THE OHIO STATE UNIVERSITY

CSE 5525: Foundations of Speech and Language Processing

Syntax I

Huan Sun (CSE@OSU)

Many thanks to Prof. Greg Durrett @ UT Austin for sharing his slides.
Some slides adapted from Dan Klein, UC Berkeley

This Lecture

- Constituency formalism
- Context-free grammars and the CKY algorithm
- Refining grammars
- Discriminative parsers

Syntax & Constituency

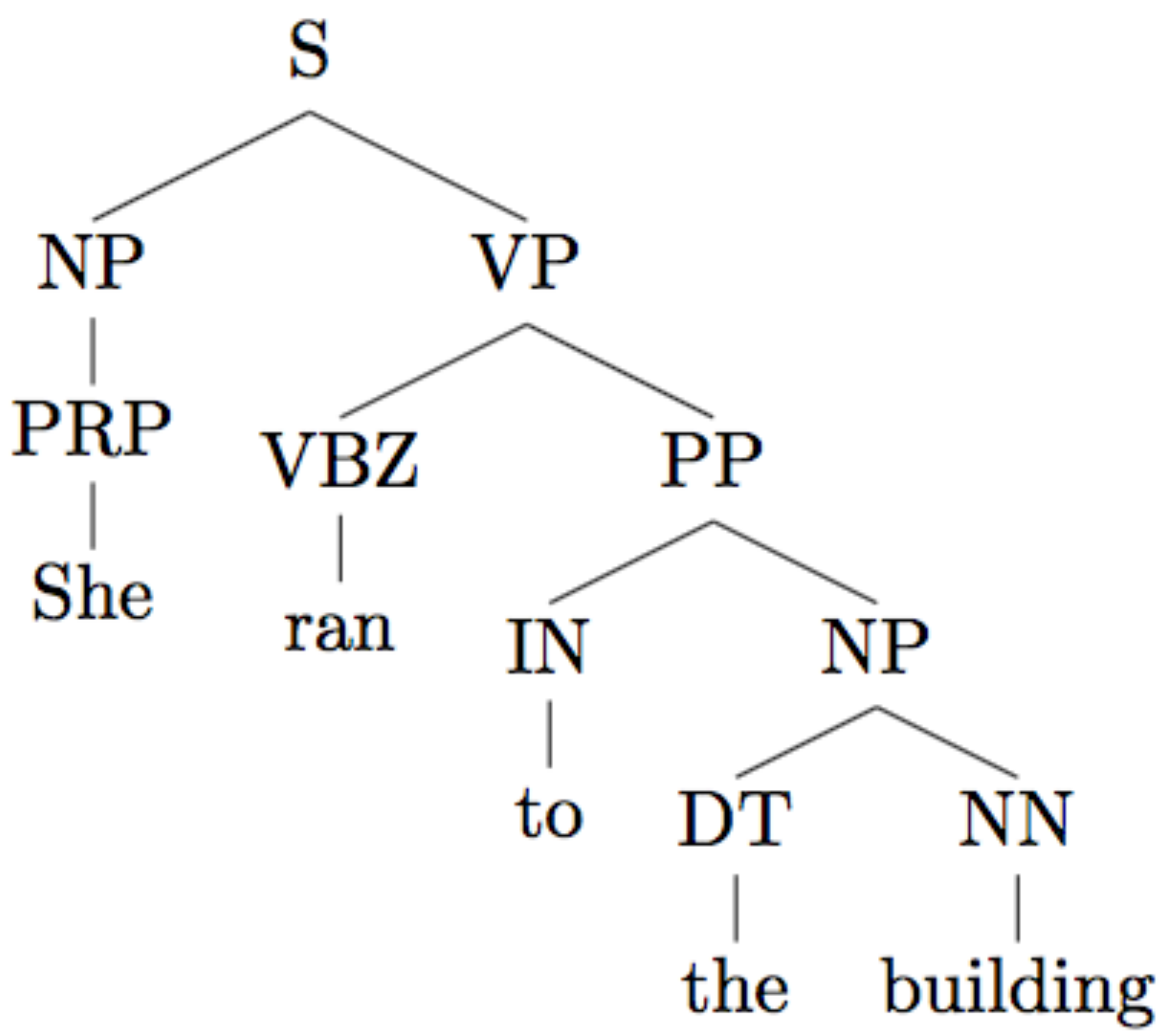
Syntax

- Study of word order and how words form sentences
- Why do we care about syntax? (useful for many tasks like phrase extraction)
  - Multiple interpretations of words (noun or verb?)
  - Recognize verb-argument structures (who is doing what to whom?)
  - Higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV, parsing can canonicalize

Constituency Parsing

- Tree-structured syntactic analyses of sentences

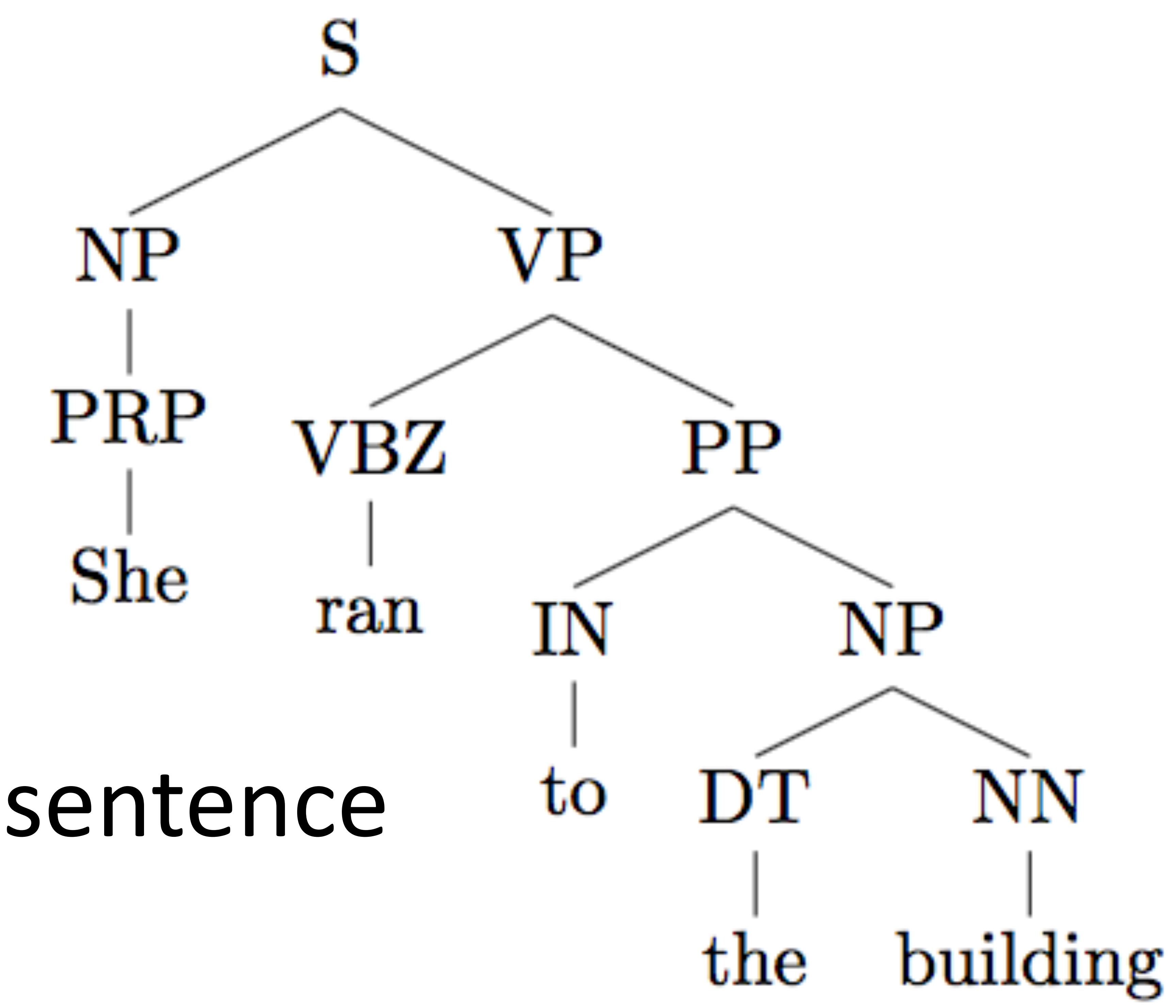
Constituency parsing is the task of breaking a text into sub-phrases, or constituents. Non-terminals in the parse tree are types of phrases, the terminals are the words in the sentence. —AllenNLP demo



Constituency Parsing

- Tree-structured syntactic analyses of sentences

Constituency parsing is the task of breaking a text into sub-phrases, or constituents. Non-terminals in the parse tree are types of phrases, the terminals are the words in the sentence. —AllenNLP demo



- Bottom layers are POS tags & words in the sentence

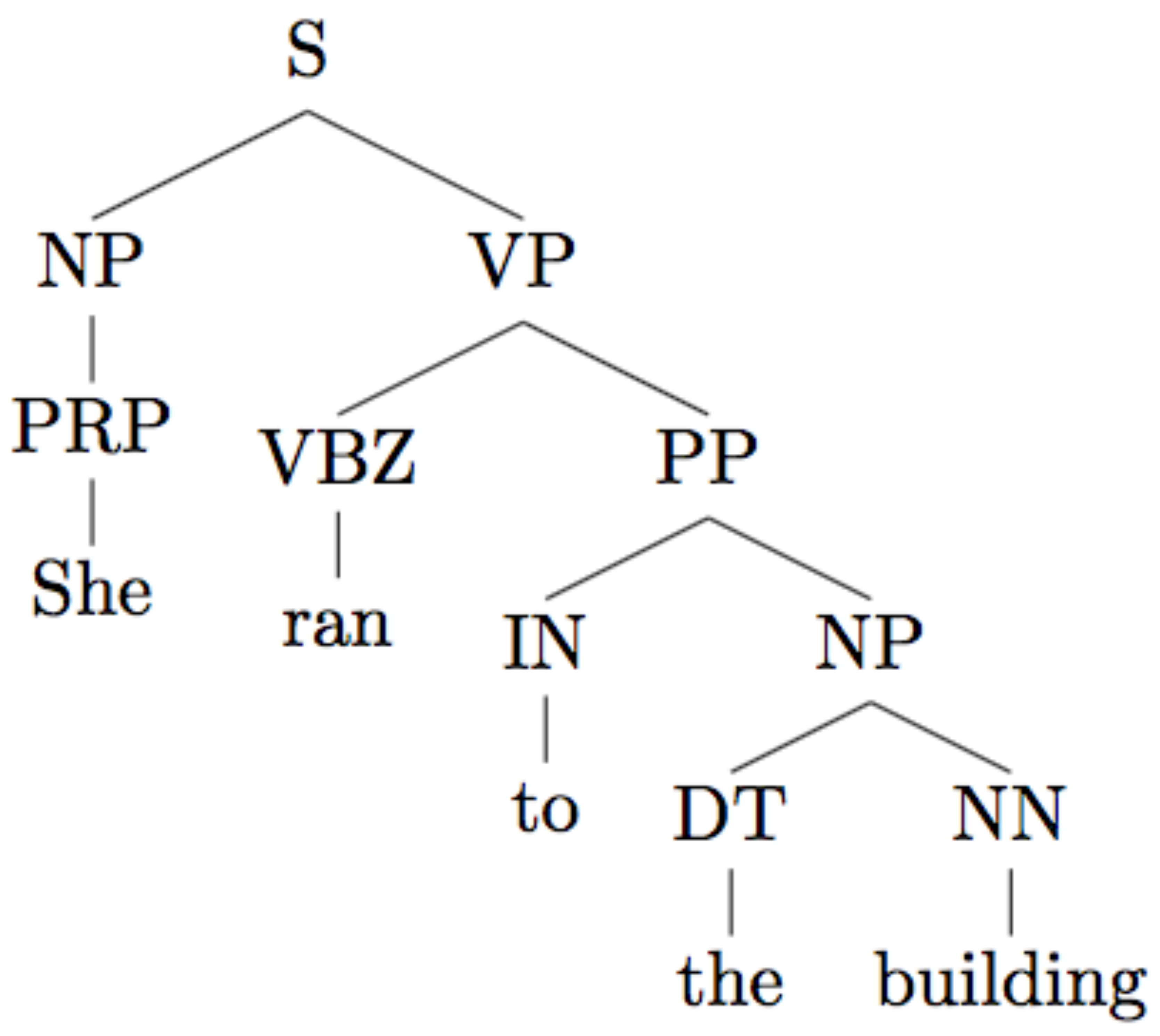
- Common things: noun phrases, verb phrases, prepositional phrases

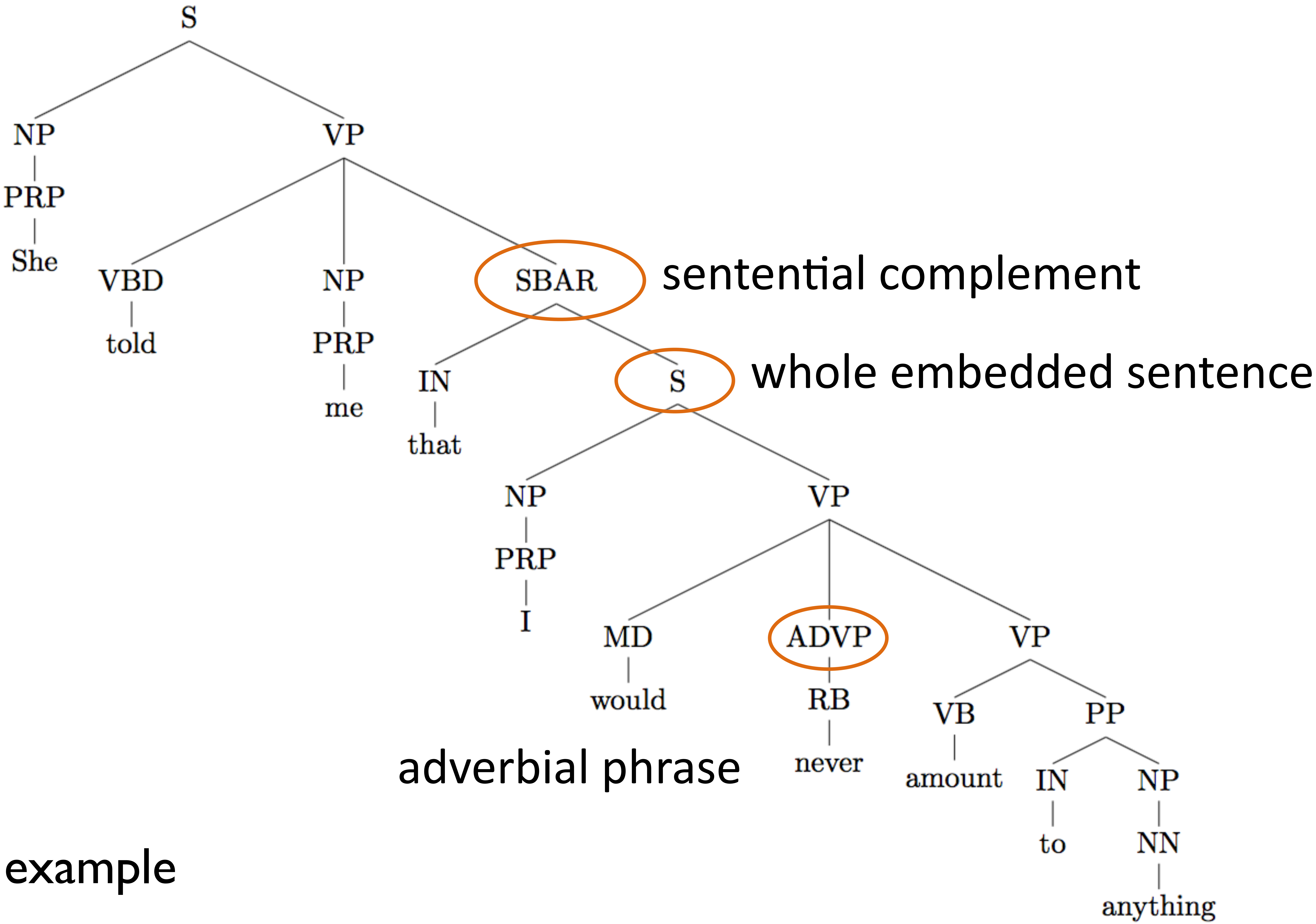
Constituency Parsing

- Tree-structured syntactic analyses of sentences

- Common things: noun phrases, verb phrases, prepositional phrases
- Bottom layers are POS tags & words in the sentence

- We will use English sentences as examples, but note that constituency makes sense for a lot of languages

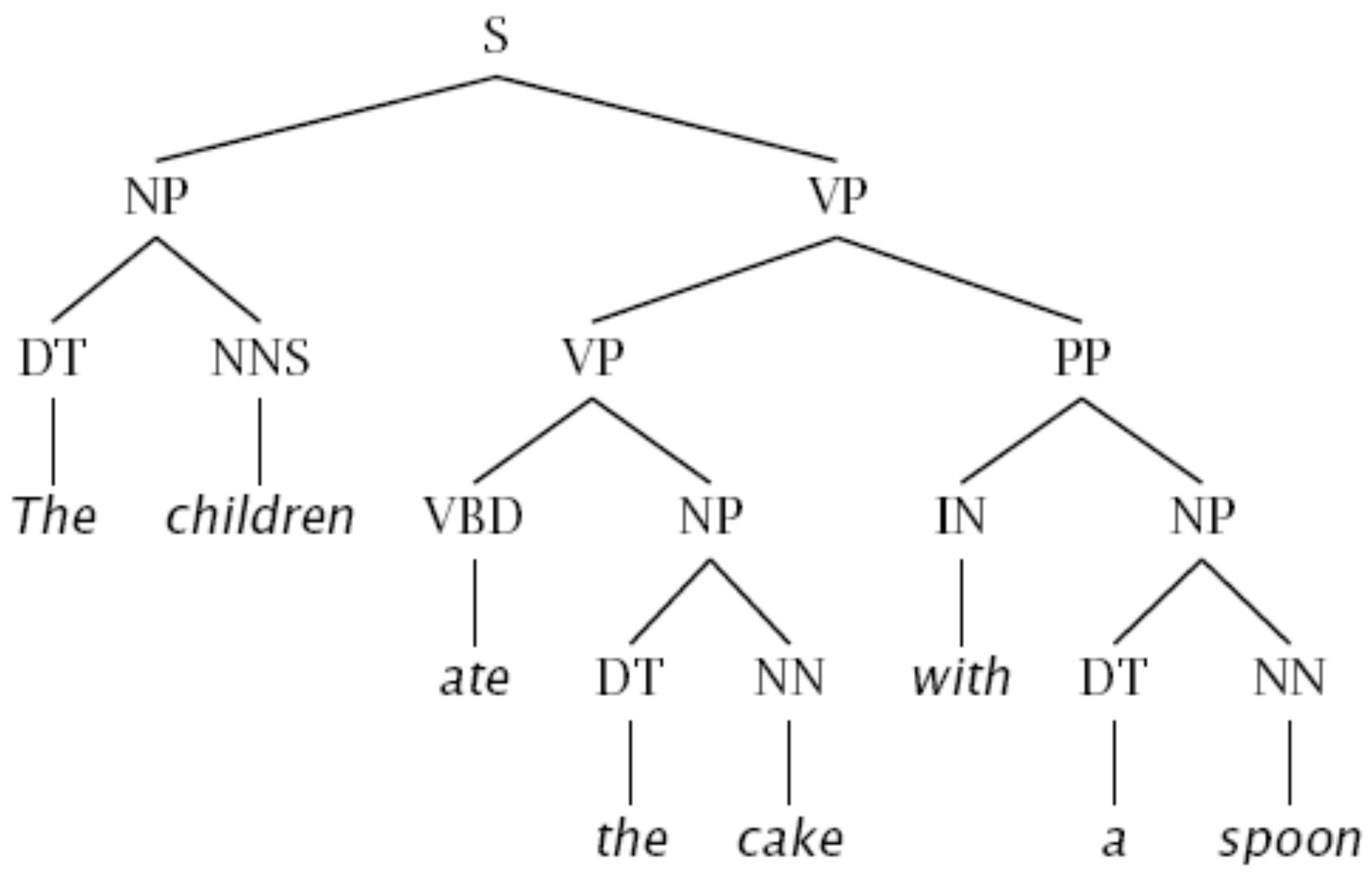




Another example

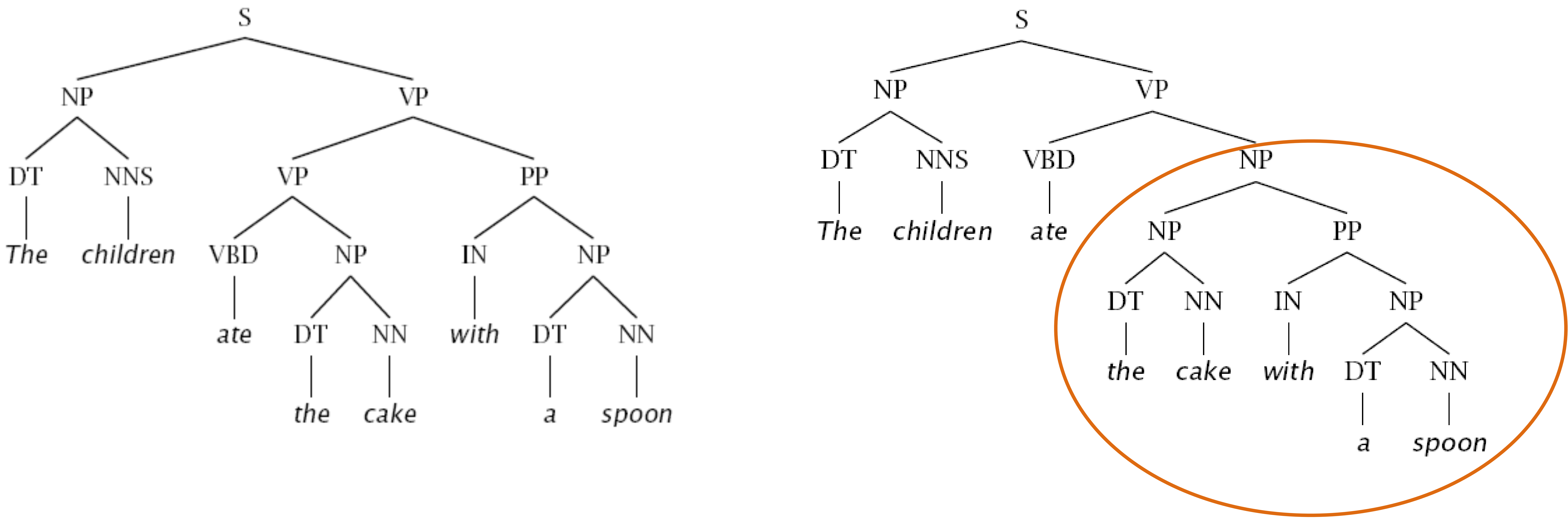
Challenges

- PP (Prepositional Phrase) attachment ambiguity



Challenges

- PP (Prepositional Phrase) attachment ambiguity

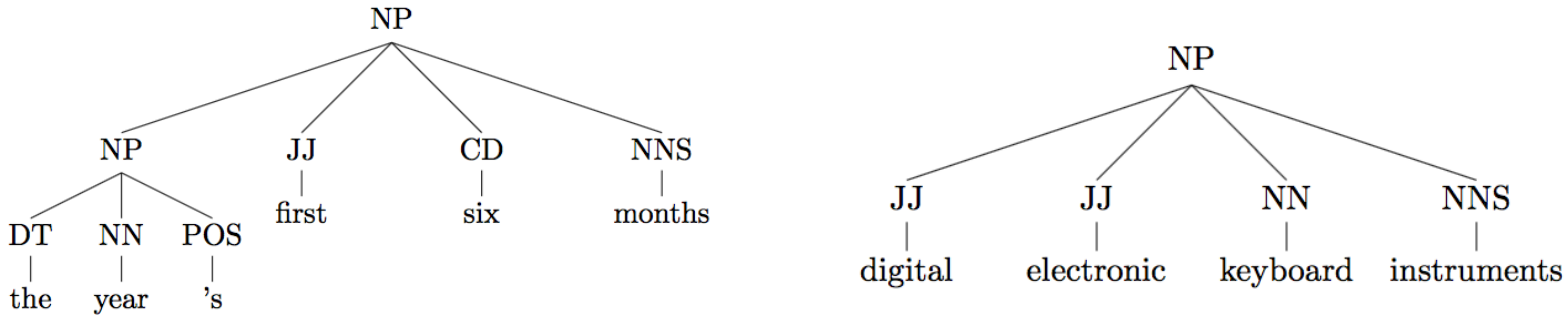


same parse as “the cake with some icing”

Challenges

- NP internal structure: tags + depth of analysis

There are a variety of components an NP can contain



Review tags, if needed: https://cs.nyu.edu/grishman/jet/guide/PennPOS.html

Coordination ambiguity

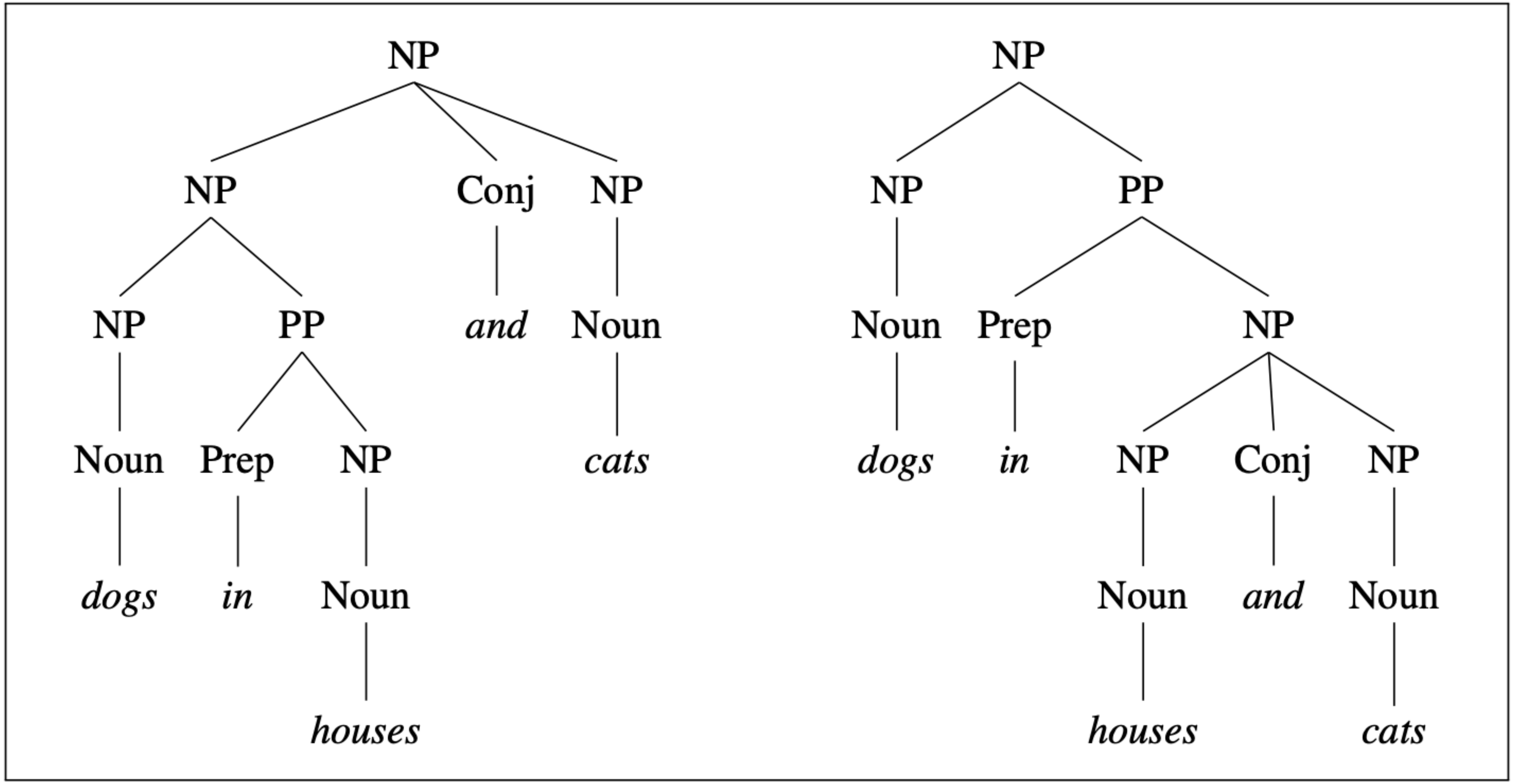
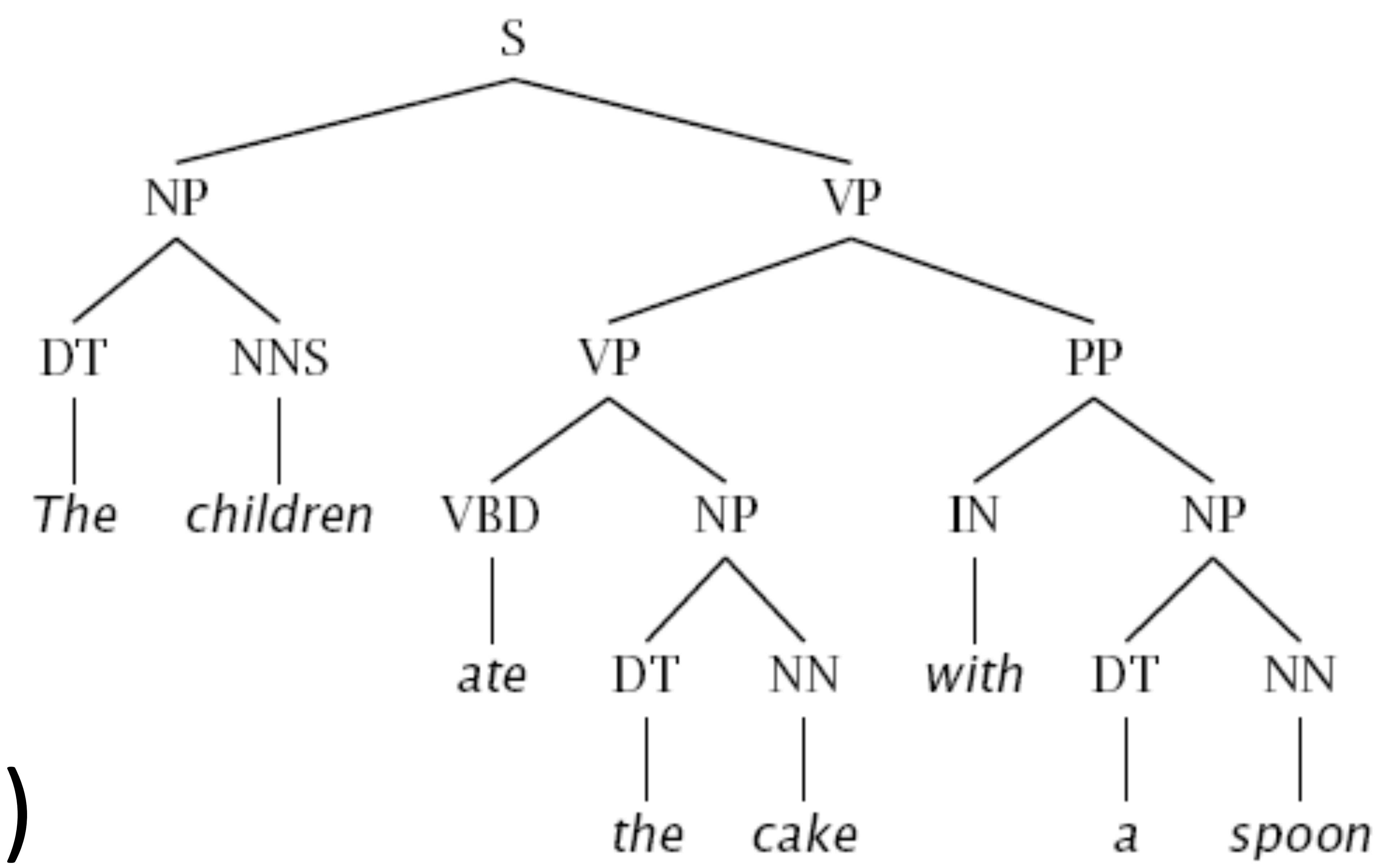


Figure 14.7 An instance of coordination ambiguity. Although the left structure is intuitively the correct one, a PCFG will assign them identical probabilities since both structures use exactly the same set of rules. After Collins (1999).

https://web.stanford.edu/~jurafsky/slp3/14.pdf

Constituency

- How do we humans know what the constituents are?
- "Constituency tests":
  - Substitution by *pro-form* (e.g., pronoun)
  - Clefting (*with a spoon is how the children …*)
  - Answer ellipsis (What did they eat? *the cake*)
    (How? *with a spoon*)



https://people.umass.edu/nconstan/201/Constituency%20Tests.pdf

Context-Free Grammars

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

Also called Phrase-Structure Grammars

Chapter 12.2 in textbook JM: https://web.stanford.edu/~jurafsky/slp3/12.pdf

CFGs

Rules (or, productions)		Lexicon
ROOT $\rightarrow$ S	NP $\rightarrow$ NP PP	NN $\rightarrow$ interest
S $\rightarrow$ NP VP	VP $\rightarrow$ VBP NP	NNS $\rightarrow$ raises
NP $\rightarrow$ DT NN	VP $\rightarrow$ VBP NP PP	VBP $\rightarrow$ interest
NP $\rightarrow$ NN NNS	PP $\rightarrow$ IN NP	VBZ $\rightarrow$ raises

- Rules/productions: symbols which rewrite as one or more symbols
  - each rule expresses the ways that symbols can be grouped and ordered together

- Lexicon consists of “preterminals” (POS tags) rewriting as terminals (words)

CFGs

Rules/productions		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises

The following rules express that an **NP** (or **noun phrase**) can be composed of either a *ProperNoun* or a determiner (*Det*) followed by a *Nominal*; a *Nominal* in turn can consist of one or more *Noun*s.

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun \mid Nominal\ Noun \end{aligned}$$

Context-free rules can be hierarchically embedded, so we can combine the previous rules with others, like the following, that express facts about the lexicon:

Example

$$\begin{aligned} Det &\rightarrow a \\ Det &\rightarrow the \\ Noun &\rightarrow flight \end{aligned}$$

CFGs

Rules/productions		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises

- Rules/productions: symbols which rewrite as one or more symbols
- Lexicon consists of “preterminals” (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
  - Terminals: words in the language. Non-terminals: symbols that express abstractions over these terminals. What is the item to the left of the arrow?

CFGs

Rules/productions		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises

- Rules/productions: symbols which rewrite as one or more symbols
- Lexicon consists of "preterminals" (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
- Probabilistic CFG (PCFG): conditional probabilities associated with rules

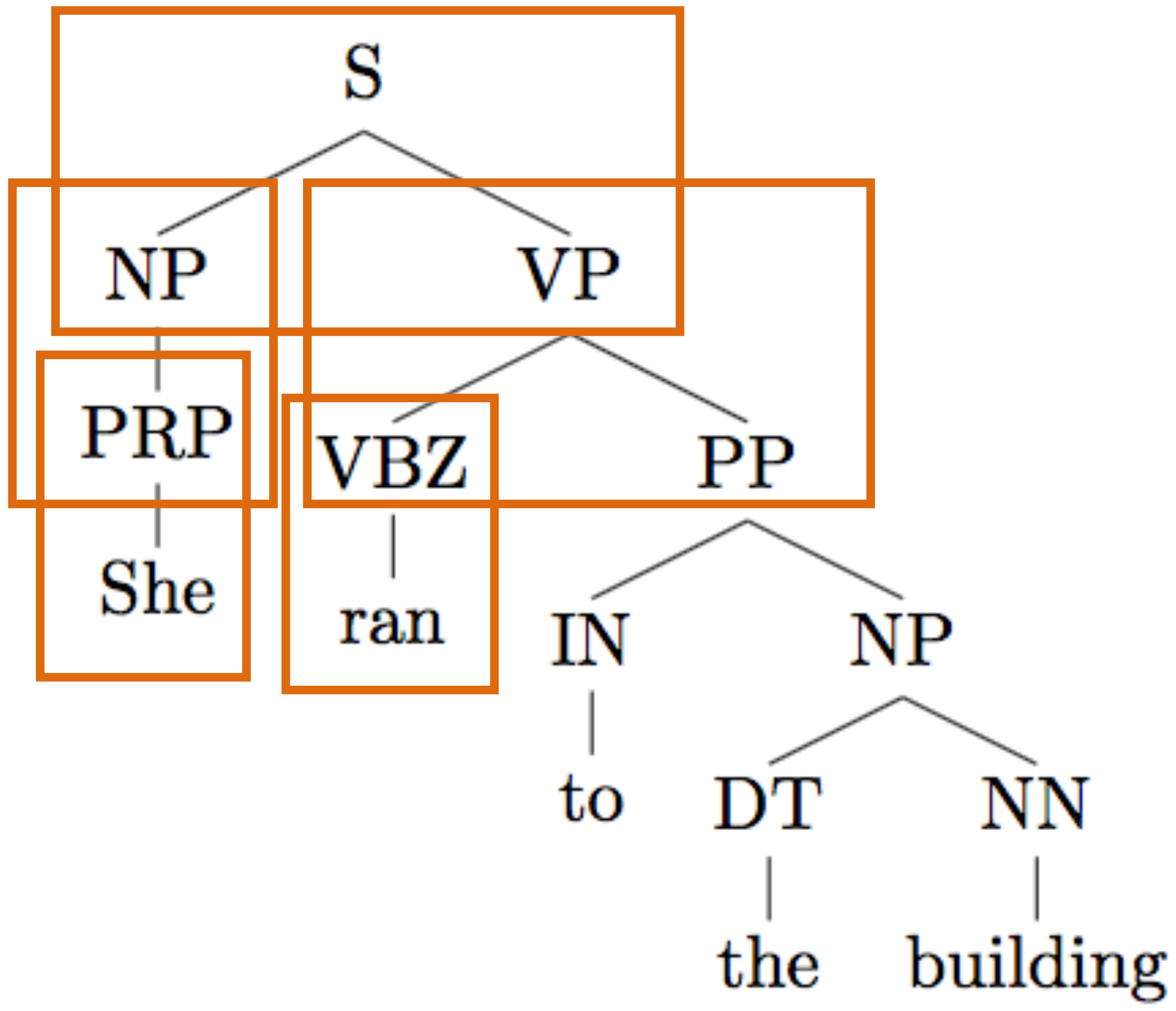
CFGs

Rules/productions				Lexicon	
ROOT → S	1.0	NP → NP PP	0.3	NN → interest	1.0
S → NP VP	1.0	VP → VBP NP	0.7	NNS → raises	1.0
NP → DT NN	0.2	VP → VBP NP PP	0.3	VBP → interest	1.0
NP → NN NNS	0.5	PP → IN NP	1.0	VBZ → raises	1.0

- Rules/productions: symbols which rewrite as one or more symbols
- Lexicon consists of "preterminals" (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
- PCFG: conditional probabilities associated with rules (num. are made up)

Estimating PCFGs

- Tree *T* is a series of rule applications *r*. (Each rule expands a non-terminal node.)



T includes:

S → NP VP

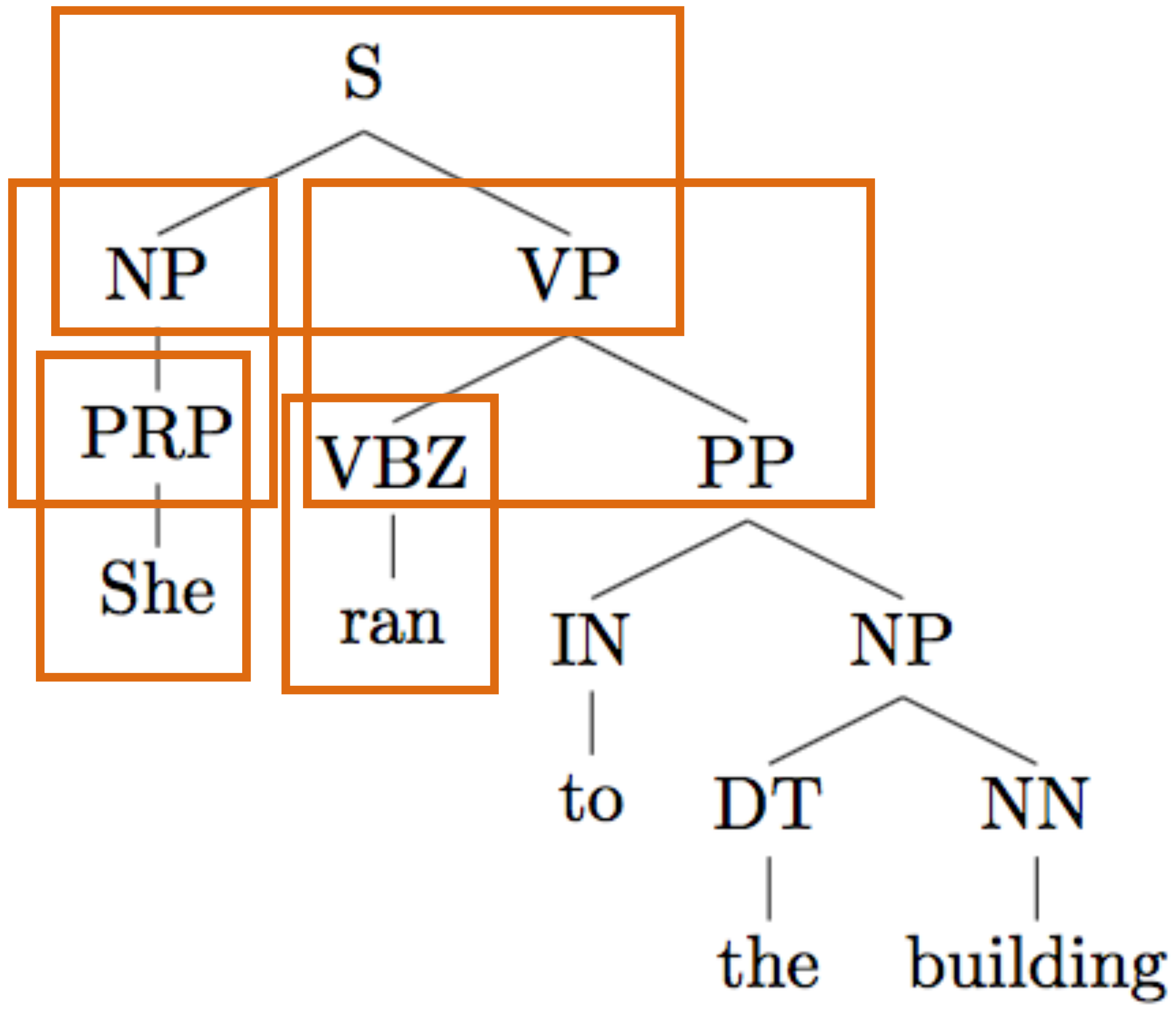
NP → PRP

NP → DT NN

…

Estimating PCFGs

- Tree *T* is a series of rule applications *r*. $P(T) = \prod_{r \in T} P(r|\text{parent}(r))$



T includes:

S → NP VP

NP → PRP

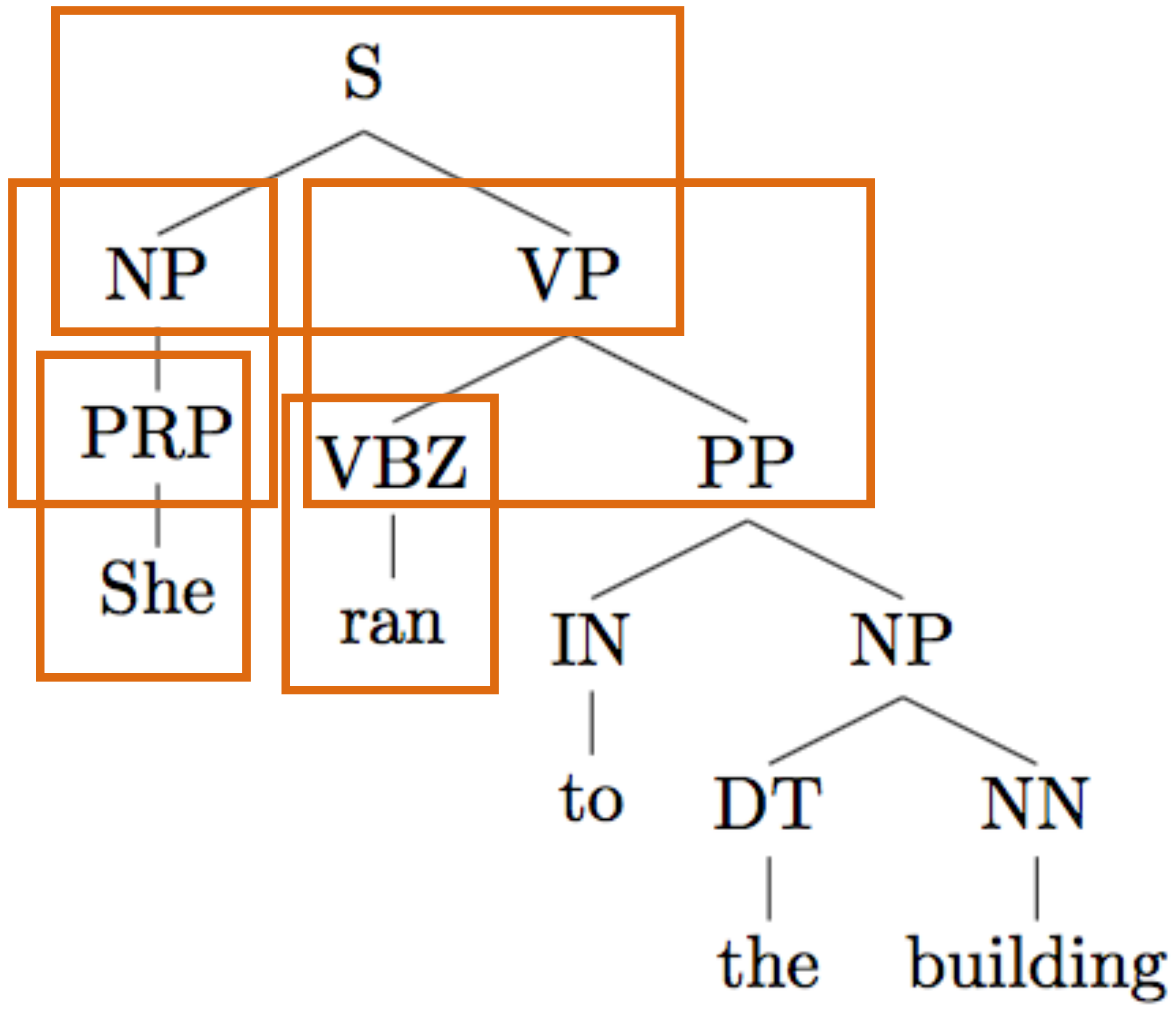
NP → DT NN

...

The probability of a particular parse tree T is defined as the product of the probabilities of all the rules used to expand each of the non-terminal nodes in the parse tree T

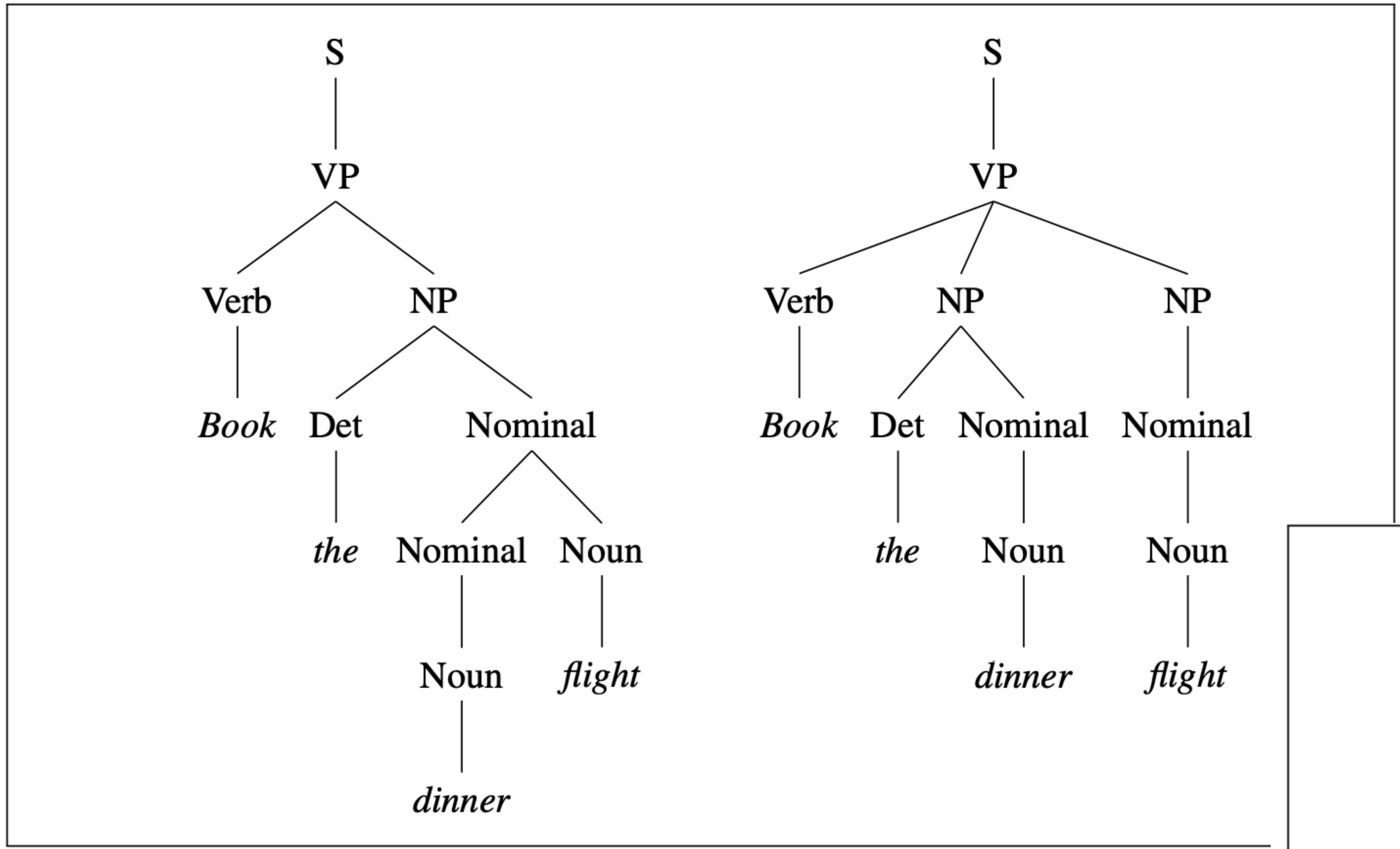
Estimating PCFGs

- Tree T is a series of rule applications r. $P(T) = \prod_{r \in T} P(r | \text{parent}(r))$



S → NP VP	1.0
NP → PRP	0.5
NP → DT NN	0.5
…	

- Maximum likelihood PCFG: count and normalize! Same as HMMs / Naive Bayes



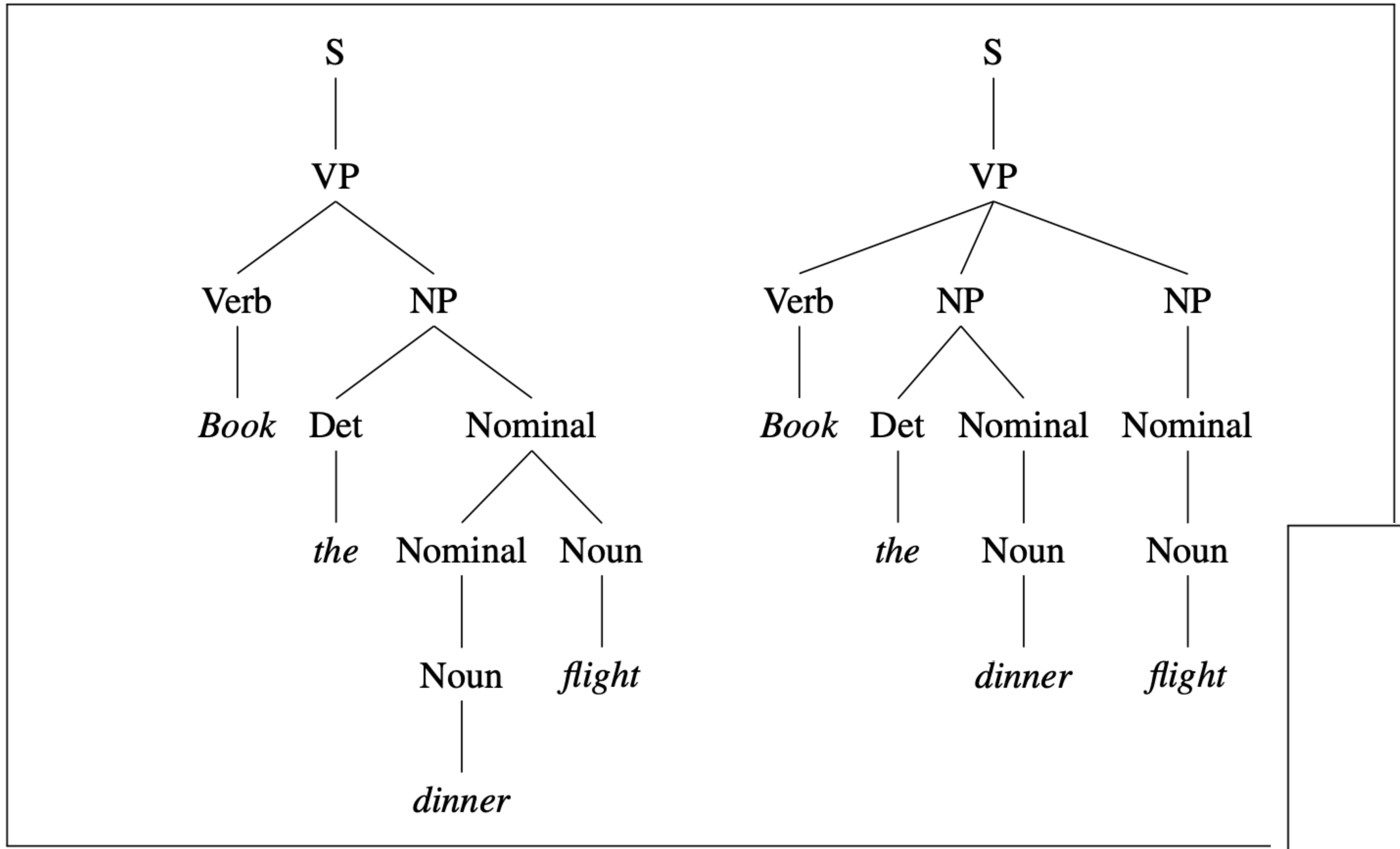
Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Figure 14.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' ".

$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$

$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

How to produce the most-likely parse T?



Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Figure 14.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' ".

$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$

$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

CKY

- Cocke-Kasami-Younger (CKY) algorithm, the most widely used dynamic-programming based approach to parsing

- Requires grammars used with it to be in Chomsky Normal Form (CNF)
  - i.e., rules of the form $A \rightarrow B\, C$ and $A \rightarrow w$
  - binary branching

CKY

- Cocke-Kasami-Younger (CKY) algorithm, the most widely used dynamic-programming based approach to parsing

- Requires grammars used with it to be in Chomsky Normal Form (CNF)
  - i.e., rules of the form $A \rightarrow B\,C$ and $A \rightarrow w$
  - binary branching

- A generic context-free grammar (CFG) can be converted to CNF

See more in Chapter13.2.1: https://web.stanford.edu/~jurafsky/slp3/13.pdf

CKY

- Requires grammars used with it to be in Chomsky Normal Form (CNF)
  - i.e., rules of the form $A \rightarrow B\ C$ and $A \rightarrow w$
  - binary branching

- Each non-terminal node will have two child nodes
- Use a two-dimensional matrix to encode the parse tree of a sentence
  - sentence length n -> (n+1) * (n+1)

Each cell [i, j] in this matrix contains the set of non-terminals that represent all the constituents that span position i through j of the input sentence

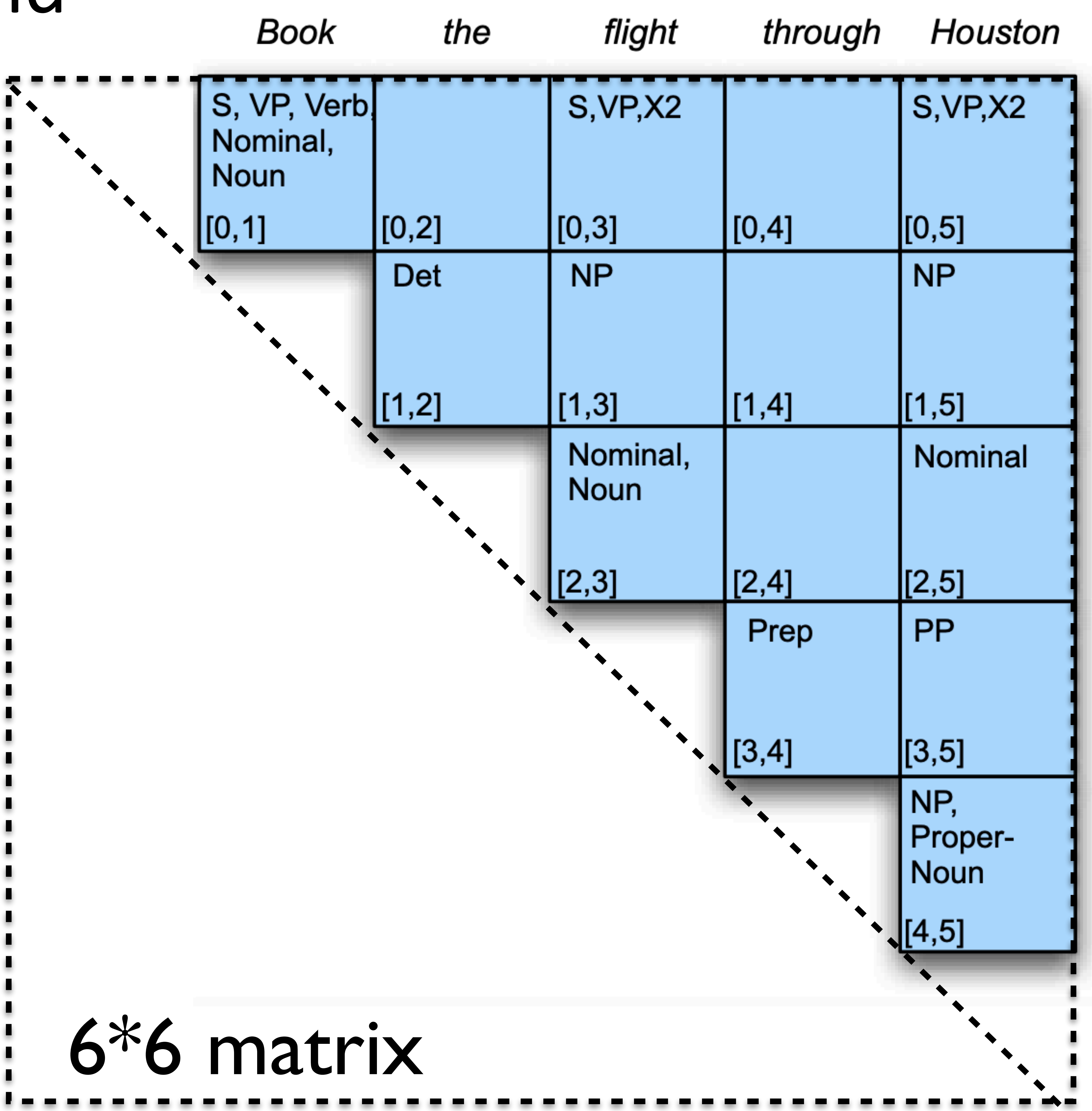
0 Book 1 the 2 flight through Houston. 5

CKY

- Requires grammars used with it to be in Chomsky Normal Form (CNF)
  - i.e., rules of the form $A \rightarrow B\ C$ and
  - binary branching

cell [i, j]:
all the constituents that span position i through j

0 1 2 5
Book the flight through Houston.

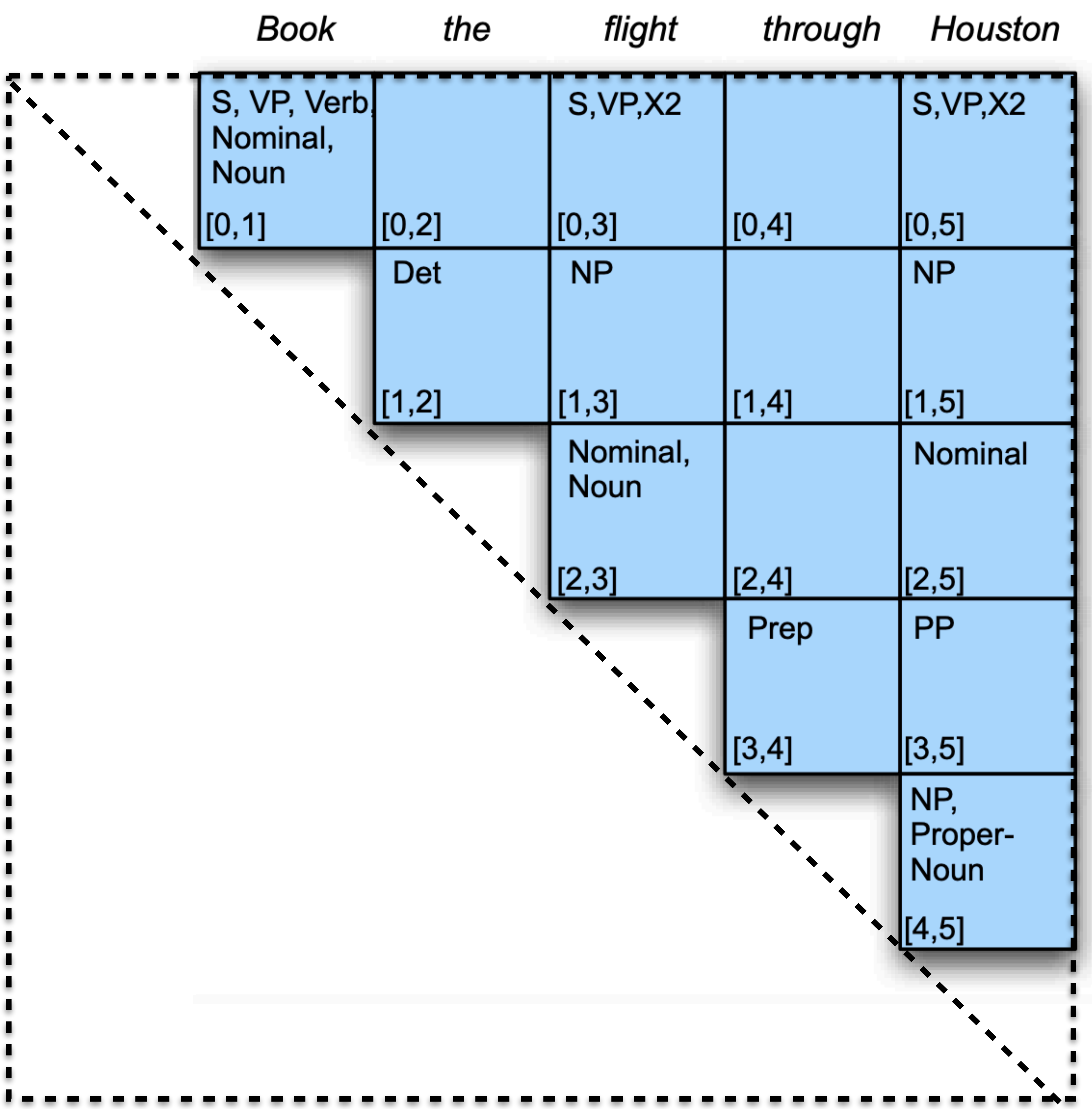


$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

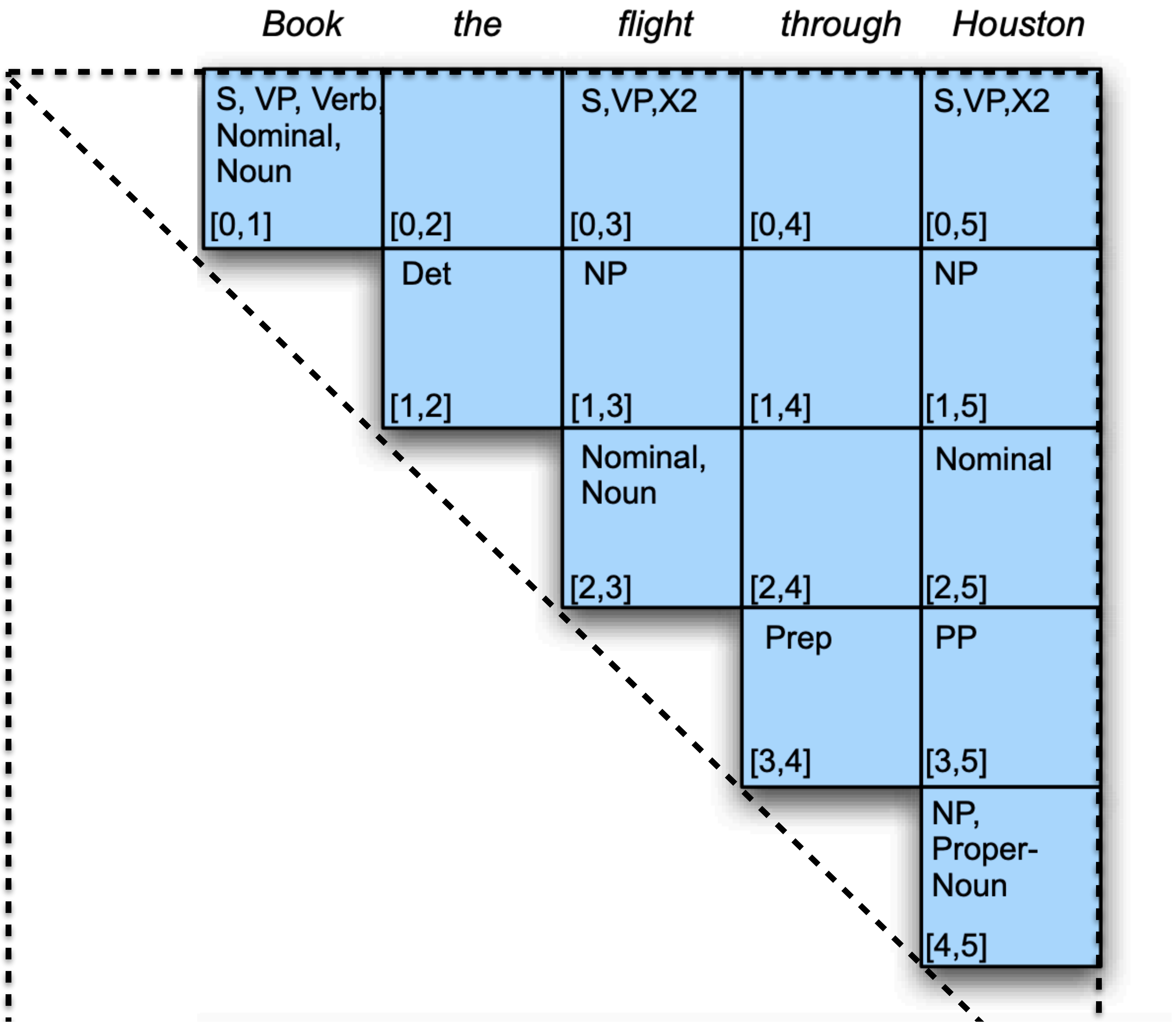
Given Grammar

0 1 2 5
Book the flight through Houston.



$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Given Grammar



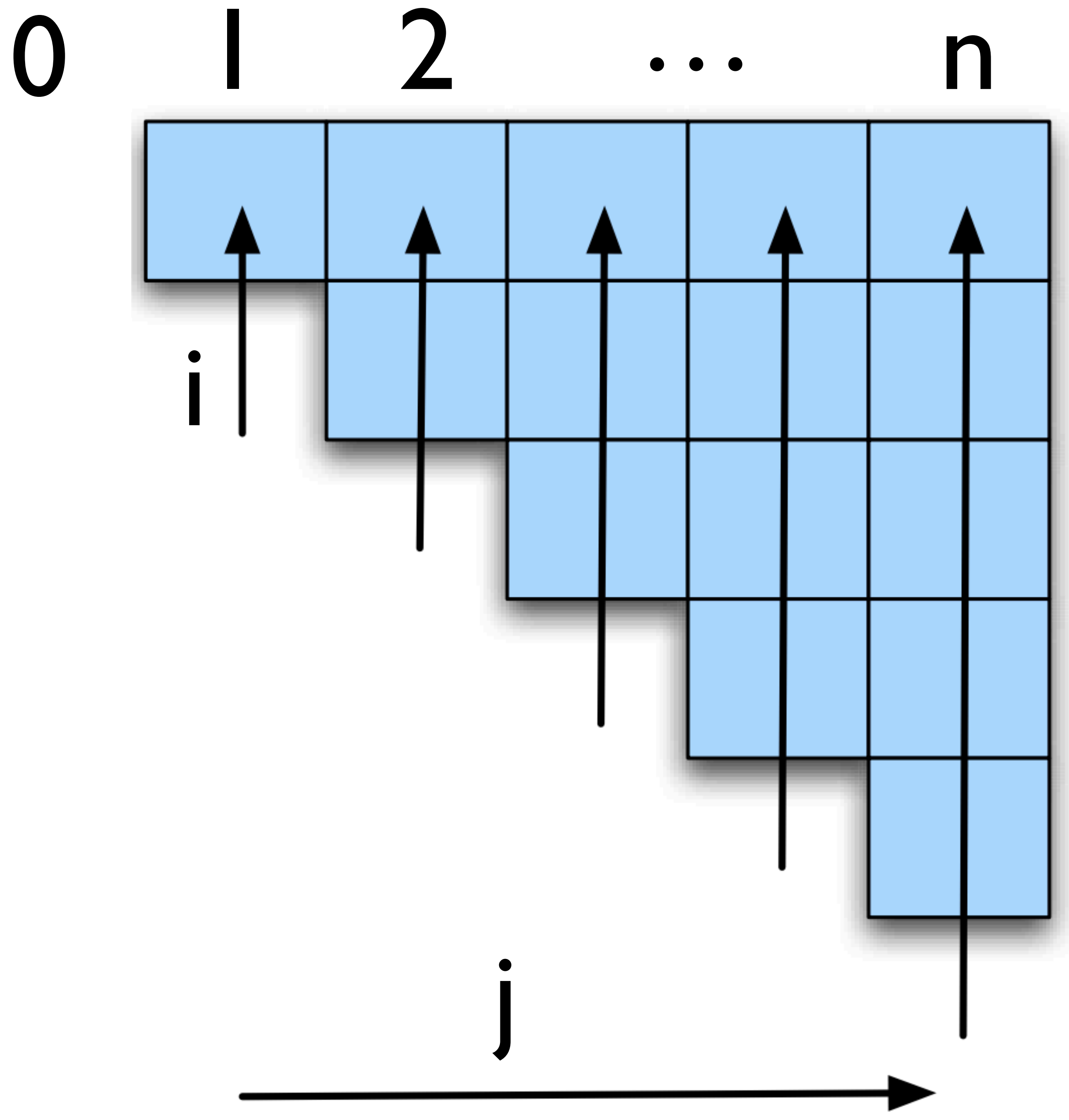
0 1 2 5

Book the flight through Houston.

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

Figure 13.5 The CKY algorithm.



function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 13.5 The CKY algorithm.

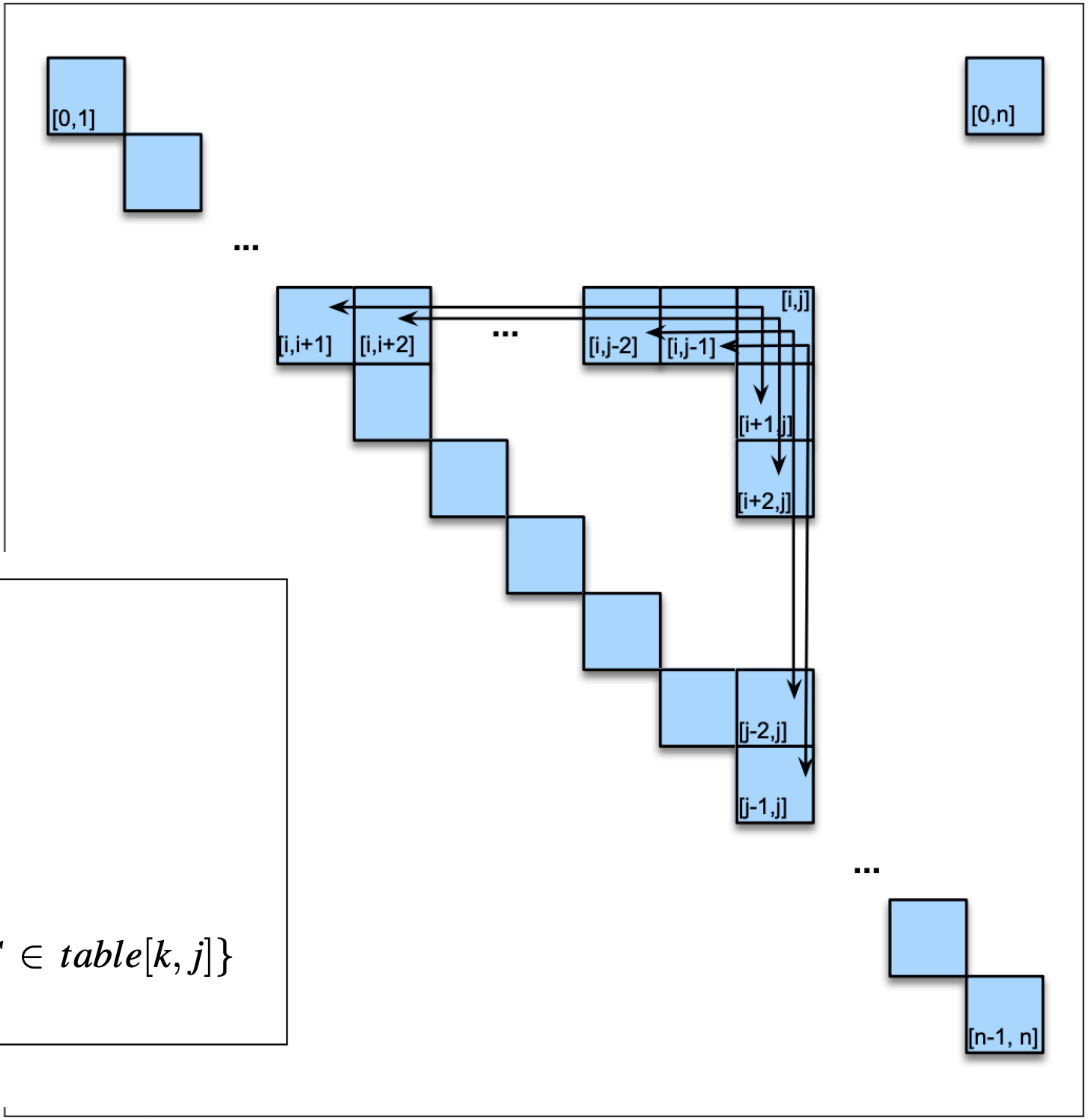


Figure 13.6 All the ways to fill the $[i, j]$th cell in the CKY table.

Example of filling the cells in column 5 (j=5):

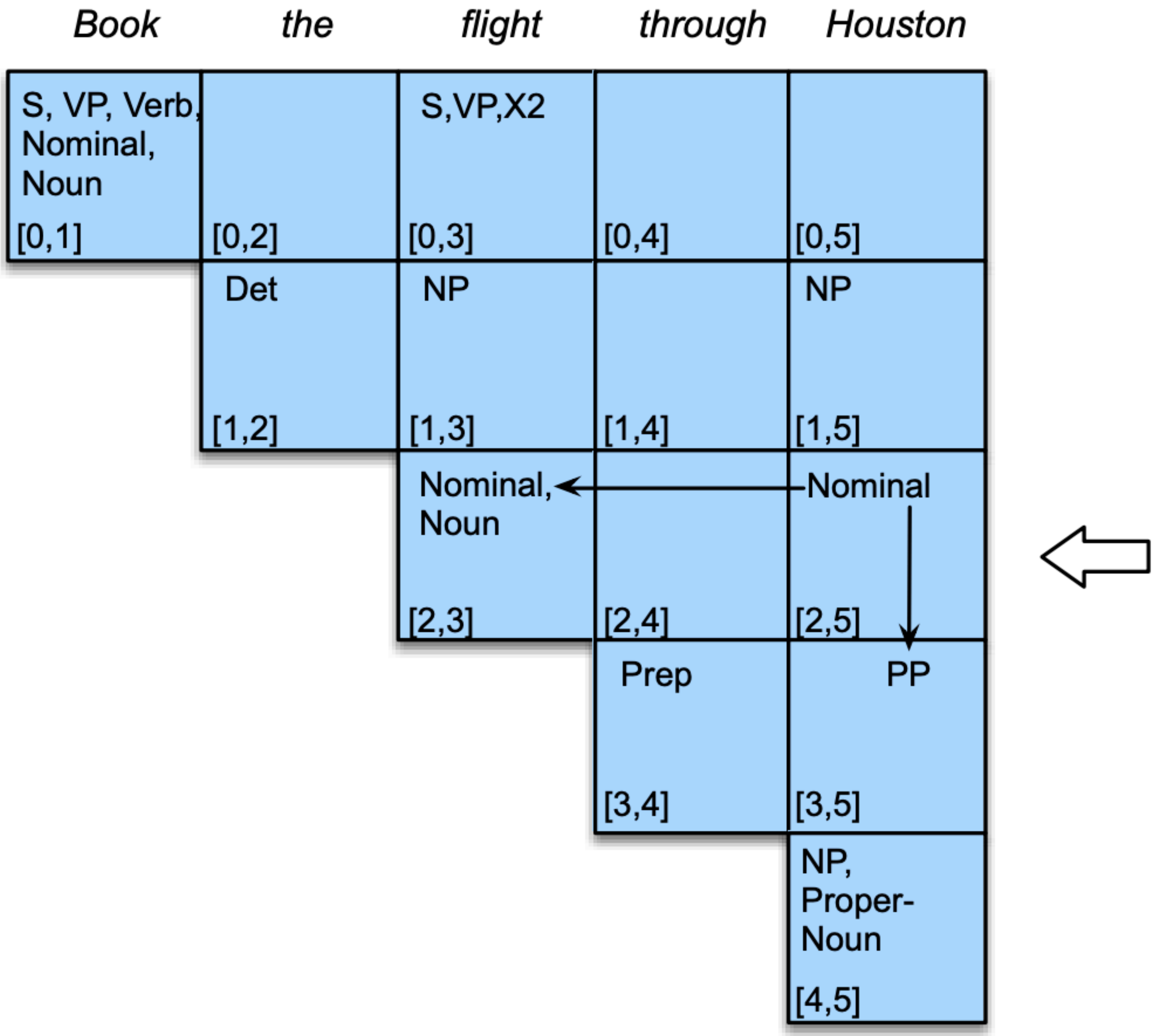
Book	*the*	*flight*	*through*	*Houston*
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	[3,5]
				NP, Proper-Noun [4,5]

table[j-1, j]

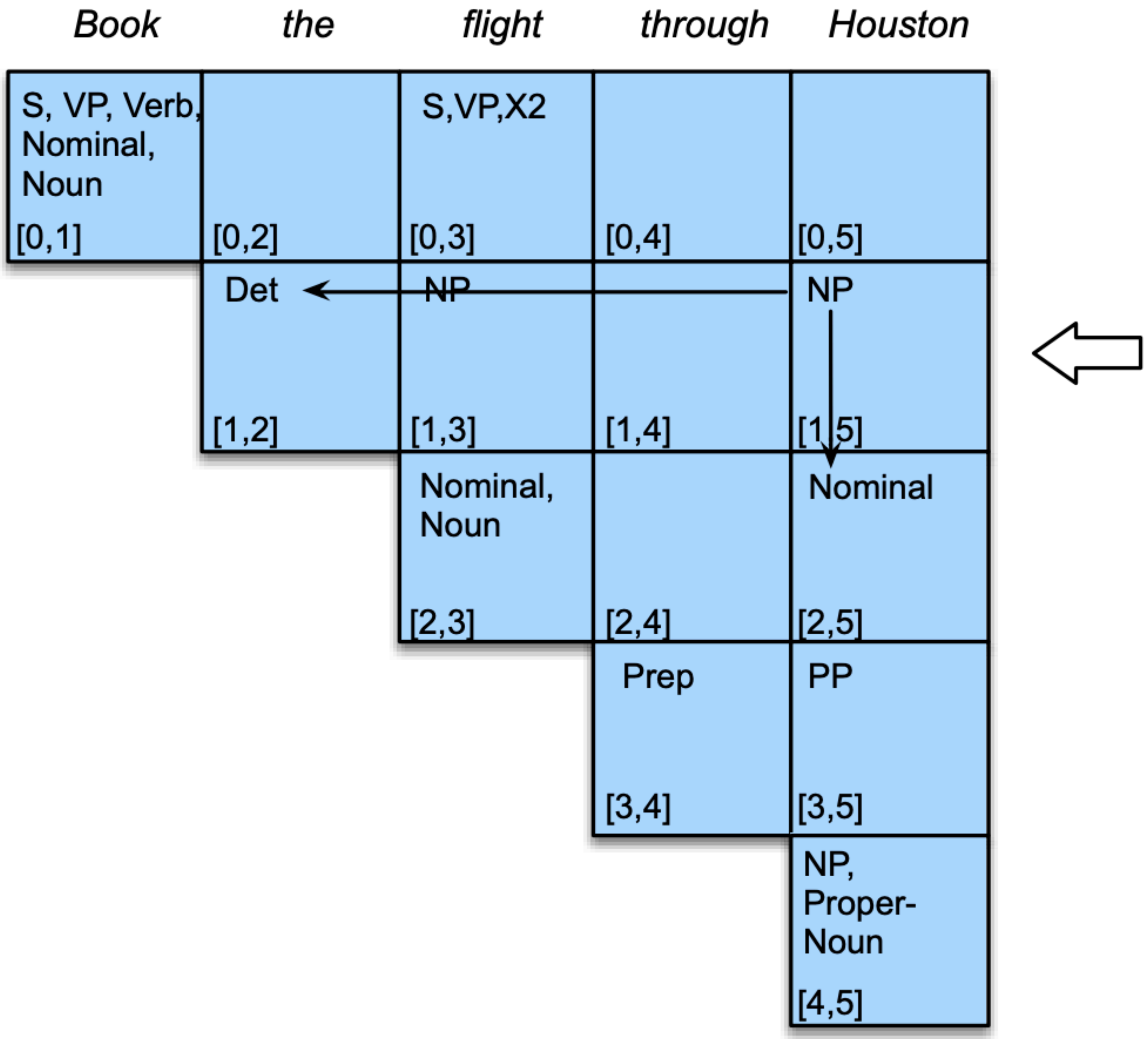
Book	*the*	*flight*	*through*	*Houston*
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	[2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

i=3

Example of filling the cells in column 5:

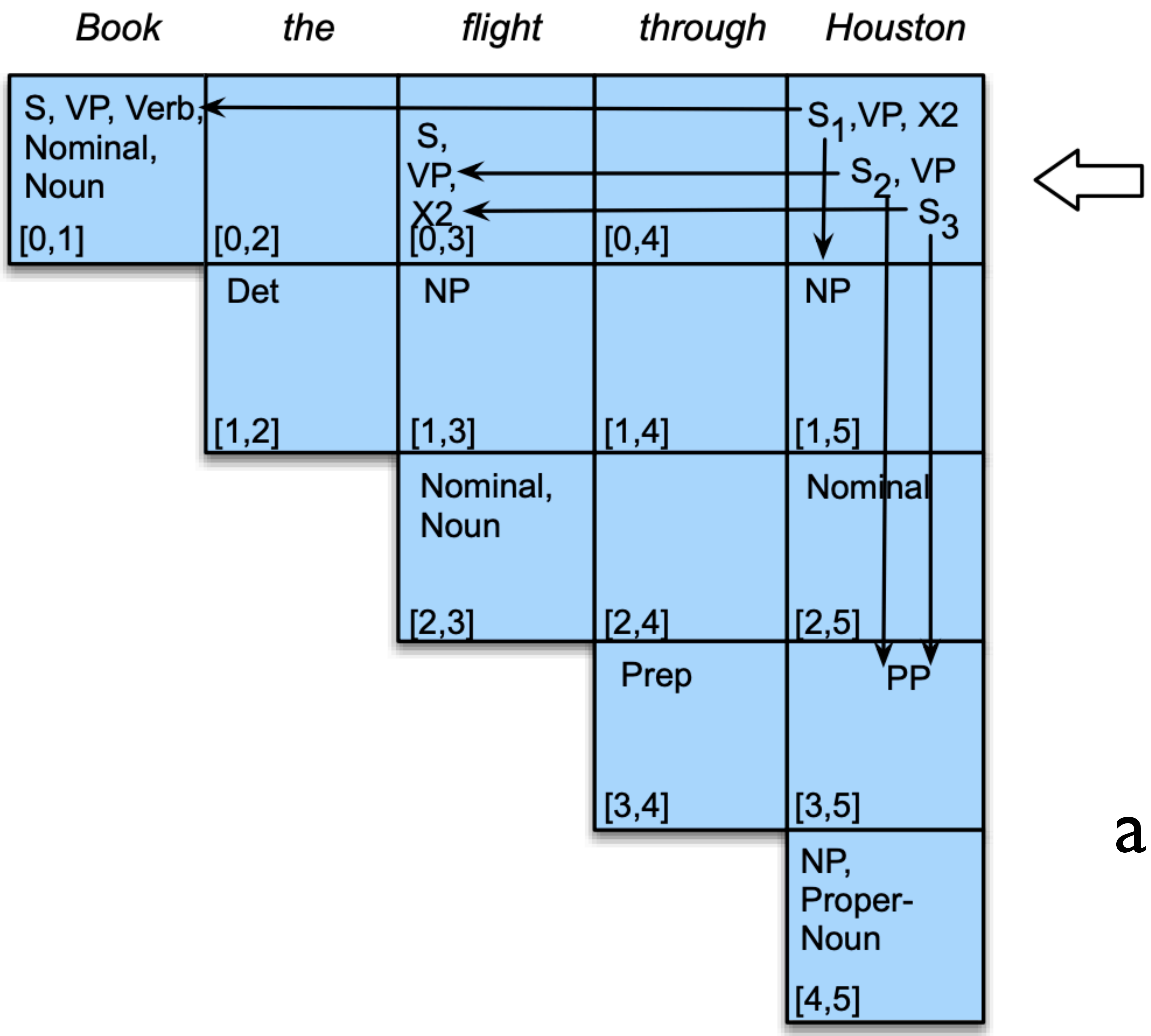


i=2



i=1

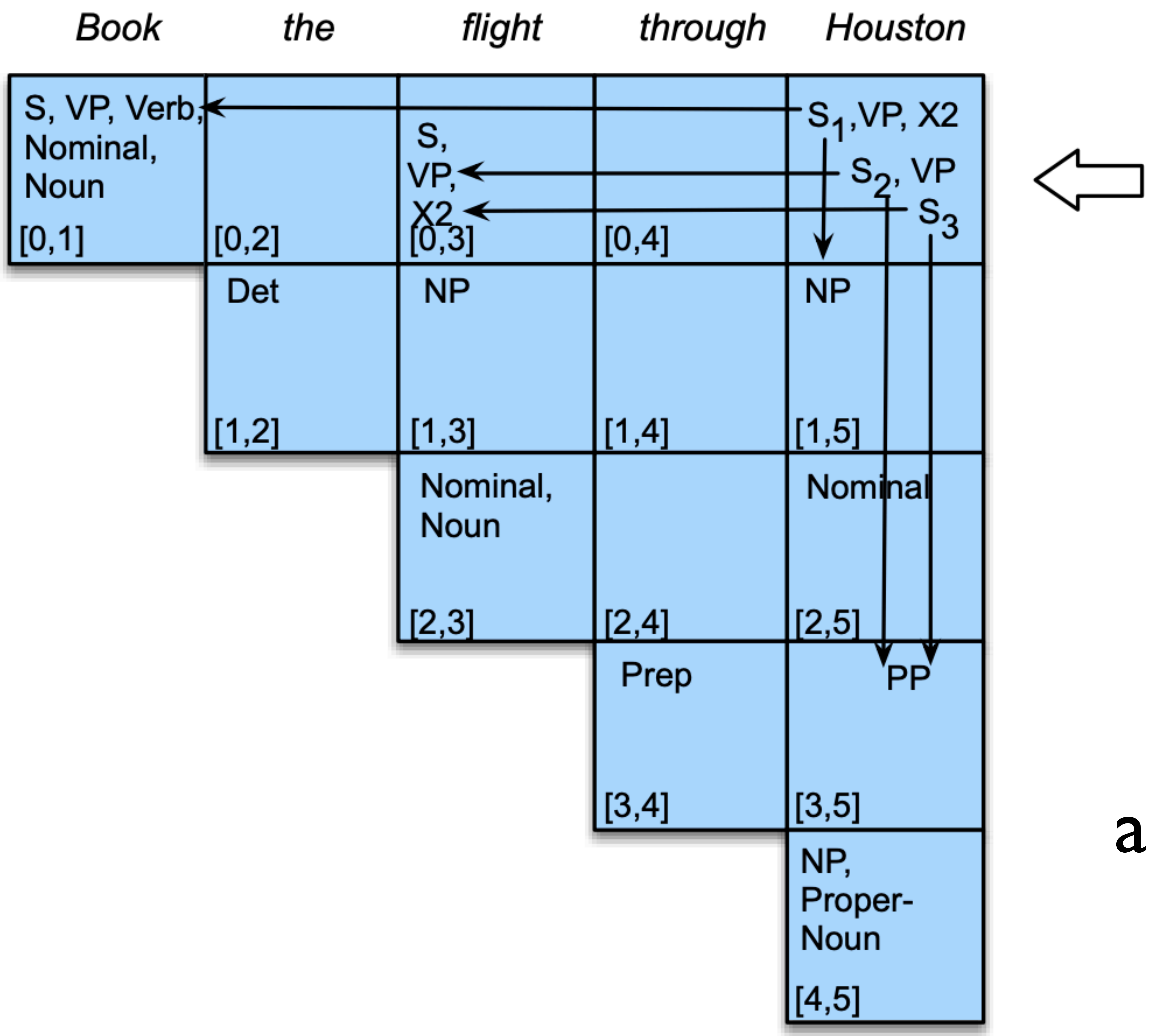
Example of filling the cells in column 5:



all parse trees can be obtained

i=0

Example of filling the cells in column 5:



i=0

all parse trees can be obtained

How to produce the most-likely parse T?

Probabilistic CKY

V: the number of non-terminals (i.e., POS tags, constituent types)

For CKY, each cell $table[i, j]$ contained a list of constituents that could span the sequence of words from i to j. For probabilistic CKY, it's slightly simpler to think of the constituents in each cell as constituting a third dimension of maximum length V. This third dimension corresponds to each non-terminal that can be placed in this cell, and the value of the cell is then a probability for that non-terminal/constituent rather than a list of constituents. In summary, each cell $[i, j, A]$ in this $(n+1) \times (n+1) \times V$ matrix is the probability of a constituent of type A that spans positions i through j of the input.

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                   and its probability
  for j ← from 1 to LENGTH(words) do
     for all { A | A → words[j] ∈ grammar}
        table[j−1, j, A] ← P(A → words[j])
     for i ← from j−2 downto 0 do
        for k ← i+1 to j−1 do
              for all { A | A → BC ∈ grammar,
                             and table[i,k,B] > 0 and table[k,j,C] > 0 }
                    if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                        table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                        back[i,j,A] ← {k,B,C}
     return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse.

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                   and its probability
  for j ← from 1 to LENGTH(words) do
     for all { A | A → words[j] ∈ grammar}
        table[j − 1, j,A] ← P(A → words[j])
     for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
           for all { A | A → BC ∈ grammar,
                         and table[i,k,B] > 0 and table[k,j,C] > 0 }
              if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                 table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                 back[i,j,A] ← {k,B,C}
     return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Independence assumption
on rules

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse.

Probabilistic CKY

- Chart: T[i,j,X] = best score

- Base: T[j,j-1,X] = log P(X $\rightarrow$ w_j)

- Loop over all split points k, apply rules X -> Y Z to build X in every possible way

- Recurrence:

$$T[i,j,X] = \max_{k} \max_{r:\, X \rightarrow X1\, X2} T[i,k,X1] + T[k,j,X2] + \log P(X \rightarrow X1\ X2)$$

- Runtime: $O(n^3G)$ G = grammar constant

Example

	The	*flight*	*includes*	*a*	*meal*
	Det: .40 [0,1]	NP: .30 *.40 *.02 = .0024 [0,2]	[0,3]	[0,4]	[0,5]
		N: .02 [1,2]	[1,3]	[1,4]	[1,5]
			V: .05 [2,3]	[2,4]	[2,5]
				Det: .40 [3,4]	[3,5]
					N: .01 [4,5]

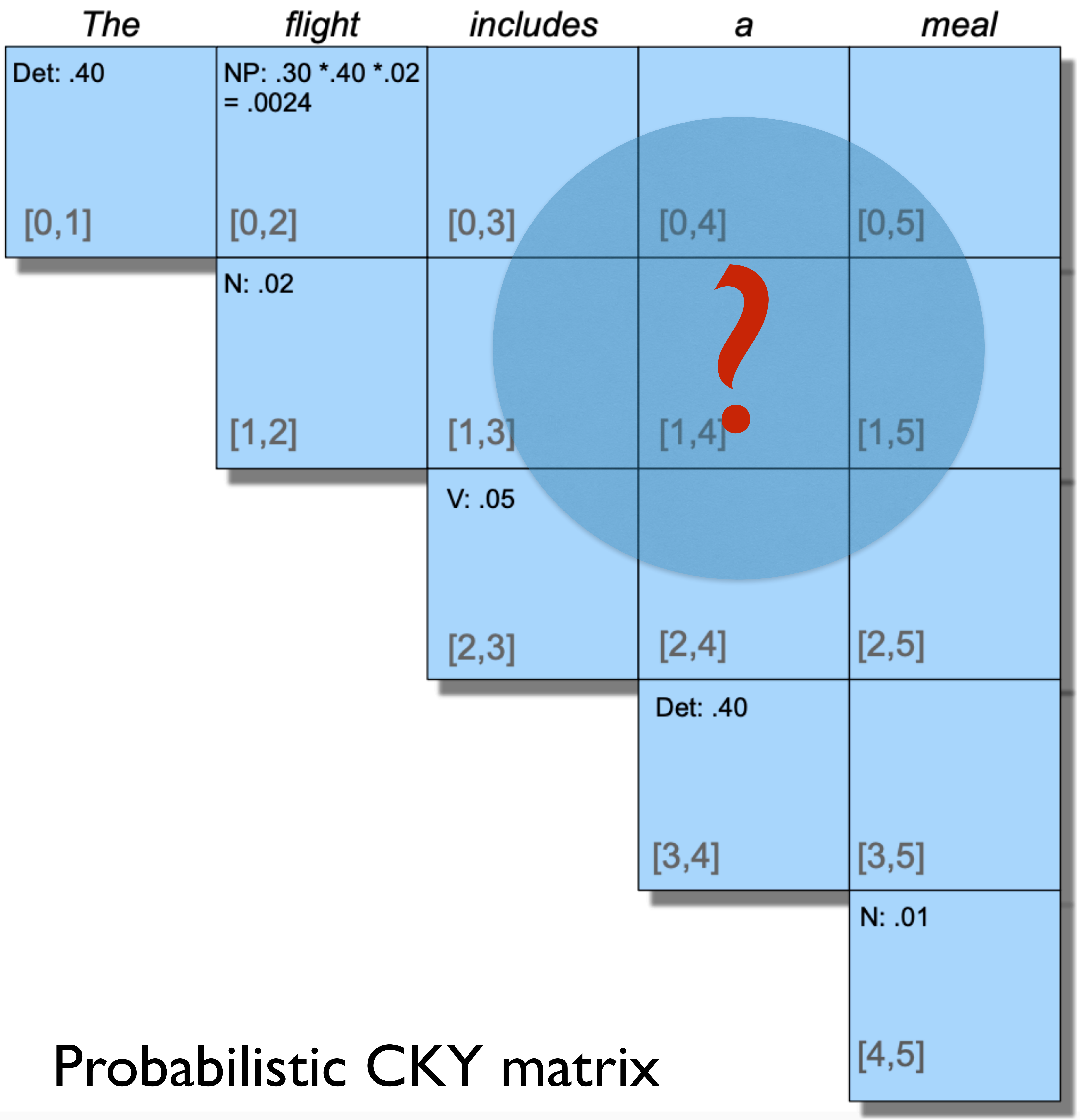
Probabilistic CKY matrix

S	$\rightarrow$	$NP\ VP$	.80
NP	$\rightarrow$	$Det\ N$	.30
VP	$\rightarrow$	$V\ NP$	.20
V	$\rightarrow$	*includes*	.05

Det	$\rightarrow$	*the*	.40
Det	$\rightarrow$	*a*	.40
N	$\rightarrow$	*meal*	.01
N	$\rightarrow$	*flight*	.02

Given this grammar

Example



Probabilistic CKY matrix

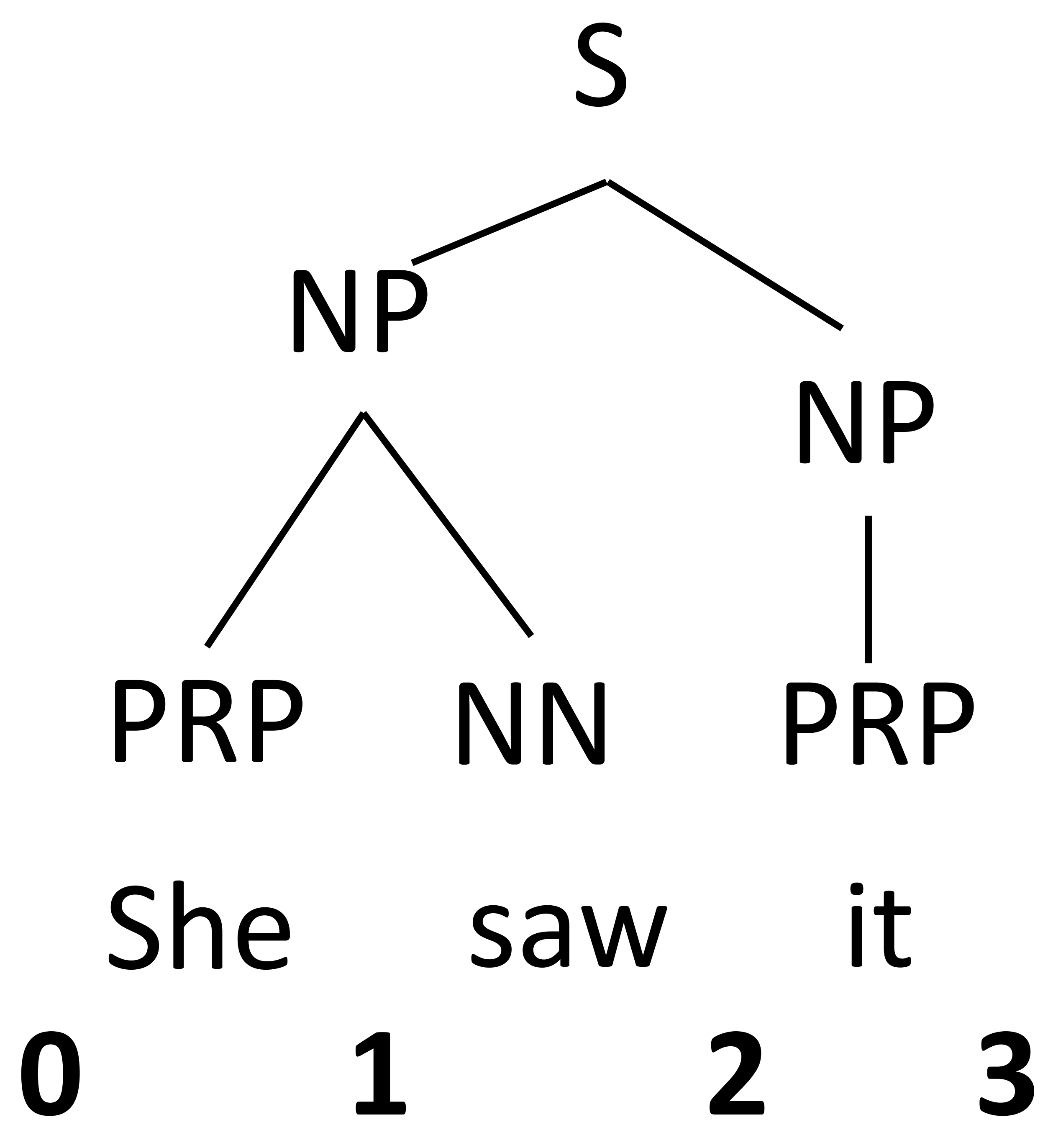
S	$\rightarrow$	$NP\ VP$	.80
NP	$\rightarrow$	$Det\ N$	.30
VP	$\rightarrow$	$V\ NP$	.20
V	$\rightarrow$	*includes*	.05

Det	$\rightarrow$	*the*	.40
Det	$\rightarrow$	*a*	.40
N	$\rightarrow$	*meal*	.01
N	$\rightarrow$	*flight*	.02

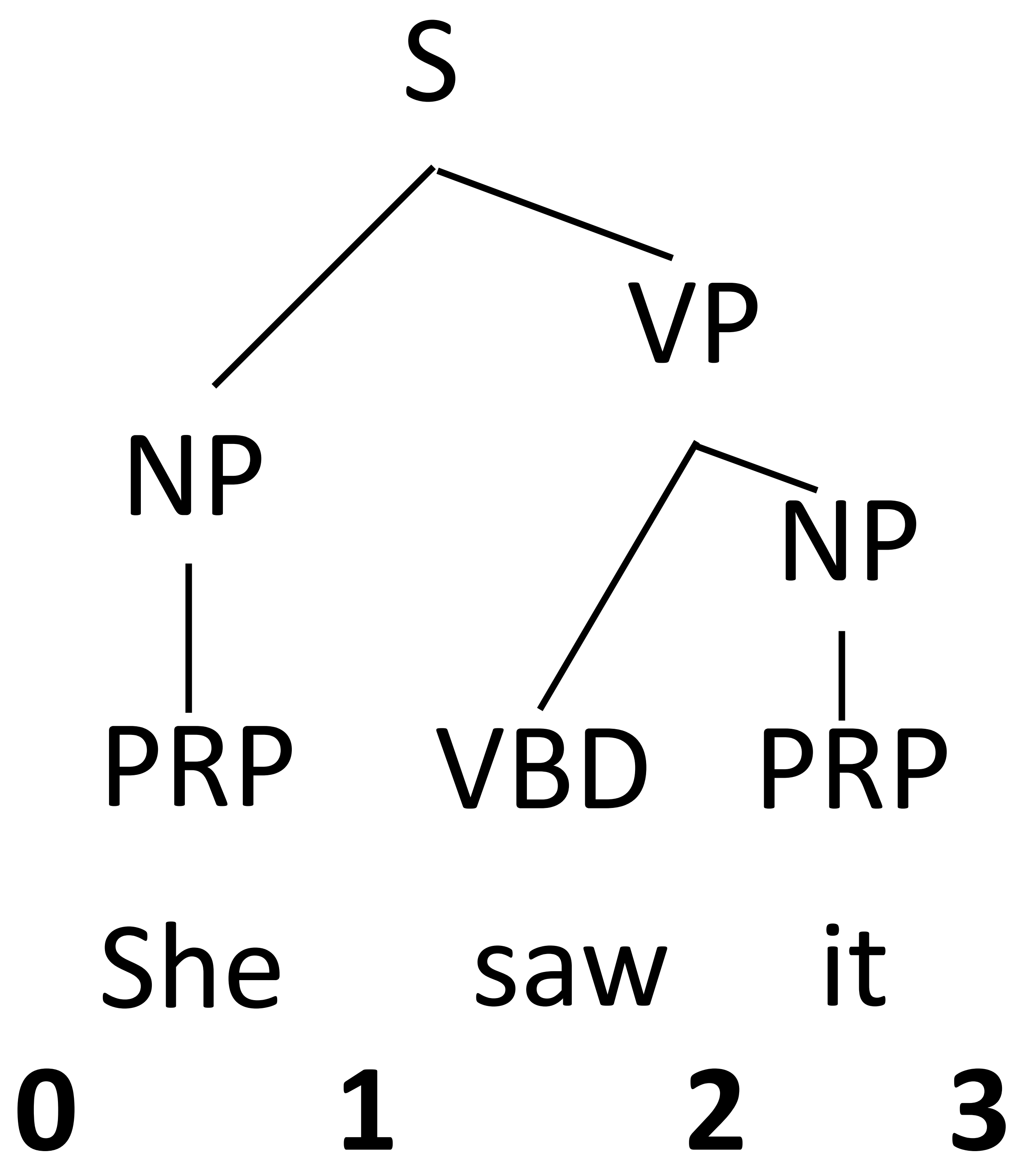
Given this grammar

For offline exercises

Parser Evaluation



S(0,3), **NP(0,2)**, **NP(2,3)**, ~~PRP(0,1), NN(1,2), PRP(2,3)~~



S(0,3), NP(0,1), VP(1,3), NP(2,3), ~~PRP(0,1), VBD(1,2), PRP(2,3)~~

- Precision: number of correct brackets / num pred brackets = 2/3
- Recall: number of correct brackets / num of gold brackets = 2/4
- F1: harmonic mean of precision and recall = $(1/2 * ((2/4)^{-1} + (2/3)^{-1}))^{-1}$

  $= 0.57$

Results

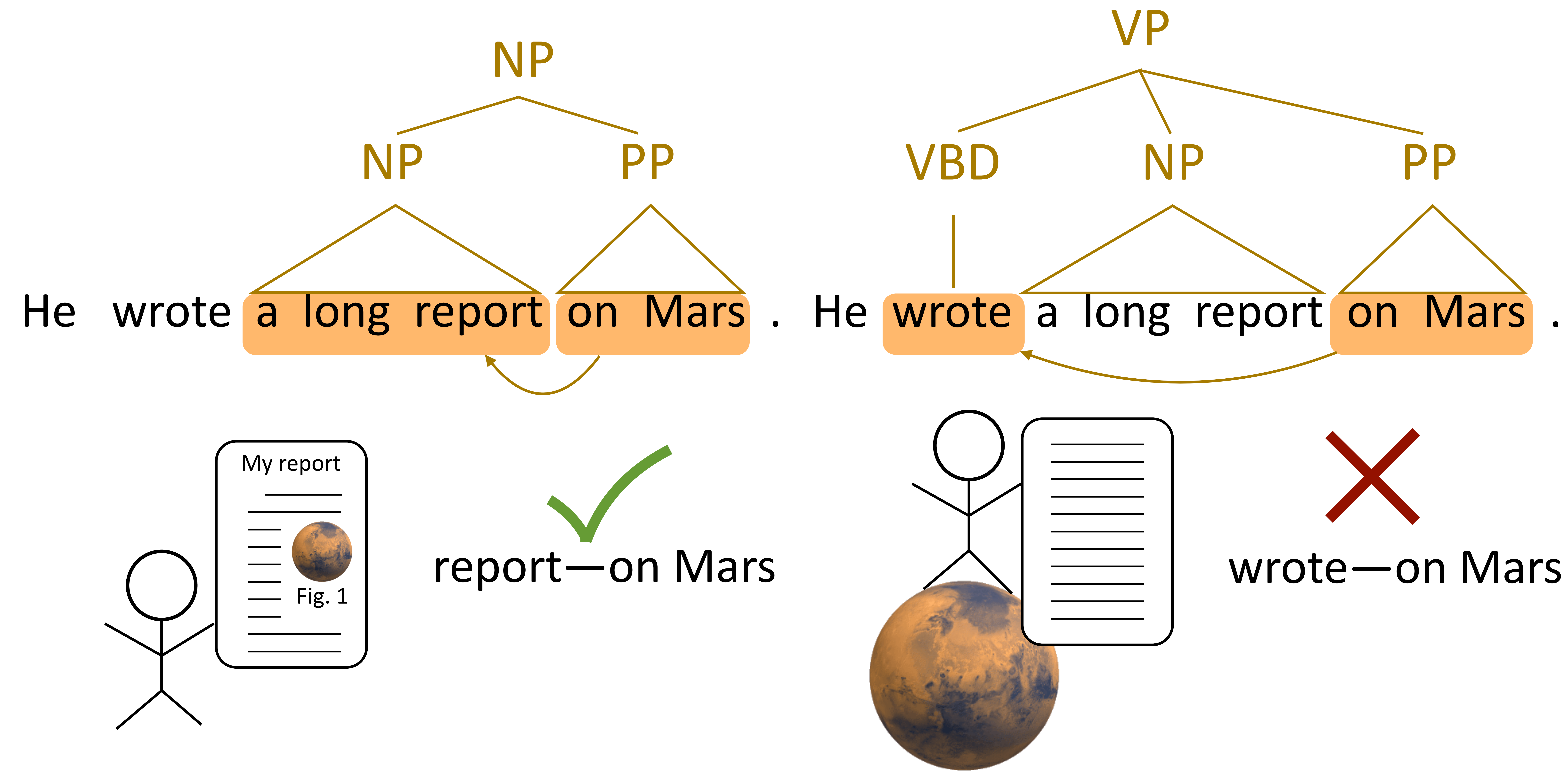
- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
  - Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best PCFGs for English: ~90 F1 Klein and Manning (2003)
- Nowadays SOTA (discriminative models): 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

Discriminative Parsers

for offline reading

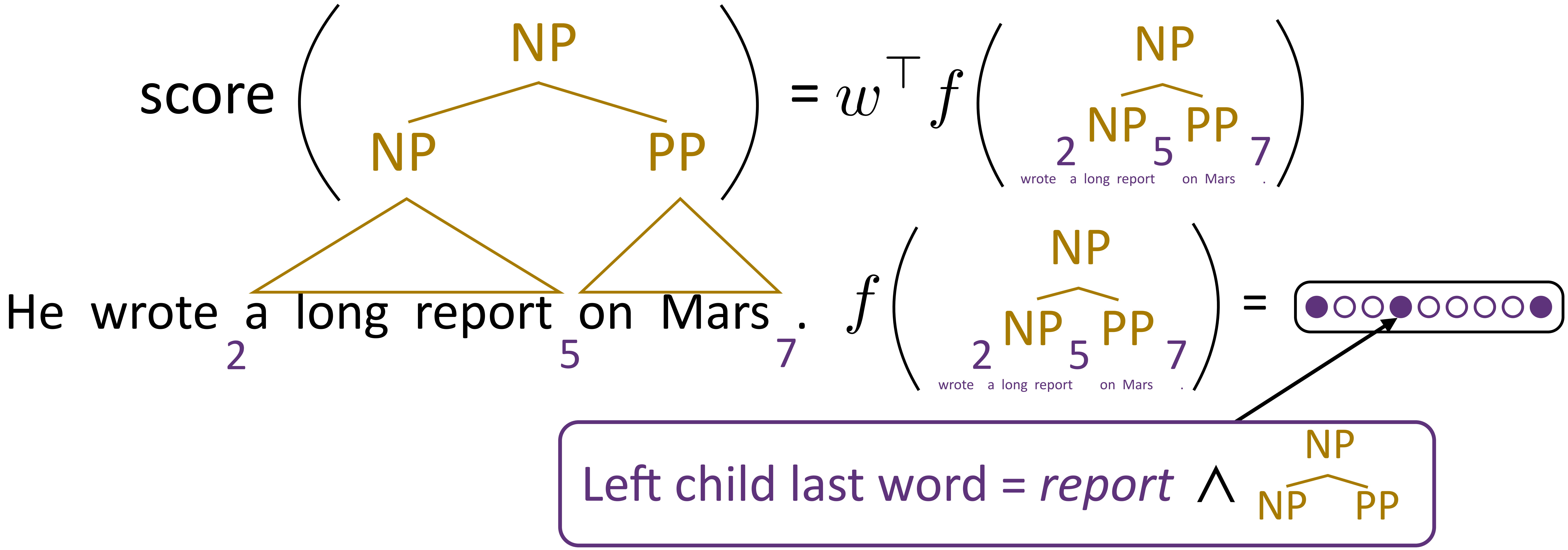
CRF Parsing

for offline reading



CRF Parsing

for offline reading

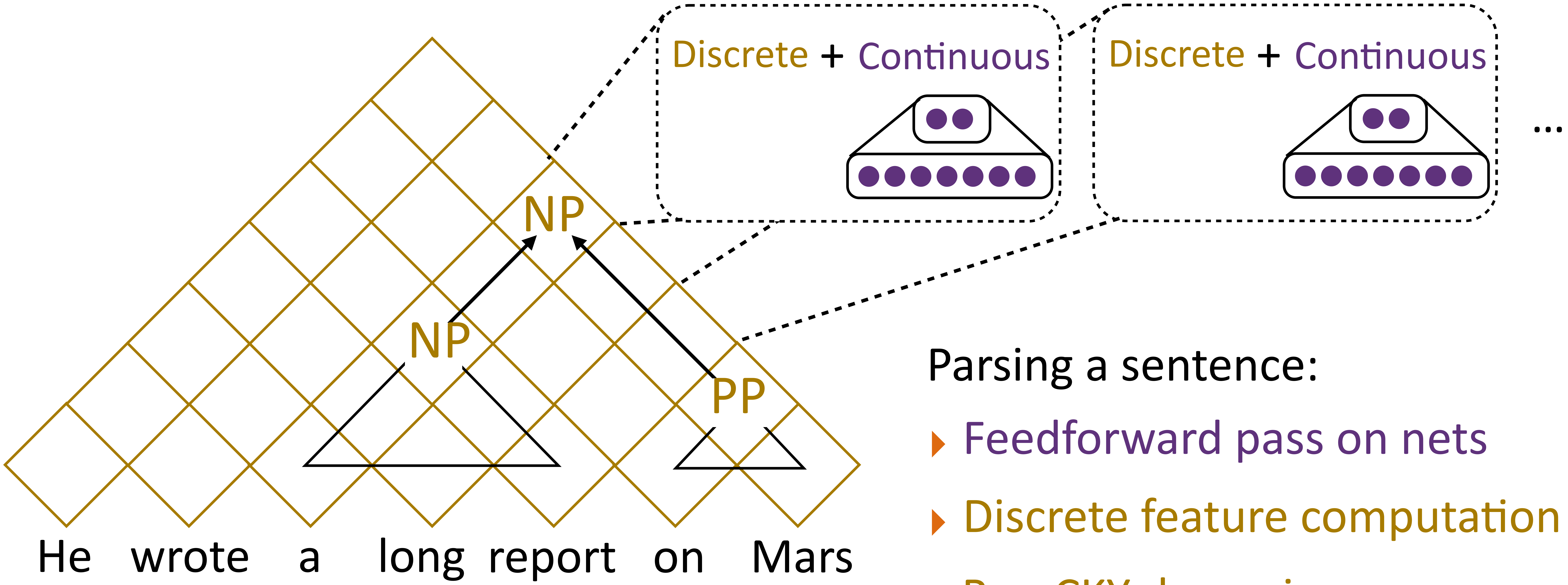


- Can learn that we *report* [PP], which is common due to *reporting on* things
- Can "neuralize" this as well like neural CRFs for NER

Taskar et al. (2004)
Hall, Durrett, and Klein (2014)
Durrett and Klein (2015)

Joint Discrete and Continuous Parsing

- Chart remains discrete!



Parsing a sentence:

- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

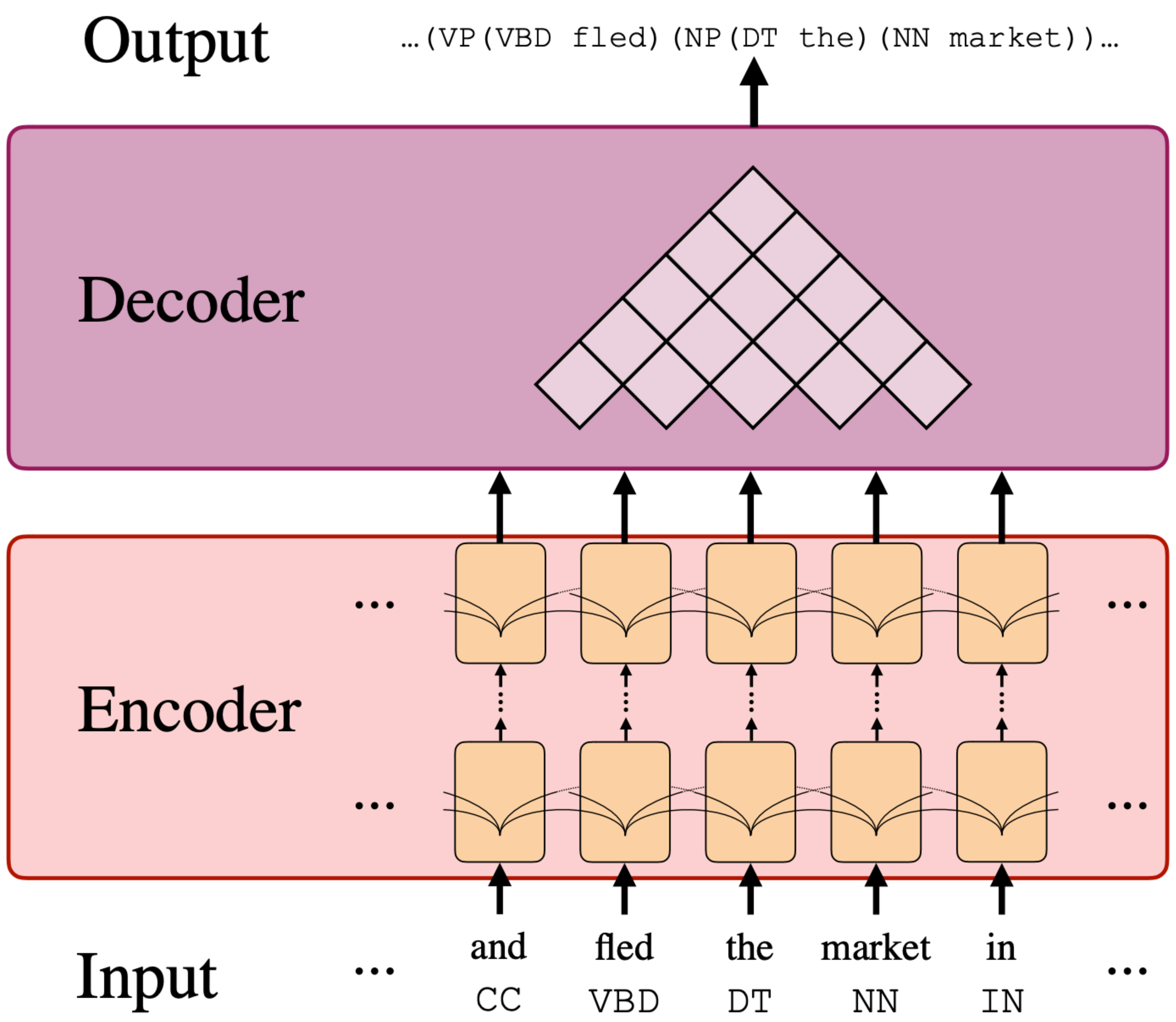
for offline reading

Durrett and Klein (ACL 2015)

Parsing with ELMo

Encoder Architecture	F1 (dev)	Δ
LSTM (Gaddy et al., 2018)	92.24	-0.43
Self-attentive (Section 2)	92.67	0.00
+ Factored (Section 3)	93.15	0.48
+ CharLSTM (Section 5.1)	93.61	0.94
+ ELMo (Section 5.2)	95.21	2.54

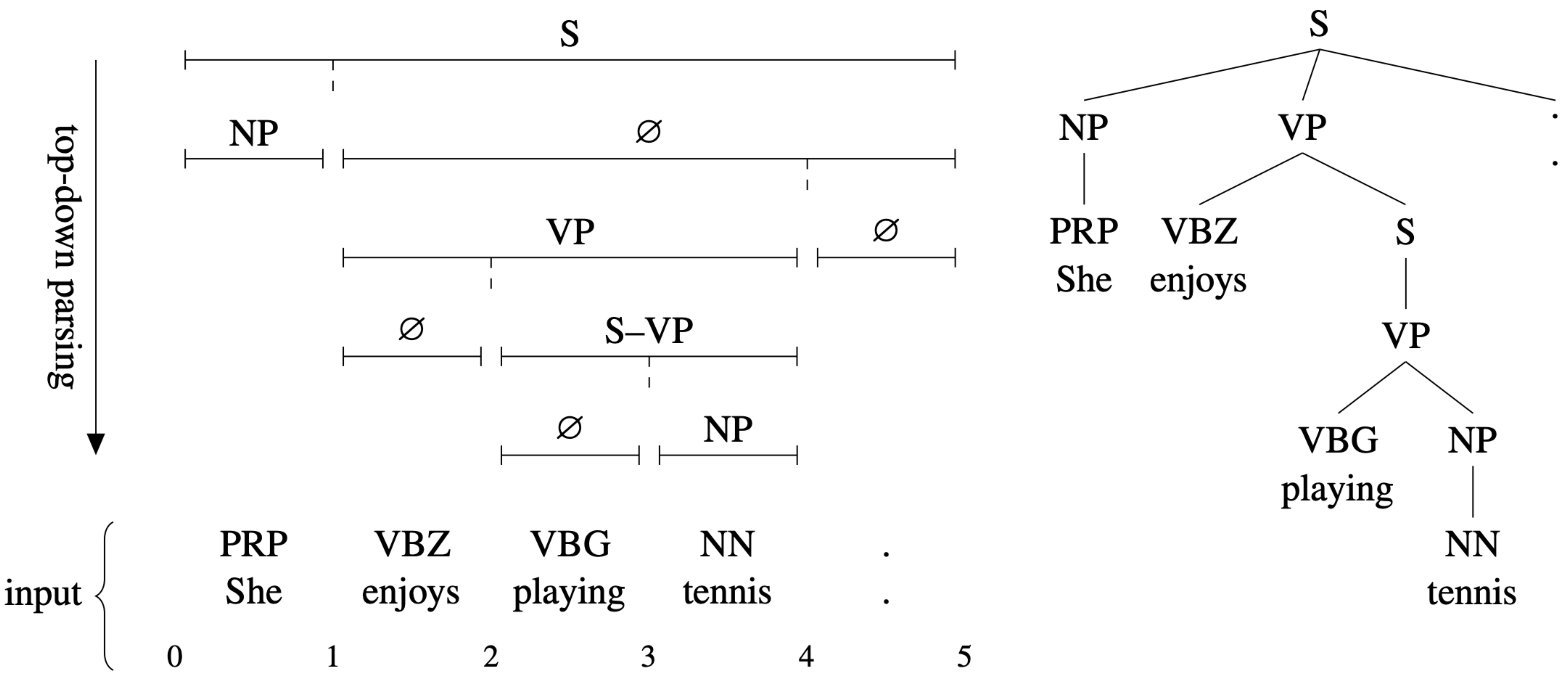
- Improves the neural CRF by using a transformer layer (self-attentive), character-level modeling, and ELMo



for offline reading

Kitaev and Klein (2018)

Top-down Parsing



(a) Execution of the top-down parsing algorithm. (b) Output parse tree.

- Greedily predict bracketing at next stage of the tree. Like a neural CRF but with no dynamic program (CKY) pass

for offline reading

Stern et al. (2017)

Takeaways

- PCFGs estimated generatively can perform well if sufficiently engineered

- Neural CRFs (offline reading) work well for constituency parsing

Midterm Review

- Basically everything we covered in class (except those briefly mentioned or marked as offline reading)
- Topics including:

Chapter 4 in JM

- Naive Bayes: model, estimation from data (including smoothing), computing posterior probabilities
- Bag-of-words features: how these feature spaces look and how they work for classification

- Logistic regression (binary classification)
- Sentiment analysis
- Multiclass classification: how weights and features work in this setting

how to derive gradients

- Feedforward neural networks
- Training neural networks

how backpropagation works

- Word embeddings: CBOW, skip-gram, skip-gram with negative sampling
- POS tagging
- Hidden Markov Models
- Viterbi algorithm
- Beam search

- basic seq2seq
- seq2seq + attention

Midterm Review

- You can check out some problems related to the topics in the previous slide in the following midterm:

  https://www.cs.utexas.edu/~gdurrett/courses/sp2020/sp19-midterm-solutions.pdf

- Unlike the above midterm, we won't have questions on code snippets on deep NNs or on topics we haven't covered so far.
- The format will be similar: (1) Multi-choice Questions (2) Short/Long answers (3) Computation