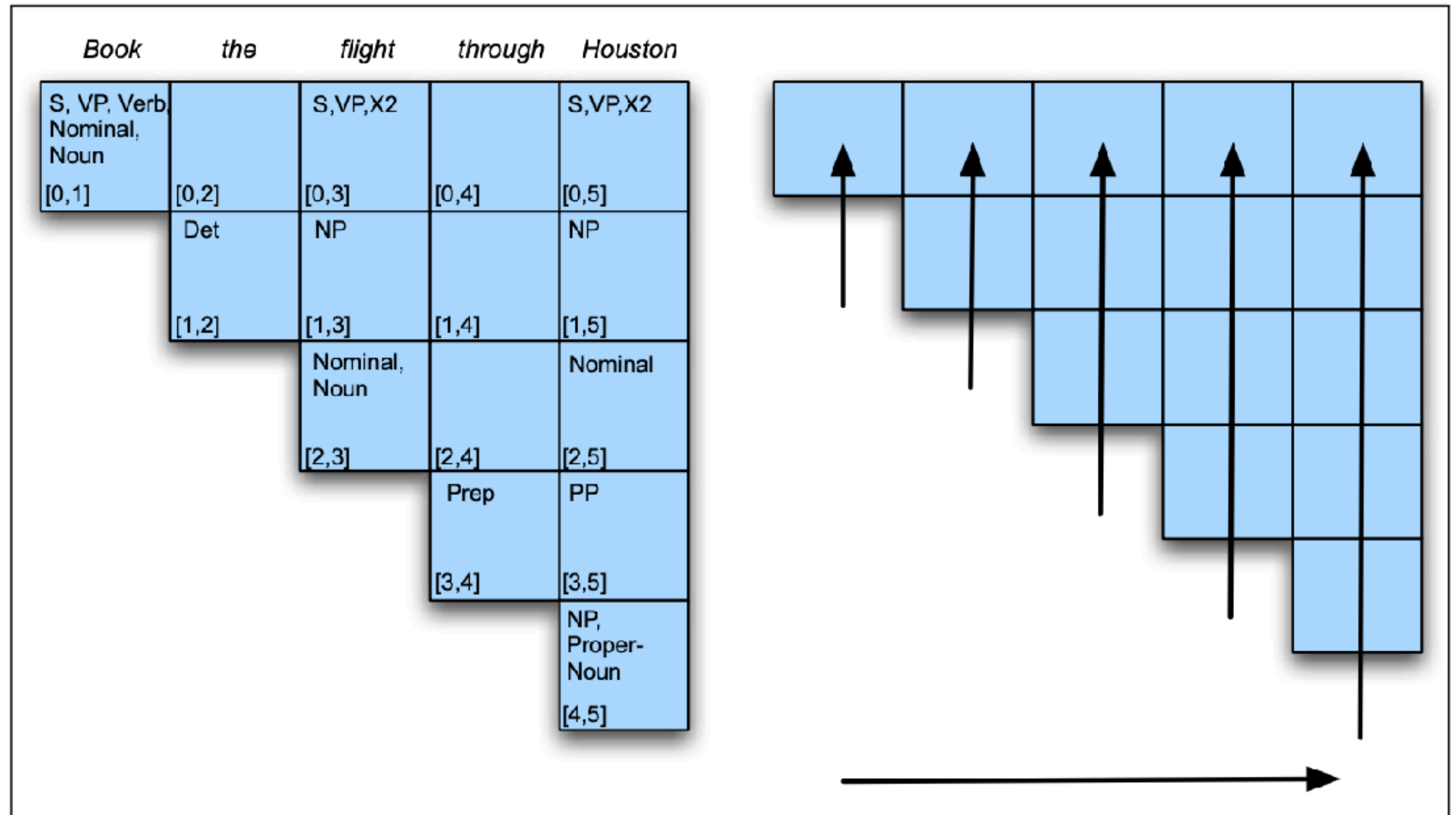
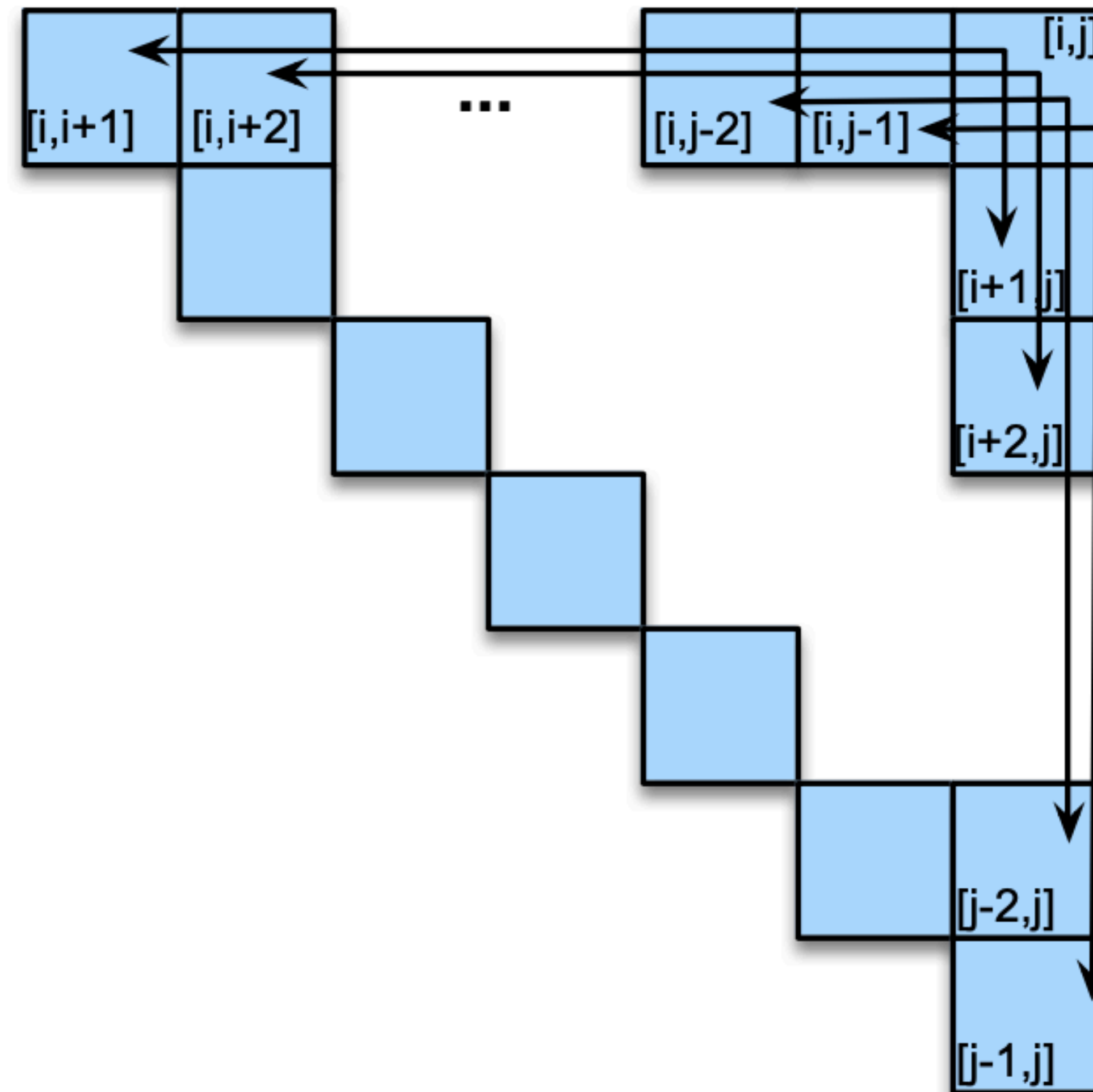


Figure 13.4 Completed parse table for *Book the flight through Houston*.

Fill the $[i, j]$ -th Cell in the CKY Table


- Since each non-terminal entry in our table has two daughters in the parse, it follows that for each constituent represented by an entry $[i, j]$, there must be a position in the input, k , where it can be split into two parts such that $i < k < j$.
- Given such a position k , the first constituent $[i, k]$ must lie to the left of entry $[i, j]$ somewhere along row i , and the second entry $[k, j]$ must lie beneath it, along column j .



Fill Column 5

- Each non-terminal is paired with pointers to the table entries from which it was derived
- Permit multiple versions of the same non-terminals to be entered into the table

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	[3,5]
				NP, Proper- Noun [4,5]

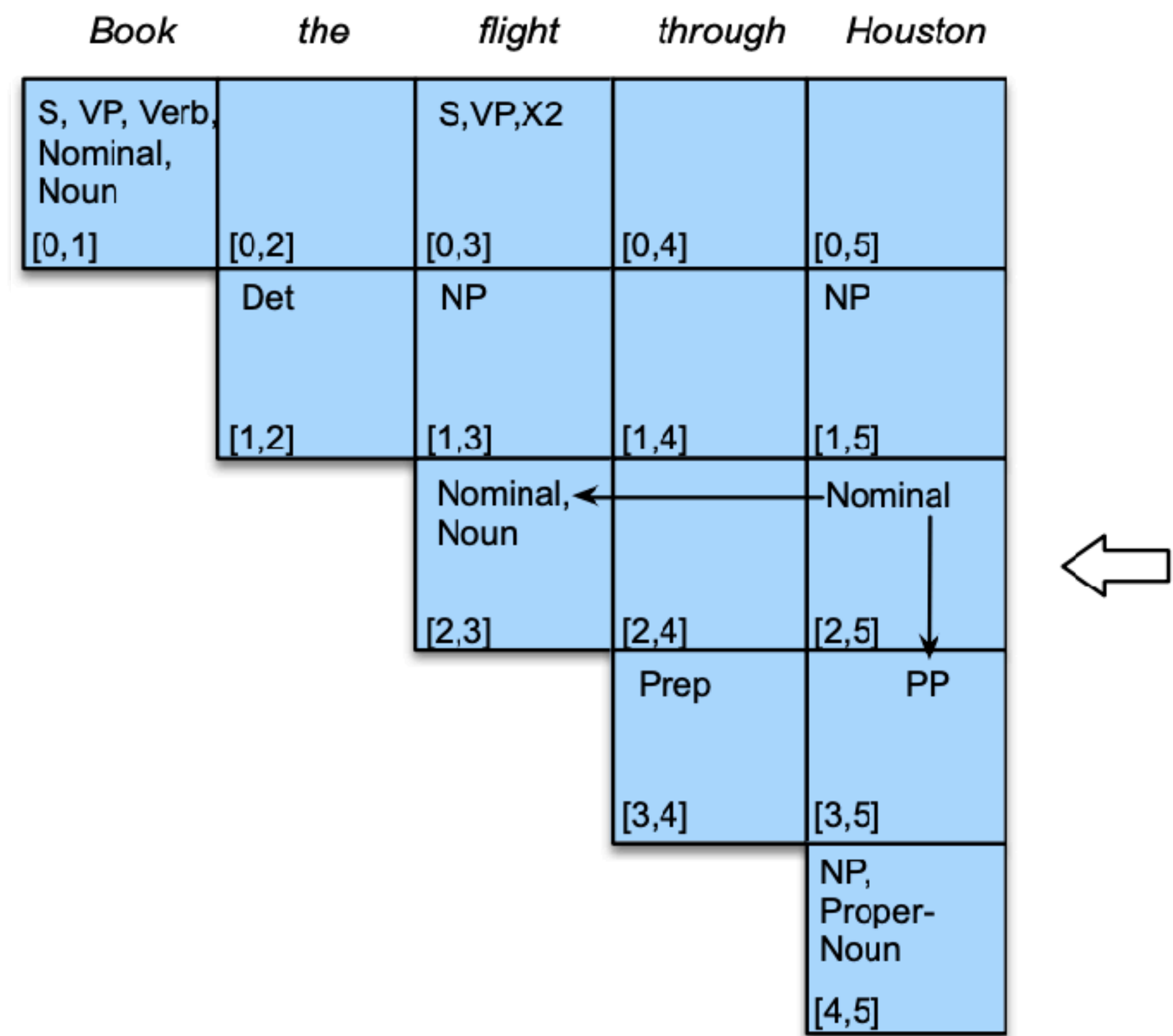


Fill Column 5

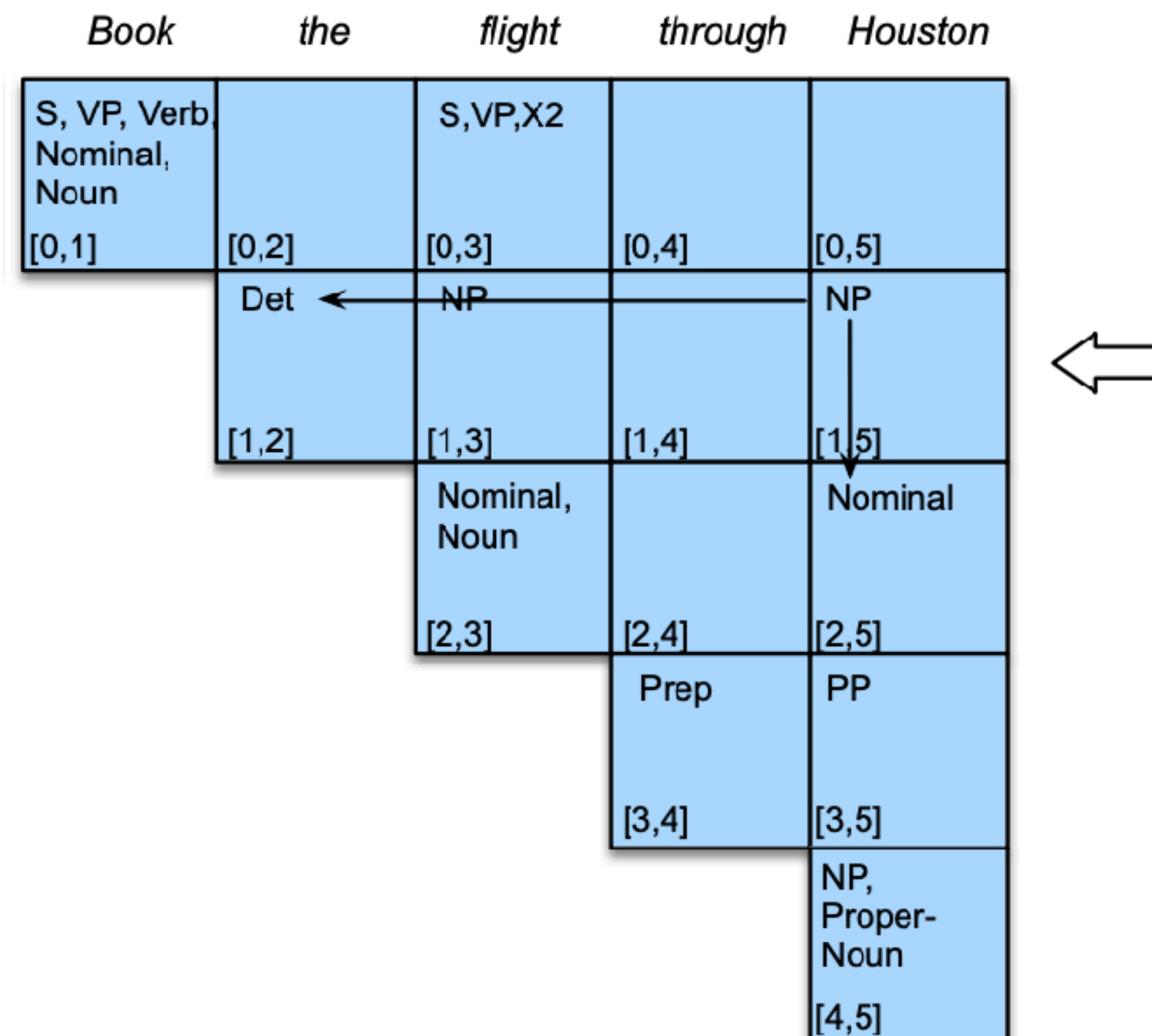
	<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]	
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]	
		Nominal, Noun [2,3]	[2,4]	[2,5]	
			Prep [3,4]	PP [3,5]	
				NP, Proper- Noun [4,5]	



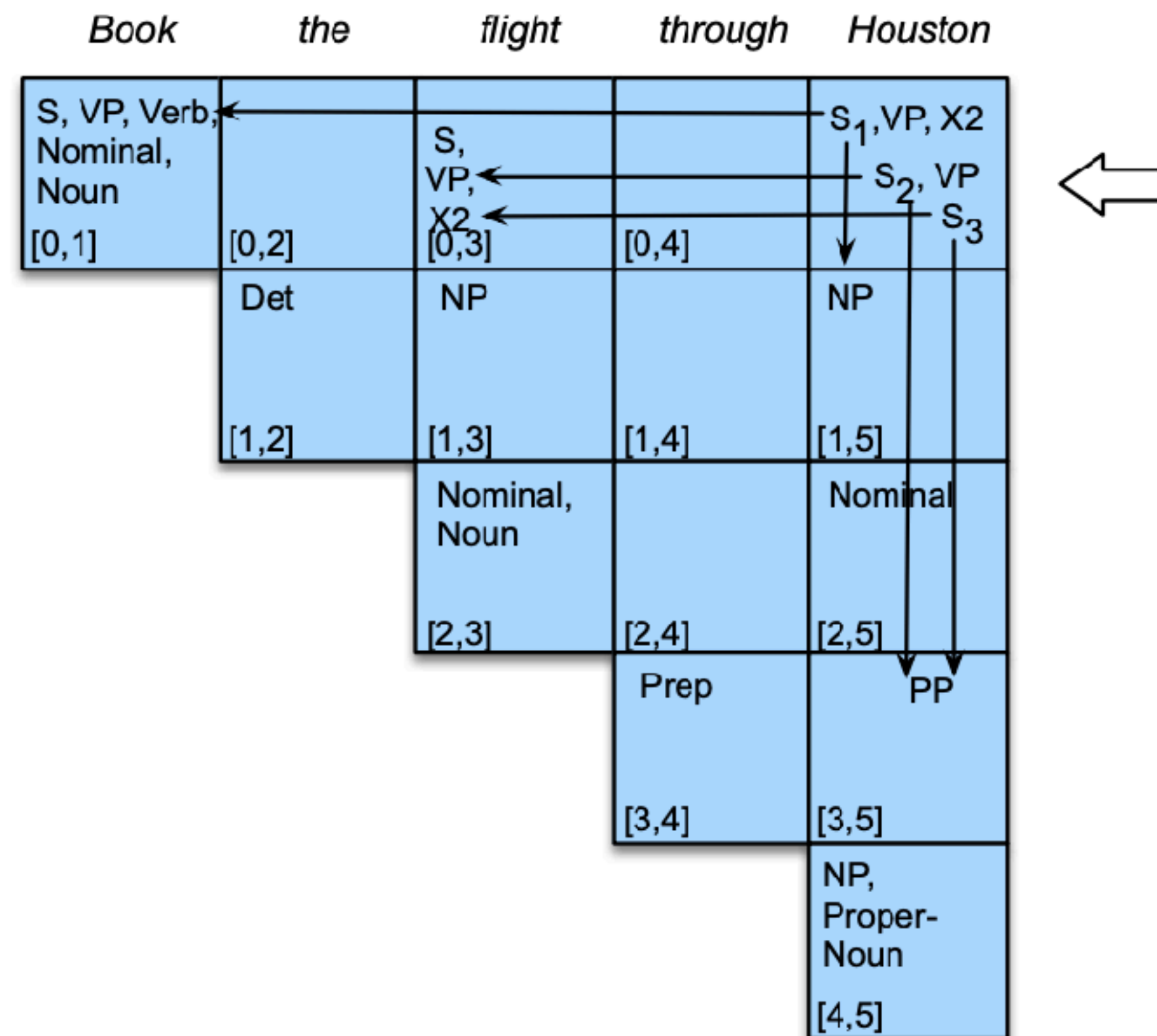
Fill Column 5



Fill Column 5



57 Fill Column 5



Evaluating Parsers

- The **PARSEVAL** metrics: measures how much the constituents in the hypothesis parse tree look like the constituents in a hand-labeled, reference parse.

$$\text{labeled recall} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of total constituents in reference parse of } s}$$

$$\text{labeled precision} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of total constituents in hypothesis parse of } s}$$

As usual, we often report a combination of the two, F_1 :

$$F_1 = \frac{2PR}{P + R}$$

- **Treebank**: a syntactically annotated corpus where every sentence in the collection is paired with a corresponding parse tree.
- A wide variety of treebanks have been created.
- The **Penn Treebank** project has produced treebanks from the Brown, Switchboard, ATIS, and Wall Street Journal corpora of English, as well as treebanks in Arabic and Chinese.

60 Penn Treebank

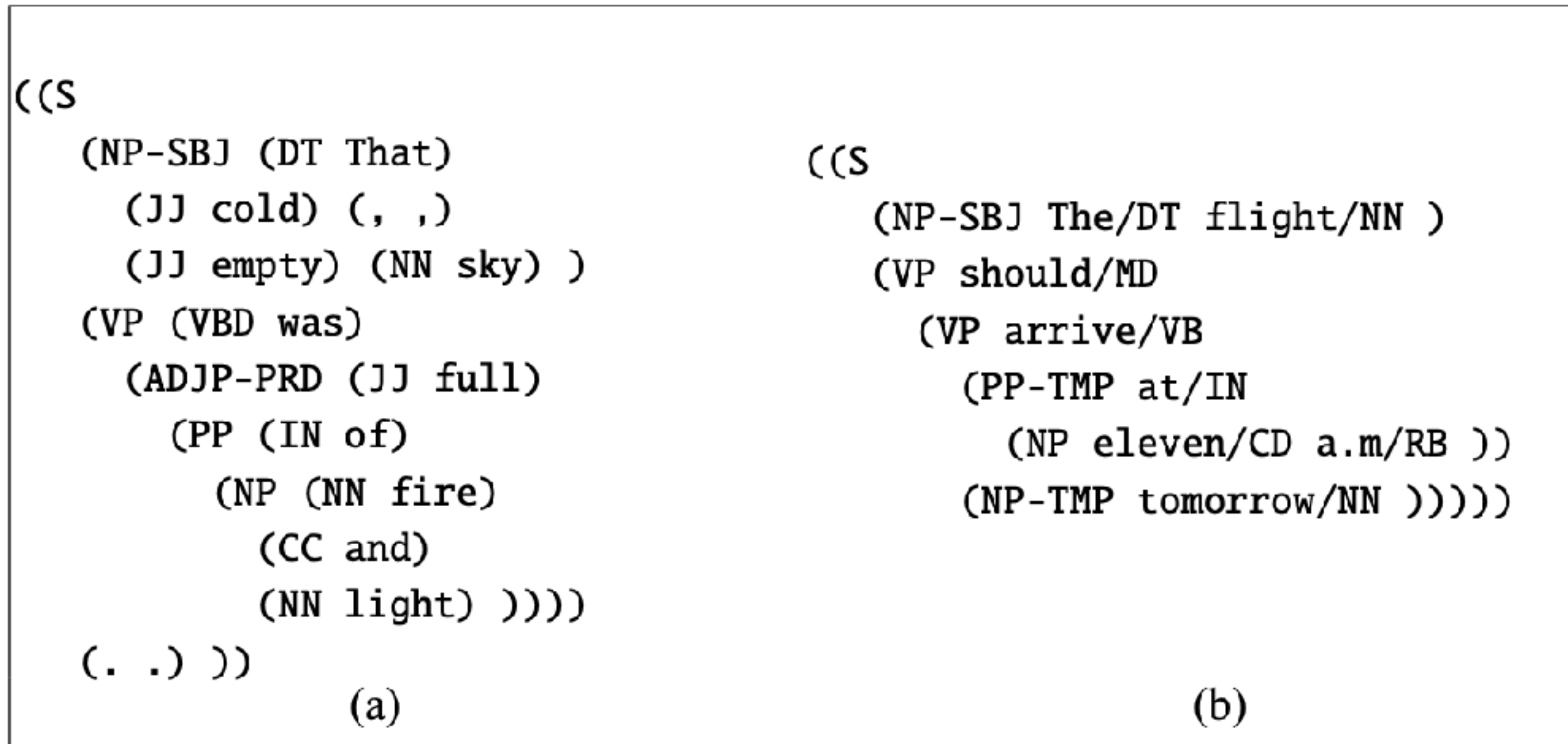


Figure 12.7 Parsed sentences from the LDC Treebank3 version of the (a) Brown and (b) ATIS corpora.

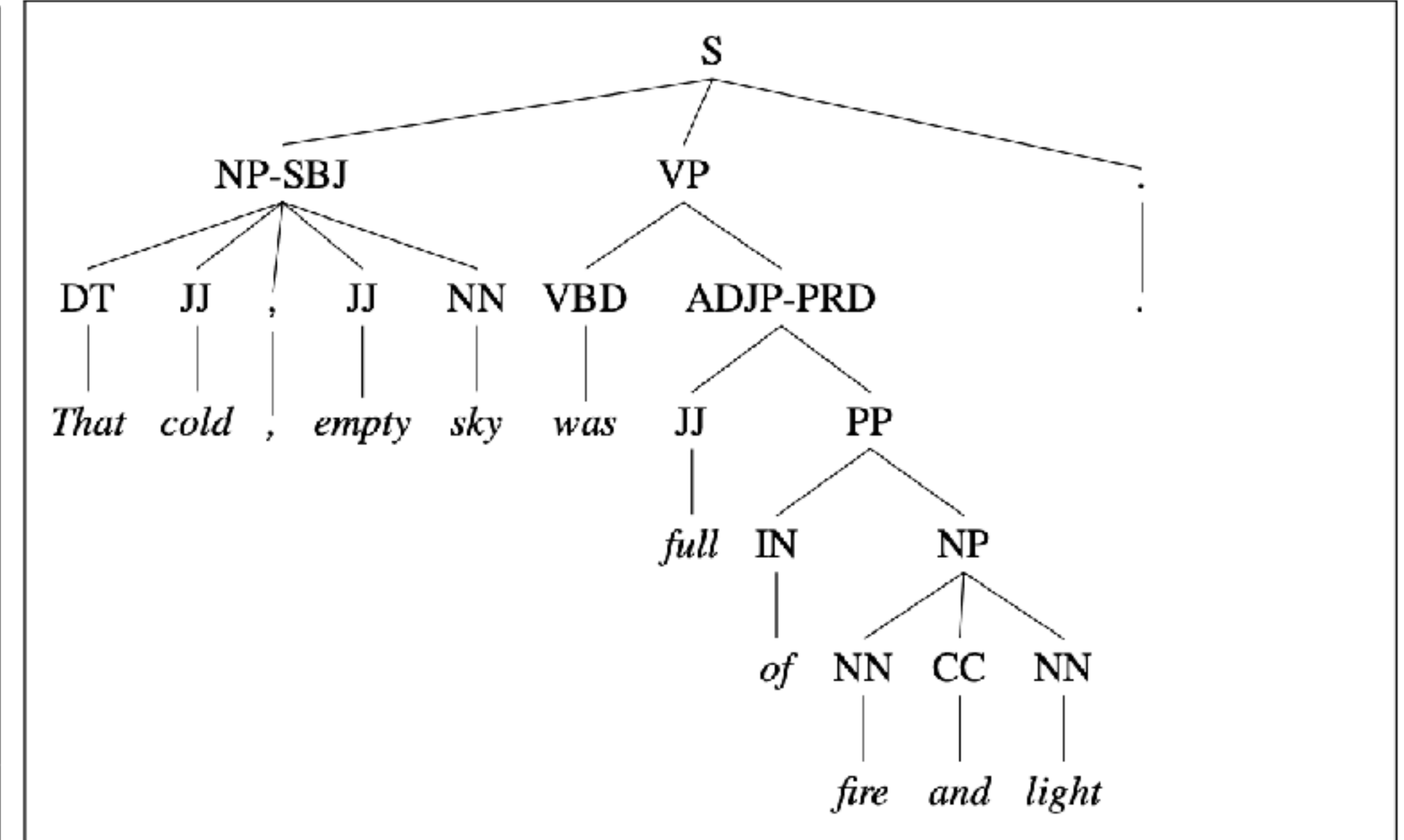
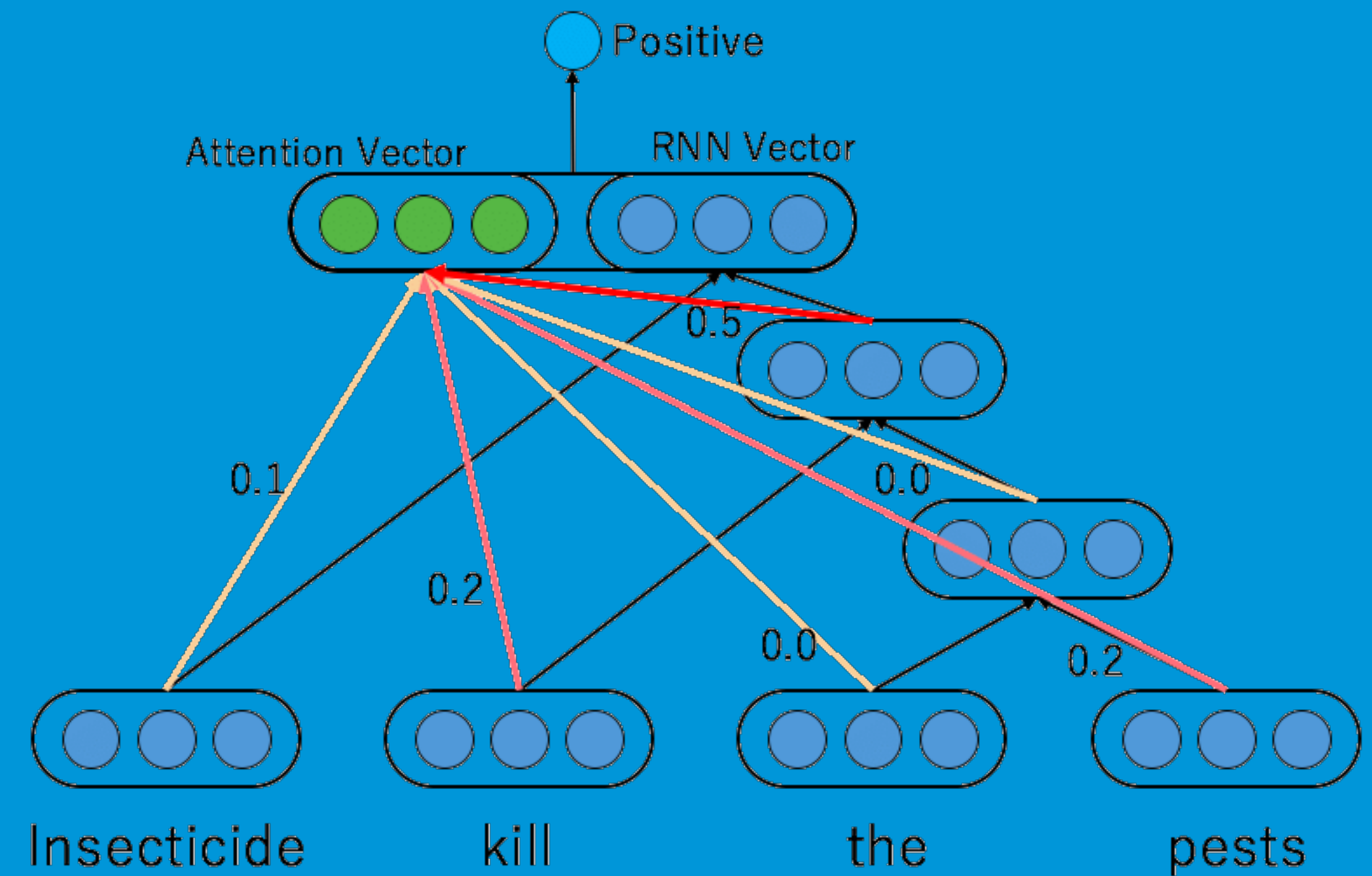


Figure 12.8 The tree corresponding to the Brown corpus sentence in the previous figure.

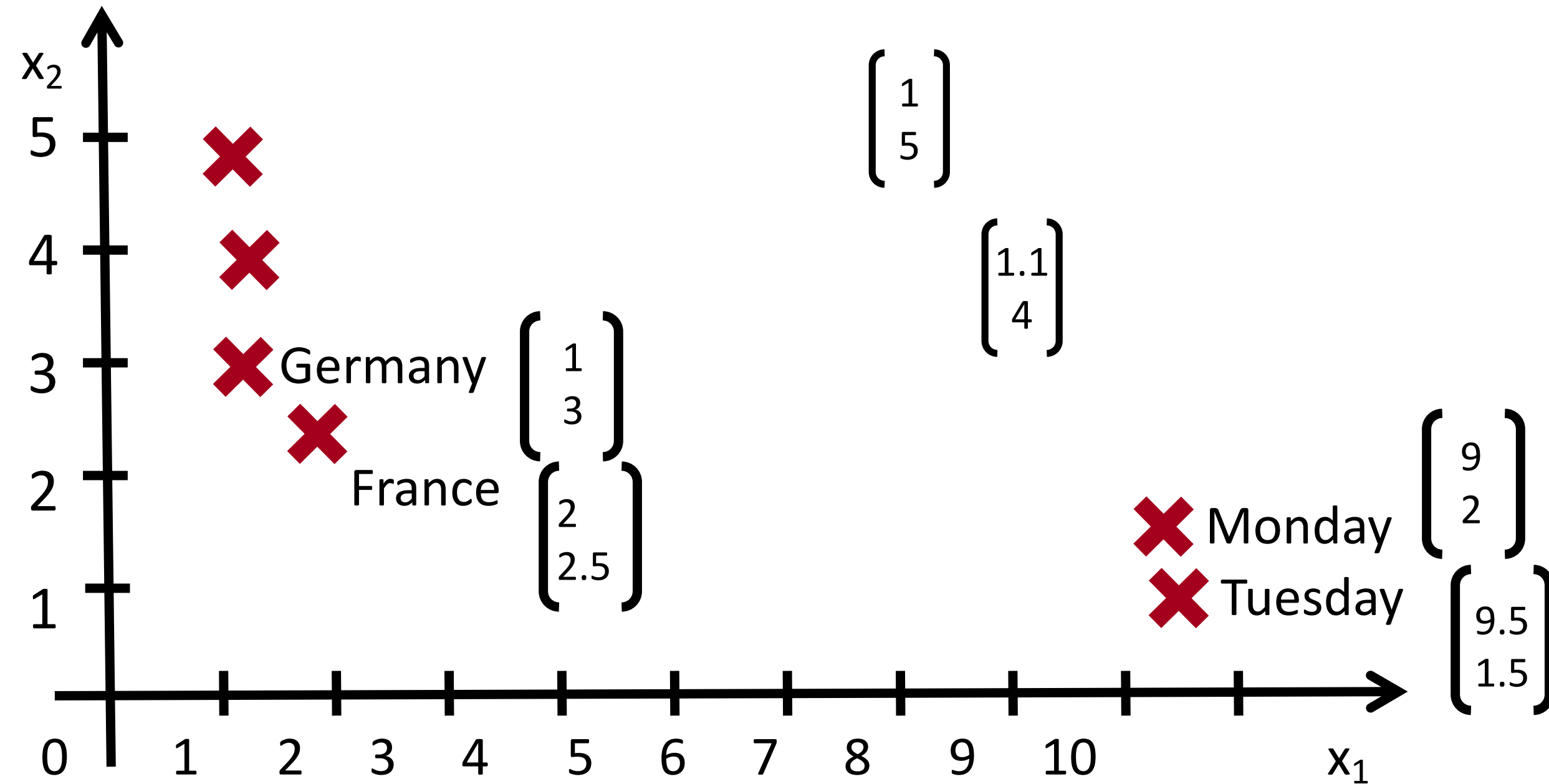
61 Span-Based Neural Constituency Parsing

- **CKY parsing** does great at **enumerating all the possible parse trees** for a sentence
- But it doesn't tell us which parse is the correct one!
- That is, **it doesn't disambiguate** among the possible parses
- To solve the disambiguation problem we'll use a simple **neural extension** of the CKY algorithm.
- The intuition of such parsing algorithms (often called span-based constituency parsing, or neural CKY), is to train a neural classifier to assign a score to each constituent, and then use a modified version of CKY to combine these constituent scores to find the best-scoring parse tree.

Parsing by TreeRNN



63 Building on Word Vector Space Models



the country of my birth
the place where I was born

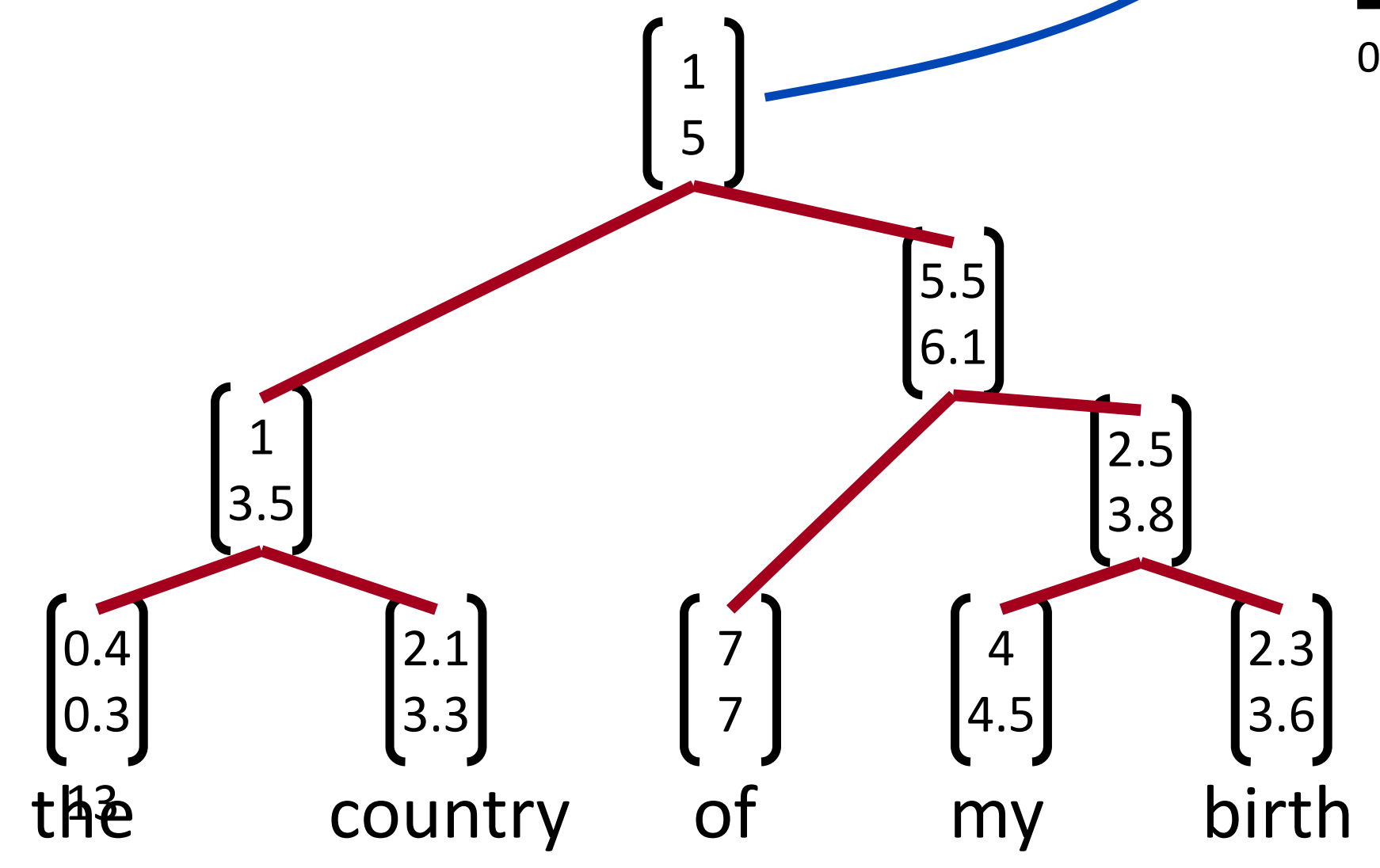
How can we represent the meaning of longer phrases?

By mapping them into the same vector space!

64 How should we map phrases into a vector space?

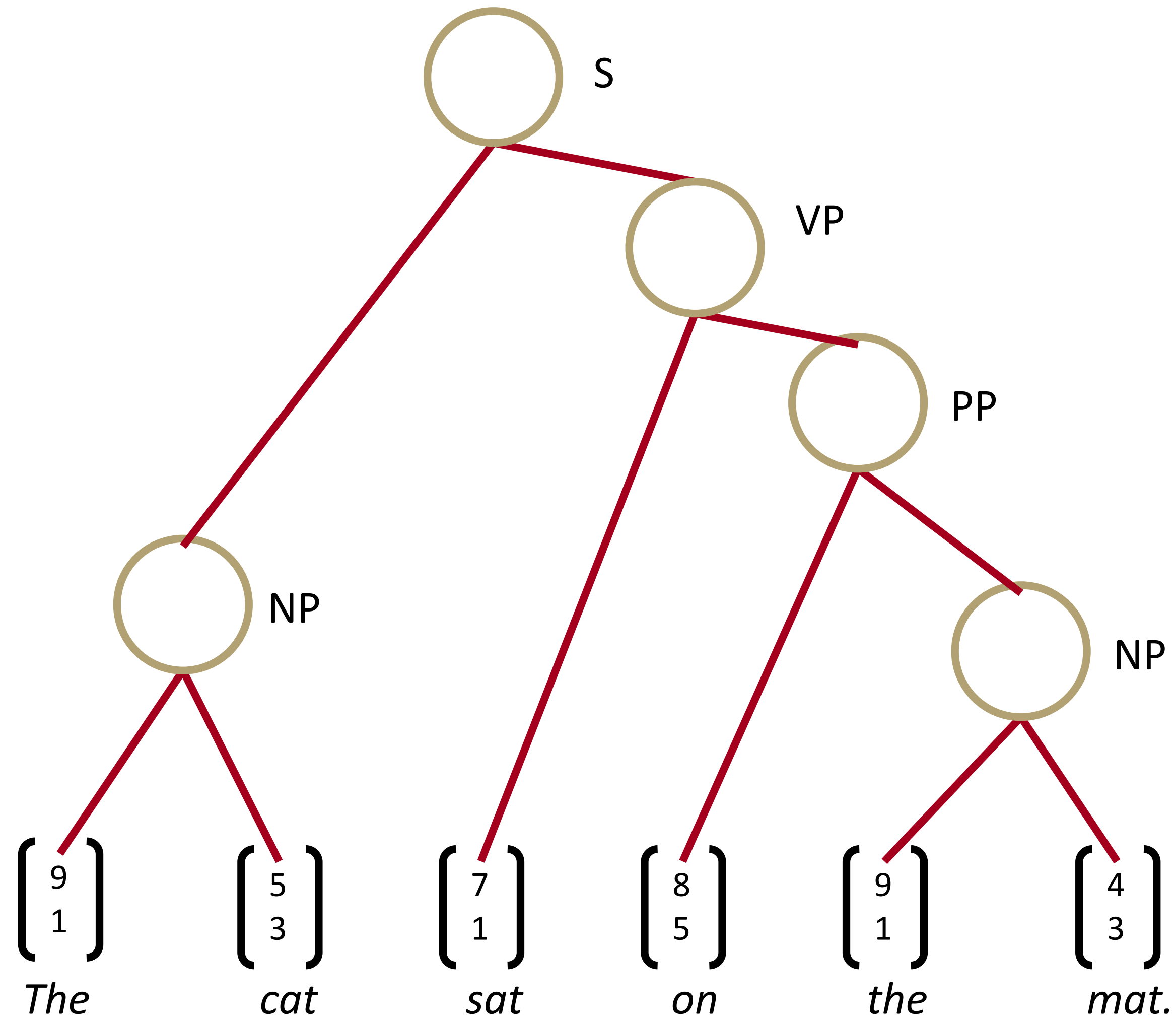
Use principle of compositionality
The meaning (vector) of a sentence is determined by
(1) the meanings of its words and
(2) the rules that combine them.

Socher, Manning, and Ng. ICML, 2011

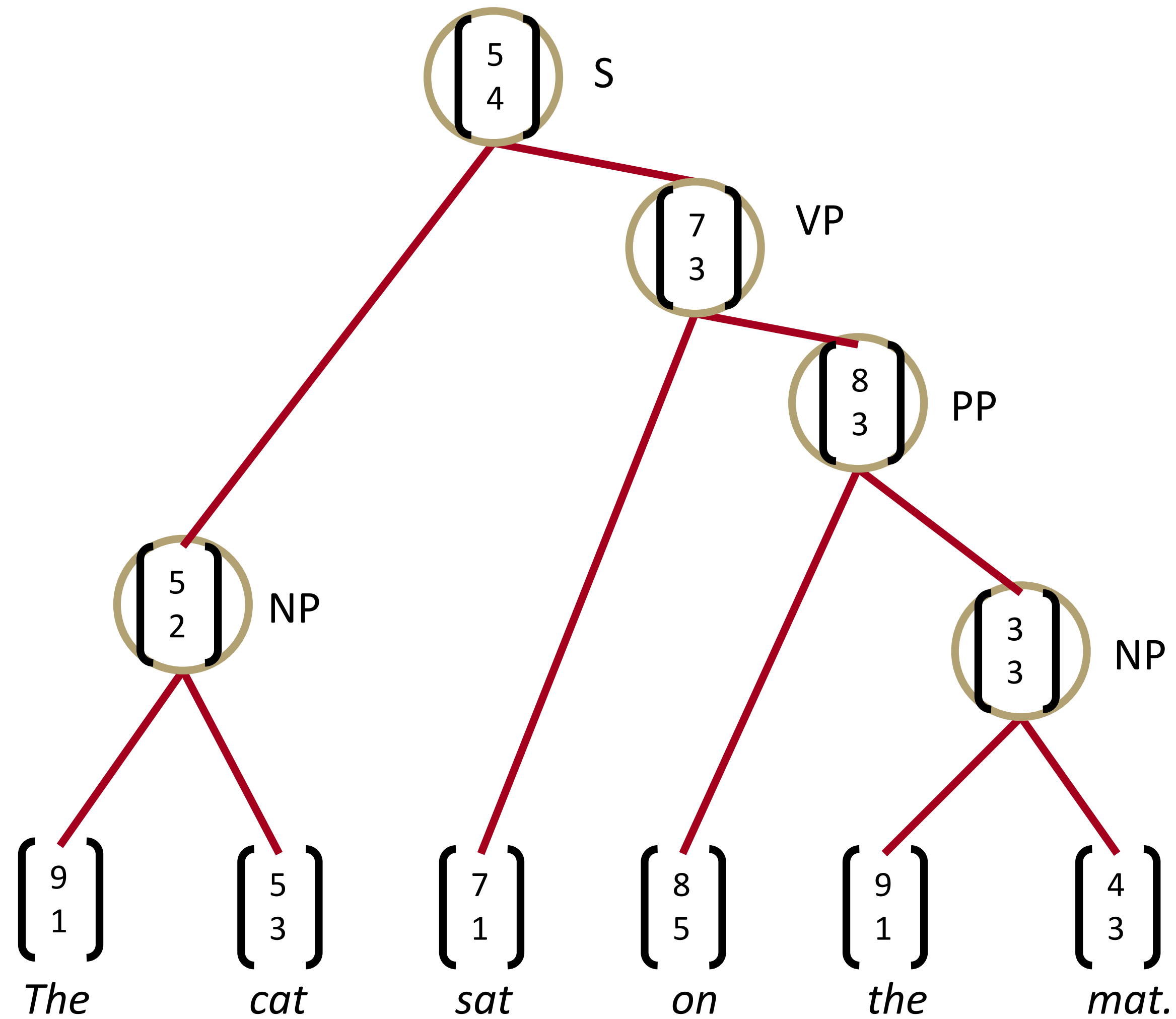


Models in this section can jointly learn parse trees and compositional vector representations

65 Constituency Sentence Parsing: What we want

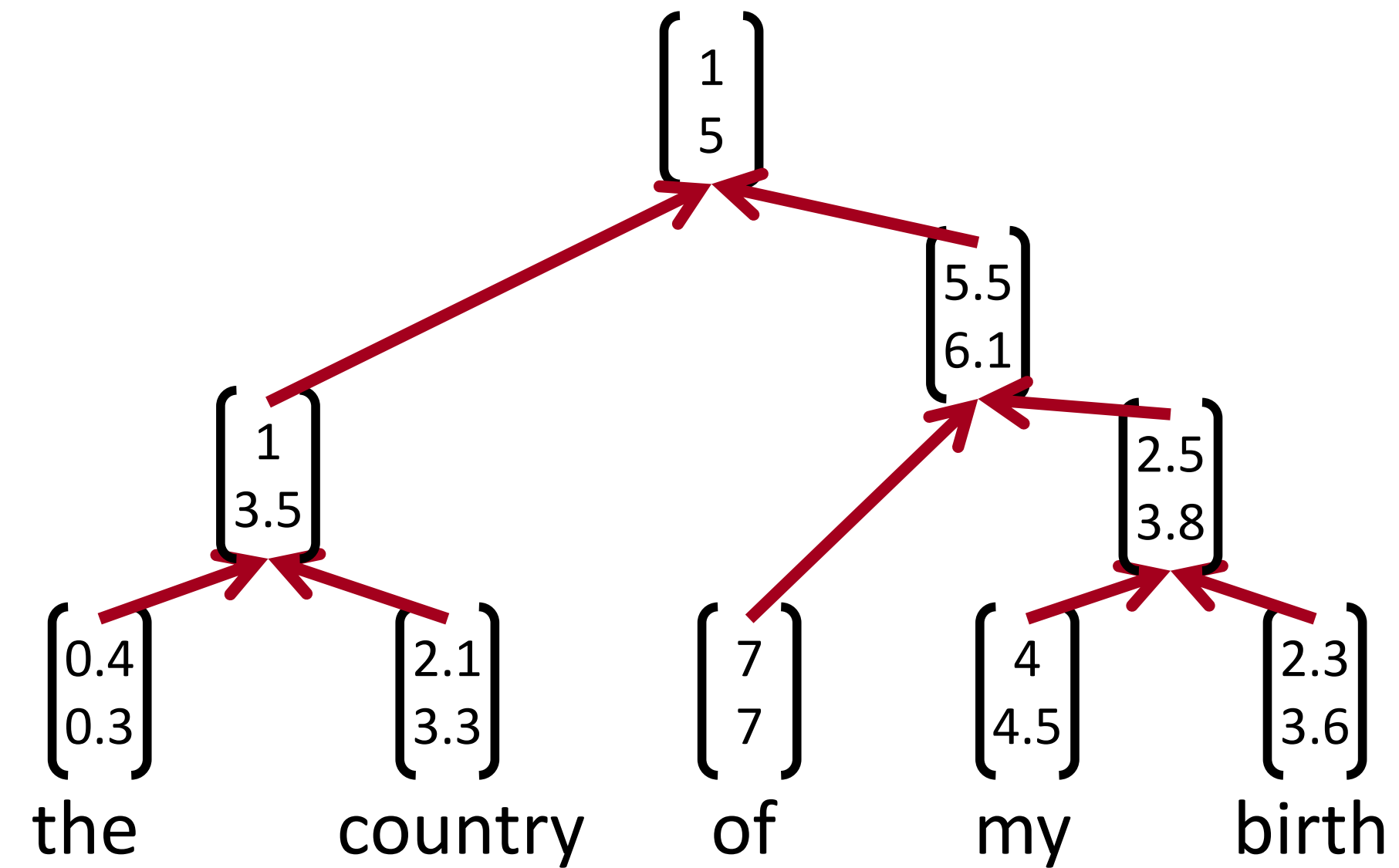


66 Learn Structure and Representation

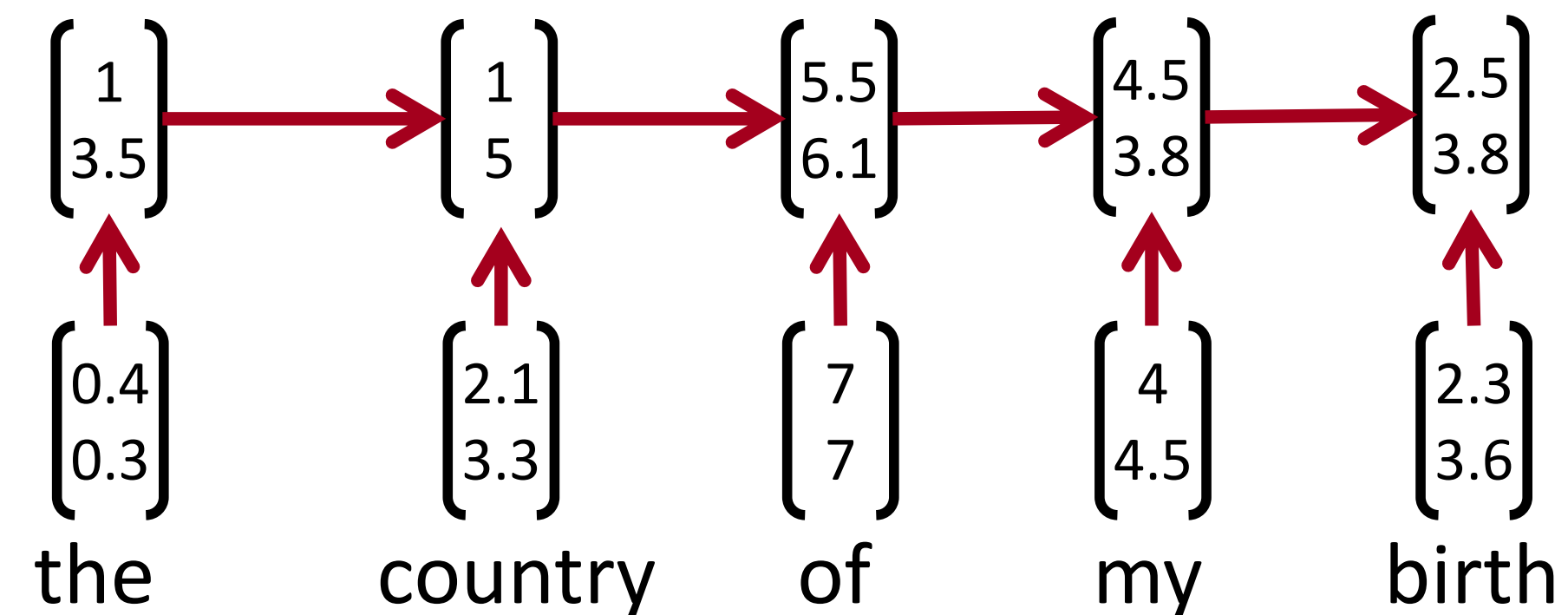


67 Recursive vs. recurrent neural networks

- Recursive neural nets require a tree structure



- Recurrent neural nets cannot capture phrases without prefix context and often capture too much of last words in final vector

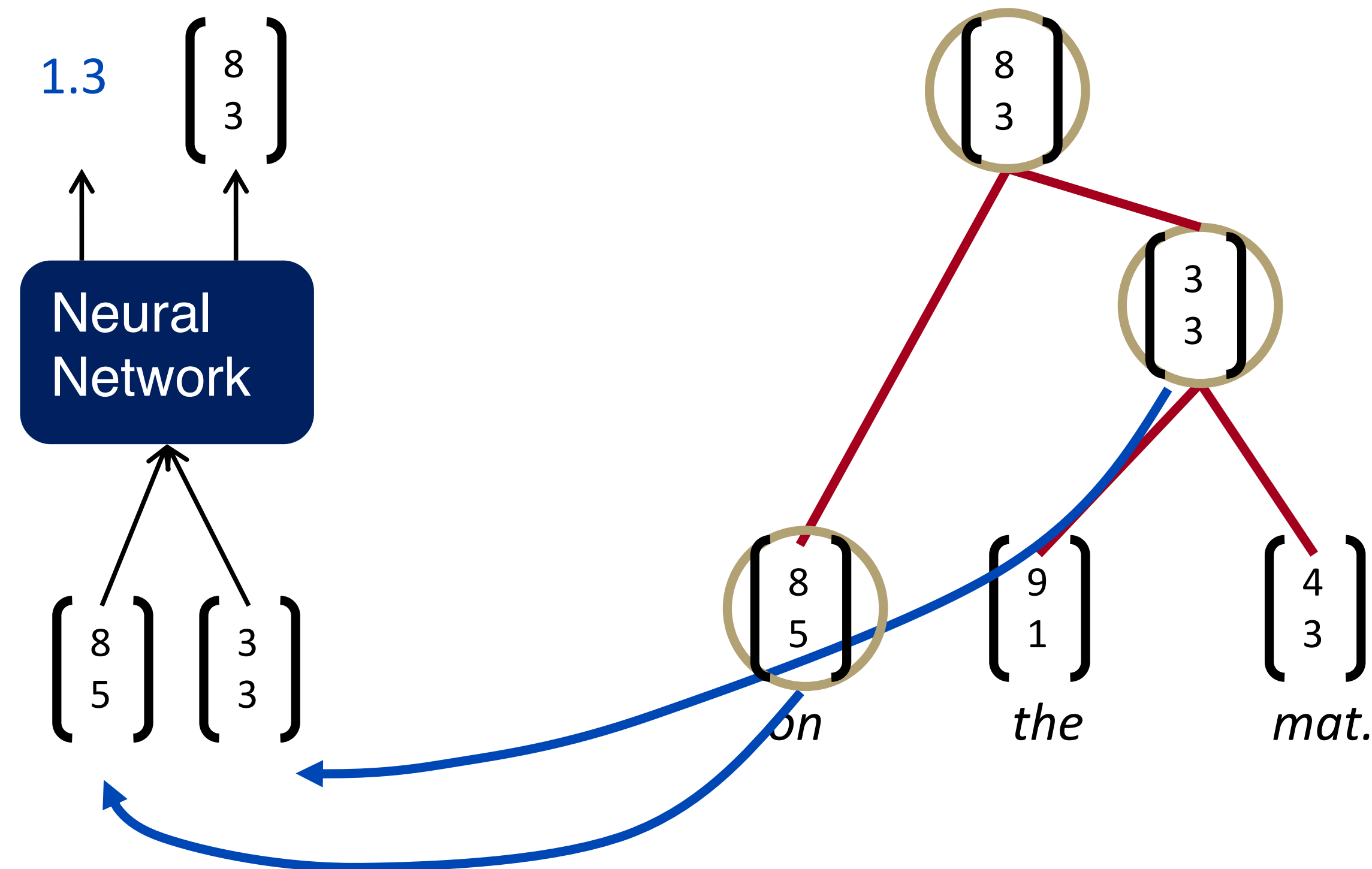


68 Recursive NNs for Structure Prediction

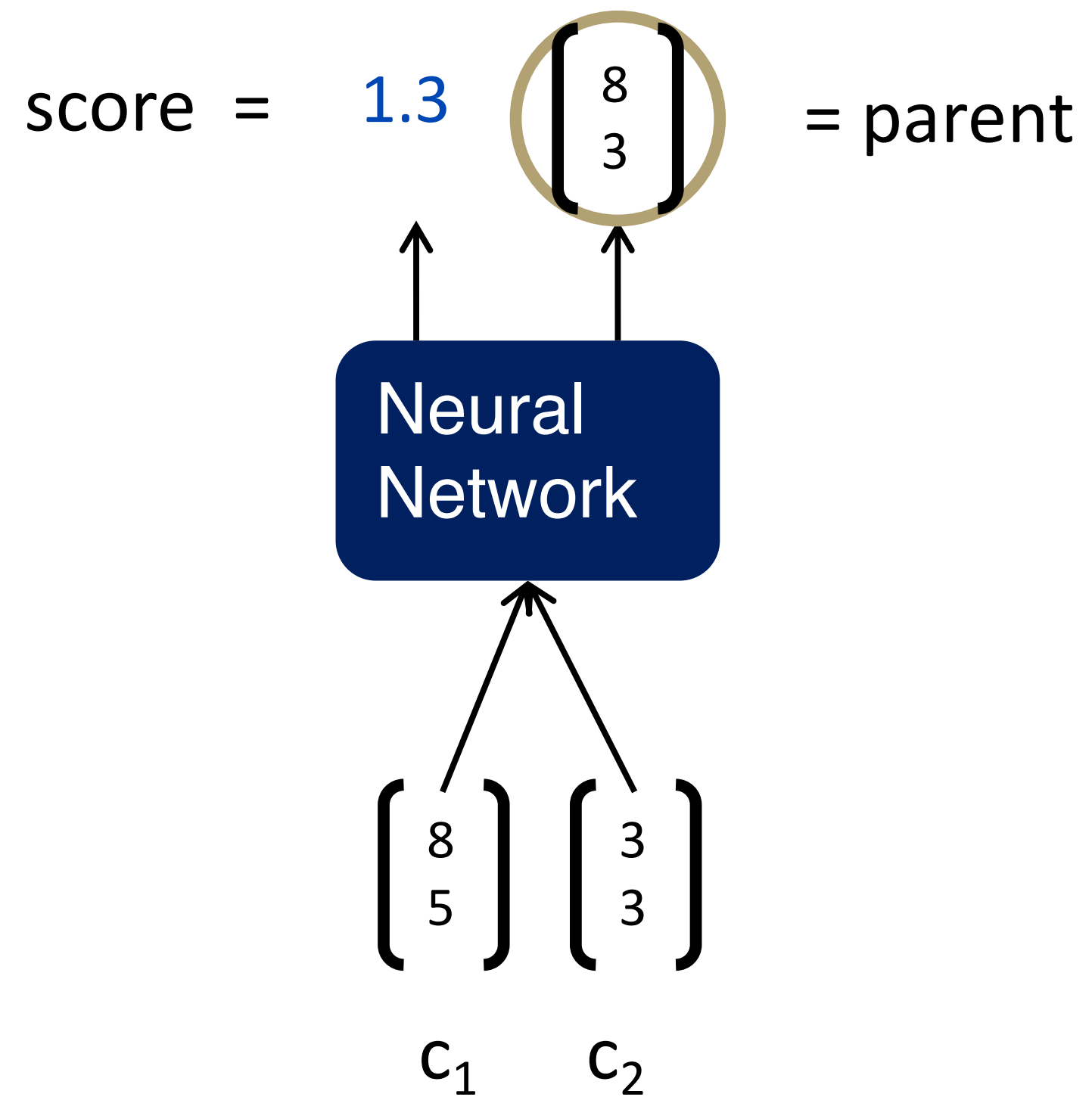
Inputs: two candidate children's representations

Outputs:

1. The semantic representation if the two nodes are merged.
2. Score of how plausible the new node would be.



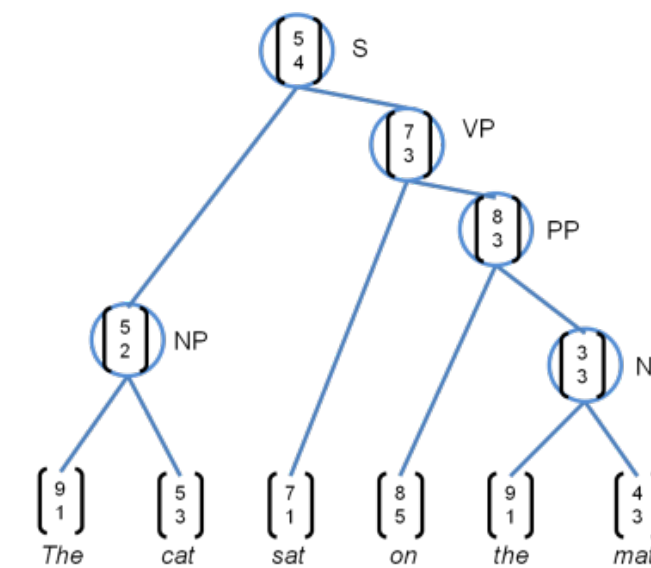
69 Recursive Neural Network Definition



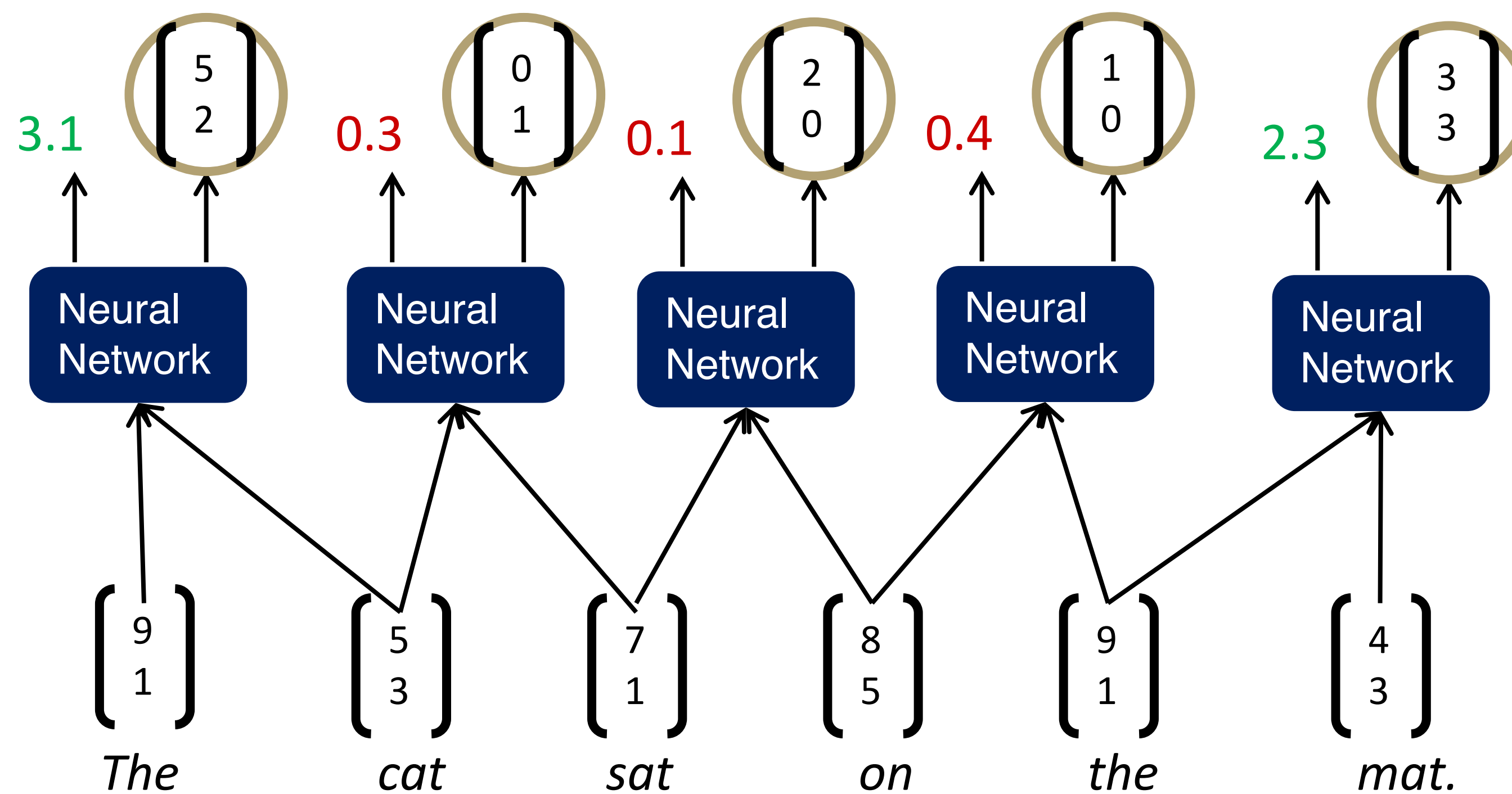
$$\text{score} = U^T p$$

$$p = \tanh\left(W \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + b\right),$$

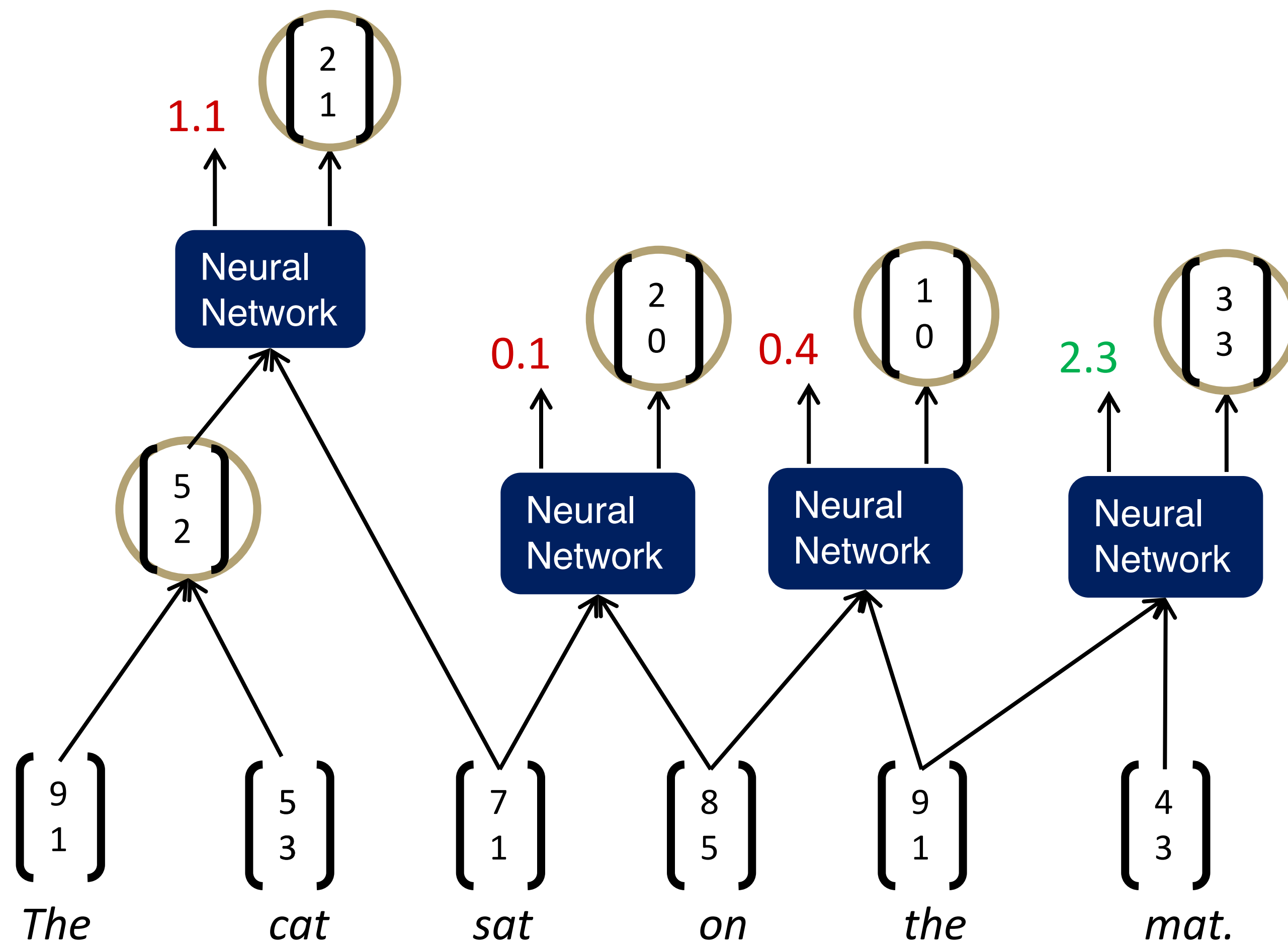
Same W parameters at all nodes of the tree



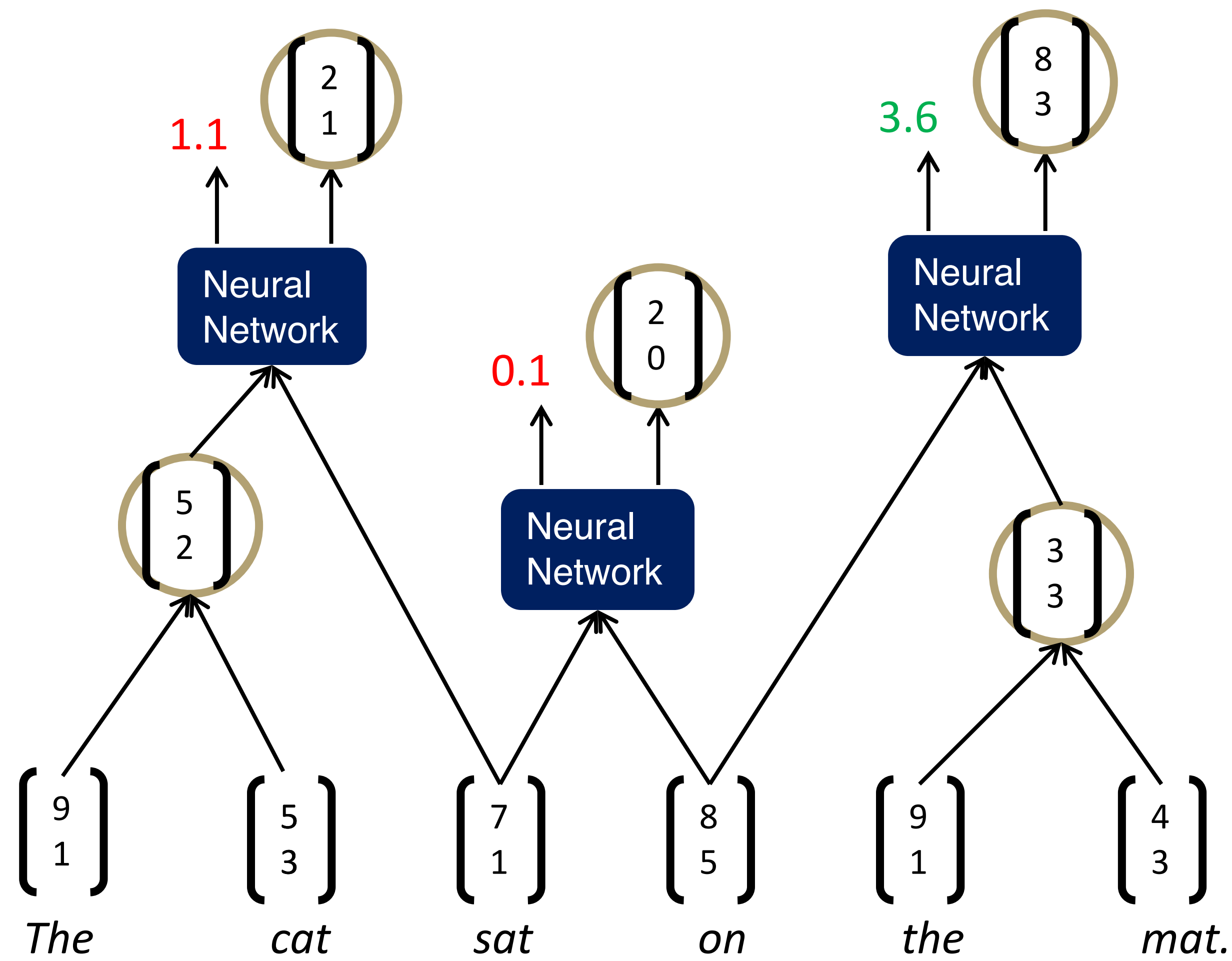
70 Parsing a sentence with an RNN (greedily)



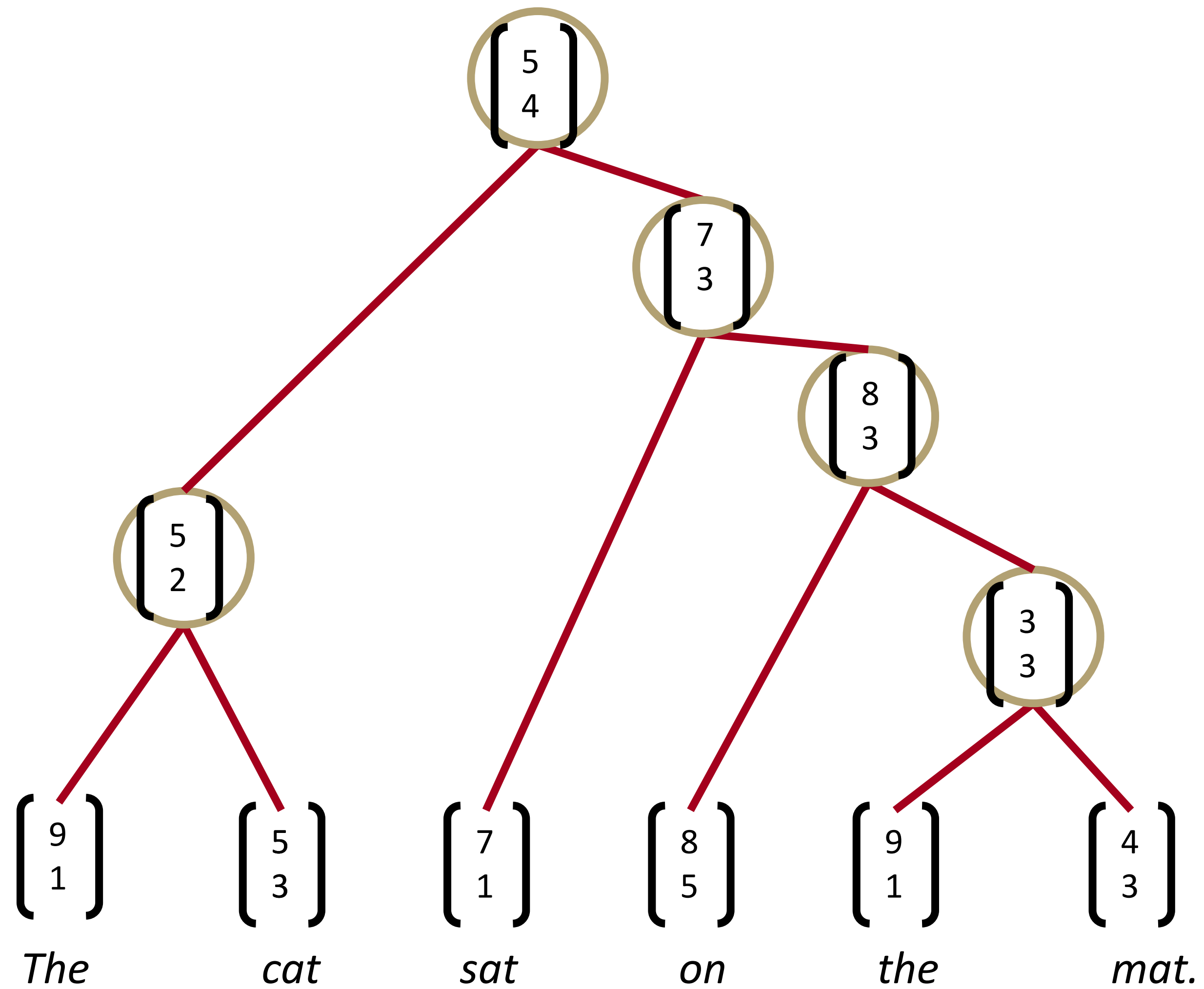
71 Parsing a sentence



72 Parsing a sentence



73 Parsing a sentence



74 Score of a tree

- The score of a tree is computed by the sum of the parsing decision scores at each node:

$$s(x, y) = \sum_{n \in \text{nodes}(y)} s_n$$

- x is sentence; y is parse tree



75 Max-Margin Framework

- Similar to max-margin parsing (Taskar et al. 2004), a supervised max-margin objective

$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} (s(x_i, y) + \Delta(y, y_i))$$

- The loss $\Delta(y, y_i)$ penalizes all incorrect decisions
- Structure search for $A(x)$ was greedy (join best nodes each time)
 - Instead: Beam search with chart

Max-margin parsing: <https://www.aclweb.org/anthology/W04-3201.pdf>

76 Backpropagation Through Structure

Introduced by **Goller & Küchler (1996)** – old stuff!

Principally the same as general backpropagation

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)} \right) \circ f'(z^{(l)}),$$

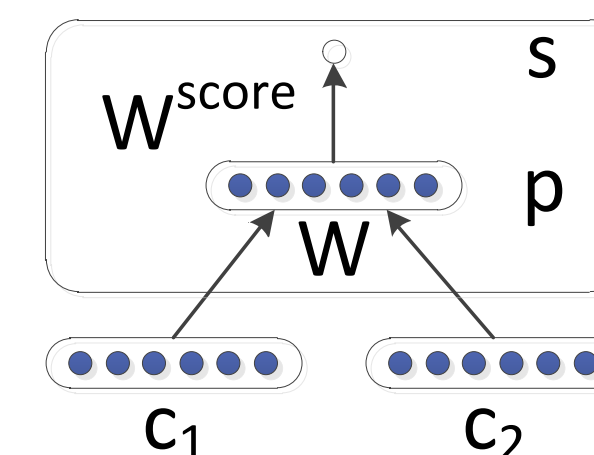
$$\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}$$

Calculations resulting from the recursion and tree structure:

1. Sum derivatives of W from all nodes (like RNN)
2. Split derivatives at each node (for tree)
3. Add error messages from parent + node itself

77 Discussion: Simple TreeRNN

- Decent results with single layer TreeRNN
- Single weight matrix TreeRNN could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences
- There is no real interaction between the input words
- The composition function is the same for all syntactic categories, punctuation, etc.



78 TreeLSTM

<https://www.aclweb.org/anthology/P15-1150.pdf>

[Tai et al., ACL 2015; also Zhu et al. ICML 2015]

Goals:

- Still trying to represent the meaning of a sentence as a location in a (high-dimensional, continuous) vector space
- In a way that accurately handles semantic composition and sentence meaning
- Generalizing the widely used chain-structured LSTM to trees

<https://www.aclweb.org/anthology/P15-1150.pdf>

$$i_t = \sigma \left(W^{(i)} x_t + U^{(i)} h_{t-1} + b^{(i)} \right), \quad (1)$$

$$f_t = \sigma \left(W^{(f)} x_t + U^{(f)} h_{t-1} + b^{(f)} \right),$$

$$o_t = \sigma \left(W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)} \right),$$

$$u_t = \tanh \left(W^{(u)} x_t + U^{(u)} h_{t-1} + b^{(u)} \right),$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1},$$

$$h_t = o_t \odot \tanh(c_t),$$

Original LSTM

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \quad (2)$$

$$i_j = \sigma \left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right), \quad (3)$$

$$f_{jk} = \sigma \left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right), \quad (4)$$

$$o_j = \sigma \left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right), \quad (5)$$

$$u_j = \tanh \left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right), \quad (6)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \quad (7)$$

$$h_j = o_j \odot \tanh(c_j), \quad (8)$$

Child-Sum Tree-LSTMs

<https://www.aclweb.org/anthology/P15-1150.pdf>

$$i_t = \sigma \left(W^{(i)} x_t + U^{(i)} h_{t-1} + b^{(i)} \right), \quad (1)$$

$$f_t = \sigma \left(W^{(f)} x_t + U^{(f)} h_{t-1} + b^{(f)} \right),$$

$$o_t = \sigma \left(W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)} \right),$$

$$u_t = \tanh \left(W^{(u)} x_t + U^{(u)} h_{t-1} + b^{(u)} \right),$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1},$$

$$h_t = o_t \odot \tanh(c_t),$$

Original LSTM

$$i_j = \sigma \left(W^{(i)} x_j + \sum_{\ell=1}^N U_{\ell}^{(i)} h_{j\ell} + b^{(i)} \right), \quad (9)$$

$$f_{jk} = \sigma \left(W^{(f)} x_j + \sum_{\ell=1}^N U_{k\ell}^{(f)} h_{j\ell} + b^{(f)} \right), \quad (10)$$

$$o_j = \sigma \left(W^{(o)} x_j + \sum_{\ell=1}^N U_{\ell}^{(o)} h_{j\ell} + b^{(o)} \right), \quad (11)$$

$$u_j = \tanh \left(W^{(u)} x_j + \sum_{\ell=1}^N U_{\ell}^{(u)} h_{j\ell} + b^{(u)} \right), \quad (12)$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^N f_{j\ell} \odot c_{j\ell}, \quad (13)$$

$$h_j = o_j \odot \tanh(c_j), \quad (14)$$

N-ary Tree-LSTMs

Parsing by Pre-trained LM



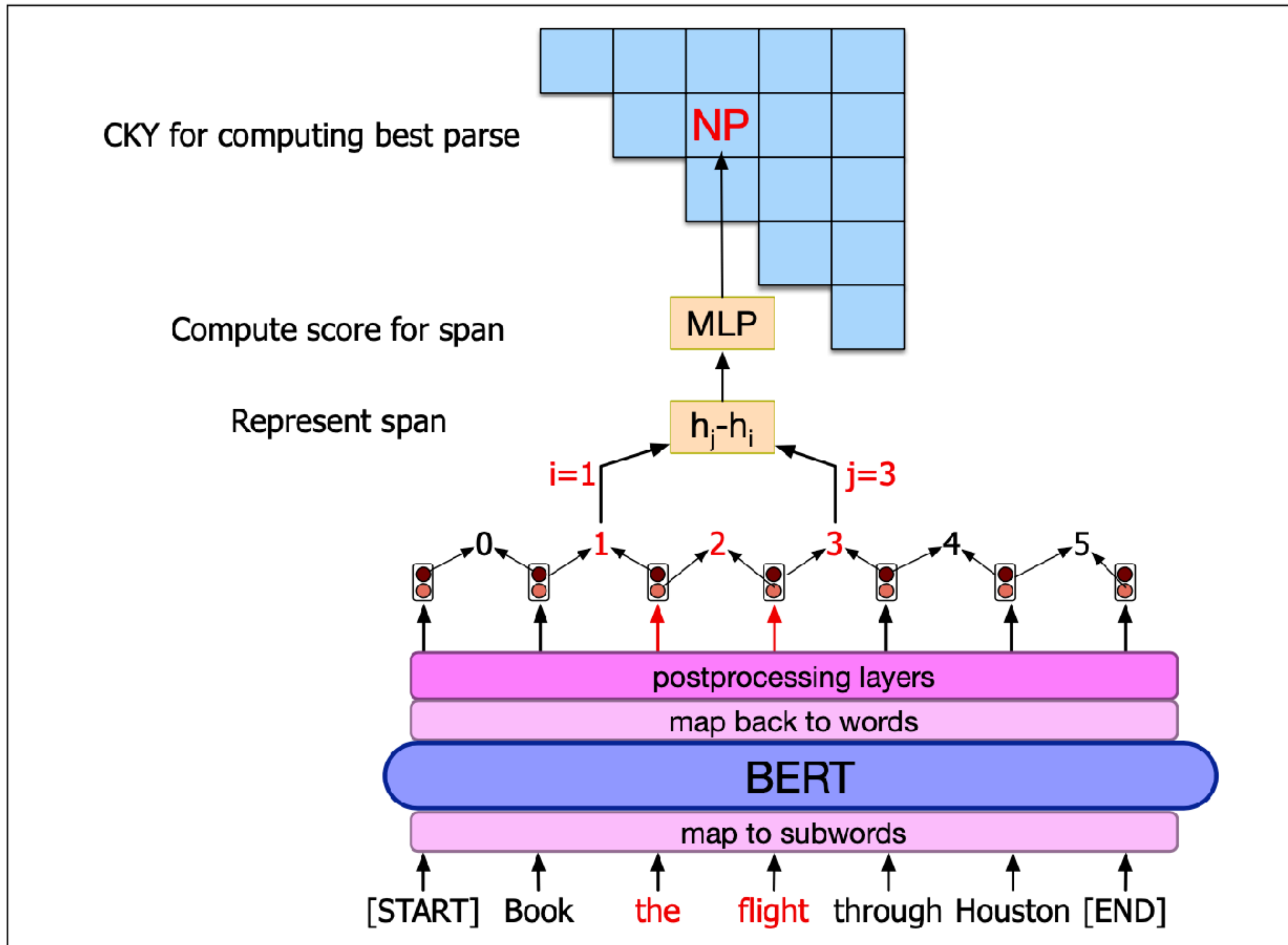
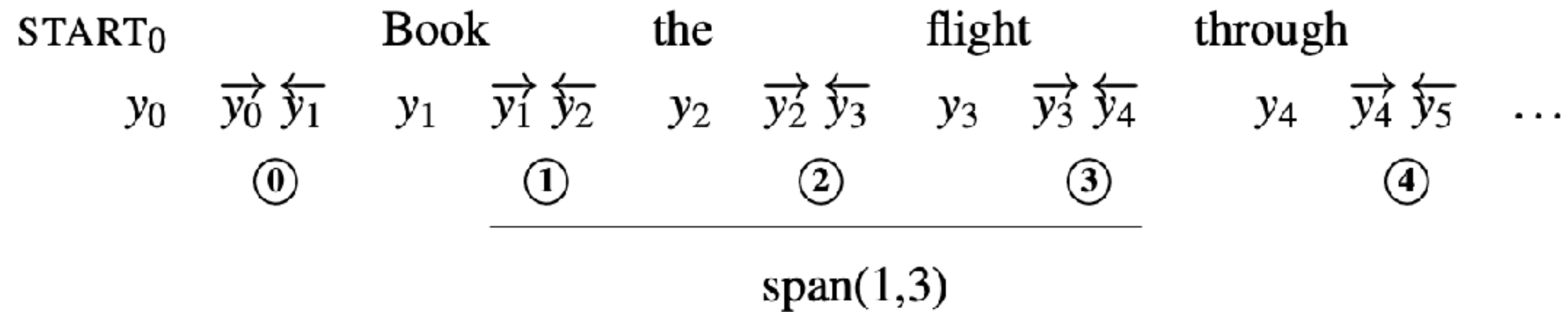


Figure 13.8 A simplified outline of computing the span score for the span *the flight* with the label NP.

83 Calculating Span Score

- Map each word encoding to two span encodings



- Represent a span by the difference between its start and end fenceposts

$$v(i, j) = [\vec{y}_j - \vec{y}_i ; \overleftarrow{y}_{j+1} - \overleftarrow{y}_{i+1}]$$

- Calculate span score by MLP

$$s(i, j, \cdot) = \mathbf{W}_2 \text{ReLU}(\text{LayerNorm}(\mathbf{W}_1 \mathbf{v}(i, j)))$$

84 Integrating Span Scores into a Parse

- A tree T is a set of spans

$$T = \{(i_t, j_t, l_t) : t = 1, \dots, |T|\}$$

- Score of T is the sum of its constituent spans

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l)$$

- Choose the final parse tree with the maximum score

$$\hat{T} = \operatorname{argmax}_T s(T)$$

- A variant the CKY algorithm to find the full parse.

$$s_{\text{best}}(i, i+1) = \max_l s(i, i+1, l)$$

$$s_{\text{best}}(i, j) = \max_l s(i, j, l) + \max_k [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)]$$

85 More details on span-based parsing

- Stern, M., Andreas, J., and Klein, D. (2017). A minimal span-based neural constituency parser. ACL.
- Gaddy, D., Stern, M., and Klein, D. (2018). What's going on in neural constituency parsers? an analysis. NAACL HLT
- Kitaev, N. and Klein, D. (2018). Constituency parsing with a self-attentive encoder. ACL.
- Kitaev, N., Cao, S., and Klein, D. (2019). Multilingual constituency parsing with self-attention and pre-training. ACL.

86 Todo

- **Reading Assignment 9:** [Speech and Language Processing \(3rd ed. draft\)](#), Dan Jurafsky and James H. Martin, [Chapter 12: Constituency Grammars](#) **Due:** April 10 23:59 pm, 2022
- **Suggested Readings:**
 - [Constituency Parsing with a Self-Attentive Encoder](#)
 - [Chapter 13: Constituency Parsing](#)
 - The papers listed on the page “**More details on span-based parsing**”

1. **Stanford CS224N, Winter 2019:** <https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture18-TreeRNNs.pdf>
2. **UIUC CS447, Fall 2018:** <https://courses.engr.illinois.edu/cs447/fa2018/Slides/Lecture09.pdf>
3. **UIUC CS447, Fall 2018:** <https://courses.engr.illinois.edu/cs447/fa2018/Slides/Lecture08.pdf>
4. **Speech and Language Processing (3rd ed. draft)**, Dan Jurafsky and James H. Martin, [Chapter 13: Constituency Parsing](#)
5. **Speech and Language Processing (3rd ed. draft)**, Dan Jurafsky and James H. Martin, [Chapter 12: Constituency Grammar](#)

Thanks! Q&A

Bang Liu

Email: bang.liu@umontreal.ca

Homepage: <http://www-labs.iro.umontreal.ca/~liubang/>