Morphology and Syntax, II

Constituencies and Context-free Grammars

Aaron Mueller
CAS CS 505: Natural Language Processing
Boston University
Spring 2026

(Some slides inspired by a lecture by Julia Hockenmaier.)

Admin

- **HW3** is due in 3 days!

- A **practice exam** and its **solutions** have been released. Use this to get a feel for the format of the exam.
  - Check the list of topics on the syllabus before you take it. Use this list to help you write your note sheet.
  - Recommendation: time yourself while taking the exam, and note any topics you struggled with. Use these to guide your studying.

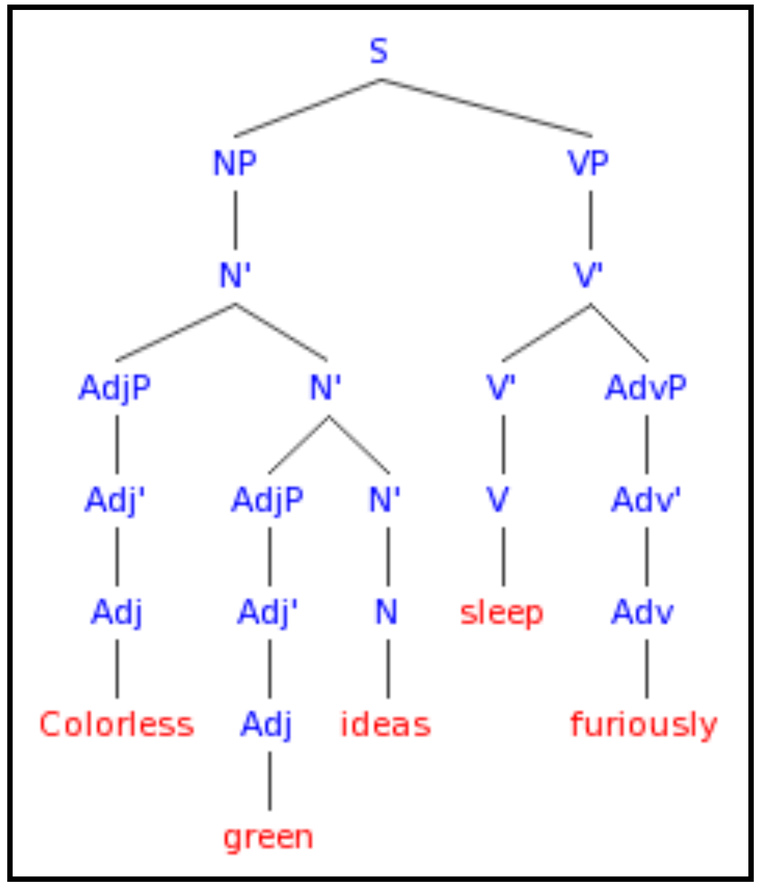
Overview of Concepts

Constituencies are syntactic units composed of one or more words.

Parse trees are graphical representations of parses.

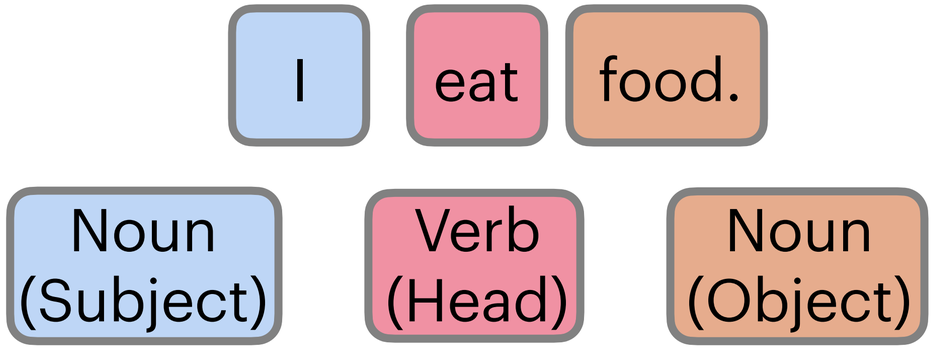
Context-free grammars are sets of symbols and rules that express ways in which symbols can be grouped.

The **CKY algorithm** is a constituency parsing algorithm.



How is language structured?

Syntax is the study of word order, and how words compose to form larger units.

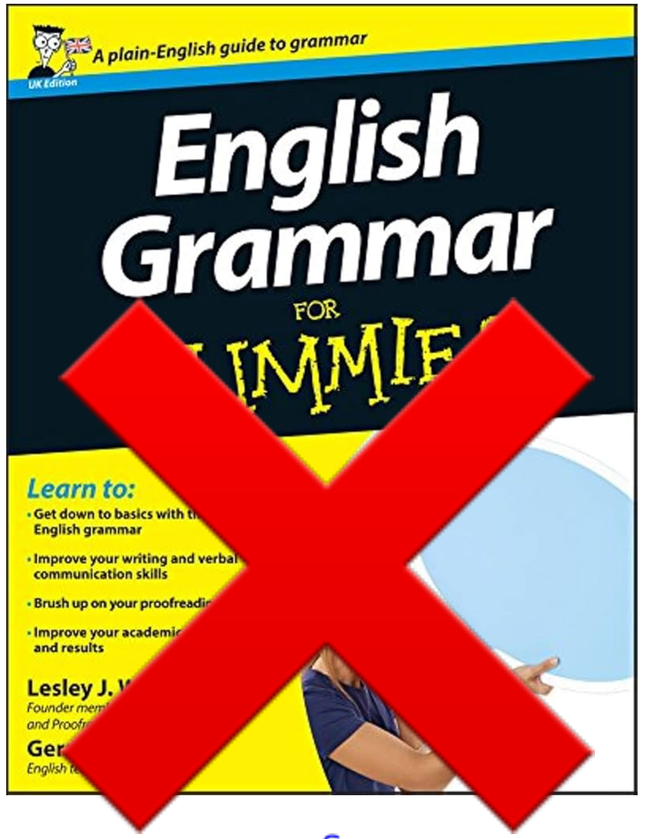


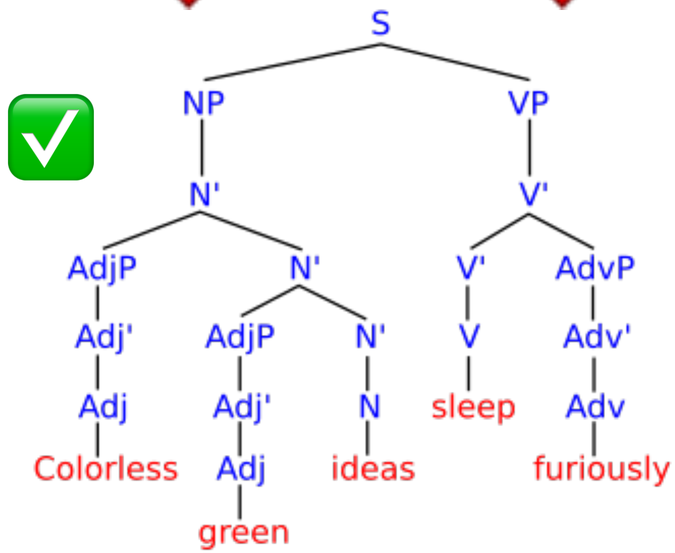
What is grammar?

- There is a *big* difference between the way “grammar” is used in school/ESL classes and the way we will use the term.

- In this class, we care about **grammar formalisms**:
  - A precise way of defining and describing the structure of sentences

- We will also talk about possible **grammars**:
  - A specific implementation in a particular formalism of a particular language





Why care about syntax?

- Most words have many interpretations—syntax helps disambiguate
  - Enables verb argument structure analysis
  - Allows us to classify languages

- Can be used for grammar checkers or as part of semantic analysis pipelines

- Context-free grammars come up *a lot* in NLP research and industry!
  - Data generation
  - Measuring linguistic complexity
  - Formal language theory

Words vs. Phrases

All sentences consist of a **noun phrase** (NP) and a **verb phrase** (VP).

noun

[The big red **ball]** is near the basketball court.

NP

verb

Pets **[eat** food multiple times a day**]**.

VP

These phrases are **constituents**.

Constituents

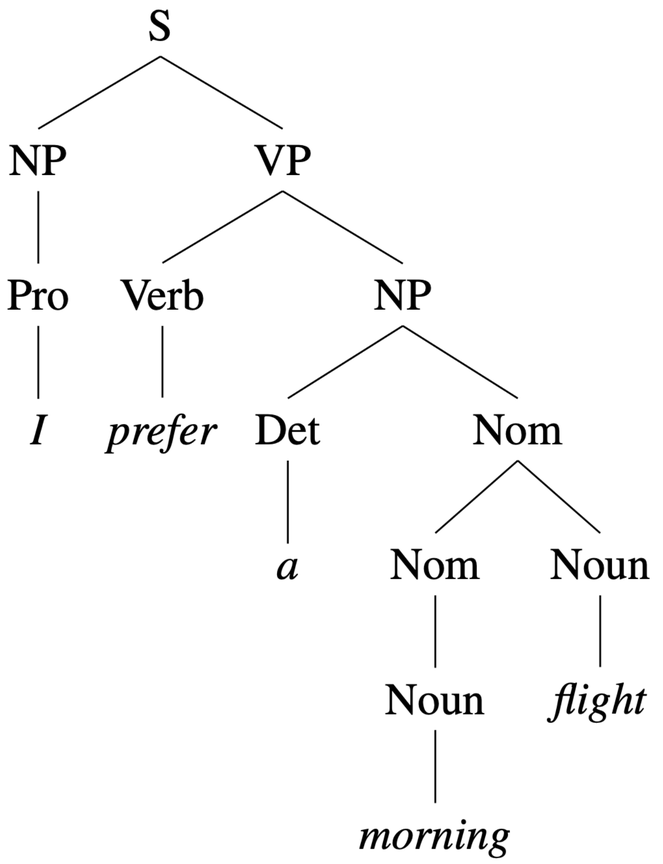
- Constituents are the basic units of syntax.
- All constituents have **heads**. Optionally, they can have *dependents*:

Noun phrase	Verb phrase	Prepositional phrase
the **man**	**sleeps**	**with** *glasses*
Gansu	**sleeps** *soundly*	***near*** *the desert*
a **dog** *with fur*	**eats** *the food*	

Constituency Parsing

- Tree-structured syntactic analysis of sentences
- *Constituents:*
  - Sentence (S)
  - Noun Phrase (NP)
  - Verb Phrase (VP)
  - Prepositional Phrase (PP)
  - etc.

- The leaf nodes are POS tags or words

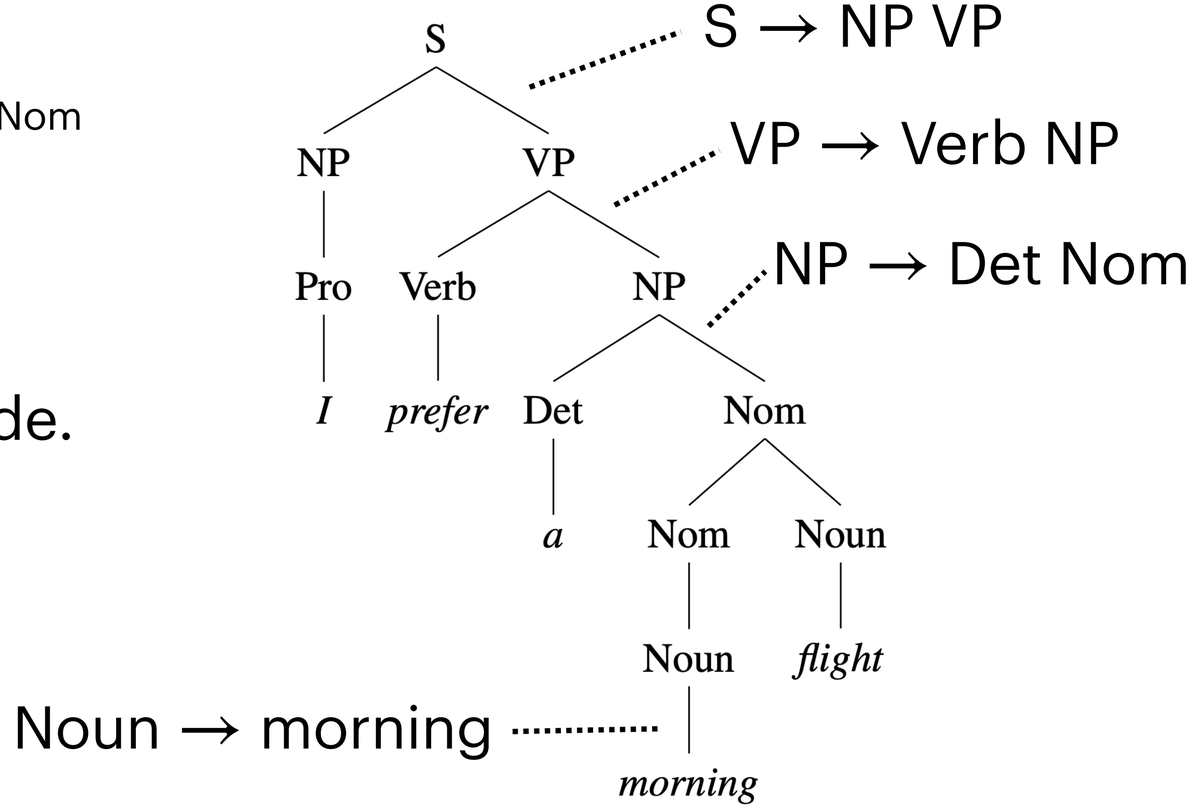


Notation

- We can express a constituency tree using **bracketed notation:**

  [S [NP [Pro I]] [VP [V prefer] [NP [Det a] [Nom [Nom [N morning]] [N flight]]]]]]

- We can express a rule in this grammar as a left-hand side generating a right-hand side.



Heads, Arguments, and Adjuncts

- **Heads:** each constituent has one head. It has the same category as the phrase:

  VP → *Verb* NP

  NP → Det *Noun*

- **Arguments:** *required* by the head

  VP → Verb **NP** ⟵ NPs are often arguments of verbs

  he [saw **a ball**]

- **Adjuncts**: optional

  VP → VP **PP** ⟵ PPs are optional adjuncts

  he [ran **by the building**]

Is this substring a constituent?

A dog with fur sleeps **[near the forest]**.

- Substitution test: can substring α be replaced by a word?
  - A dog with fur sleeps [here].

- Movement test: can α be moved around in the sentence?
  - [Near the forest], a dog with fur sleeps.

- Answer test: can α be a complete answer to a question?
  - Where does a dog with fur sleep? [Near the forest].

Language is recursive.

the ball

the **red** ball

the **big** red ball

the big, red, **heavy** ball

the big, red, heavy ball **that I saw**

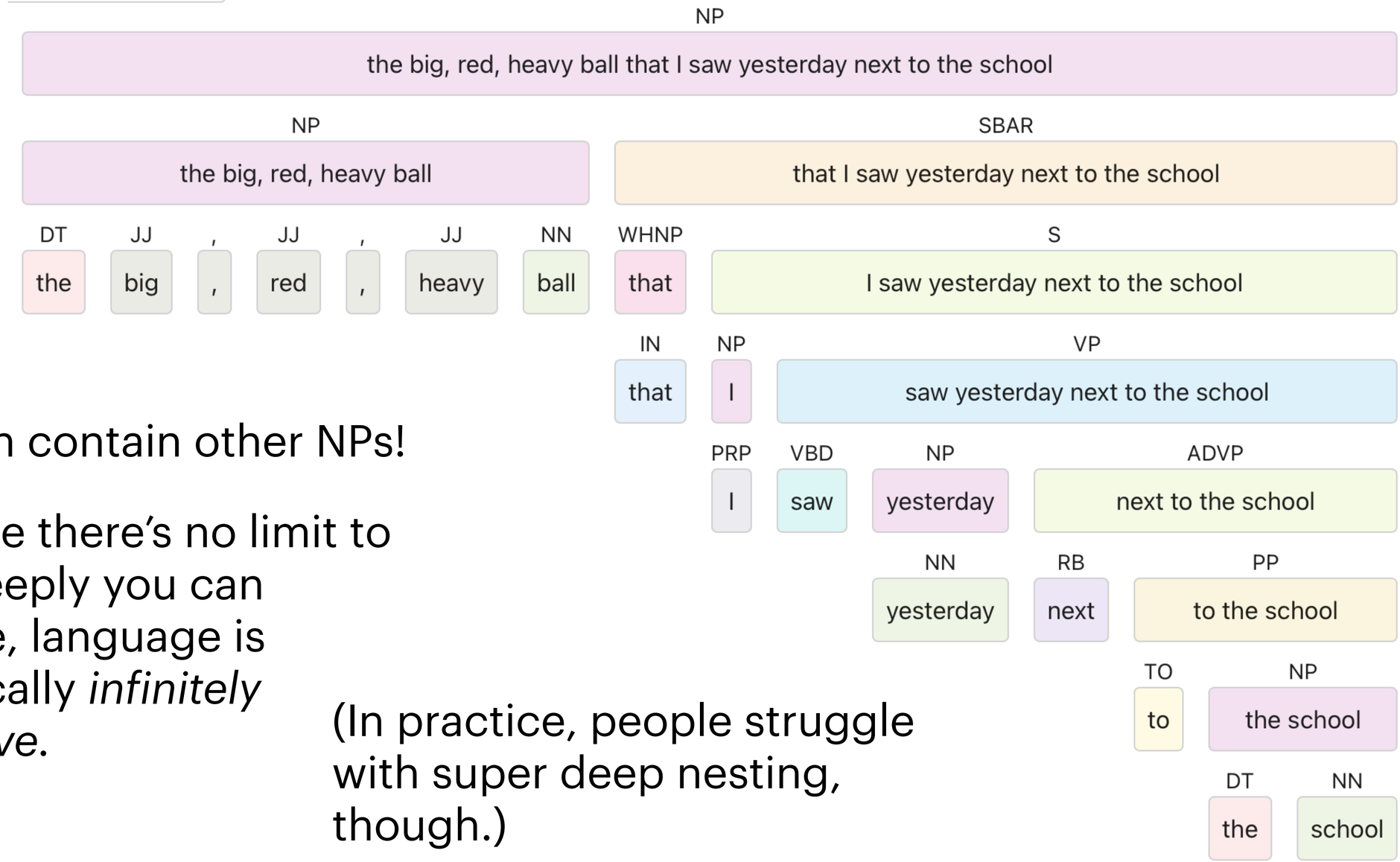
the big, red, heavy ball that I saw **yesterday**

the big, red, heavy ball that I saw yesterday **next to the school**

Constituents often contain other constituents.

There is no theoretical limit to how many constituents a constituent can contain.

Language is recursive.



NPs can contain other NPs!

Because there's no limit to how deeply you can recurse, language is technically *infinitely recursive*.

(In practice, people struggle with super deep nesting, though.)

Definition of Context-free Grammar (CFG)

- All **context-free grammars (CFGs)** consist of the following:

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
S	a designated **start symbol** and a member of N

e.g. N = {S, NP, VP, PP, Noun, Verb, ...}

Start symbol $S \in N$

e.g. Σ = {I, you, eat, drink, the, a}

R = {$A \rightarrow \beta$ with left-hand side $A \in N$, and right-hand side $\beta \in (N \cup \Sigma)^*$}

A Context-free Grammar

- Given a grammar like this one, we say that a sequence is **grammatical** if it can be parsed by the grammar.
  - We say it is **ungrammatical** if not.

Grammar Rules		
S	$\rightarrow$	$NP\ VP$
NP	$\rightarrow$	$Pronoun$
	$\mid$	$Proper\text{-}Noun$
	$\mid$	$Det\ Nominal$
$Nominal$	$\rightarrow$	$Nominal\ Noun$
	$\mid$	$Noun$
VP	$\rightarrow$	$Verb$
	$\mid$	$Verb\ NP$
	$\mid$	$Verb\ NP\ PP$
	$\mid$	$Verb\ PP$
PP	$\rightarrow$	$Preposition\ NP$

$Noun$	$\rightarrow$	$flights \mid flight \mid breeze \mid trip \mid morning$
$Verb$	$\rightarrow$	$is \mid prefer \mid like \mid need \mid want \mid fly \mid do$
$Adjective$	$\rightarrow$	$cheapest \mid non\text{-}stop \mid first \mid latest$ $\mid other \mid direct$
$Pronoun$	$\rightarrow$	$me \mid I \mid you \mid it$
$Proper\text{-}Noun$	$\rightarrow$	$Alaska \mid Baltimore \mid Los\ Angeles$ $\mid Chicago \mid United \mid American$
$Determiner$	$\rightarrow$	$the \mid a \mid an \mid this \mid these \mid that$
$Preposition$	$\rightarrow$	$from \mid to \mid on \mid near \mid in$
$Conjunction$	$\rightarrow$	$and \mid or \mid but$

Generating with CFGs

- By writing down only tens of rules, we can generate many hundreds of thousands of sentences!
- We can sample from our CFG by randomly picking a rule for each non-terminal
  - (For now, we'll assume each rule is equally likely)

S → NP VP

NP_1 → Proper-Noun

VP → Verb NP_2

NP_2 → Det Nominal

Nominal → Noun

Proper-Noun → Chicago

Verb → is

Det → a

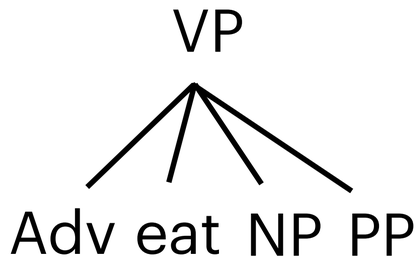
Noun → breeze

Grammar Rules		
S	→	*NP VP*
NP	→	*Pronoun*
	|	*Proper-Noun*
	|	*Det Nominal*
Nominal	→	*Nominal Noun*
	|	*Noun*
VP	→	*Verb*
	|	*Verb NP*
	|	*Verb NP PP*
	|	*Verb PP*
PP	→	*Preposition NP*

Chomsky Normal Form (CNF)

- The RHS of a standard CFG can contain an arbitrary number of symbols:

VP → Adv eat NP PP



- A CFG in **Chomsky Normal Form (CNF)** allows only two kinds of right-hand sides:
  - *Two nonterminals:* VP → Adv VP
  - *One terminal:* VP → eat

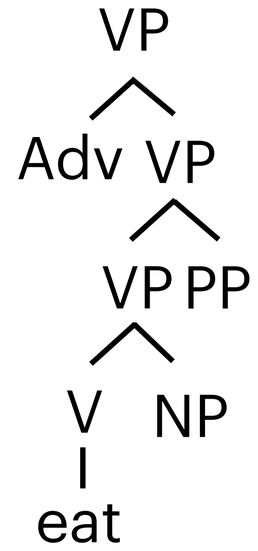
- All CFGs can be converted into an equivalent CNF.

VP → Adv VP

VP → VP PP

VP → V NP

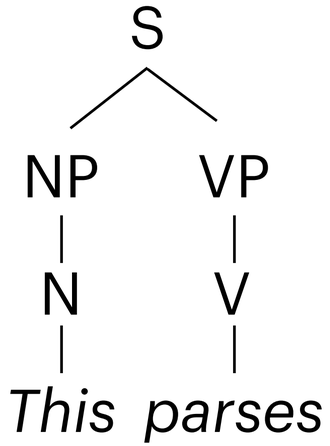
V → eat



Syntactic Parsing

- **Parsing:** mapping from a string of words to its parse tree
- Given a sentence and a CFG, we want to search through the space of possible parses to find:
  - If there exists any valid parse (**recognition**)
  - All valid parses
  - The most probable parse

This parses →



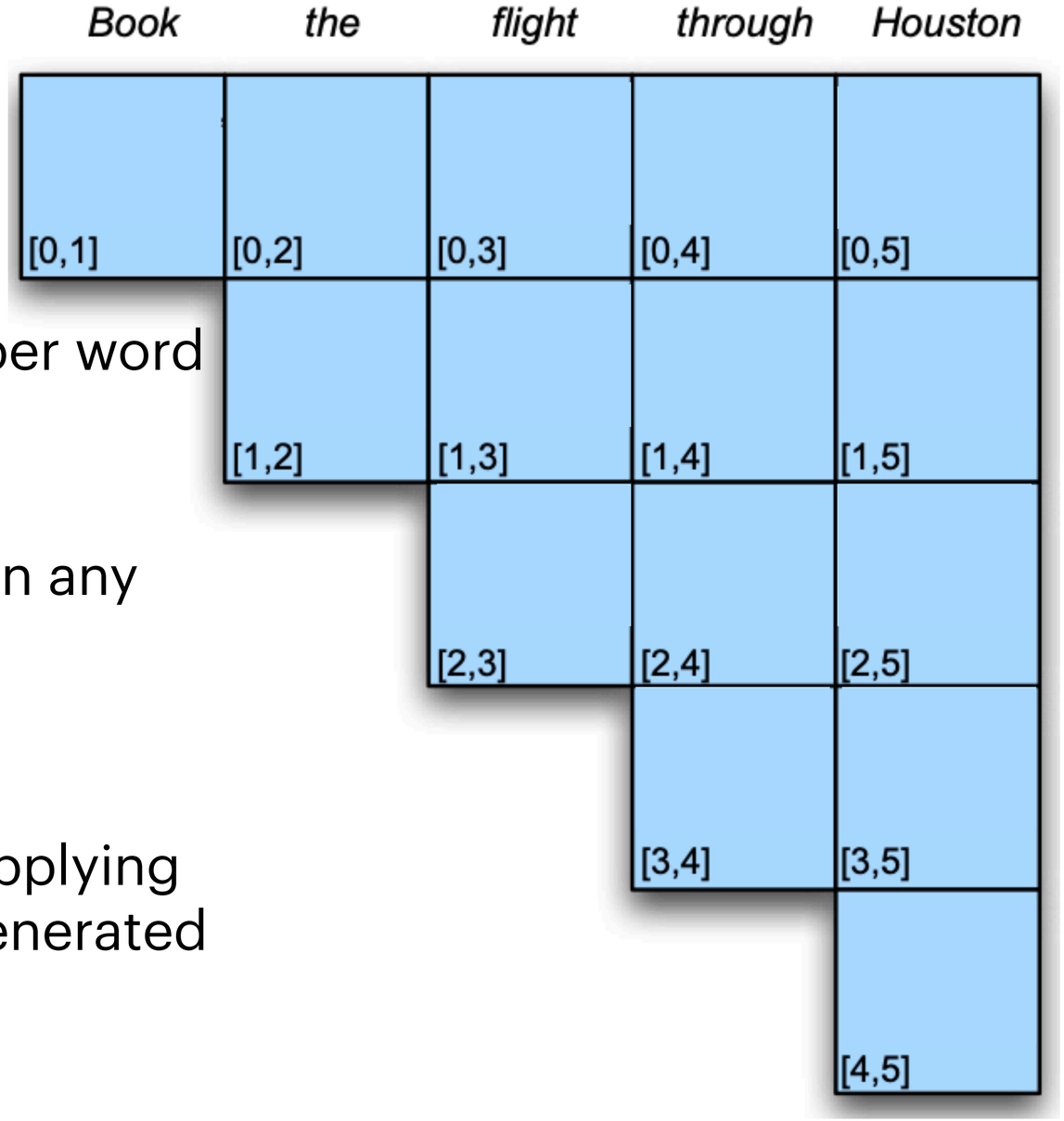
CKY Algorithm

- The **CKY algorithm** is a bottom-up dynamic programming approach to constituency parsing
  - Start with words
  - Save results in a table
  - Re-use these results to find larger constituents

- $O(n^3 |G|)$ time, where $n =$ word length of sentence, $G =$ size of grammar

- Requires a CFG to be in Chomsky normal form

CKY Recognition



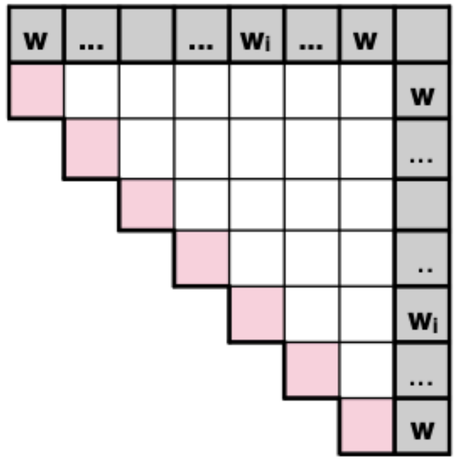
1. Draw a triangular table with one column per word

2. For each word's diagonal entry, write down any terminals that it could correspond to

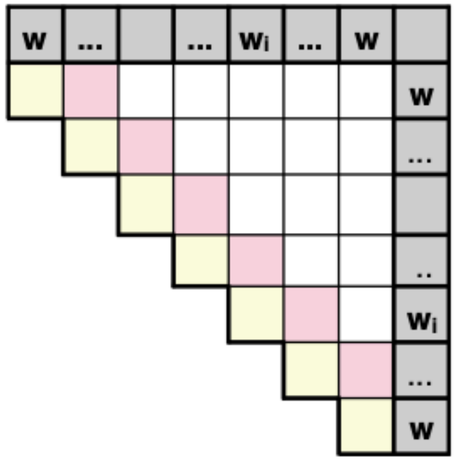
3. Progressively fill the rest of the table by applying possible non-terminals that could have generated observed pairs of terminals

Filling the CKY Chart

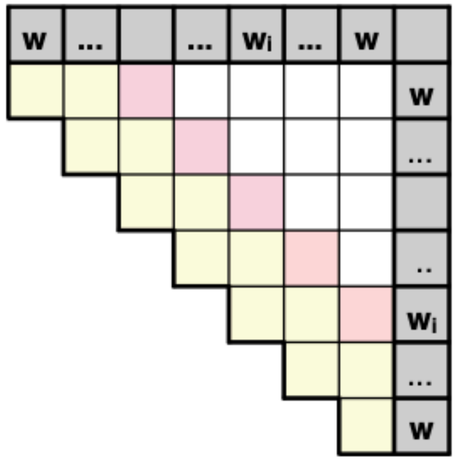
1



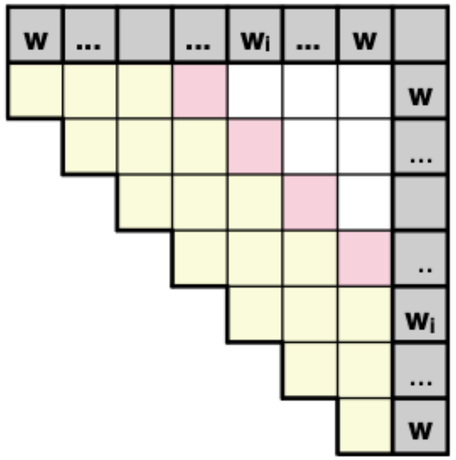
2



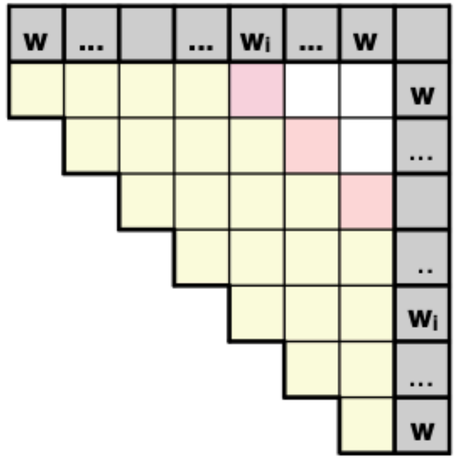
3



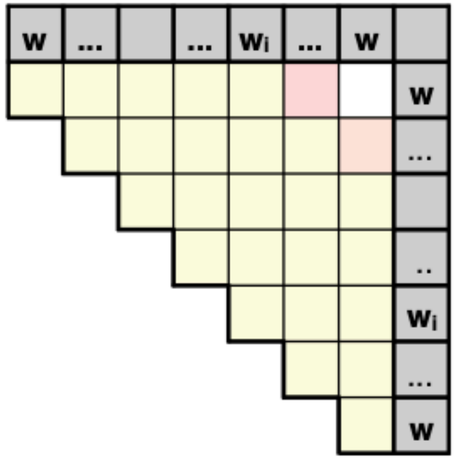
4



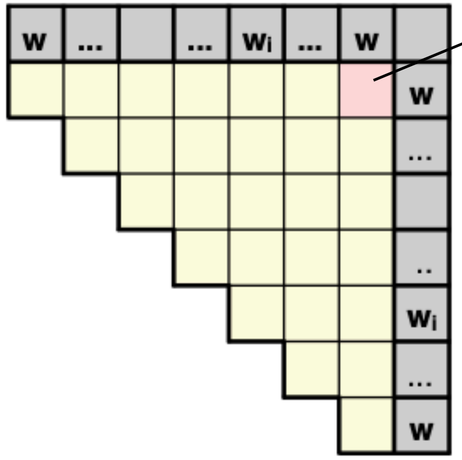
5



6

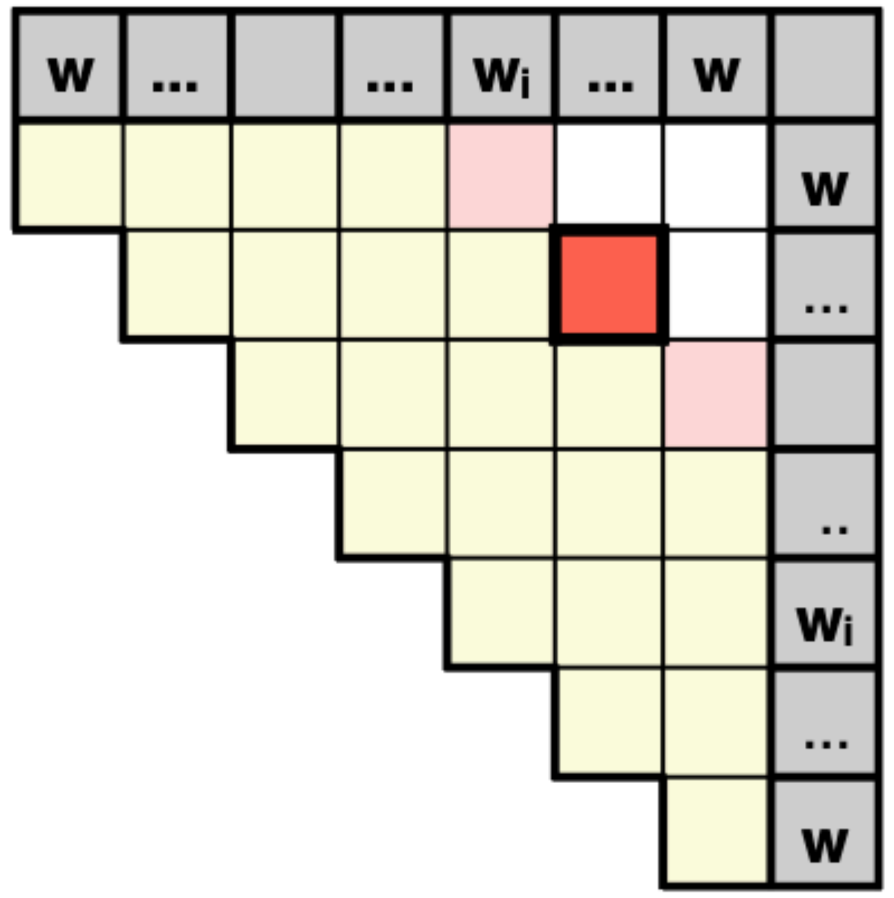


7



If the top-right cell contains the start symbol, the sentence can be parsed

Filling a CKY Cell

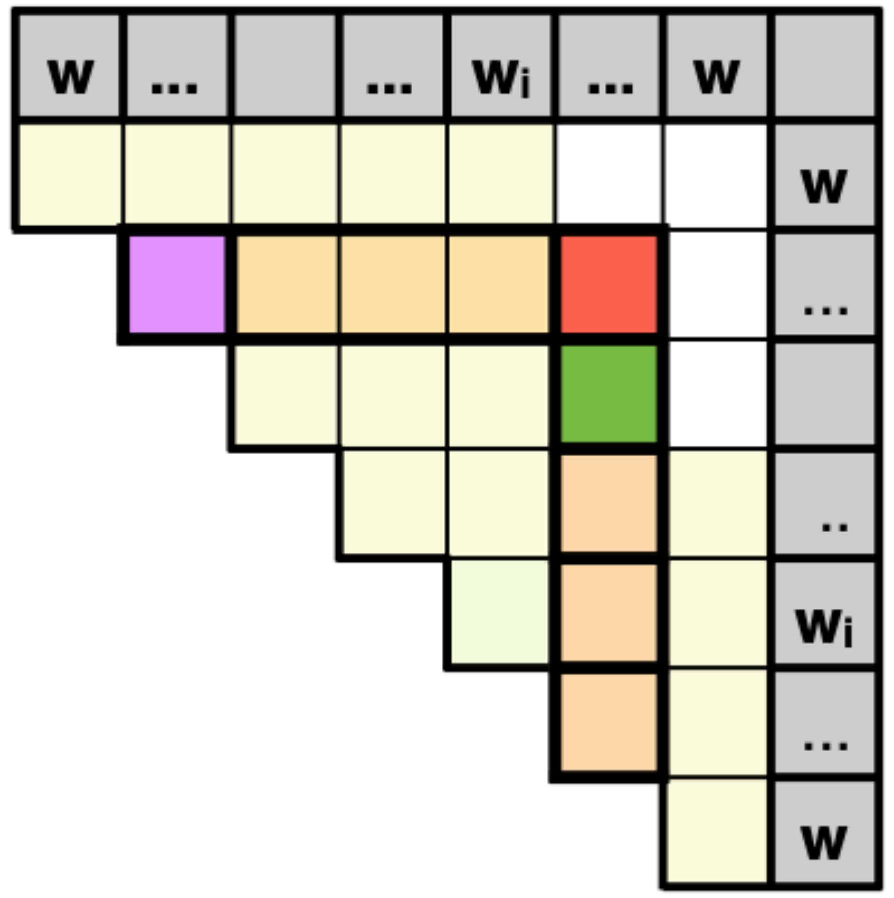


chart[2][6]:

w_1 **w_2 w_3 w_4 w_5 w_6** w_7

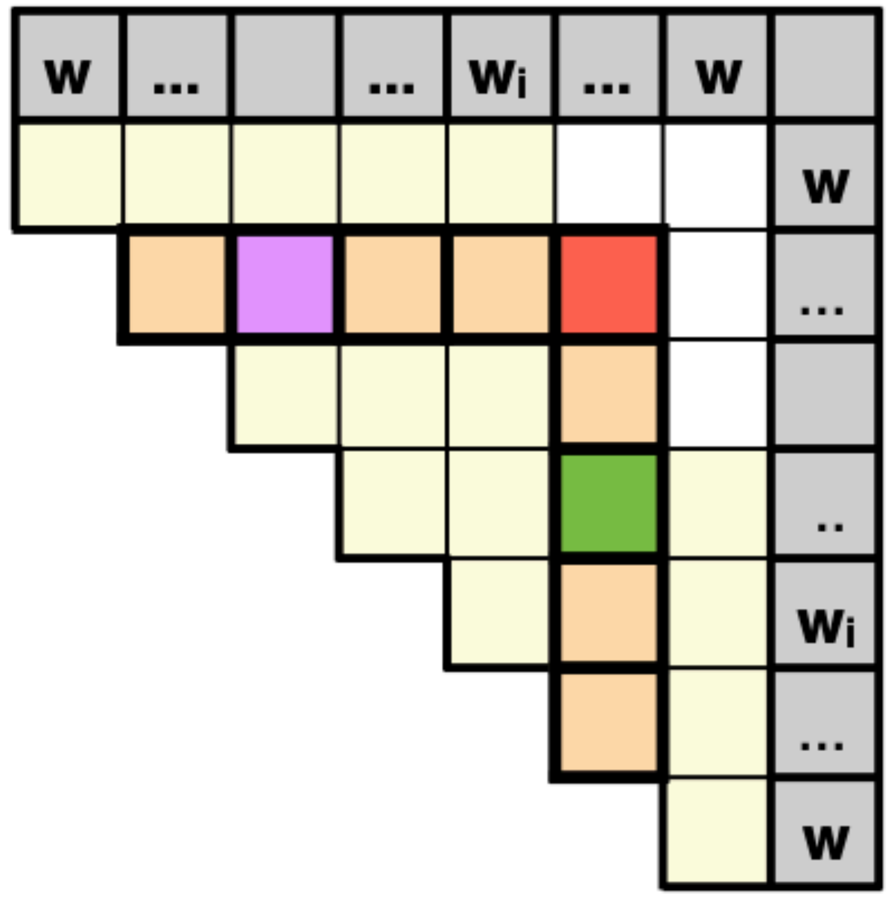
chart[2][6]:

w_1 **$w_2 w_3 w_4 w_5 w_6$** w_7



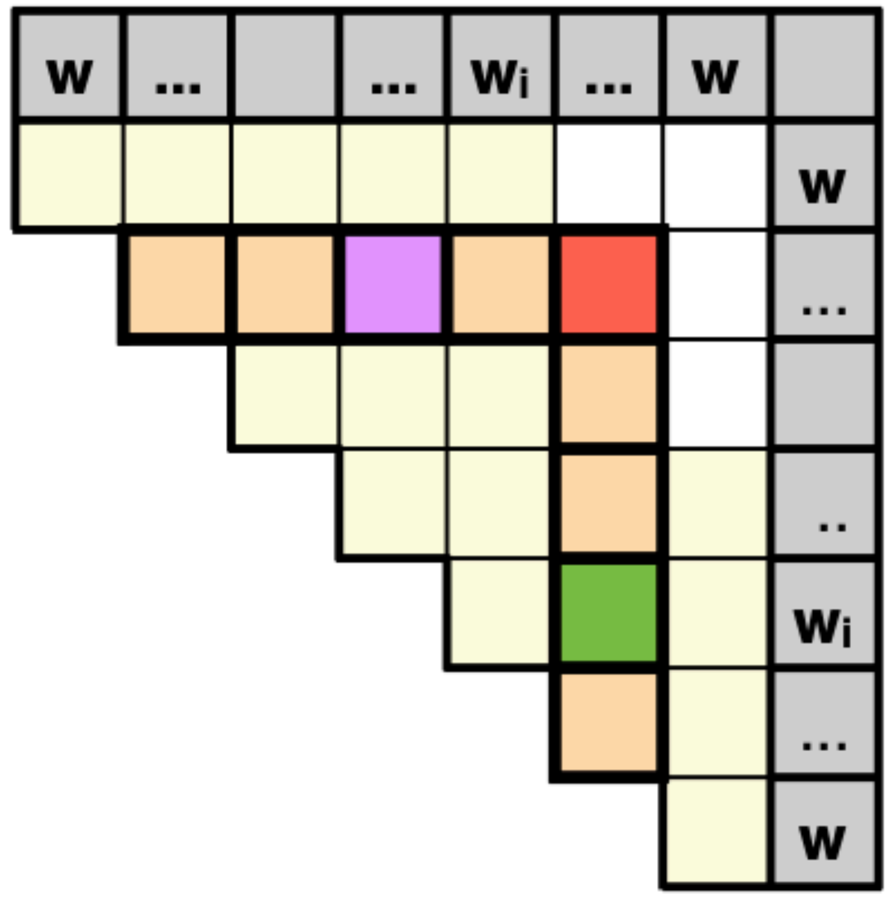
chart[2][6]:

w_1 **$w_2 w_3 w_4 w_5 w_6$** w_7



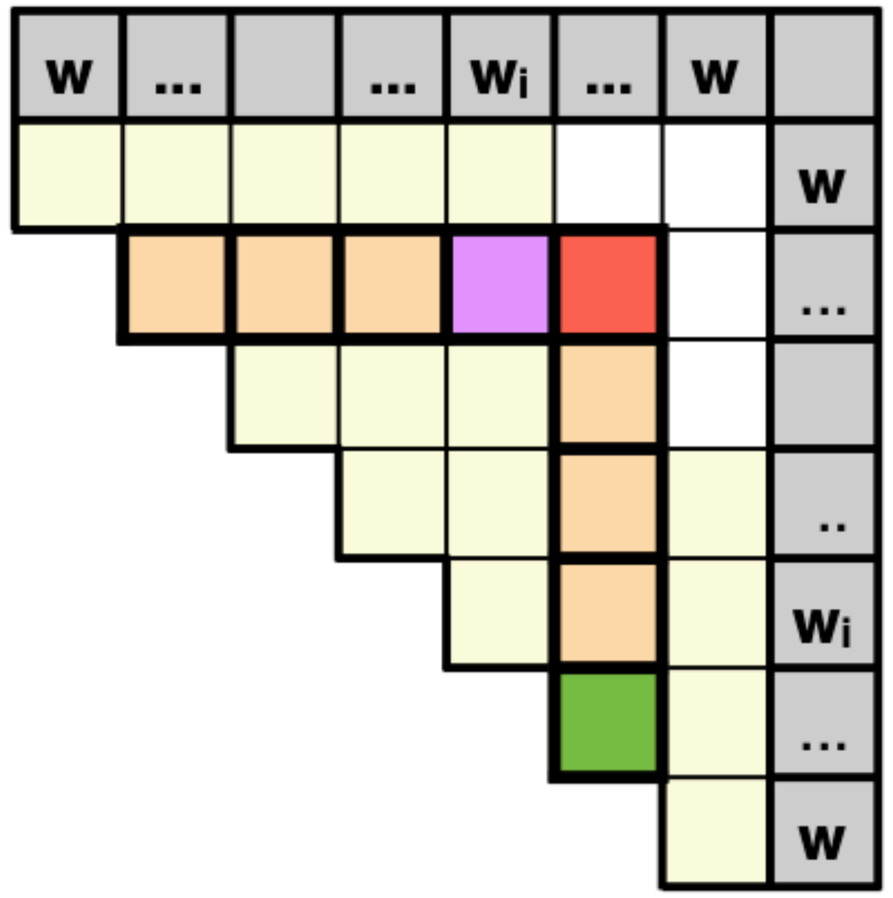
chart[2][6]:

w_1 **$w_2 w_3 w_4 w_5 w_6$** w_7



chart[2][6]:

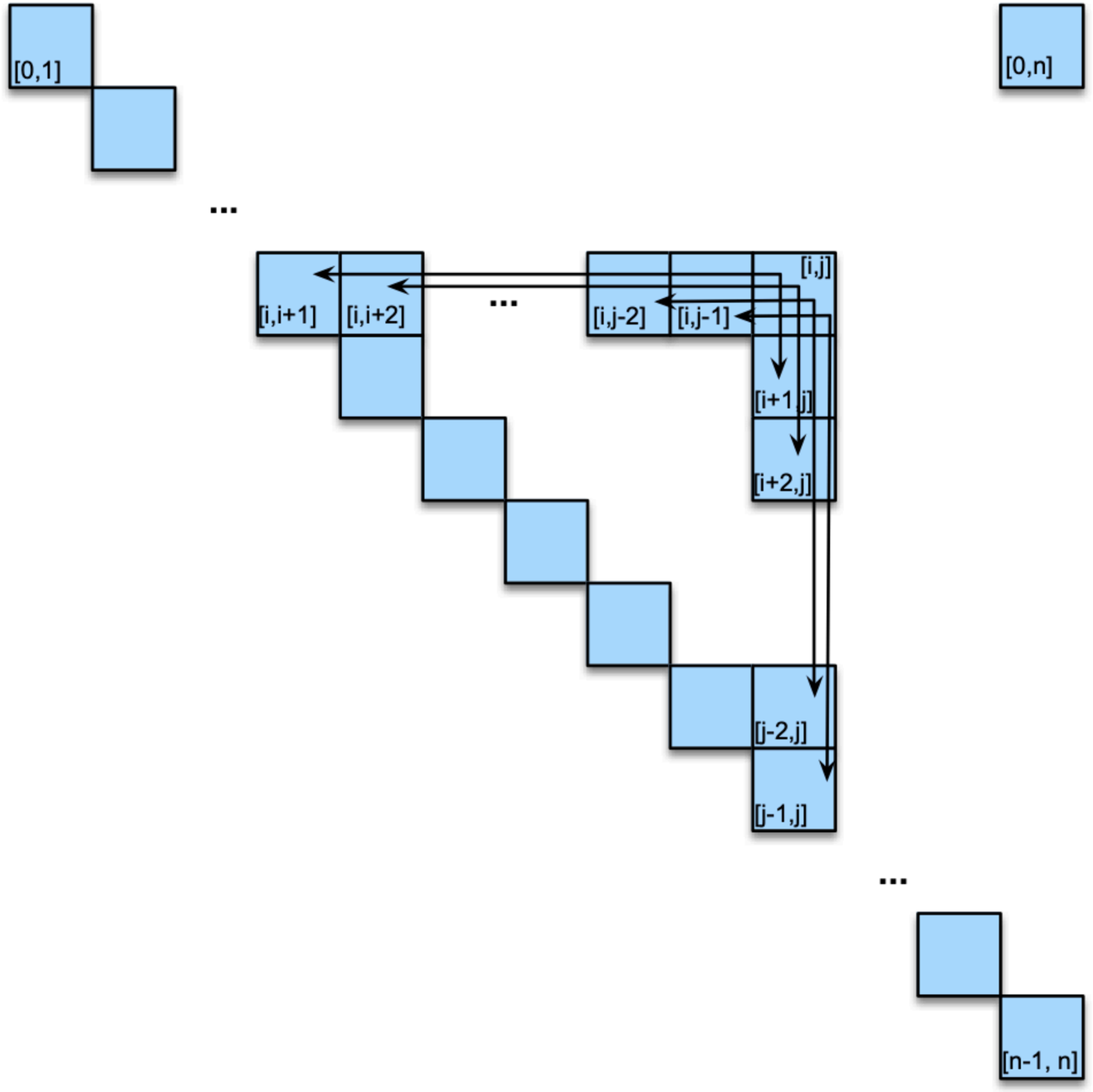
w_1 **$w_2 w_3 w_4 w_5 w_6$** w_7

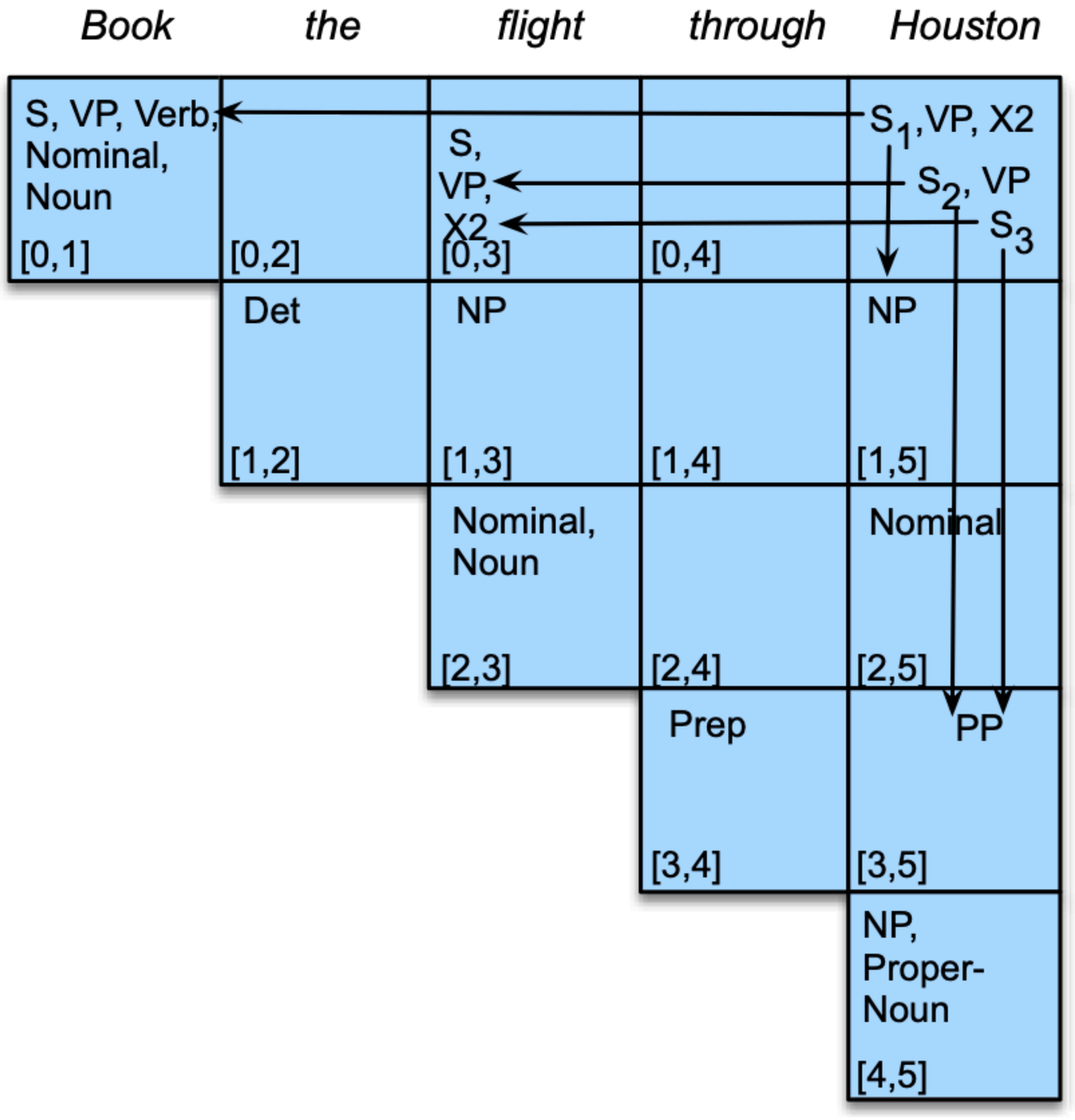


CKY Parsing

- The previous algorithm was a **recognizer**, not a **parser**.
  - It tells us whether any valid parse exists, but not what the valid parses actually are.

- To make this a parser, we have to add **backpointers**.


[0,1]
[0,n]
...
[i,i+1]
[i,i+2]
...
[i,j-2]
[i,j-1]
[i,j]
[i+1,j]
[i+2,j]
[j-2,j]
[j-1,j]
...
[n-1, n]


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
[0,1]
[0,2]
S, VP, X2
[0,3]
[0,4]
S1, VP, X2
S2, VP
S3
Det
[1,2]
NP
[1,3]
[1,4]
NP
[1,5]
Nominal, Noun
[2,3]
[2,4]
Nominal
[2,5]
Prep
[3,4]
PP
[3,5]
NP, Proper-Noun
[4,5]

The CKY Algorithm

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

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

Loop over words in sentence

  for all $\{A \mid A \rightarrow words[j] \in grammar\}$

    $table[j-1, j] \leftarrow table[j-1, j] \cup A$

Loop over rules that can generate this word, add LHS to table

  for $i \leftarrow$ **from** $j-2$ **down to** 0 **do**

    for $k \leftarrow i+1$ **to** $j-1$ **do**

Loop over pairs of non-terminals

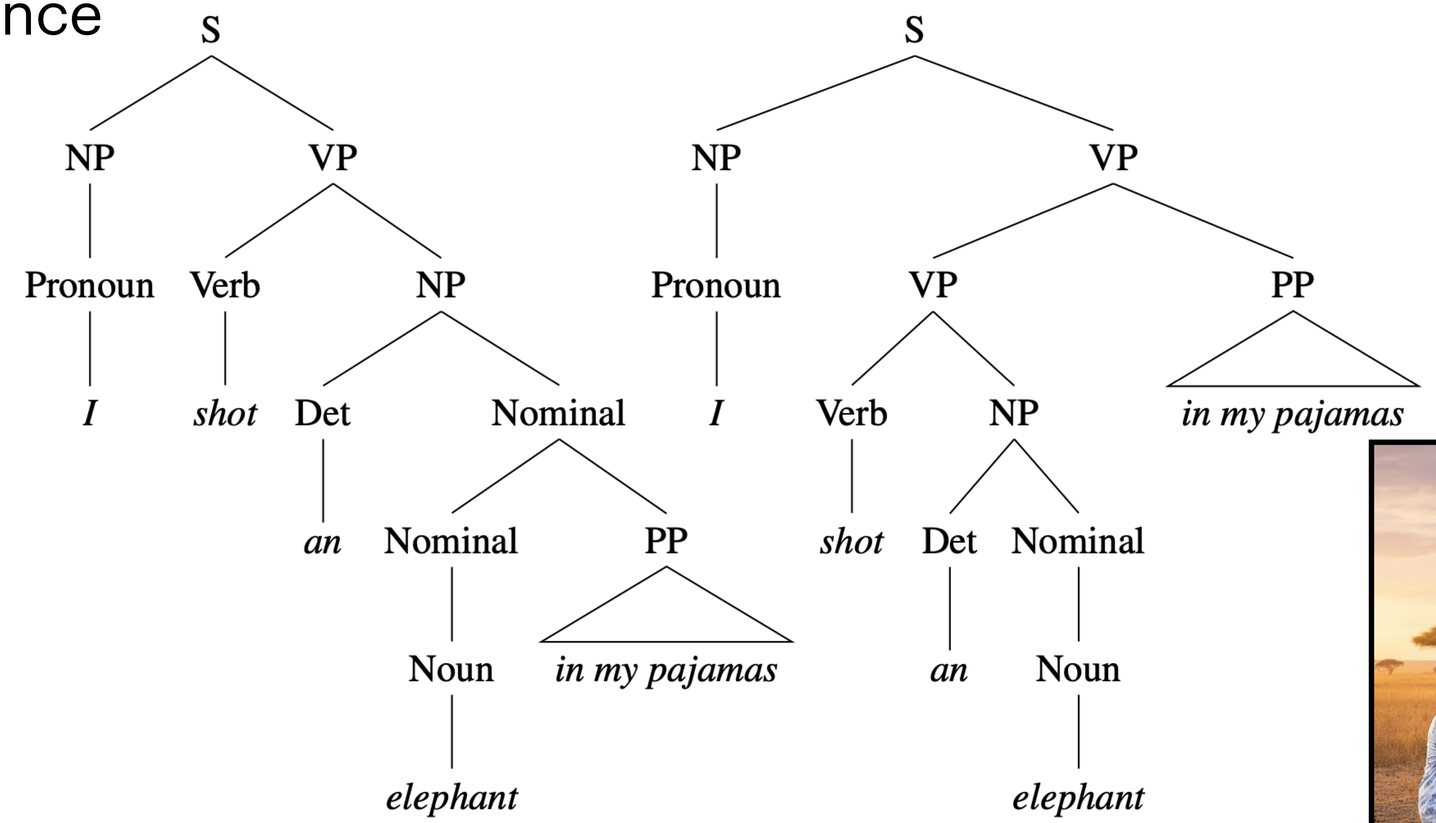
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$

        $table[i,j] \leftarrow table[i,j] \cup A$

If there's a rule that could generate these nonterminals, add LHS to table

Ambiguity

- **Structural ambiguity:** the same grammar can often assign more than one parse to the same sentence



This is an *attachment ambiguity*. It's not clear what the constituent attaches to.

What is the *best* parse?

- Our CKY parser associates allows each table cell [i][j] to be associated with multiple possible pairs of backpointers
  - When we get the parses at the end, we get **all possible parses**.
  - *How can we know which parse is best?*

- *Solution:* associate each rule with a probability. In other words, we'll use **probabilistic context-free grammars** (PCFGs)

Probabilistic Context-free Grammars (PCFGs)

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$ $[p]$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$,
	and p is a number between 0 and 1 expressing $P(\beta \mid A)$
S	a designated **start symbol**

p is the probability that non-terminal A will be expanded into β:

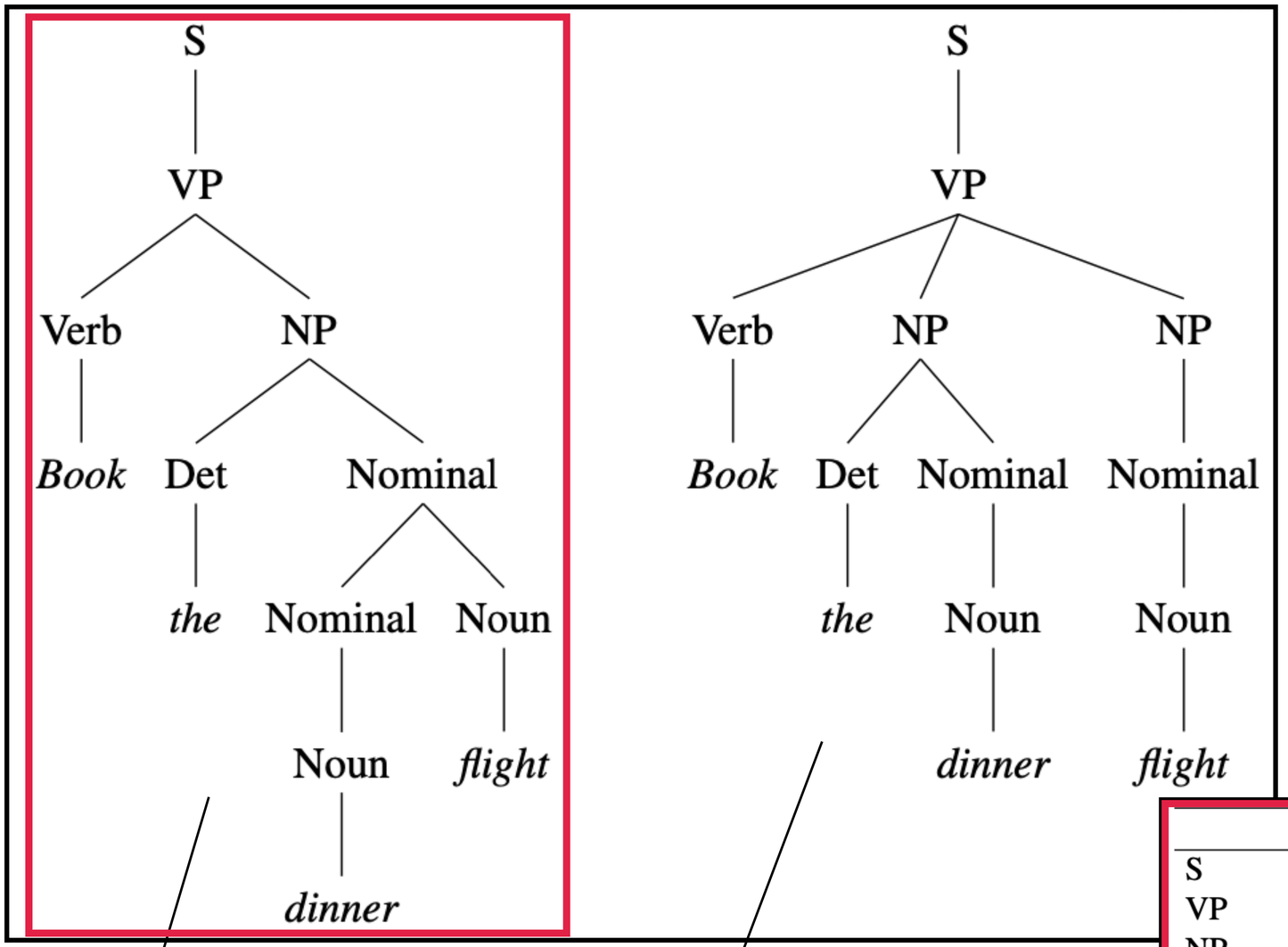
$$p(A \rightarrow \beta) \equiv p(A \rightarrow \beta \mid A) \equiv p(\text{RHS} \mid \text{LHS})$$

Thus, $\sum_{\beta} p(A \rightarrow \beta) = 1.$

Sums to 1

Sums to 1

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\ [.10] \mid a\ [.30] \mid the\ [.60]$
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book\ [.10] \mid trip\ [.30]$
$S \rightarrow VP$	[.05]	$\mid meal\ [.05] \mid money\ [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flight\ [.40] \mid dinner\ [.10]$
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book\ [.30] \mid include\ [.30]$
$NP \rightarrow Det\ Nominal$	[.20]	$\mid prefer\ [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I\ [.40] \mid she\ [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me\ [.15] \mid you\ [.40]$
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston\ [.60]$
$Nominal \rightarrow Nominal\ PP$	[.05]	$\mid NWA\ [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does\ [.60] \mid can\ [.40]$
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from\ [.30] \mid to\ [.30]$
$VP \rightarrow Verb\ NP\ PP$	[.10]	$\mid on\ [.20] \mid near\ [.15]$
$VP \rightarrow Verb\ PP$	[.15]	$\mid through\ [.05]$
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	



Which parse is more probable?

$$p(T,S) = \prod_{i=1}^{n} p(\text{RHS}_i \,|\, \text{LHS}_i)$$

$p = 2.2 \times 10^{-6}$ $p = 6.1 \times 10^{-7}$

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

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

"Book the flight on behalf of the dinner."

"Book the flight that serves dinner."

Picking the Best Parse

- We'll assume the best parse is the most probable one according to the PCFG:

$$\hat{T}(S) = \arg\max_{T} p(T \mid S)$$

- By Bayes' Rule, this can be rewritten as:

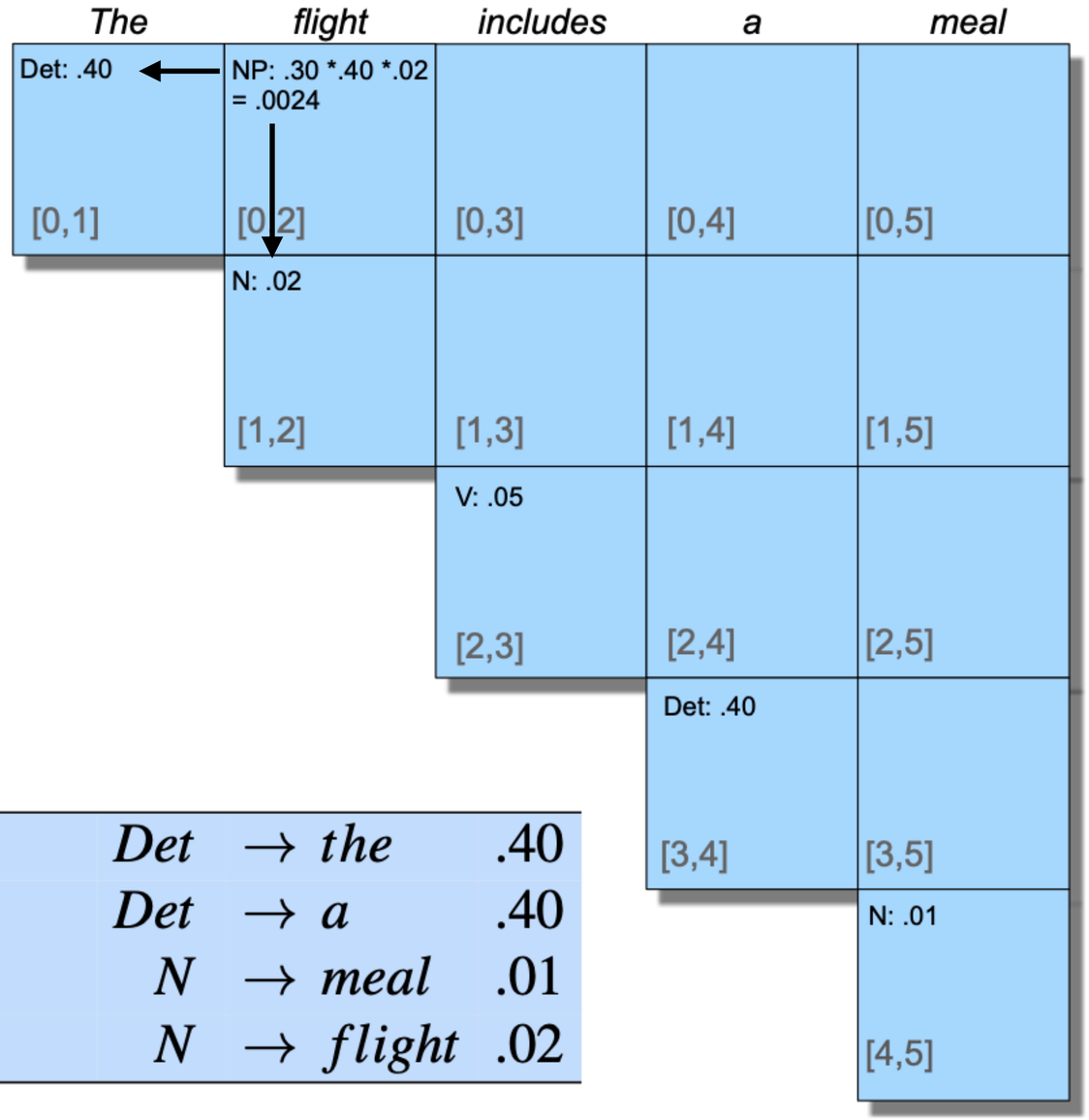
$$\hat{T}(S) = \arg\max_{T} \frac{p(T, S)}{p(S)}$$

- But $p(S)$ is constant for all possible trees, so we just need the numerator:

$$\hat{T}(S) = \arg\max_{T} p(T, S)$$

Probabilistic CKY Parsing

Like CKY, but we also store the probability of each (intermediate) parse



S	$\rightarrow NP\ VP$	.80	Det	$\rightarrow the$	.40
NP	$\rightarrow Det\ N$	.30	Det	$\rightarrow a$	.40
VP	$\rightarrow V\ NP$	.20	N	$\rightarrow meal$	.01
V	$\rightarrow includes$	.05	N	$\rightarrow flight$	.02

Probabilistic CKY Parsing

- We can use the same **Viterbi algorithm** as last time!
    1. Initialize: Every cell containing a terminal (cells on the diagonal) has probability $p(X[i][j]) = 1$
    2. Recurrence: For every non-terminal cell (cells off the diagonal), keep only the highest-scoring backpointer pair
    3. Final step: Use the backpointers to return the Viterbi parse for the start symbol S in the top-right cell [1][n]

Example

Alan	eats	potatoes	with	butter	
Noun 1.0					
	Verb 1.0				
		Noun 1.0			
			Prep 1.0		
				Noun 1.0	

1. Initialize with terminals.

```
S      → NP VP          0.8
S      → S ConjS        0.2
NP     → Noun           0.2
NP     → Det Noun       0.4
NP     → NP PP          0.2
NP     → NP ConjNP      0.2
VP     → Verb           0.3
VP     → Verb NP        0.3
VP     → Verb NPNP      0.1
VP     → VP PP          0.3
PP     → PP NP          1.0
Prep   → P              1.0
Noun   → N              1.0
Verb   → V              1.0
ConjS  → conj S         1.0
ConjNP → conj NP        1.0
NPNP   → NP NP          1.0
```

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 **NP 0.2**					
	Verb 1.0				
		Noun 1.0			
			Prep 1.0		
				Noun 1.0	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2					
	Verb 1.0 **VP 0.3**				
		Noun 1.0			
			Prep 1.0		
				Noun 1.0	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2					
	Verb 1.0 VP 0.3				
		Noun 1.0 **NP 0.2**			
			Prep 1.0		
				Noun 1.0	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2					
	Verb 1.0 VP 0.3				
		Noun 1.0 NP 0.2			
			Prep 1.0		
				Noun 1.0	

2. Recursive steps.

```
S      → NP VP        0.8
S      → S ConjS      0.2
NP     → Noun         0.2
NP     → Det Noun     0.4
NP     → NP PP        0.2
NP     → NP ConjNP    0.2
VP     → Verb         0.3
VP     → Verb NP      0.3
VP     → Verb NPNP    0.1
VP     → VP PP        0.3
PP     → PP NP        1.0
Prep   → P            1.0
Noun   → N            1.0
Verb   → V            1.0
ConjS  → conj S       1.0
ConjNP → conj NP      1.0
NPNP   → NP NP        1.0
```

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2					
	Verb 1.0 VP 0.3				
		Noun 1.0 NP 0.2			
			Prep 1.0		
				Noun 1.0 **NP 0.2**	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 **NP 0.2**	**S** 0.8 * 0.2 * 0.3				
	Verb 1.0 **VP 0.3**				
		Noun 1.0 NP 0.2			
			Prep 1.0		
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3				
	Verb 1.0 VP 0.3	**VP** 0.3 * 1 * 0.2			
		Noun 1.0 **NP 0.2**			
			Prep 1.0		
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3				
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0		
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3				
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0	**PP** 1 * 1 * 0.2	
				Noun 1.0 **NP 0.2**	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 **NP 0.2**	S 0.8 * 0.2 * 0.3	**S** 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	**VP** 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2			
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 **NP 0.2**		**NP** 0.2 * 0.2 * 0.2	
			Prep 1.0	**PP** **1 * 1 * 0.2**	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2			
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06			
	Verb 1.0 VP 0.3	**VP** **0.3 * 1 * 0.2**		**VP** **1 * 0.008 * 0.3** **VP** **0.06 * 0.2 * 0.3**	
		Noun 1.0 NP 0.2		**NP** **0.2 * 0.2 * 0.2**	
			Prep 1.0	**PP** **1 * 1 * 0.2**	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 **NP 0.2**	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06		**S** 0.2 * 0.0024 * 0.8 **S** 0.2 * 0.0036 * 0.8	
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2		**VP 0.0024** **VP 0.0036**	
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06		**s** 0.2 * 0.0036 * 0.8	
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2		VP 0.0036	
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06		**s** 0.2 * 0.0036 * 0.8	
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2		VP 0.0036	
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

```
S      → NP VP        0.8
S      → S ConjS      0.2
NP     → Noun         0.2
NP     → Det Noun     0.4
NP     → NP PP        0.2
NP     → NP ConjNP    0.2
VP     → Verb         0.3
VP     → Verb NP      0.3
VP     → Verb NPNP    0.1
VP     → VP PP        0.3
PP     → PP NP        1.0
Prep   → P            1.0
Noun   → N            1.0
Verb   → V            1.0
ConjS  → conj S       1.0
ConjNP → conj NP      1.0
NPNP   → NP NP        1.0
```

Example

Alan	eats	potatoes	with	butter	
Noun 1.0 NP 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06		**s** 0.2 * 0.0036 * 0.8	
	Verb 1.0 VP 0.3	VP 0.3 * 1 * 0.2		VP 0.0036	
		Noun 1.0 NP 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun 1.0 NP 0.2	

2. Recursive steps.

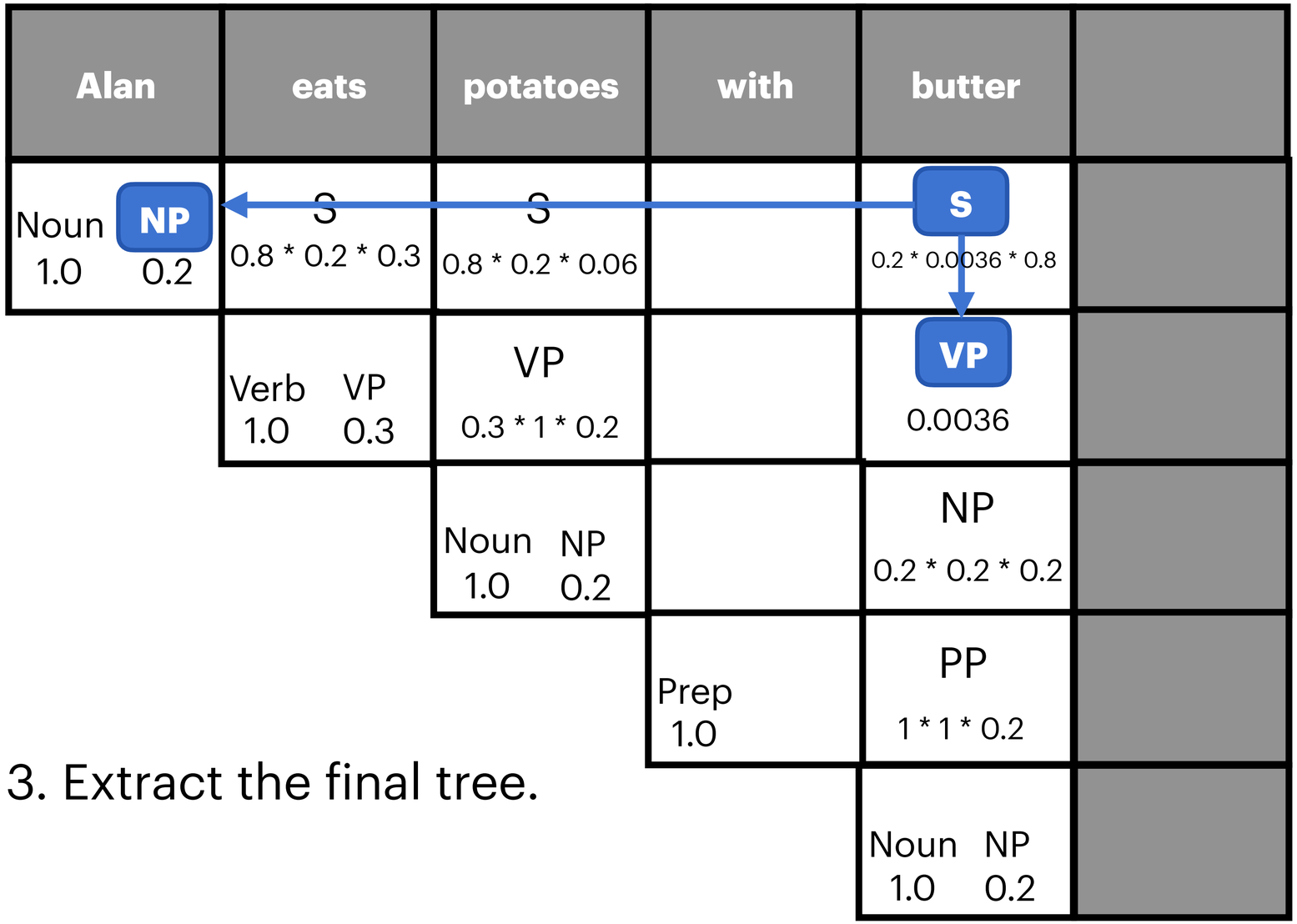
S	→	NP VP	0.8
S	→	S ConjS	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP ConjNP	0.2
VP	→	Verb	0.3
VP	→	Verb NP	0.3
VP	→	Verb NPNP	0.1
VP	→	VP PP	0.3
PP	→	PP NP	1.0
Prep	→	P	1.0
Noun	→	N	1.0
Verb	→	V	1.0
ConjS	→	conj S	1.0
ConjNP	→	conj NP	1.0
NPNP	→	NP NP	1.0

Example

Alan	eats	potatoes	with	butter	
Noun NP 1.0 0.2	S 0.8 * 0.2 * 0.3	S 0.8 * 0.2 * 0.06		S 0.2 * 0.0036 * 0.8	
	Verb VP 1.0 0.3	VP 0.3 * 1 * 0.2		VP 0.0036	
		Noun NP 1.0 0.2		NP 0.2 * 0.2 * 0.2	
			Prep 1.0	PP 1 * 1 * 0.2	
				Noun NP 1.0 0.2	

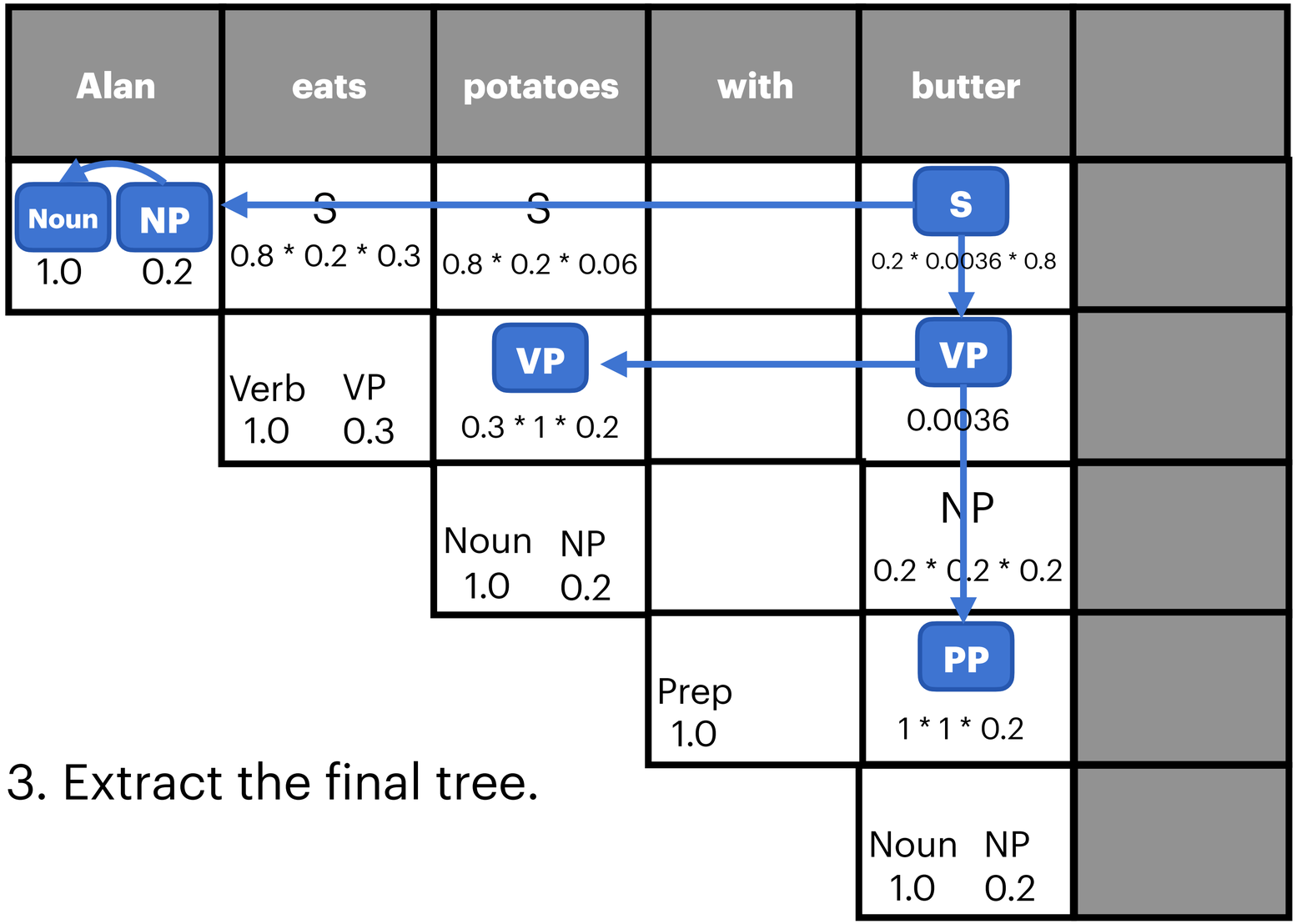
3. Extract the final tree.

Example



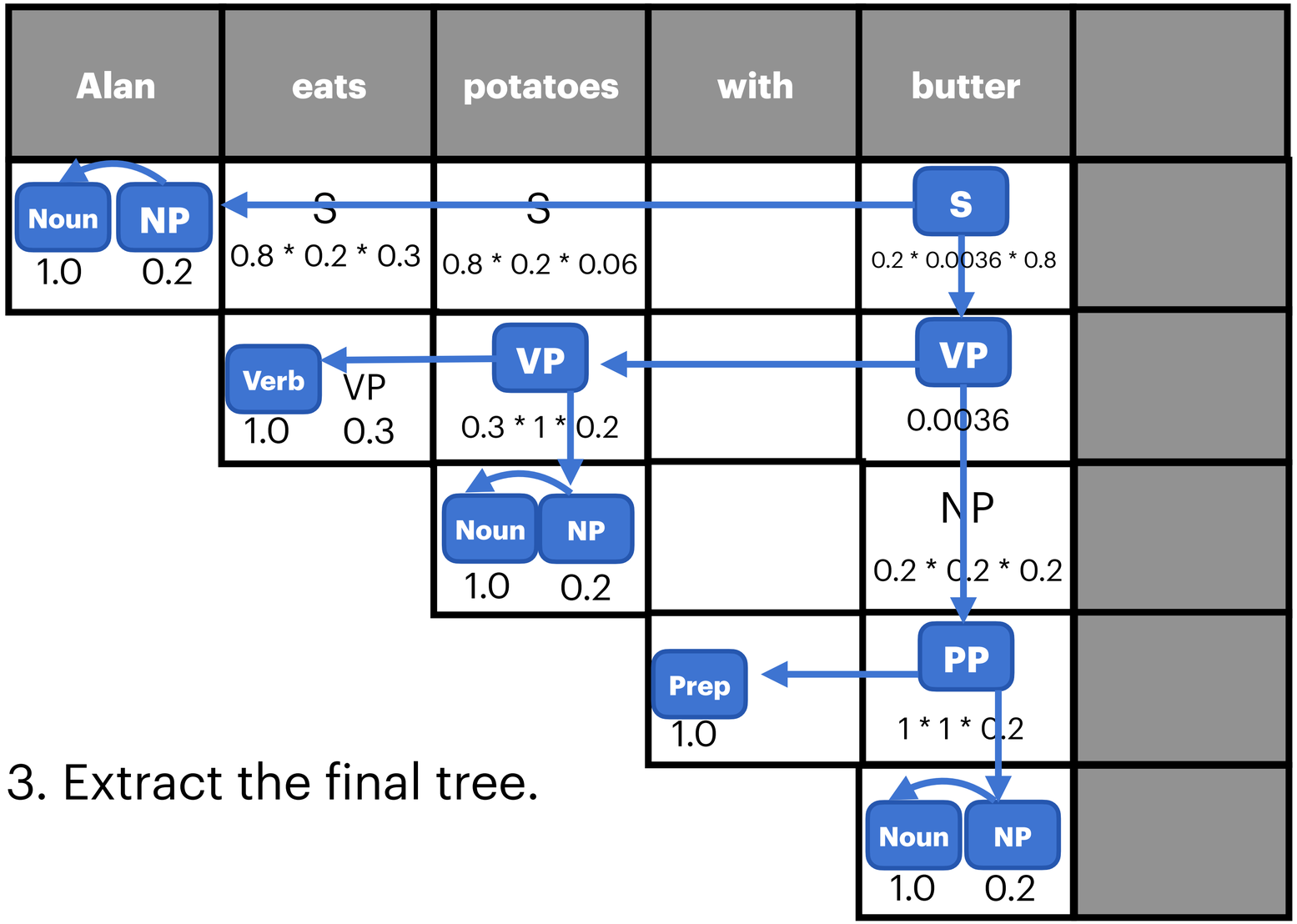
3. Extract the final tree.

Example

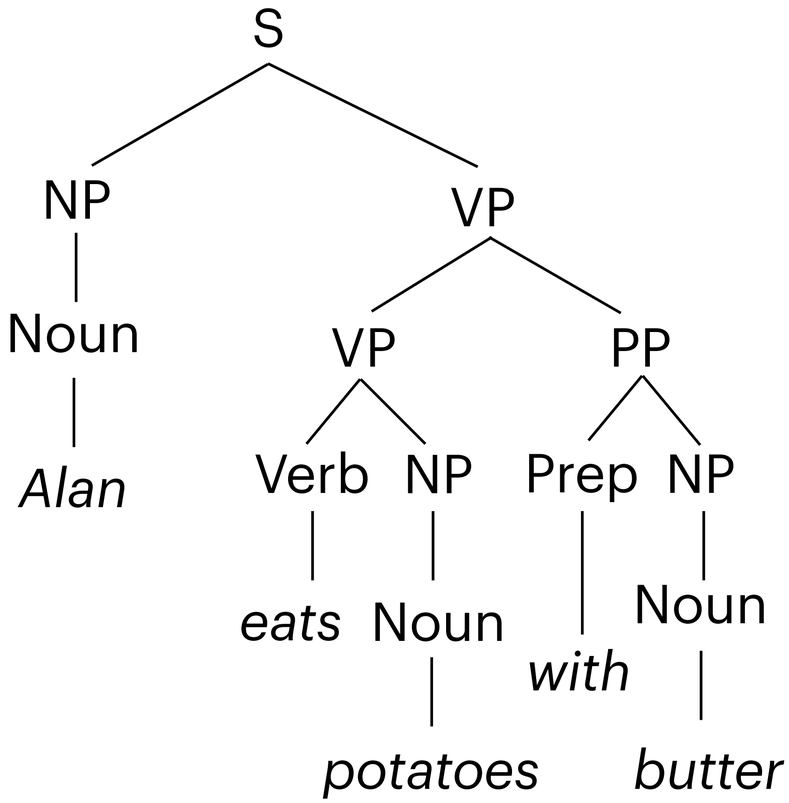


3. Extract the final tree.

Example



3. Extract the final tree.



Learning PCFG Probabilities

- A **treebank** is a dataset where every sentence is provided along with its parse.
- We can learn PCFG probabilities using a very similar approach to the one we used for learning n-gram probabilities:

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

```
((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 )))))
```

$$p(\alpha \rightarrow \beta \,|\, \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- It's also possible to learn PCFG probabilities without a treebank, but we won't cover methods for doing this

Generative Grammar

- Using formal languages with finite rules to model natural language's infinite possible sentences is called **generative grammar**.
    - The language is defined by the set of possible sentences that can be "generated" by the grammar.
- Advocates of this theory typically hold that there are some universal rules ("Universal Grammar") governing the structure of language that are biologically innate
- Still among the most common theories of syntax

Noam Chomsky

Problems with PCFGs

- PCFGs are very useful, but they have two big issues:

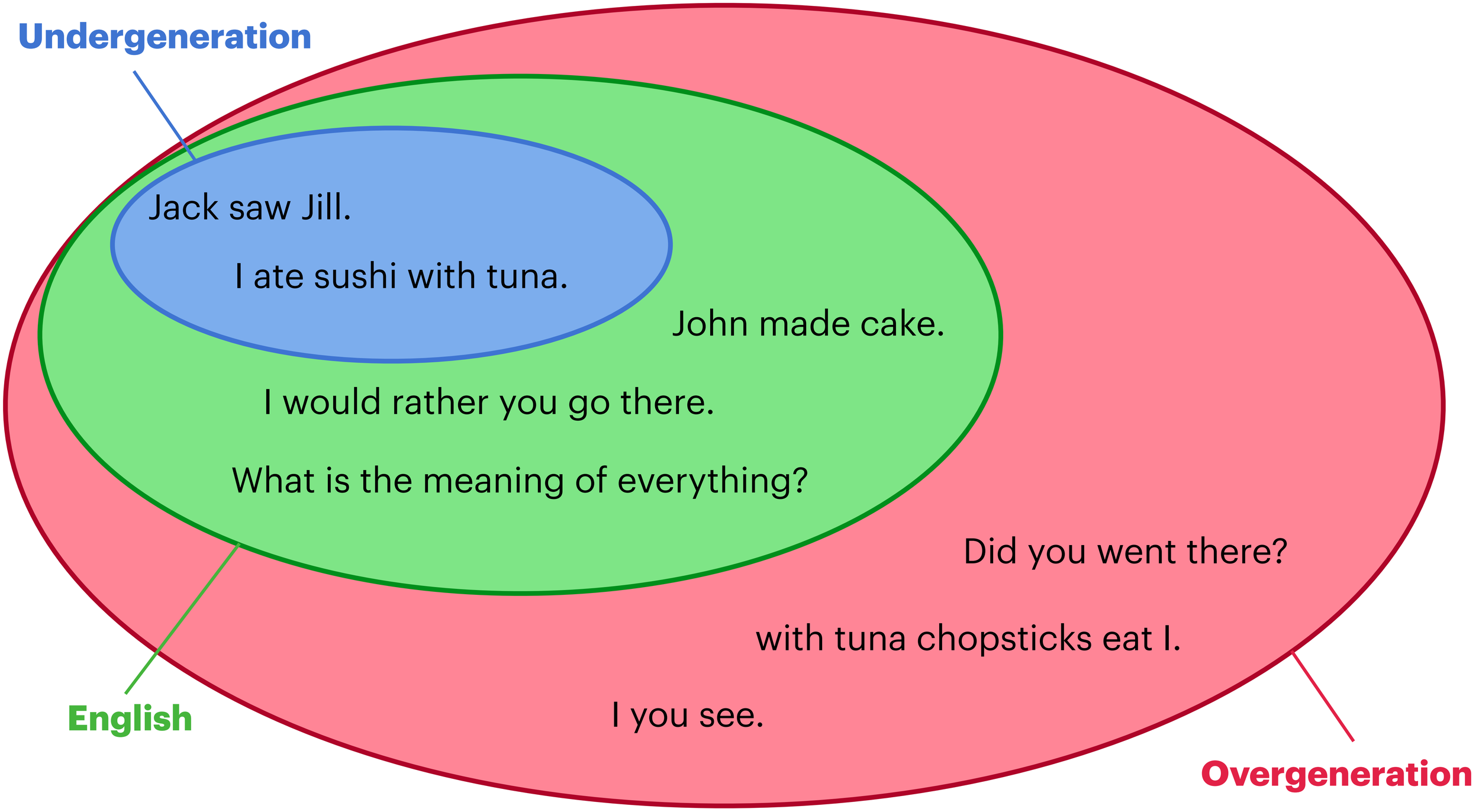
- **Poor independence assumptions:** CFG rules assume that the label of a node determines what children it can have. *This does not always work well.*

- **Lack of lexical conditioning**: CFG rules don't model facts about specific words' syntactic distributions. *Certain words might not be compatible with certain syntactic structures.*

Ways to Improve PCFGs

1. Change the grammar:
   - Parent-based annotations: not all NPs/VPs/etc. are the same—it matters where they are in a tree

2. Change the probability model:
   - **Lexicalization:** condition on specific words
   - **Markovization:** generalize the rules


Undergeneration
Jack saw Jill.
I ate sushi with tuna.
John made cake.
I would rather you go there.
What is the meaning of everything?
Did you went there?
with tuna chopsticks eat I.
I you see.
English
Overgeneration

Subject–Verb Agreement

He eats food.

*I eats food.

*They eats food.

(In linguistics papers, * before a sentence means “this is ungrammatical.”)

$S \rightarrow NP_{3sg}\ VP_{3sg}$

$S \rightarrow NP_{1sg}\ VP_{1sg}$

$S \rightarrow NP_{3pl}\ VP_{3pl}$

Need attributes to capture agreement:
number, person, case, ...

Subordination

He says [he eats food].

He says [that [he eats food]].

$VP \rightarrow V_{comp}\ S$

$VP \rightarrow V_{comp}\ SBAR$

$SBAR \rightarrow COMP\ S$

$V_{comp} \rightarrow says \mid think \mid believes$

$COMP \rightarrow that$

Relative Clauses

- Relative clauses modify noun phrases:

  the man [that eats food] NP -> NP RelC

- Relative clauses do not have an NP.
- **Subject relative clauses** lack a subject: “the man that eats food”

  RelC -> RelPron VP

- **Object relative clauses** lack an object: “the food that the man eats”.
  - Define “slash notation”: S\NP, VP\NP for sentences lacking an object NP

RelC → RelPron S\NP VP\NP → V_{trans}

S\NP → NP VP\NP VP\NP → VP\NP PP

Yes/No and Wh-Questions

- Yes/no questions consist of an auxiliary, a subject and an uninflected VP:

Does he eat food?

Have you eaten food?

YesNoQ → Aux NP VP_{inf}

YesNoQ → Aux NP $VP_{pastPart}$

Who has eaten food?

What does he eat?

WhQ → WhPron Aux $VP_{pastPart}$

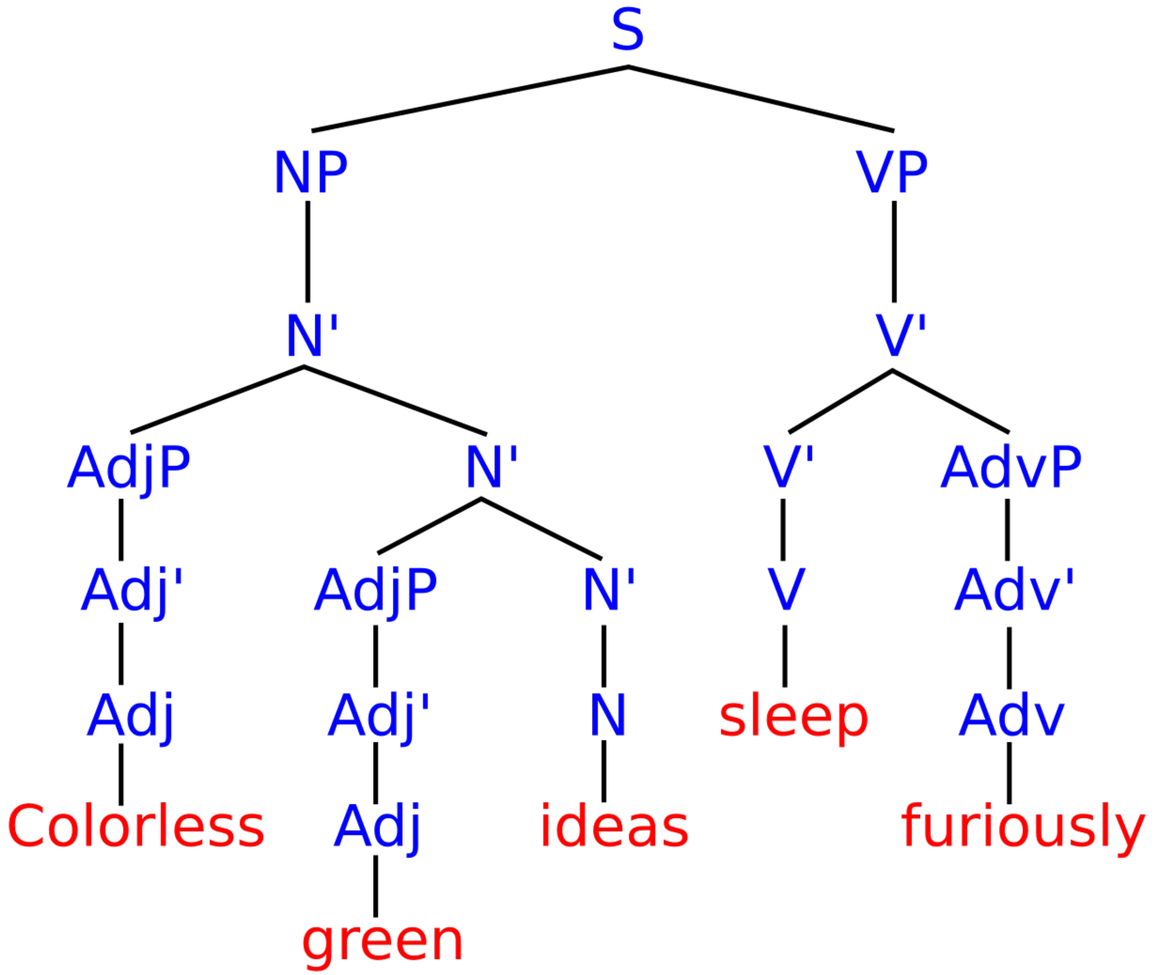
WhQ → WhPron Aux NP VP_{inf}\NP

Grammatical ≠ Acceptable ≠ Meaningful

- If you ask a person whether a sentence is grammatical, they actually will just tell you whether it's **acceptable**:

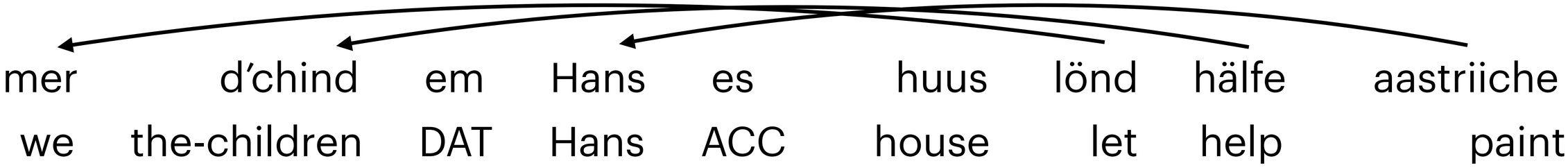
*? Colorless green ideas sleep furiously.

- **Acceptability:** whether a native speaker of the language would consider the sentence valid
- **Grammaticality:** whether a sentence conforms to a predefined formalism



Language is context-*sensitive*.

- Human language is not totally context-free **[Shieber, 1985]**
- We know this because cross-serial (crossing) dependencies are allowed in some languages:



- Cross-serial dependencies are *not possible* in fully context-free languages
  - Thus, natural language is slightly *context-sensitive*