Queen Mary
University of London

ECS763 Natural Language Processing

Unit 7: Syntax and Parsing

Lecturer: Julian Hough
School of Electronic Engineering and Computer Science

OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

Why are sequence models not enough for natural language?

- We have looked at ways to model the probability of word/category sequences and how those models work in sequence to sequence classification.

- All these models have used notions of linear, contiguous dependency from one word to the next:
  - HMMs and n-gram Language Models used the **Markov assumption**: current word/category label is only dependent on *k* previous words/categories.
  - CRFs use a fixed contiguous **window** of words of width *k* to get a category.

- How would they capture the following sentence even if k=10?:
  - "the computer I just put into the machine room on the fifth floor crashed"

Why are sequence models not enough for natural language?

KEY POINT:

Language has hierarchical structure

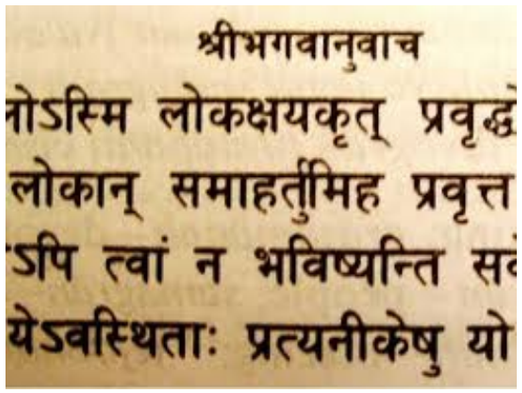
Why are sequence models not enough for natural language?

KEY POINT:

Formal grammars were designed to capture the structure of NL

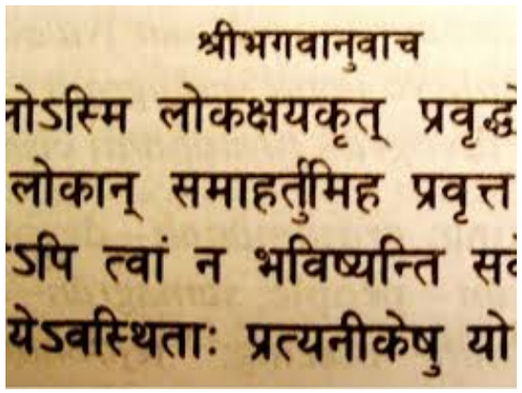
A bit of history of Formal Grammar

- The first formal grammar was written over 2000 years ago for Sanskrit by *Panini*. But it is still referenced today when teaching Sanskrit and studying its grammar.



A bit of history of Formal Grammar

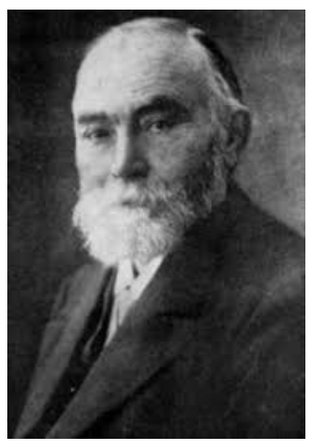
- Panini's book had just under 4000 commented algebraic statements, an example of which is the following one:



Sanskrit has ten classes of verbs divided into two broad groups: athematic and thematic. The thematic verbs are so called because an 'a', called the theme vowel, is inserted between the stem and the ending. This serves to make the thematic verbs generally more regular.

- Earliest work in **formal syntax:** e.g. characterising **the ways words are arranged together** in the sentences and other constructions of language.

A bit of history of Formal Grammar



- Modern founders of **formal** approaches to language were **G. Frege** (1848-1925) and **F. de Saussure** (1857-1913).

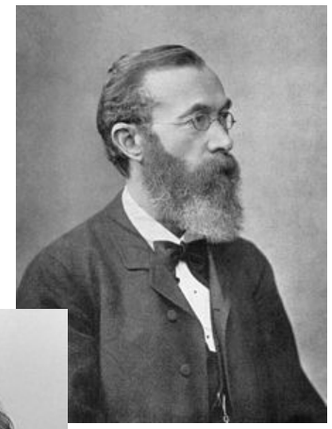
- Frege introduced predicate logic and used it to do a functional analysis of language using diagrams.

- He was also the first person who introduced a, now widely used, **principle of compositionality:**

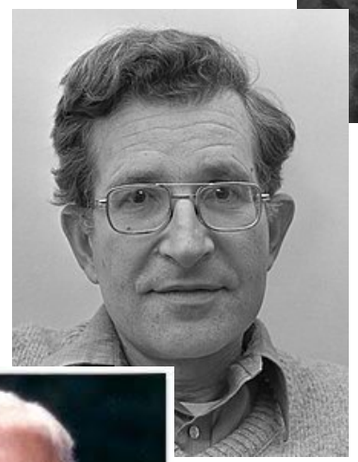
  The meaning of a complex term is a function of meanings of its parts.

Generative Grammars

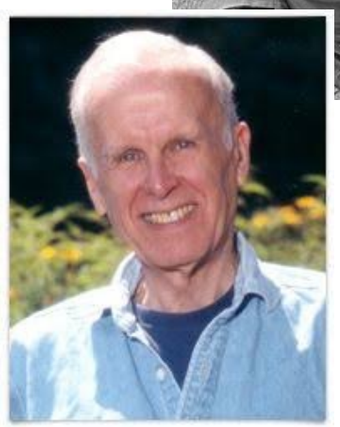
In linguistics, enumerating the rules which generate and accept only and all of the strings of a natural language is traditionally called a **generative grammar.**



The most widely used formal grammars were motivated by the work of one of the fathers of psychology:
Wilhem Wundt in 1900.



A linguist formalised these ideas using a set of generative rules, starting modern linguistics:
Noam Chomsky in 1955-57



They were later also expressed by a computer scientist:
J. W. Backus in 1959.

Generative Grammars

What is the best system/set of rules (grammar) which generates strings/sentences **in** the language?

On September seventeenth, I'd like to fly from Atlanta to Denver
I'd like to fly *on September seventeenth* from Atlanta to Denver
I'd like to fly from Atlanta to Denver *on September seventeenth*

…and rules out those strings **not in** the language (i.e. ungrammatical strings)?

*On September, I'd like to fly seventeenth from Atlanta to Denver
*On I'd like to fly September seventeenth from Atlanta to Denver
*I'd like to fly on September from Atlanta to Denver seventeenth

Generative Grammars

KEY POINT:
The main concept underlying generative grammars is **constituency**.

Formal Grammar

- Groups of words that behave as a single unit are called a **constituent.**
- A significant part of developing a grammar/grammar engineering involves discovering the constituents present in a particular corpus, or in a language more generally.

Formal Grammar

Examples of Constituents in English:

Noun Phrases

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

How do we know which words group together?

- They appear in similar syntactic positions, e.g. before a verb:

three parties from Brooklyn *arrive*...
a high-class spot such as Mindy's *attracts*...
the Broadway coppers *love*...
they *sit*

- But not true for the individual words in the constituent. It doesn't make sense to say:

*from *arrive*...	*as *attracts*...
*the *is*...	*spot *sat*...

Formal Grammar

Examples of Constituents in English:

Prepositional Phrases

On September seventeenth,

- They appear in different syntactic positions in the sentence:

On September seventeenth, I'd like to fly from Atlanta to Denver
I'd like to fly *on September seventeenth* from Atlanta to Denver
I'd like to fly from Atlanta to Denver *on September seventeenth*

- But again, not true for the individual words in the constituent. It doesn't make sense to say:

*On September, I'd like to fly seventeenth from Atlanta to Denver
*On I'd like to fly September seventeenth from Atlanta to Denver
*I'd like to fly on September from Atlanta to Denver seventeenth

Formal Grammar

Grammatical Relations

These are relationships between the constituents. Examples are **Subject** and **Object.**

For example in the sentence:

"She adores the deep blue sky",

"she" and "the deep blue sky" are noun phrase constituents that are the **subject** and the **object** of verb "adores".

Formal Grammar

Dependency Relations

Special types of **relations** between **words and phrases**.

e.g. the verb “want” can be followed by an **infinitive**:

“I want to sleep.”

It can also be followed by a noun phrase:

“I want a sleeping bag.”

This is not the case for all verbs, for example the verb “find", cannot be followed by an infinitive. One cannot say:

*“I find to fly to Edinburgh.”

OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

Context Free Grammars

Context free grammars (from the **Chomsky hierarchy** of formal languages) are the backbone of many formal models of syntax.

Reasons:

1- **Powerful** enough to formalise (most) relationships between words in a sentence.

2- **Computationally tractable enough** for implementation, so there exists lots of parsing algorithms and tools for them.

Context Free Grammars

Context free grammars or **CFG's** are also called

Phrase Structure Grammars.

Their formalisation is similar to

Backus-Naur Form (BNF)

Context Free Grammars

A CFG has:

a set of **production rules:** how constituents of language are grouped and ordered together

a lexicon: determining which constituents words of a language are part of.

Context Free Grammars

A sample lexicon:

Noun →	*flights* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

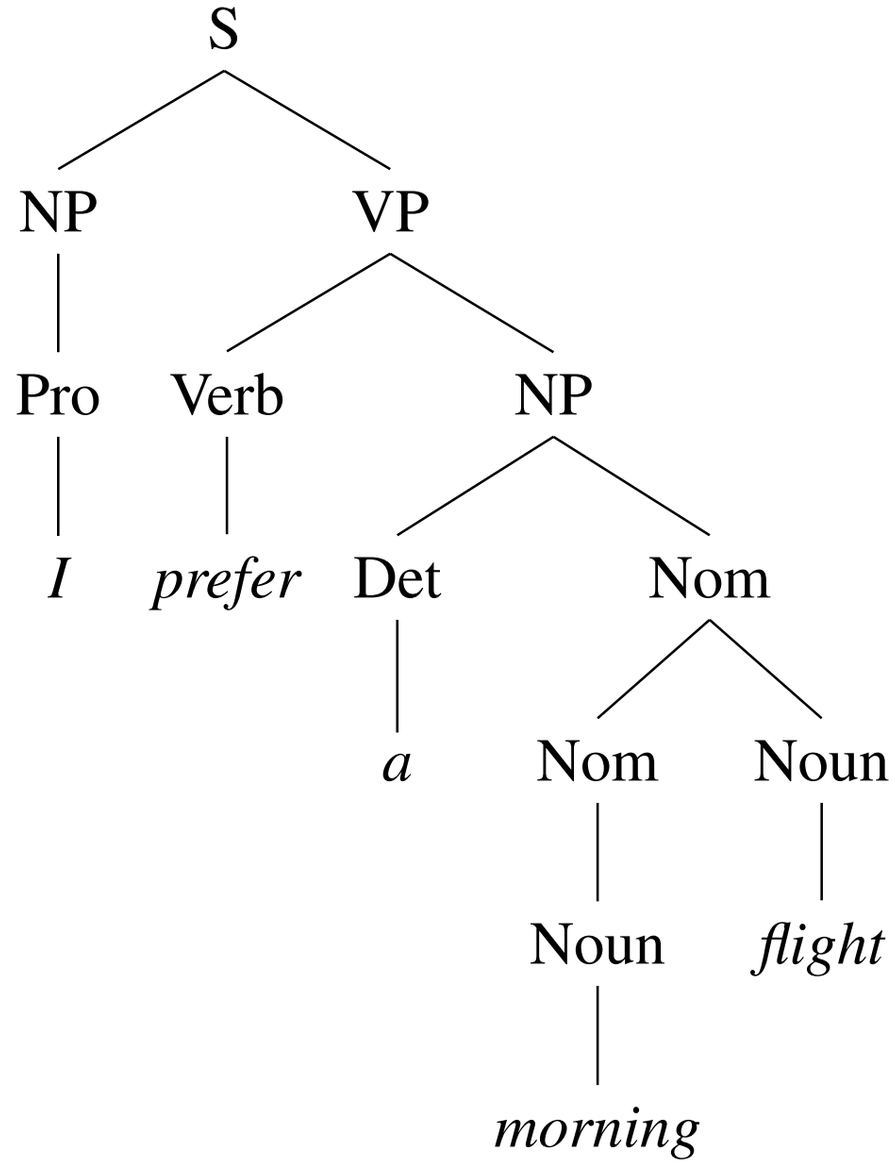
Context Free Grammars

Some production/grammar rules:

Grammar Rules			Examples
S	$\rightarrow$	*NP VP*	I + want a morning flight
NP	$\rightarrow$	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	$\rightarrow$	*Preposition NP*	from + Los Angeles

Context Free Grammars

Multiple applications of the rules can lead to a **derivation (or parse) tree** showing how the grammar rules derive the sentence, e.g.
“I prefer a morning flight.”

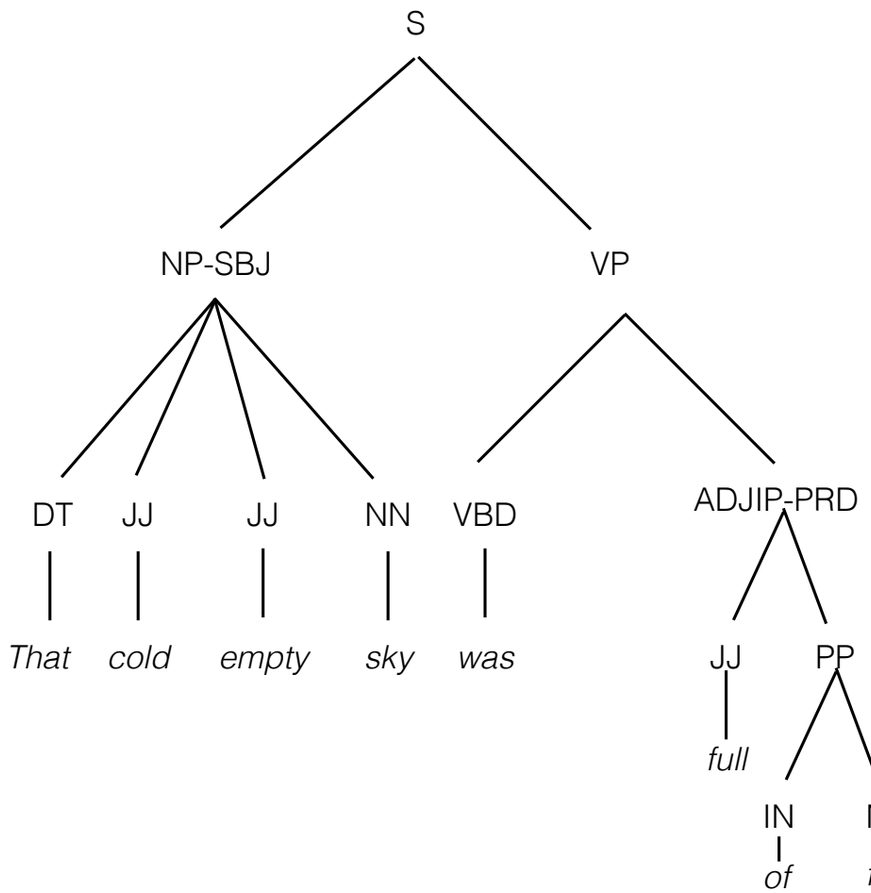


Context Free Grammars

The **bracketed notation** for the parse tree:

[$_S$ [$_{NP}$ [$_{Pro}$ I]] [$_{VP}$ [$_V$ prefer] [$_{NP}$ [$_{Det}$ a] [$_{Nom}$ [$_N$ morning] [$_{Nom}$ [$_N$ flight]]]]]]

A more complicated example:



“The cold, empty sky was full of fire”

A more complicated example
(in an example treebank form):

((S
(NP-SBJ (DT The)
(JJ long) (, ,)
(JJ lonely) (NN night))
(VP (VBD is)
(ADJP-PRD (JJ full)
(PP (IN of)
(NP (NN stars)
(CC and)
(NN moonlight)))))
(. .)))

The long, lonely night is full of stars and moonlight.

Formal Definition of a CFG

Formal definition of a CFG:

$$(N, \Sigma, R, S)$$

N **a set of non-terminal symbols**

Σ **a set of terminal symbols (disjoint from N)**

S **a designated start symbol**

R **a set of production rules of the form** $\alpha \to \beta$

α **a non-terminal**

β **a string of symbols from the strings** $(\Sigma \cup N)^*$

Formal Definition of a CFG

Formal definition of a CFG:

$$(N, \Sigma, R, S)$$

Notational conventions:

Capital letters like A, B, and S	Non-terminals
S	The start symbol
Lower-case Greek letters like α, β, and γ	Strings drawn from $(\Sigma \cup N)*$
Lower-case Roman letters like u, v, and w	Strings of terminals

Formal Definition of a CFG

The symbols of a CFG are classified into two groups:

1- <u>Terminals</u>:

These correspond to the words of language.
The words are introduced via these rules in the lexicon.
e.g. flight, morning, star, a, the, this, that

2- <u>Non-Terminals</u>:

Symbols that express generalisations of these.
a.g. S, NP, VP, Noun, Det

(and of course the -> symbol)

Formal Definition of a CFG

In a CFG:

The items to the right of **->** are:
an ordered list of one or more T's or NT's

The item to the left of **->** is a
a single NT

This is what makes the language generated by these rules **context free -** there is no context for the application of the rule: **on the left hand side we only have a single NT.**

Formal Definition of a CFG

A language is defined through the concept of **derivation.**
A string **derives** another if it can be **rewritten** as the second one by a series of rule applications.

If $A \to \beta$ is a production rule and α and γ are any two strings in $(\Sigma \cup N)^*$, then we say:

$\alpha A \gamma$ **directly derives** $\alpha \beta \gamma$

This is more formally denoted by:

$$\alpha A \gamma \implies \alpha \beta \gamma$$

obtained by substituting A by β.

Formal Definition of a CFG

A derivation is a generalisation of a direct derivation.If we have $\alpha_1 \implies \alpha_2, \alpha_2 \implies \alpha_3, \cdots, \alpha_{n-1} \implies \alpha_n$

then, we say:

α_1 **derives** α_n

and formally write:

$$\alpha_1 \overset{*}{\implies} \alpha_n$$

Parsing is the problem of mapping a string of words to its derivation.
(More later…)

Formal Definition of a CFG

- The **language** generated by a CFG is the set of strings composed of terminals that can be derived from the designated start symbol. A **context-free language** is therefore:

$$\mathcal{L}_{CFG} = \left\{ w \mid w \in \Sigma^* \quad \text{and} \quad S \overset{*}{\Longrightarrow} w \right\}$$

Formal Definition of a CFG

- Sentences (strings of words) that can be derived by a grammar are in the language defined by that grammar, and are called **grammatical sentences.**

- Sentences that cannot be derived by a given grammar are not in the language defined by that grammar and are called **ungrammatical**.

- In linguistics, this is called a **generative grammar** since the language is defined the set of possible sentences "generated" by the grammar.

- Later we will go beyond 'in' and 'out' binary notions of grammaticality and move towards **probabilistic notions of grammaticality**.

More sentence types

Imperative: S -> VP	Show the lowest fare Give me the morning flights
Yes-No Question: S -> Aux NP VP	Does American fly to Boston?
Wh Question: S -> Wh-NP VP	What airlines fly to Boston?

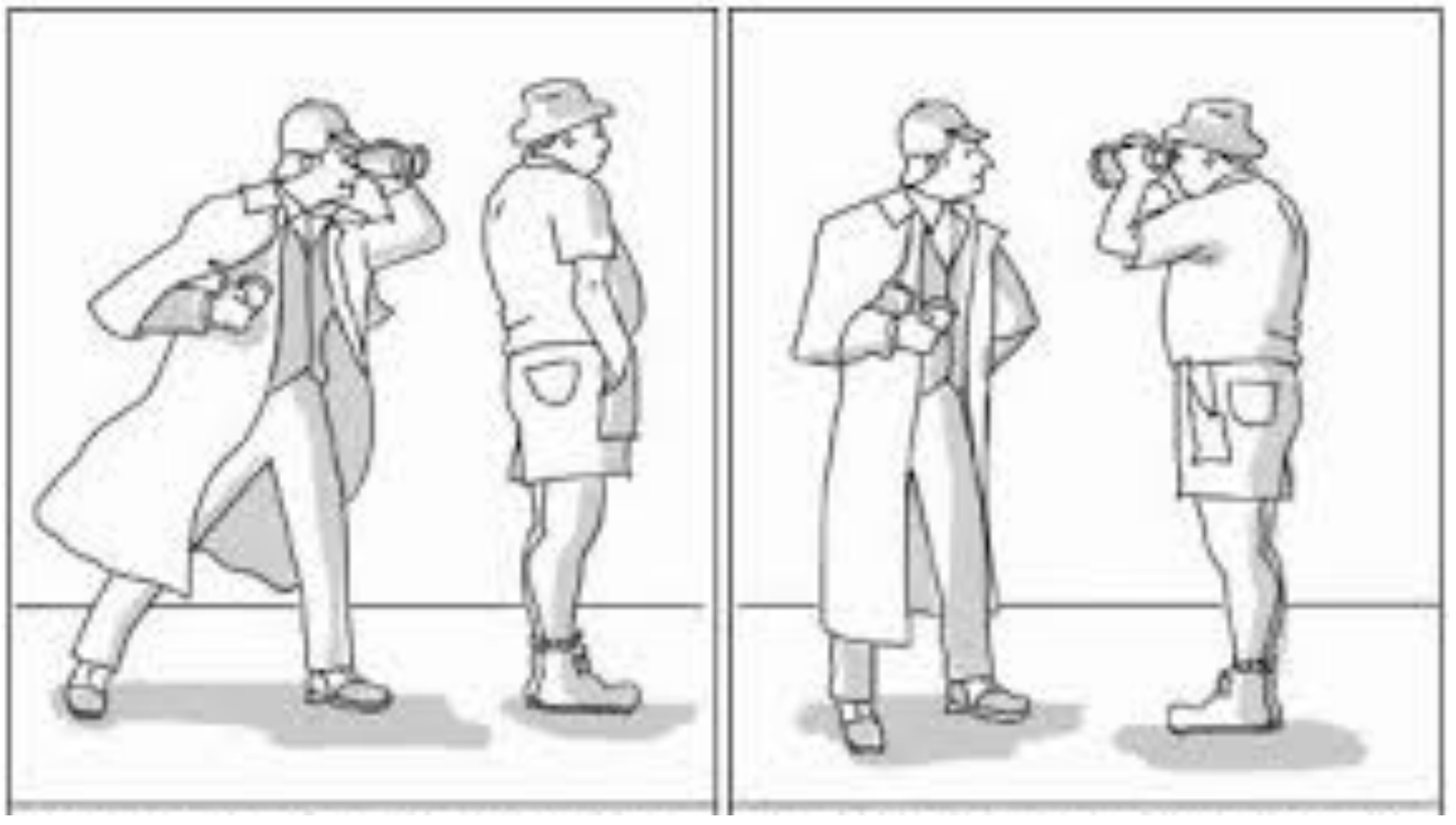
OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

Ambiguity

Syntactic Ambiguity
giving rise to Meaning (Semantic) Ambiguity

Sherlock saw a man with binoculars



Syntactic Ambiguity with Generative Grammars

Sherlock saw a man with binoculars.

S ⟶ Sherlock VP

VP ⟶ saw a man with binoculars

tV ⟶ saw

NP ⟶ a man with binoculars

Meaning 1

Syntactic Ambiguity with Generative Grammars

Sherlock saw a man with binoculars.

S ⟶ Sherlock VP PP

VP ⟶ saw a man

tV ⟶ saw

NP ⟶ a man

PP ⟶ with binoculars

Prepositional Phrase

Meaning 2

Ambiguity

- **How can we deal with the syntactic (and therefore semantic/meaning) ambiguity of the parse of 'Sherlock saw a man with binoculars'?**

- **We need to assign more than one possible structure!**

Syntactic Parsing

Parsing: The task of assigning a syntactic structure to a sentence.

<u>Applications in NLP</u>

- Grammar checking in word processing tools.
- **Semantic analysis,** which has applications to
  - Question answering
  - Information extraction
  - Dialogue systems (intent recognition)

Challenges to Parsing:
Ambiguity

Two kinds:

- **attachment ambiguity:** a constituent can be attached to the parse tree at more than one place.

- **coordination ambiguity:** different sets of phrase can be conjoined by a conjunction, e.g. and/or

Later on: Semantic Ambiguity (meanings of words)

Challenges to Parsing: Example

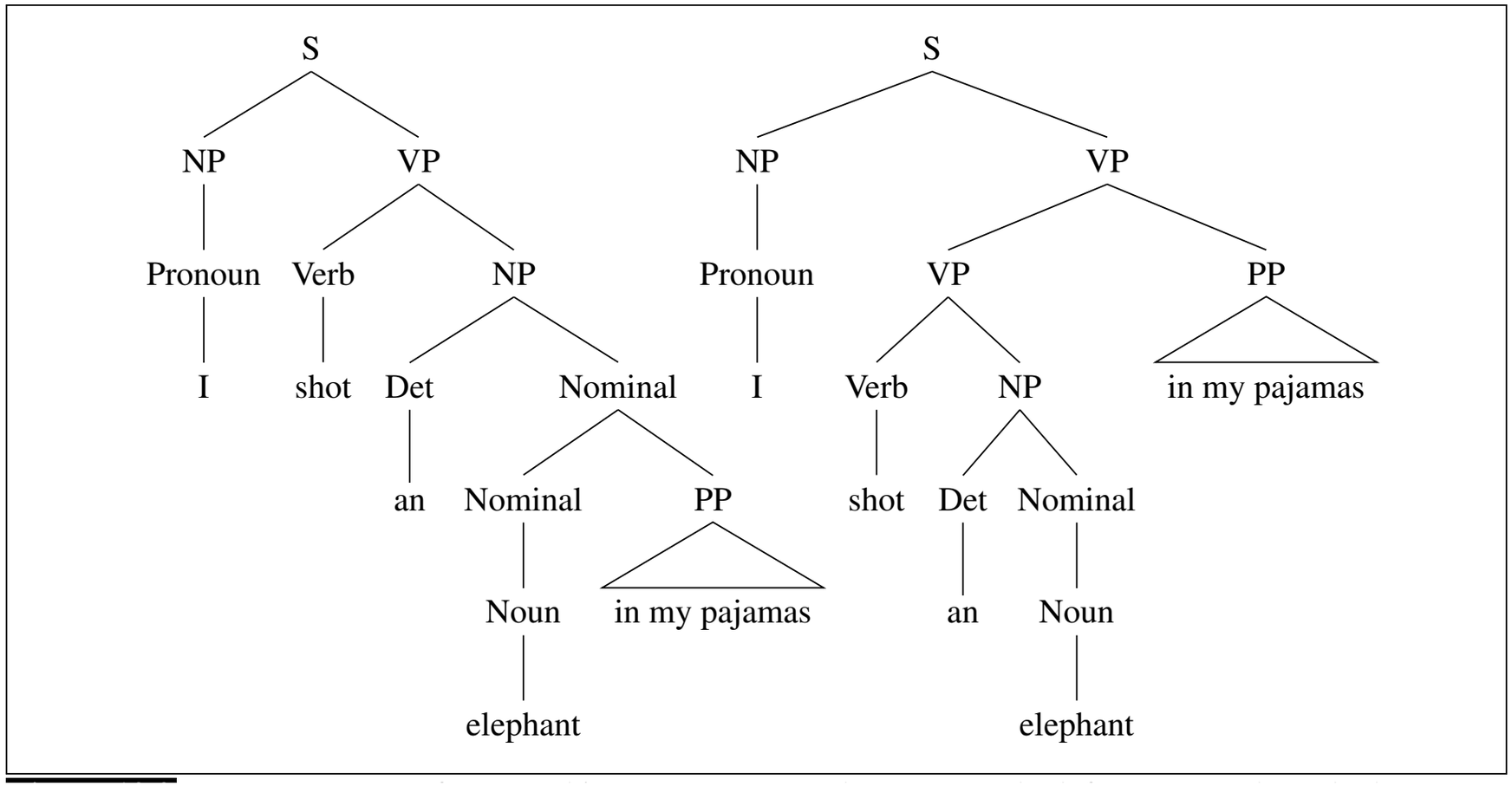
“I shot an elephant in my pyjamas.”

- “in my pyjamas” can be
- part of a NP headed by “an elephant” (the elephant was in my pyjamas!)

or

- part of a VP headed by “shot”. (I shot the elephant whilst I was wearing my pyjamas)

Challenges to Parsing:
Attachment Ambiguity



I shot an elephant in my pyjamas.

An example of **PP-attachment ambiguity.**

Challenges to Parsing: Attachment Ambiguity

"We saw the Eiffel Tower flying to Paris"

- "flying to Paris" is an example of a **VP-attachment** ambiguity, as it can have:
  - have Eiffel Tower as subject (the Eiffel Tower flies!)

  or:

  - modify the verb "saw" (we saw the Eiffel Tower while we were flying to Paris)

It does not always have to "make sense" semantically (for any type of ambiguity)

Challenges to Parsing:
Coordination Ambiguity

“old men and women dance.”

1- (old men) and women dance.

2- (old (men and women)) dance.

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

- ((nationwide tv) and (radio))
- (nationwide (tv and radio))

Challenges to Parsing: Coordination Ambiguity

- The combination of these two ambiguities and the fact that the options do not have to make sense semantically, gives rise to a large number of options.

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

Challenges to Parsing:
Syntactic Ambiguity

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

Prepositional Phrase

- (to the American people) (over nationwide tv and radio)

Challenges to Parsing:
Syntactic Ambiguity

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

Prepositional Phrase

- (to the American people) (over nationwide tv and radio)

- (to (the American people over nationwide tv and radio))

Challenges to Parsing:
Syntactic Ambiguity

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

Prepositional Phrase

- (to the American people) (over nationwide tv and radio)

- (to (the American people over nationwide tv and radio))

The American people who are over nationwide tv and radio?

Again, the parse does not always have to make sense…

Challenges to Parsing:
Syntactic Ambiguity

Show me the meals on the flight from San Fransisco.

Challenges to Parsing:
Syntactic Ambiguity

Show me the meals on the flight from San Fransisco.

when/where should you show me the meals?

Show me the meals | on the flight from San Fransisco.

meals that come from where?

Show me the meals on the flight | from San Fransisco.

Parsers

Different syntactic structures mean different meanings/semantics!

When developing parsers, we have to have the ambiguity challenge in mind. We want a parser that generates ALL possible parses.

CFG Parsers

- Paradigm: parsing as search

- Searching through the space of possible parse trees to find the correct one: one whose root is S and whose leaves are exactly the words in the input sentence.

- Classic search algorithms:

  1- Top-Down

  2- Bottom-Up

  3- Dynamic Programming

Top-Down

Given: a grammar and an input sentence

Start: the root of your parse tree: S

Continue: find all trees that can start with S

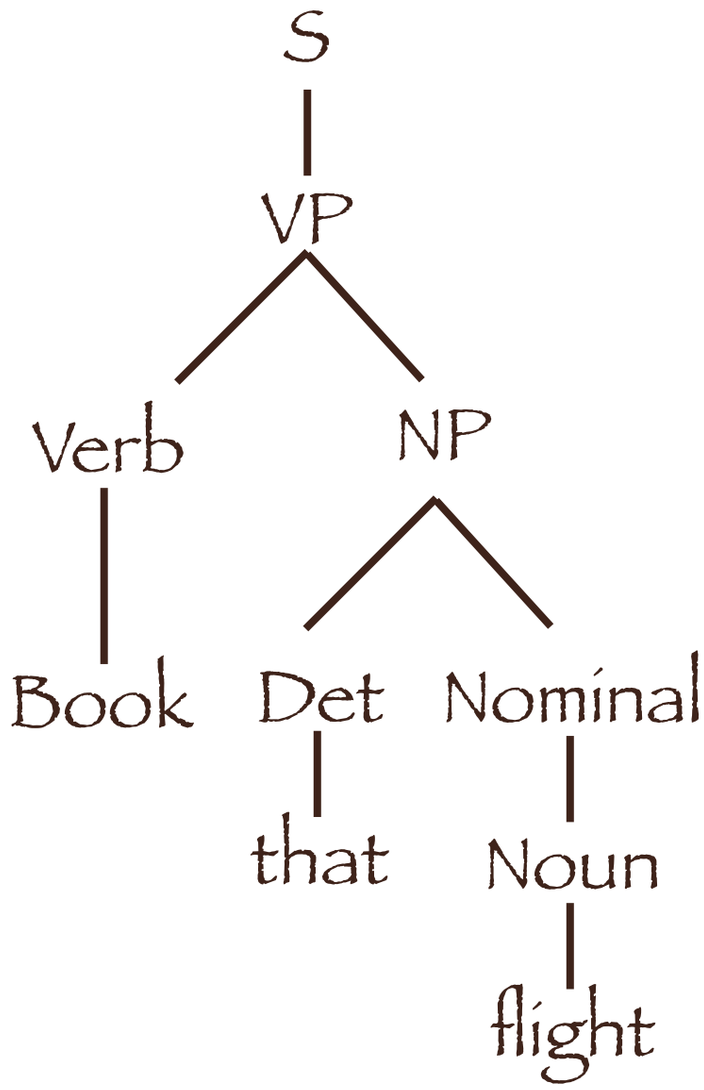
Method: look for rules with S on their left hand side

Repeat for each child

Stop: when the children are exactly the input words

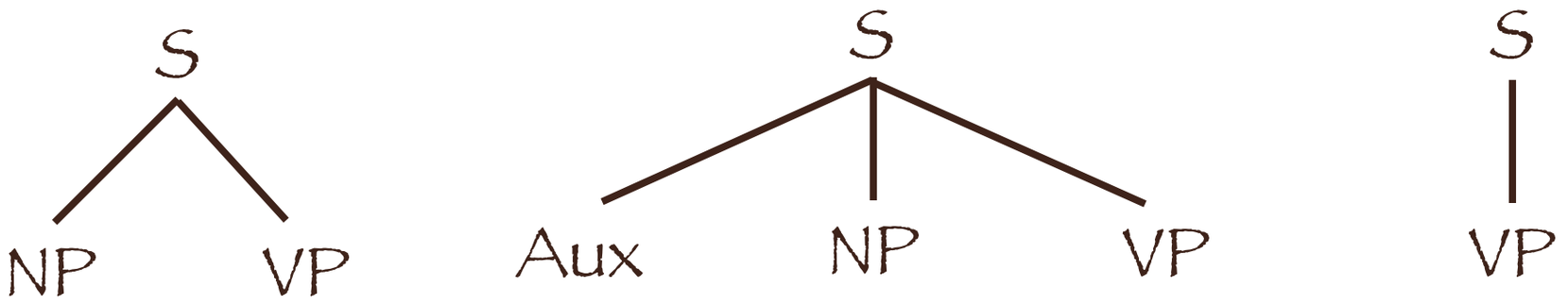
Example

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	



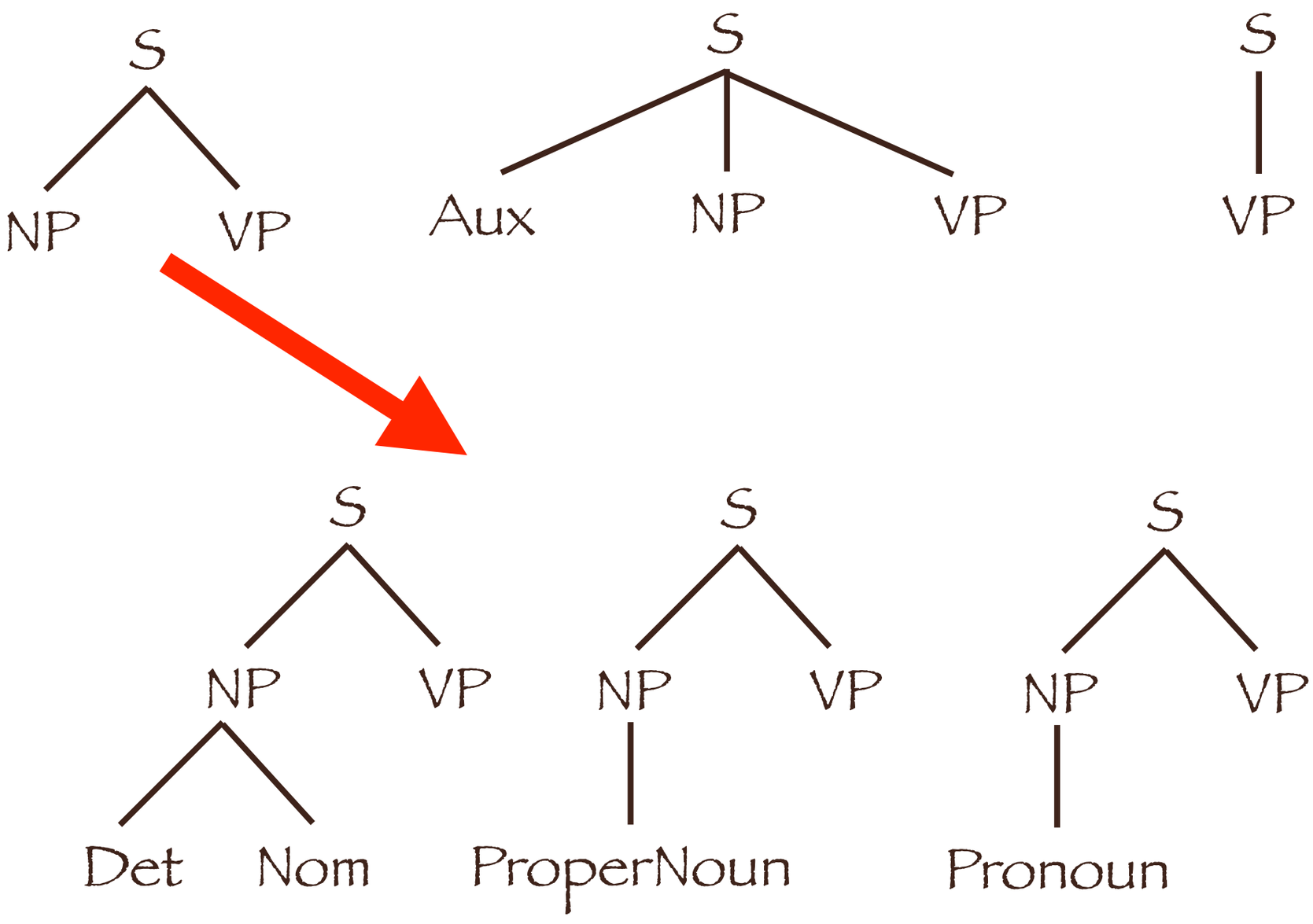
Book that flight.

Search Space



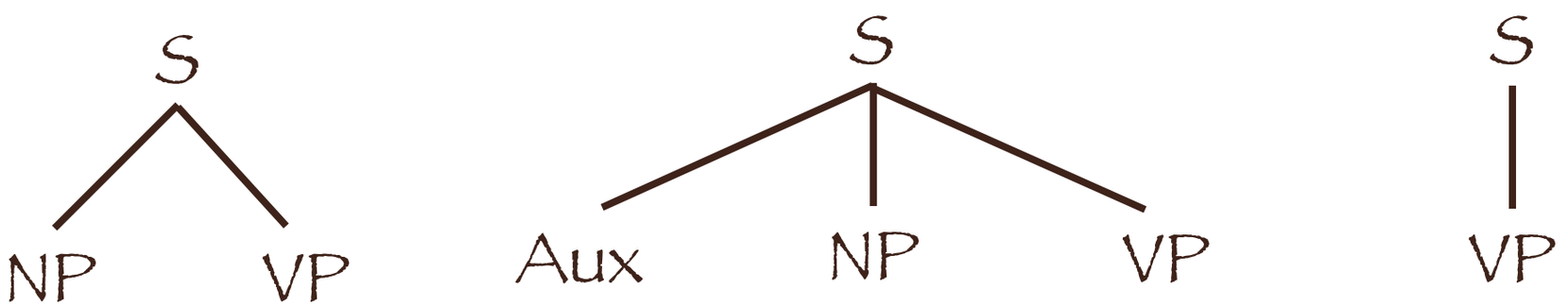
Book that flight.

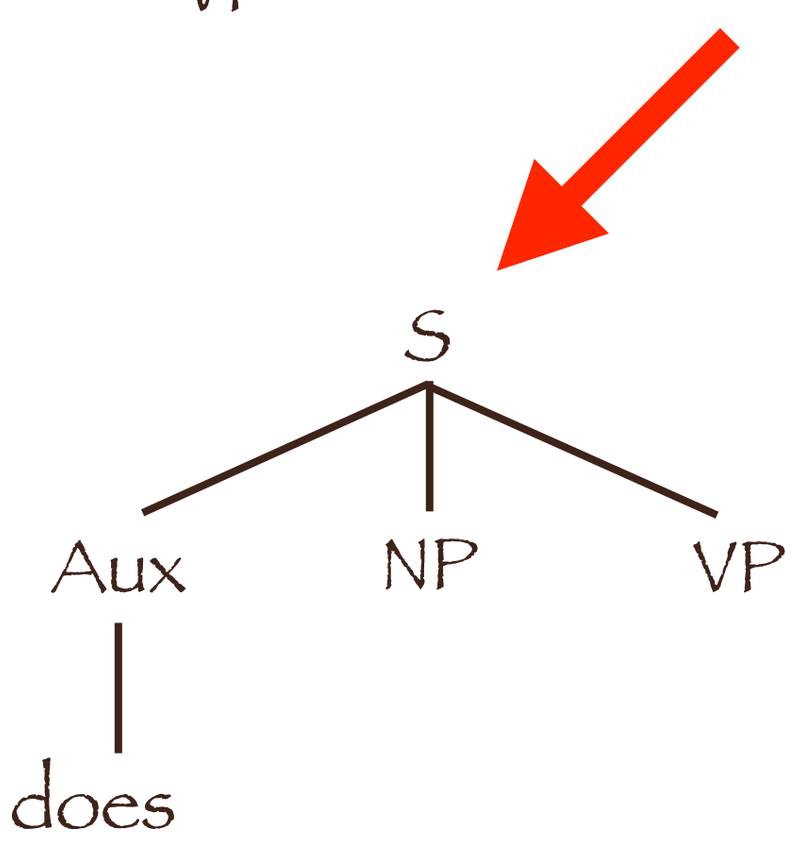
Search Space



Book that flight.

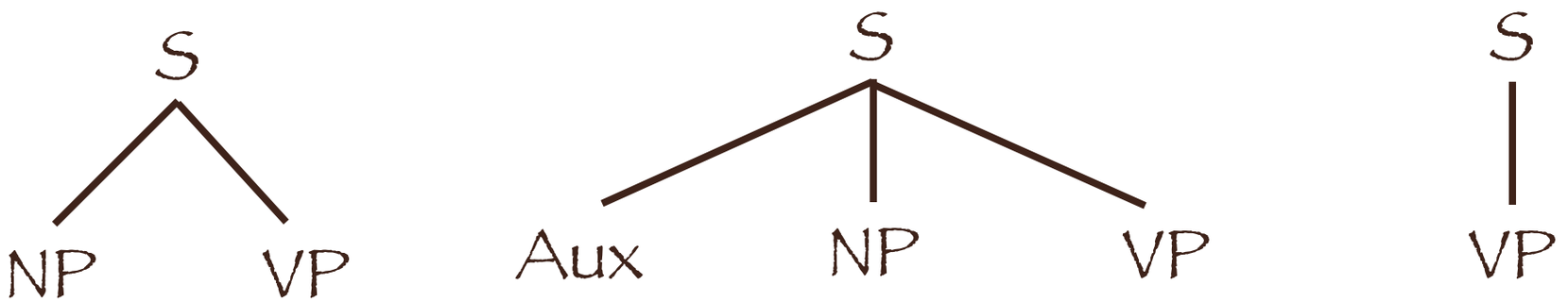
Search Space

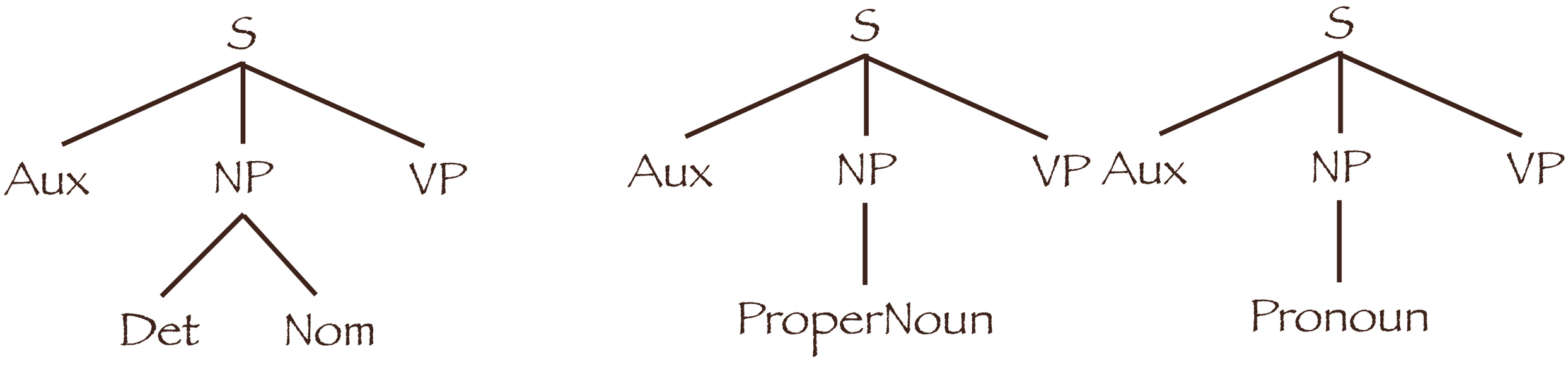




Book that flight.

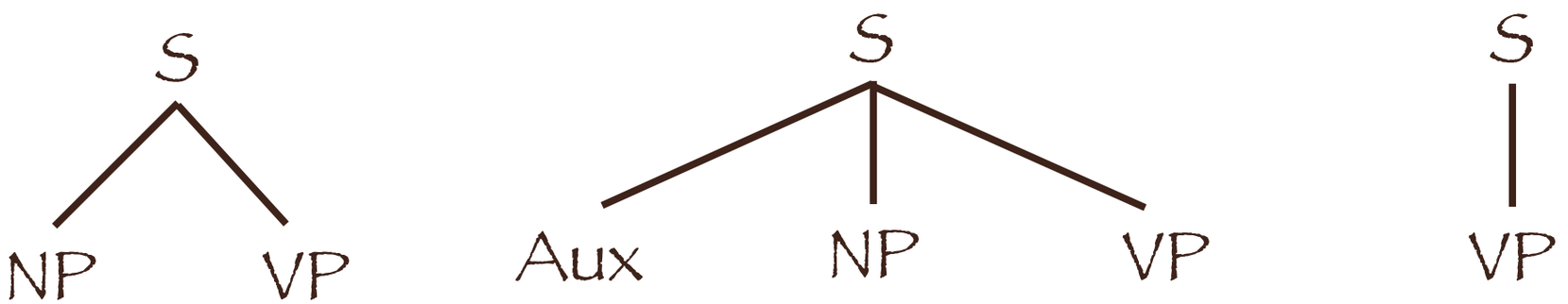
Search Space

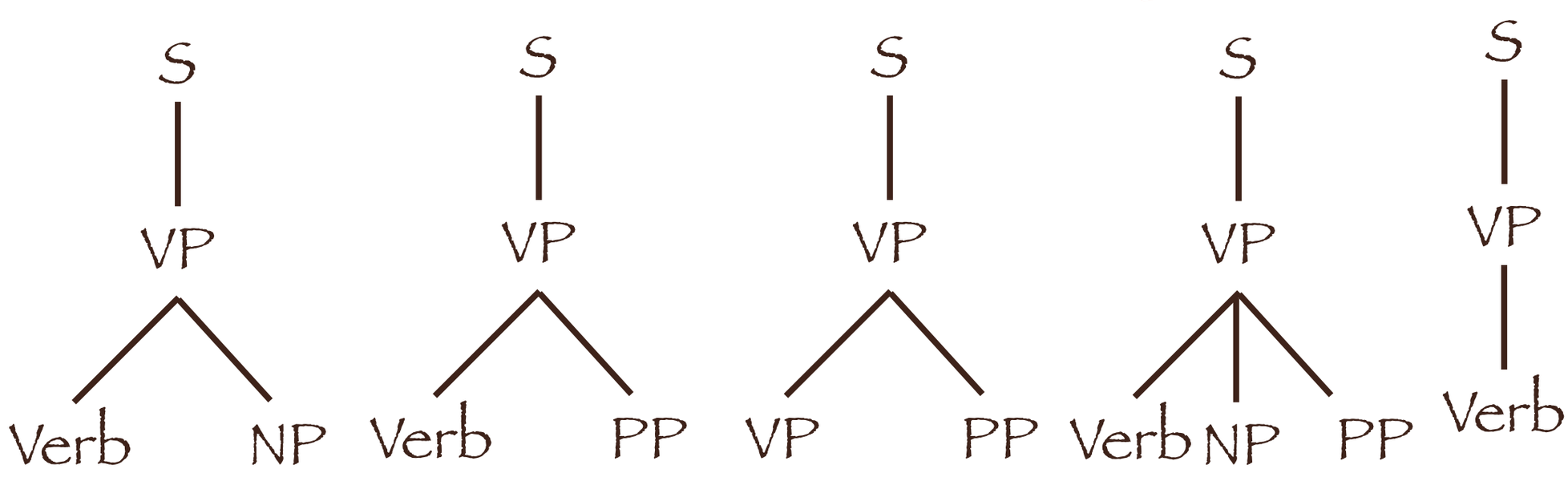




Book that flight.

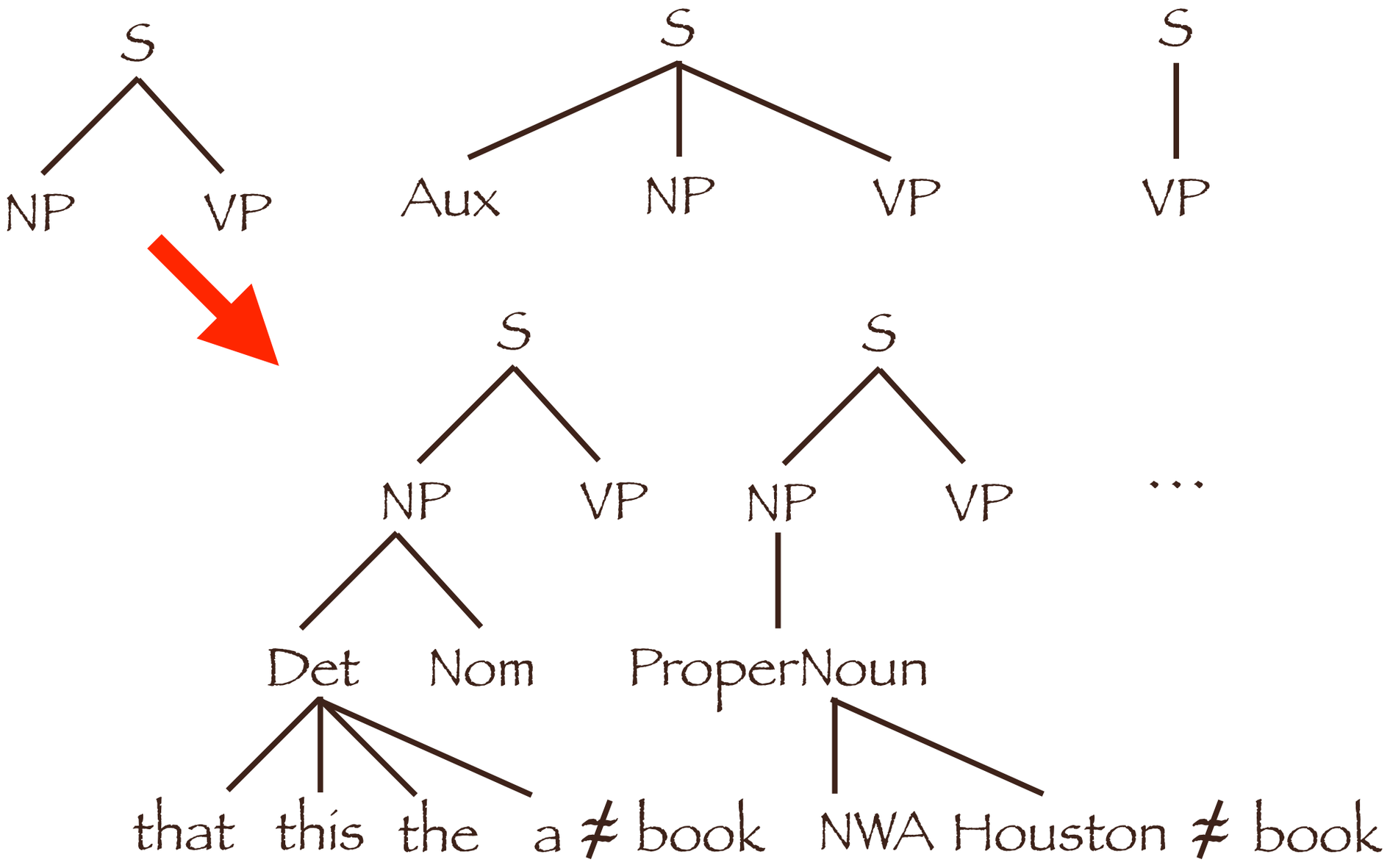
Search Space





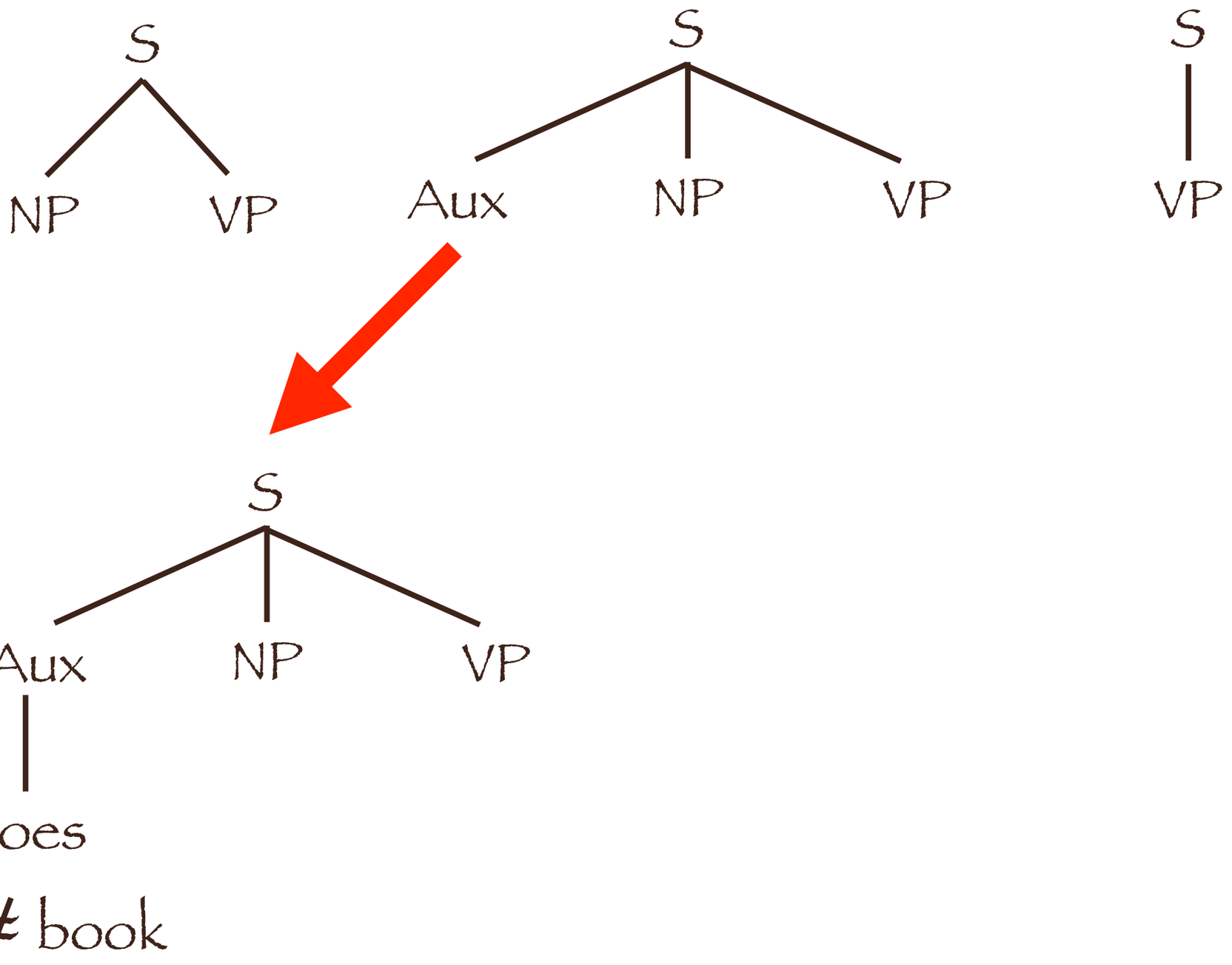
Book that flight.

Search Space



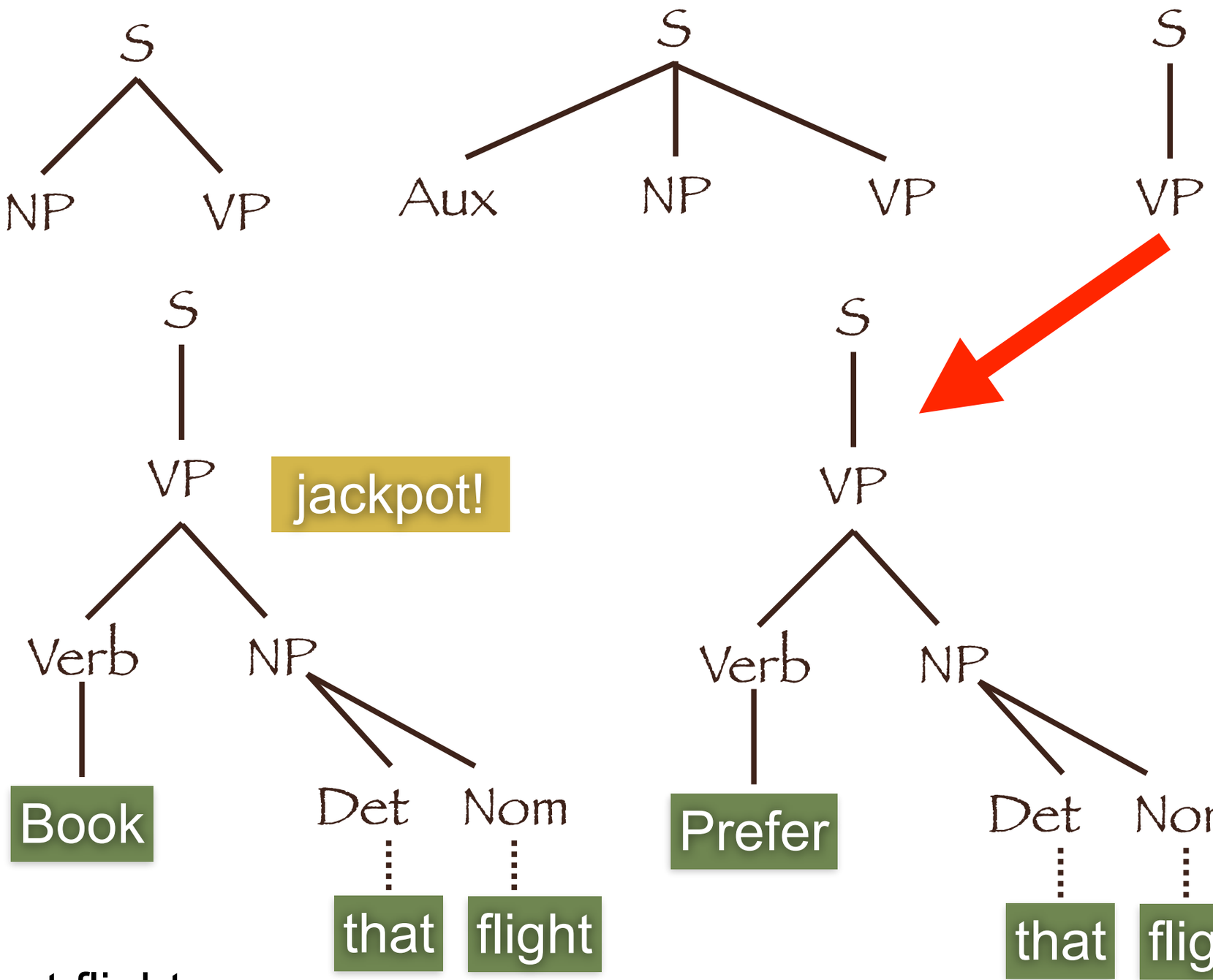
Book that flight.

Search Space



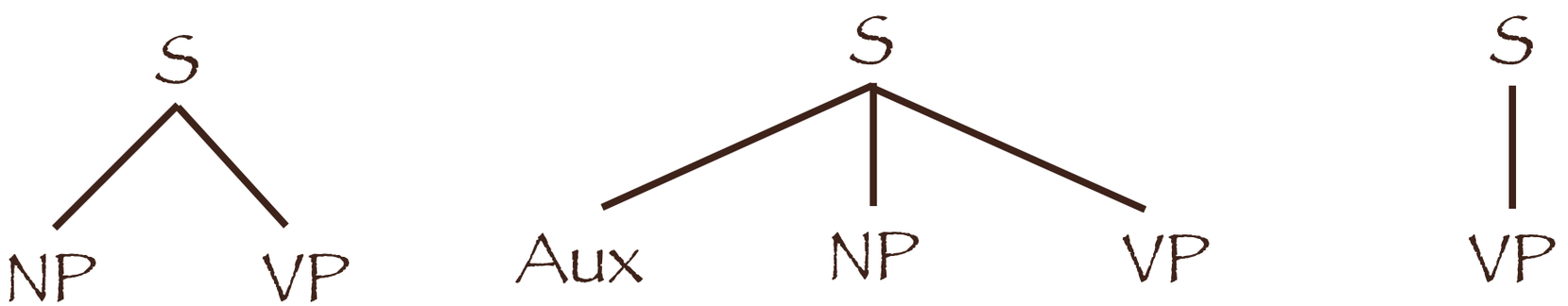
Book that flight.

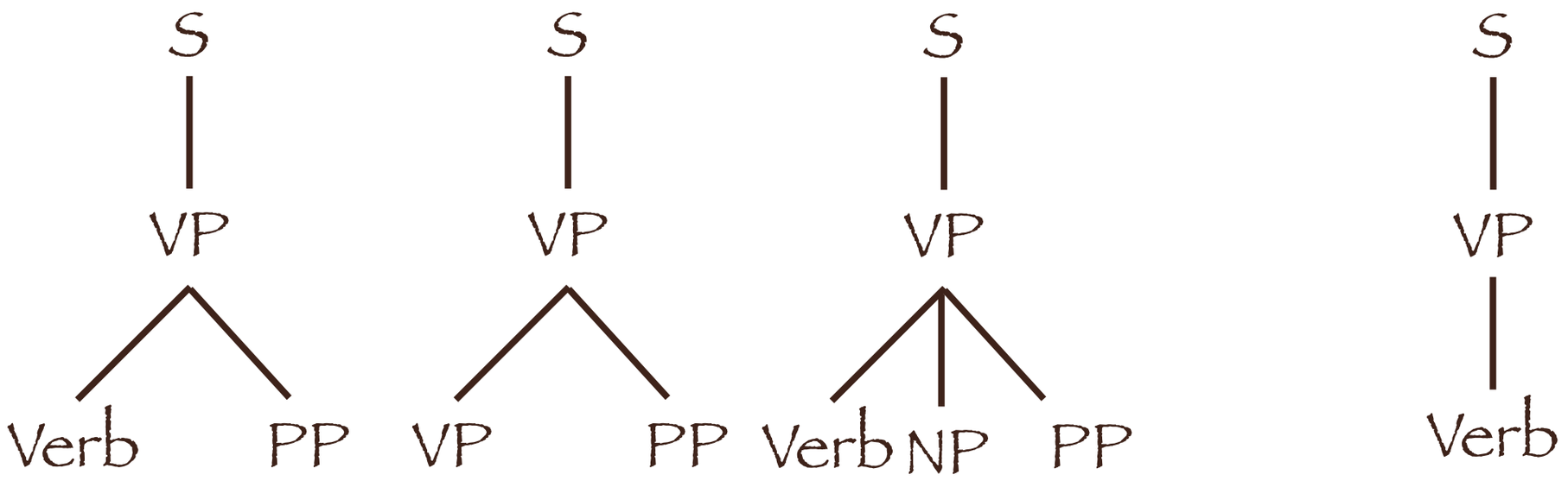
Search Space



Book that flight.

Search Space

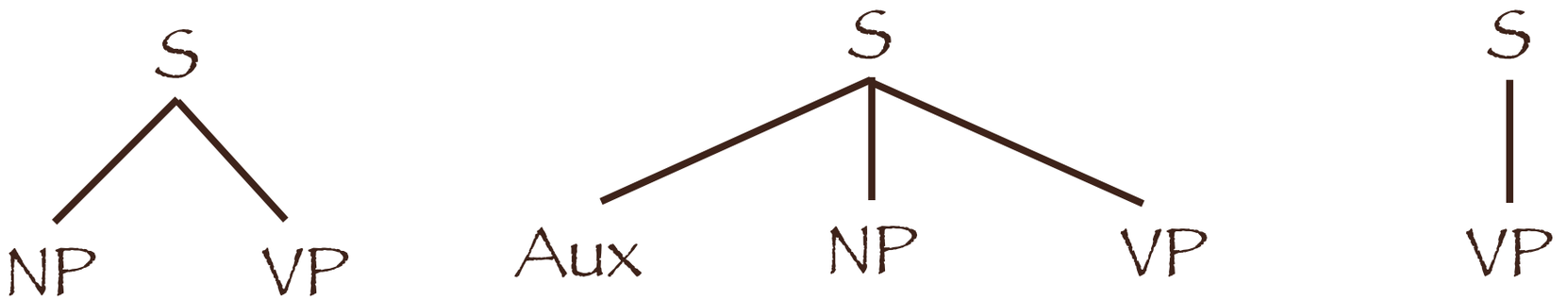


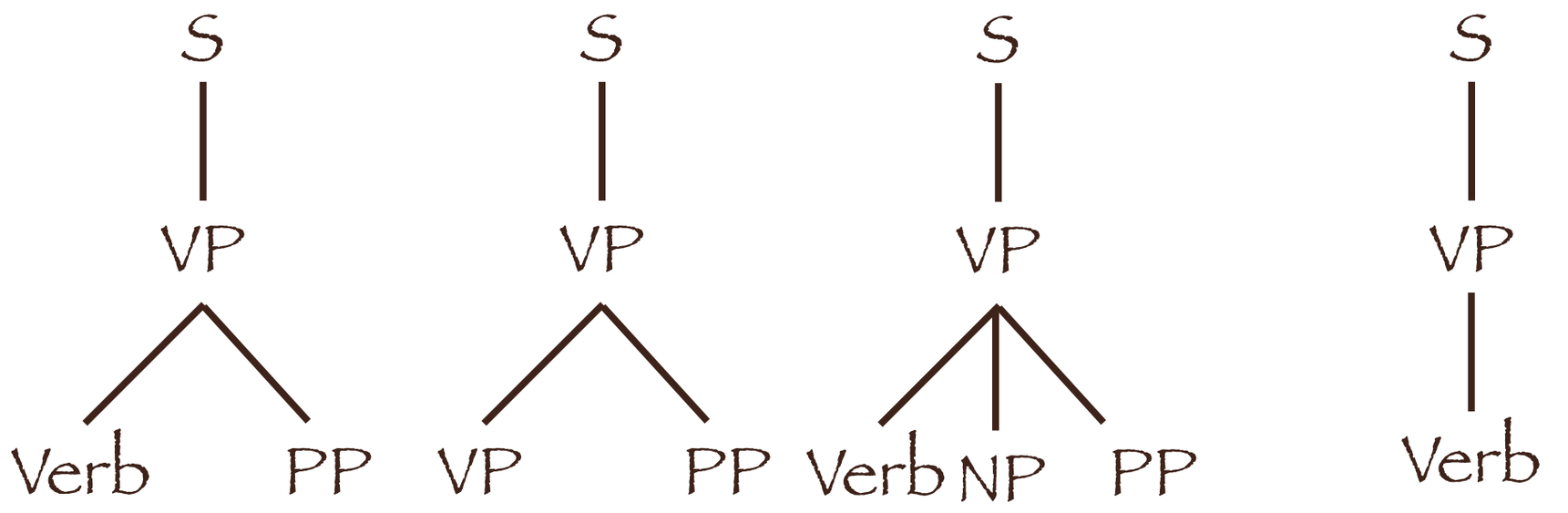


These 3 are bad: there are no PP's in S.

Book that flight.

Search Space





This is bad: we have more than a verb in S.

Book that flight.

Bottom-Up

Start: words of the sentence we are parsing

Continue: find all trees that can start with words

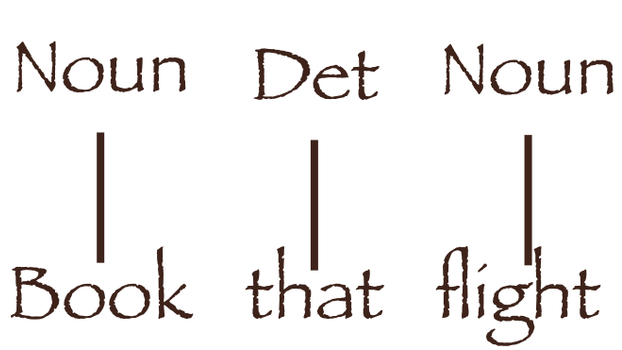
Method: look for rules with words on their right hand side

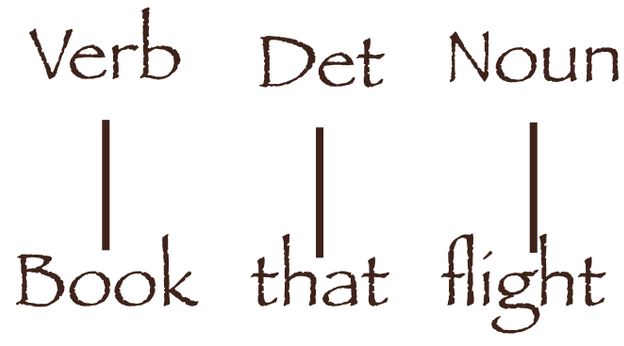
Repeat for each child

Stop: a tree with root S

Search Space

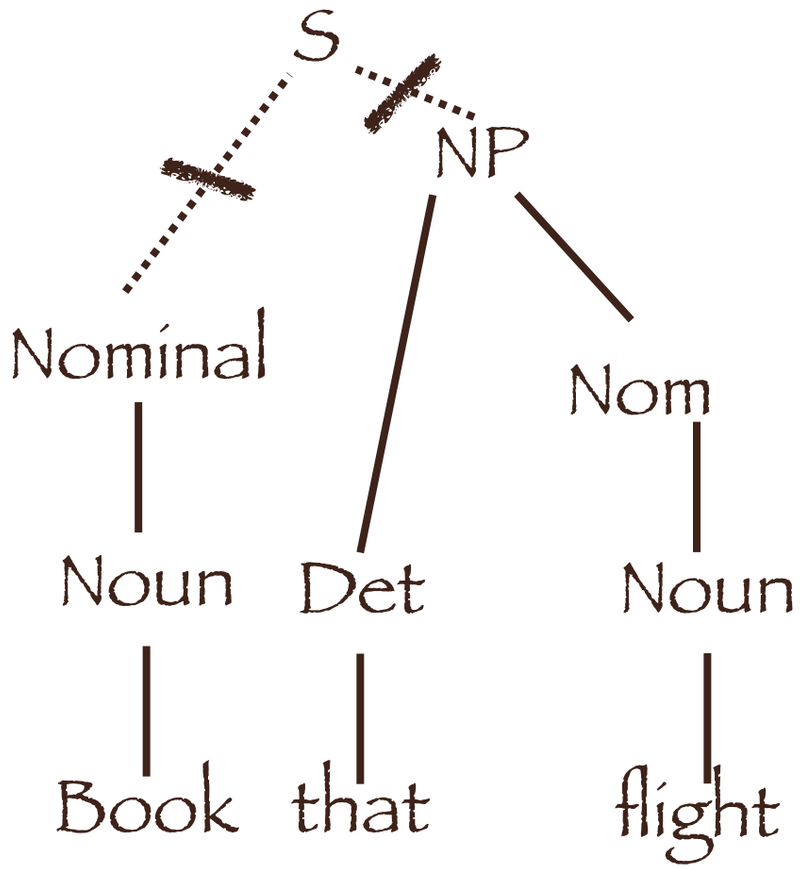
Book that flight.





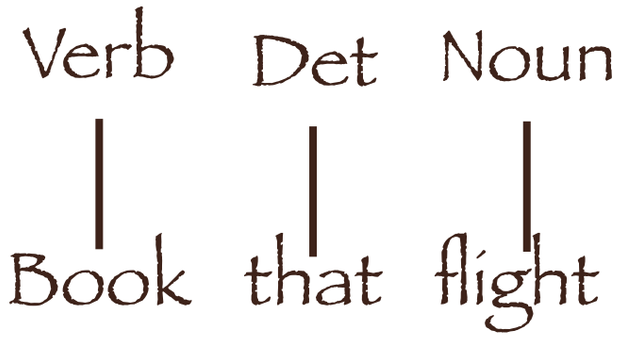
Search Space

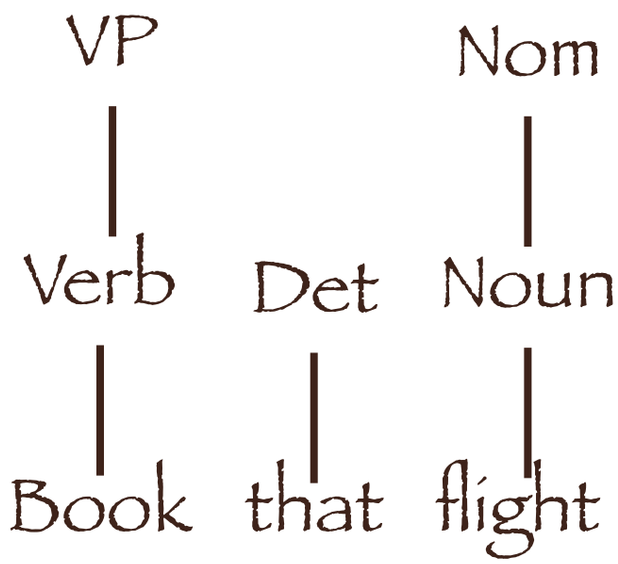
Book that flight.



Search Space

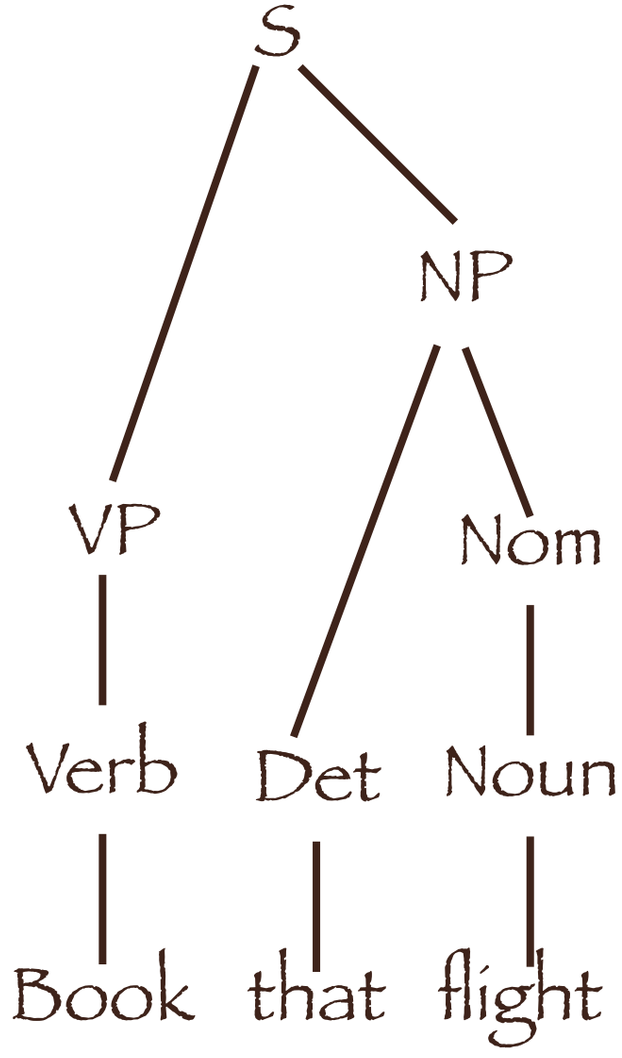
Book that flight.





Search Space

Book that flight.



Which Method?

- Each has their own advantages and disadvantages
- The Top-Down will never waste time with trees that cannot result in S.
- The Bottom-Up will not waste time with trees that cannot end in the words of input.
- Which one is worse? In our example, Bottom-Up seemed to be better. But this could be accidental.

OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

Parsing Algorithm: CKY

- CKY (or CYK) algorithm was introduced by different people. This version is due to:
  - Cocke
  - Kasami
  - Younger
- It is also called “chart parsing”.
- It stores constituents and subtrees as it finds them, so they do not need to be reconstructed again.

Chomsky Normal Form

- To use CKY, one has to work with grammars that are in **Chomsky Normal Form (CNF)**. This is when each rule expands to either of the following forms:

  - only two non-terminals
  - one single single terminal

$$A \rightarrow B\ C \qquad A \rightarrow a$$

$$A, B, C, \in N \qquad a \in \Sigma$$

- If we work with these grammars have we lost expressive power? Nothing is lost! So we have to **convert** any CFG rules not in CNF to CNF.

  - Theorem: *A CFG and its CNF accept the same language.*

Example of a Grammar and its CNF

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

Chomsky Normal Form

- Why CNF?
  - The parse trees are now all binary!
  - So they can be denoted by 2 dimensional matrices.
    - More specifically, the upper triangle of an *(n+1)x(n+1)* matrix.
    - Where *n* is the number of words in the input sentence.

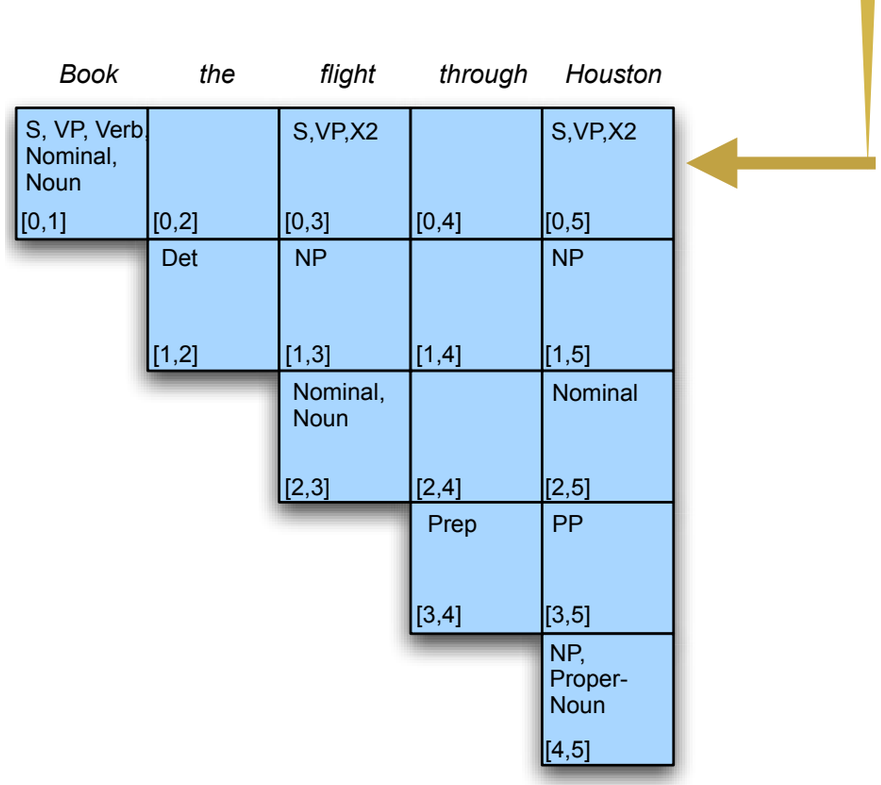
Parsing Algorithm: CKY

The cell representing the entire input is [0,n].

How do we fill in the matrix?

1- Index your input sentence by inserting a number before and after each word:

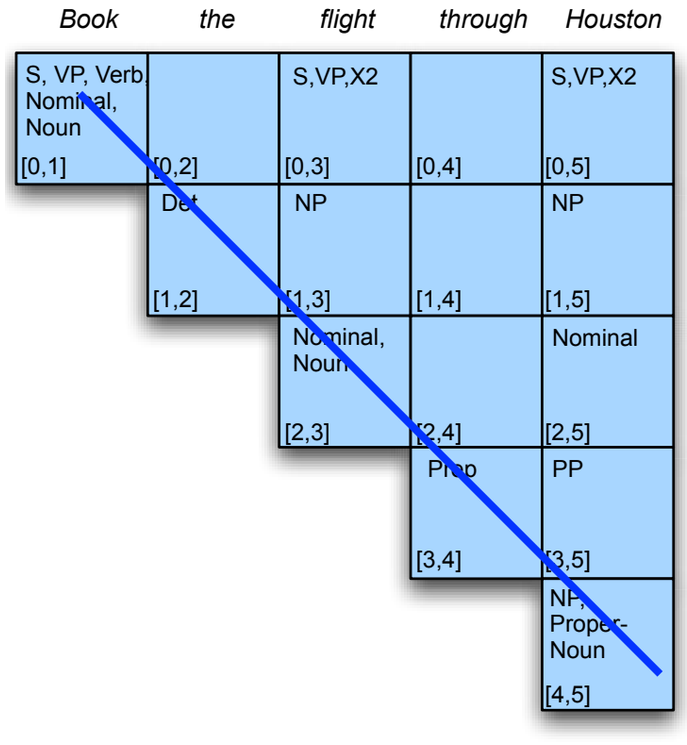
0 Book 1 the 2 flight 3 through 4 Houston 5



Parsing Algorithm: CKY

The main diagonal has the constituencies of the words of the sentence:

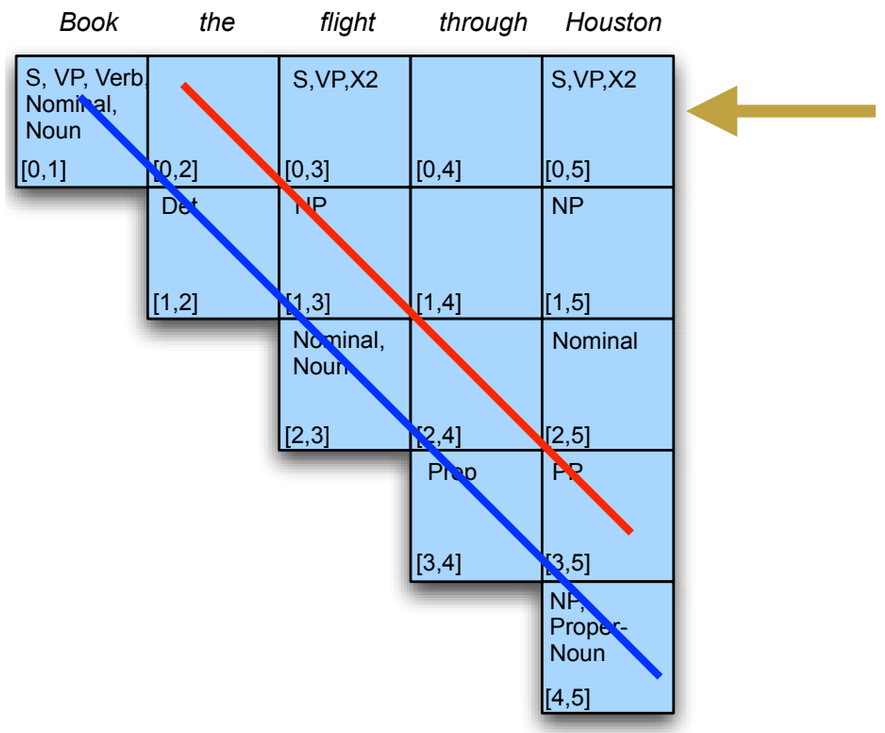
[0,1]Book,
[1,2]the,
[2,3]flight,
[3,4]through,
[4,5]Houston.



Parsing Algorithm: CKY

The second diagonals cover constituencies of combinations of two words:

[0,2]Book the,
[1,3]the flight,
[2,4]flight through,
[3,5] through Houston.

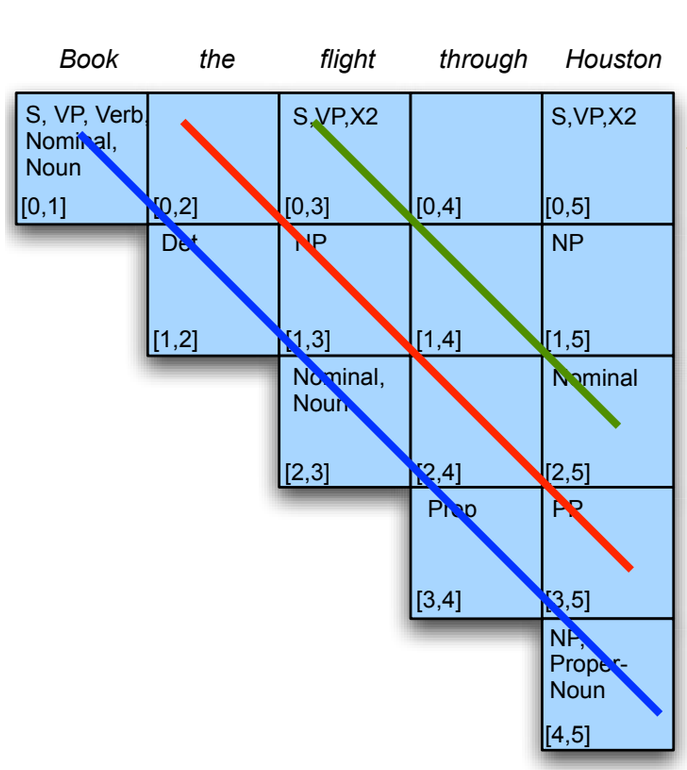


Parsing Algorithm: CKY

The cell representing the entire input is [0,n].

The third diagonals cover constituencies of combinations of three words:

[0,3]Book the flight,
[1,4]the flight through,
[2,5]flight through Houston,



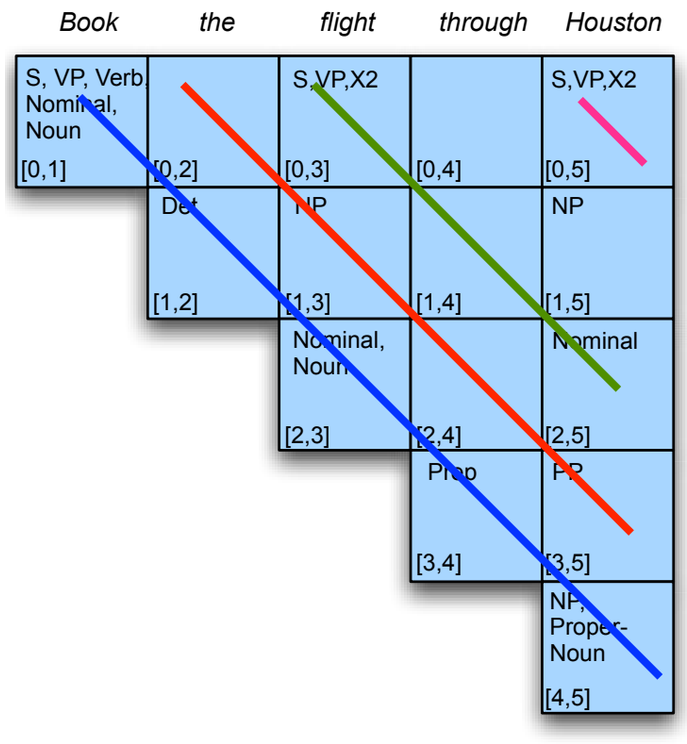
Parsing Algorithm: CKY

and so on …

The cell representing the entire input is [0,n].

The last diagonal has the constituency of the full sentence:

[0,5] Book the flight through Houston

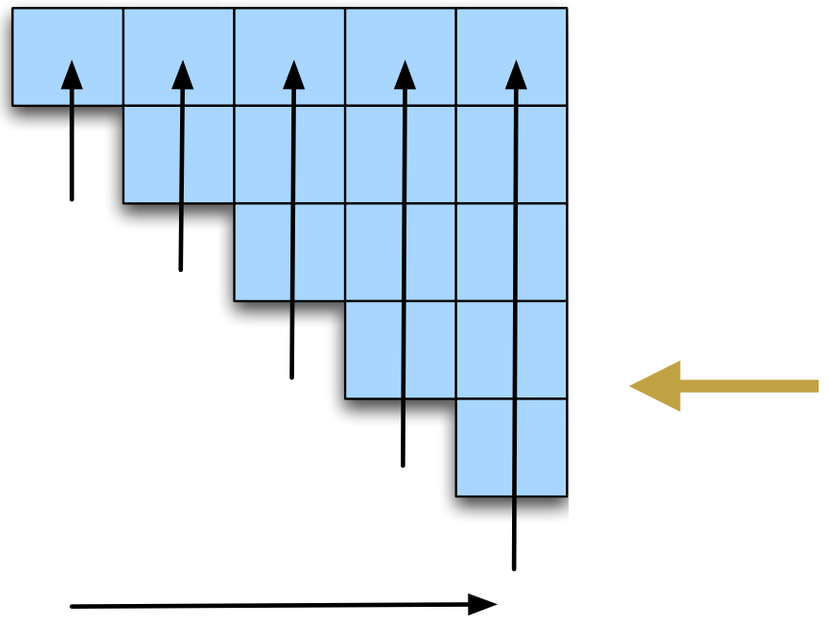


Parsing Algorithm: CKY

The matrix is filled in a bottom-up fashion:

- fill the matrix a column at a time from the left.

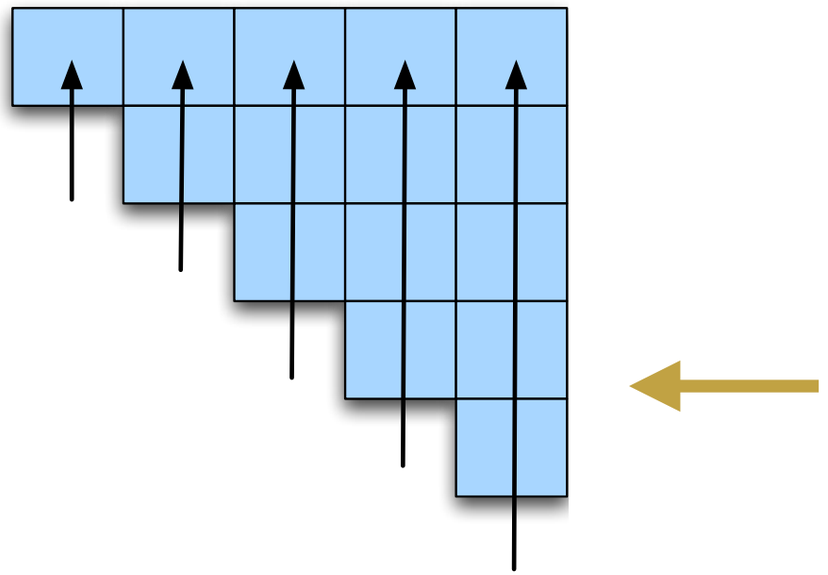
- fill each column from bottom to top.



Parsing Algorithm: CKY

The matrix is filled in a bottom-up fashion:

- fill the matrix a column at a time from the left.
- fill each column from bottom to top.



Why?

1- At each cell, we have all the info we need.

2- This is how incremental, word-by-word parsing (which is what humans are supposed to do) works.

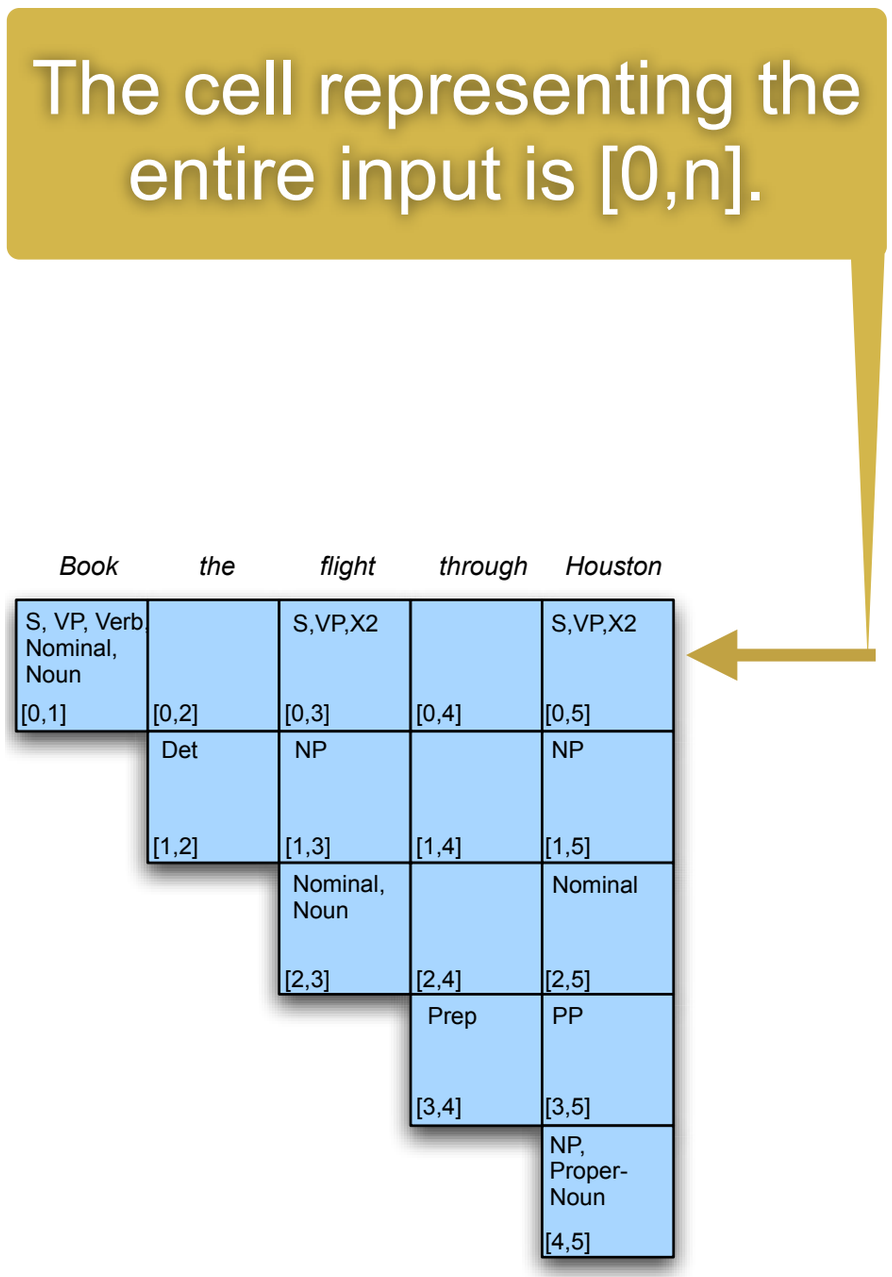
Parsing Algorithm: CKY

How do we fill in the matrix?

0 Book 1 the 2 flight 3 through 4 Houston 5

Each cell (i,j) of matrix is filled with a set of non terminals, representing all constituents from position i to j.

The cell representing the entire input is [0,n].



Parsing Algorithm: CKY

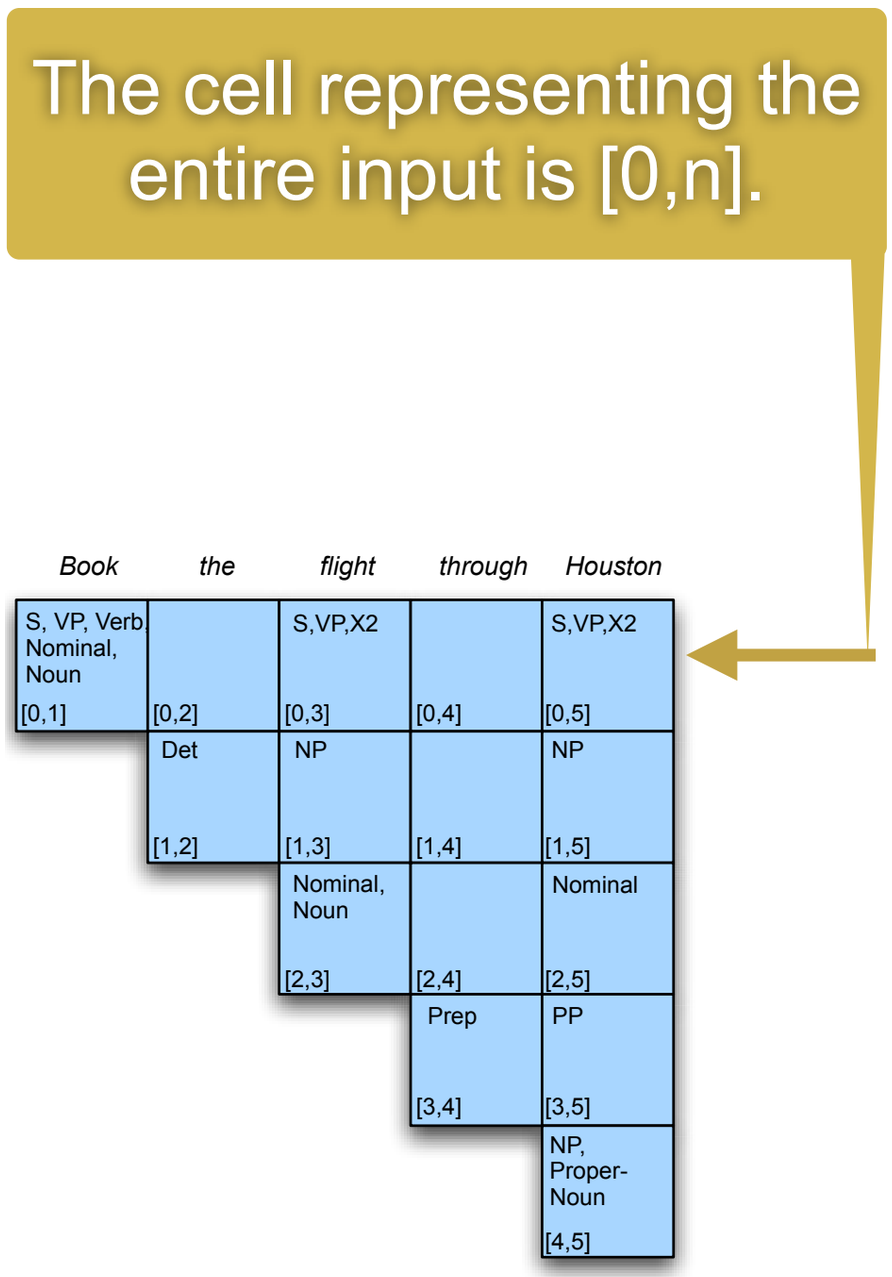
How do we fill in the matrix?

0 Book 1 the 2 flight 3 through 4 Houston 5

Each cell (i,j) of matrix is filled with a set of non terminals, representing all constituents from position i to j.

Consider the cell (0,1). Fill it with the non terminals that represent the constituency of the word from position 0 to 1, i.e. “Book”.

The cell representing the entire input is [0,n].



Parsing Algorithm: CKY

How do we fill in the matrix?

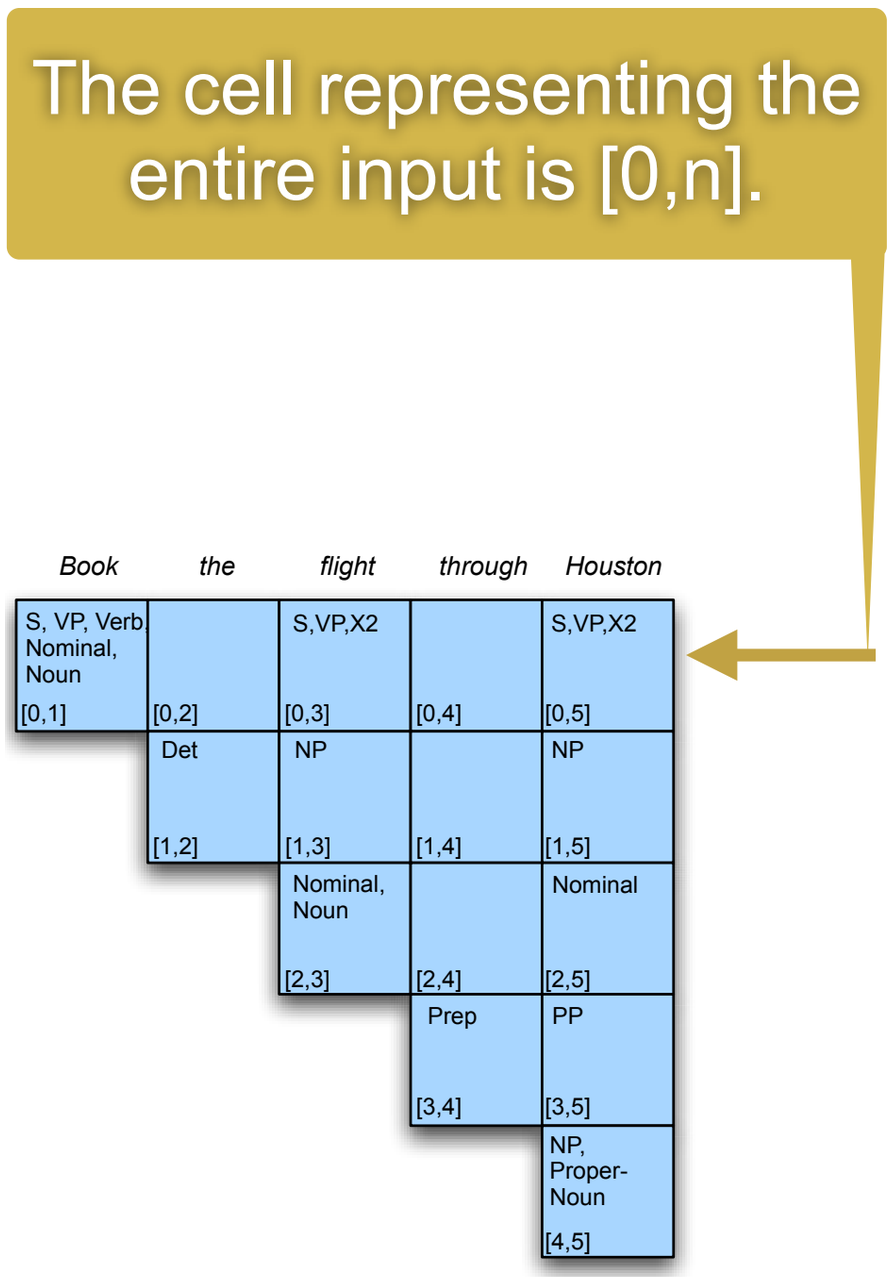
0 Book 1 the 2 flight 3 through 4 Houston 5

Each cell (i,j) of matrix is filled with a set of non terminals, representing all constituents from position i to j.

Consider the cell (0,1). Fill it with the non terminals that represent the constituency of the word from position 0 to 1, i.e. “Book”.

“Book” can be an imperative sentence S, it can be a VP (book a seat), or just a Verb (book from home). It can also be part of a group of nouns (book worm)- so a Nominal, or only one Noun.

The cell representing the entire input is [0,n].



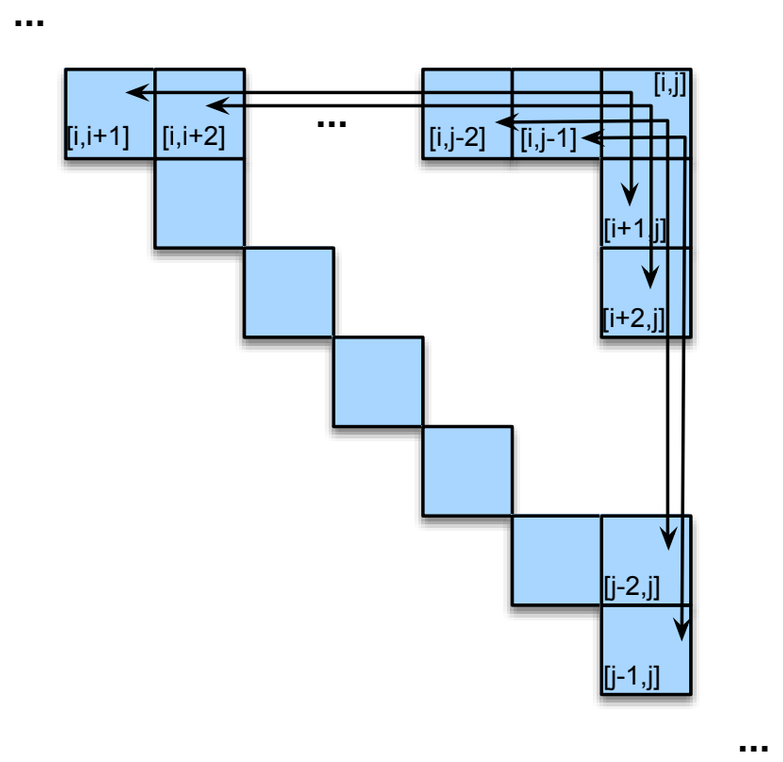
Parsing Algorithm: CKY

We have the constituents, but haven't recorded the structural relation they are in to each other as we parse.

We need to record these to get the different parses/syntactic trees.

Parsing Algorithm: CKY

After filling the table, start with the S cell of the table, cell (0,n). In the presence of ambiguity, we want to return all possible parses of S.



Make each non-terminal point to cells from which it was derived in the whole matrix.

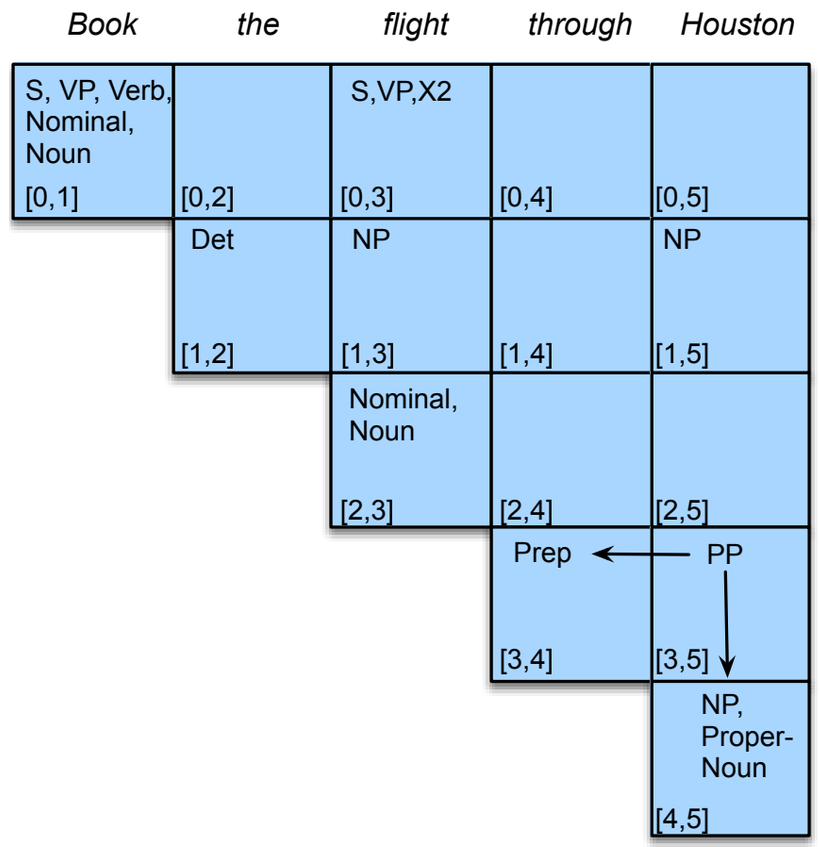
Now all parses are returned by choosing the S from cell [0,n] and retrieving all of its component constituents from the matrix, by following the pointers.

Examples

[3,5] points to [3,4] and [4,5].

This means that the PP in [3,5], i.e. “through Houston” can be generated by
Prep NP
or
Prep Proper-Noun

In this case, it is Prep Proper-Noun

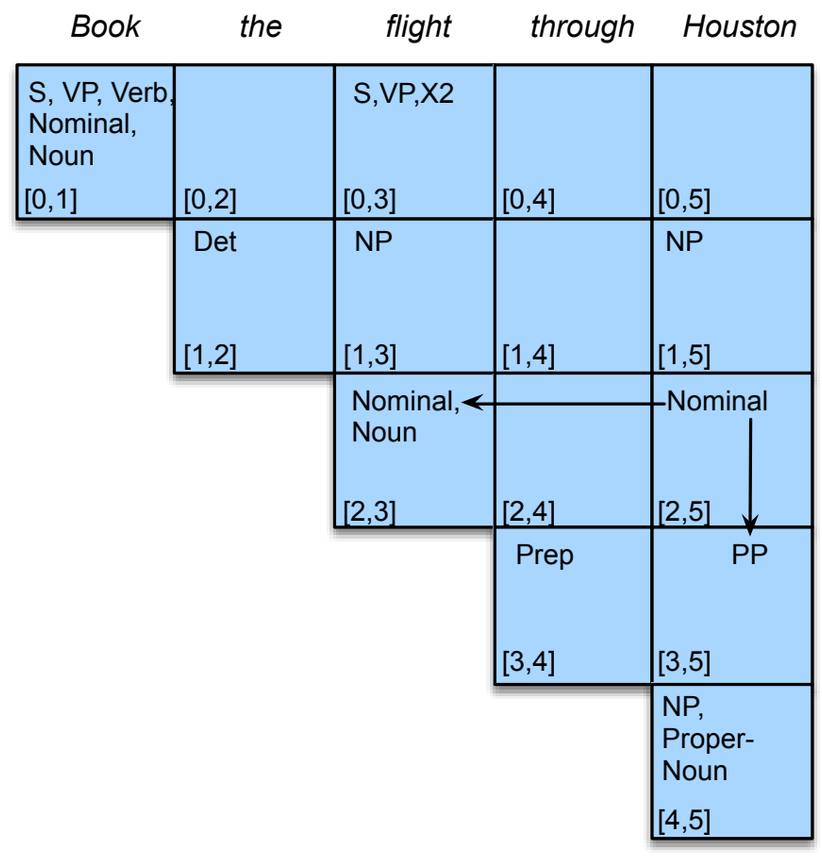


Examples

[2,5] points to [2,3] and [3,5].

This means that the Nominal in [2,5], i.e. “flight through Houston” can be generated by
Nominal PP
or
Noun PP

In this case, it is Noun PP

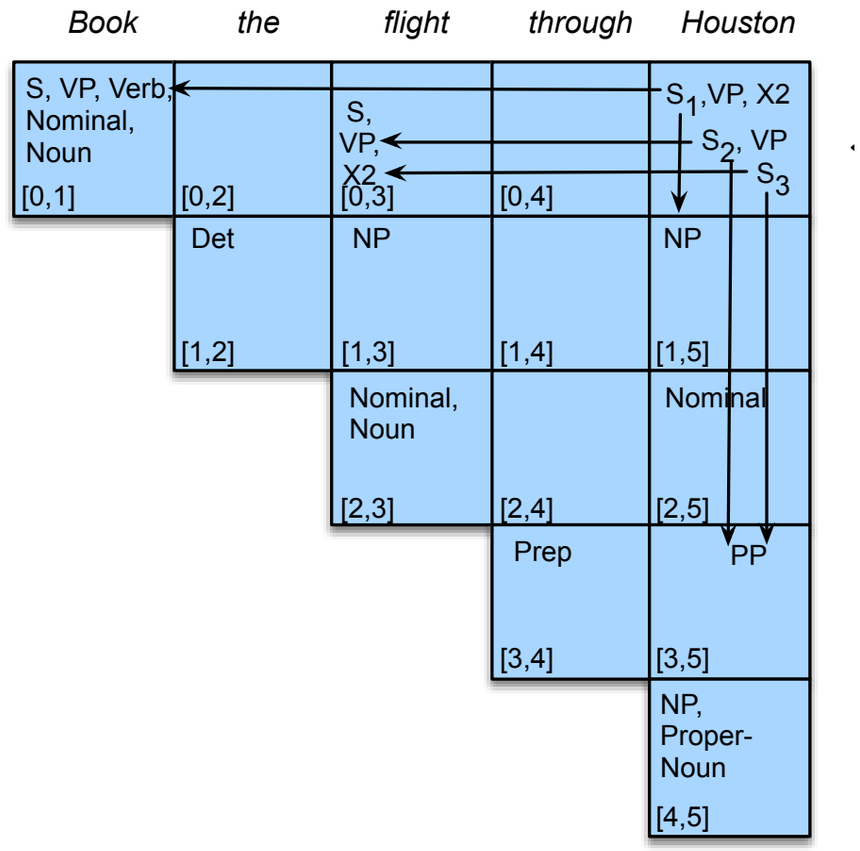


Examples

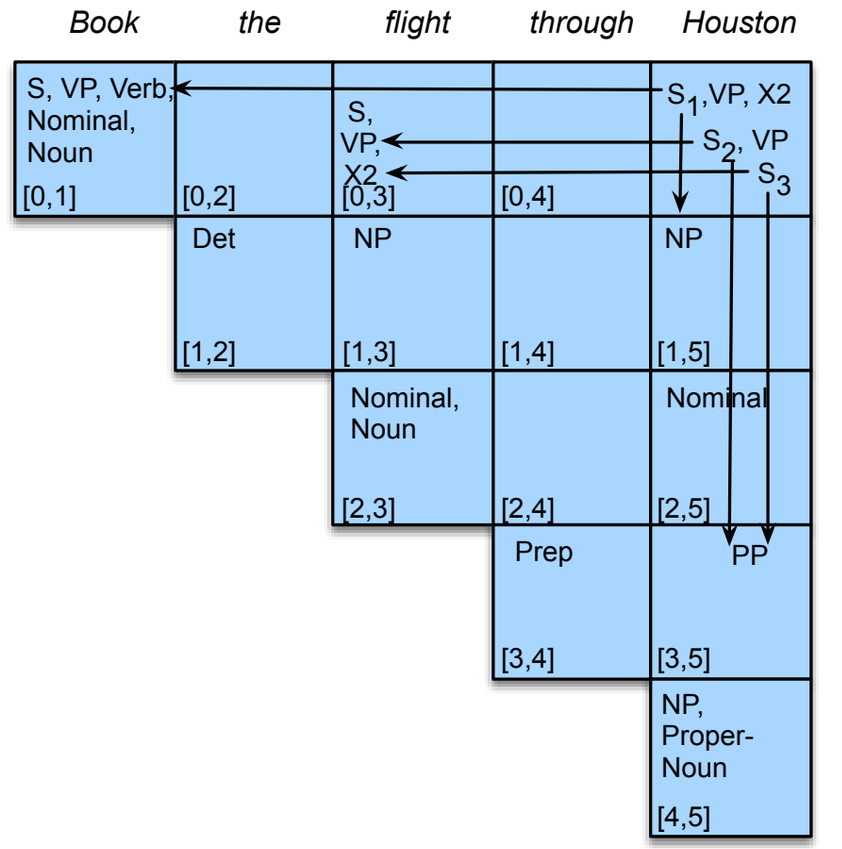
and so on … in particular …

The S1 in [0,5] points to Verb in [0,1] and NP in [1,5], which gives the parse Verb NP (the correct one).

Book, which flight? the flight that goes through Houston.



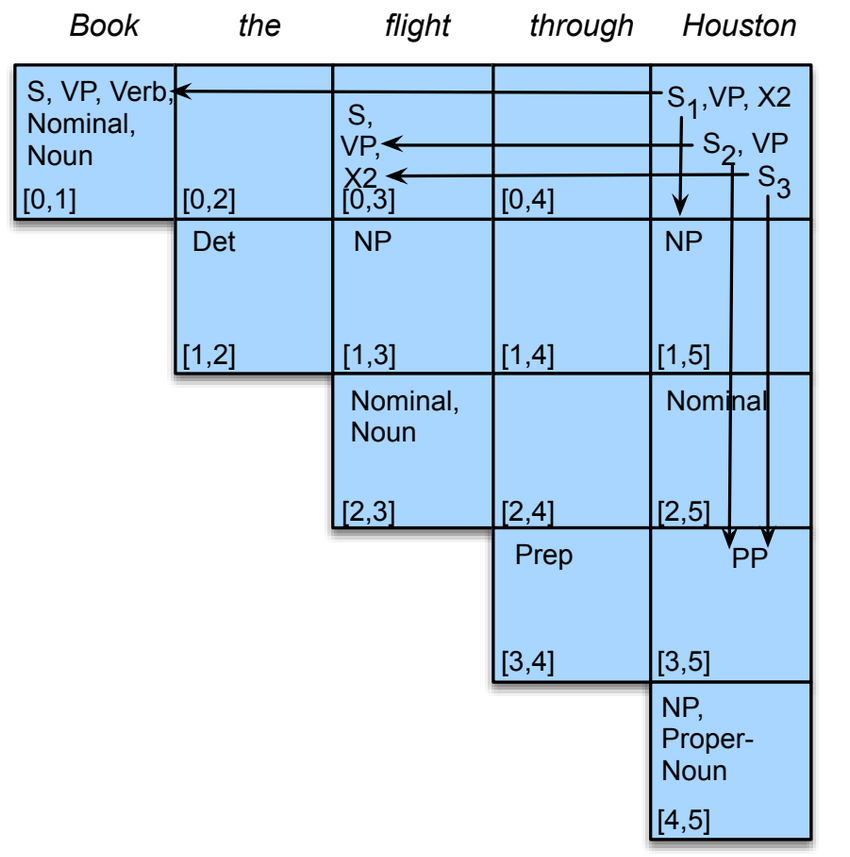
Examples



The S2 in [0,5] points to VP in [0,3] and PP in [3,5], which gives the parse VP PP (which is slightly wrong).

Book the flight, how? through Houston.

Examples



The S2 in [0,5] points to VP in [0,3] and PP in [3,5], which gives the parse VP PP (which is slightly wrong).

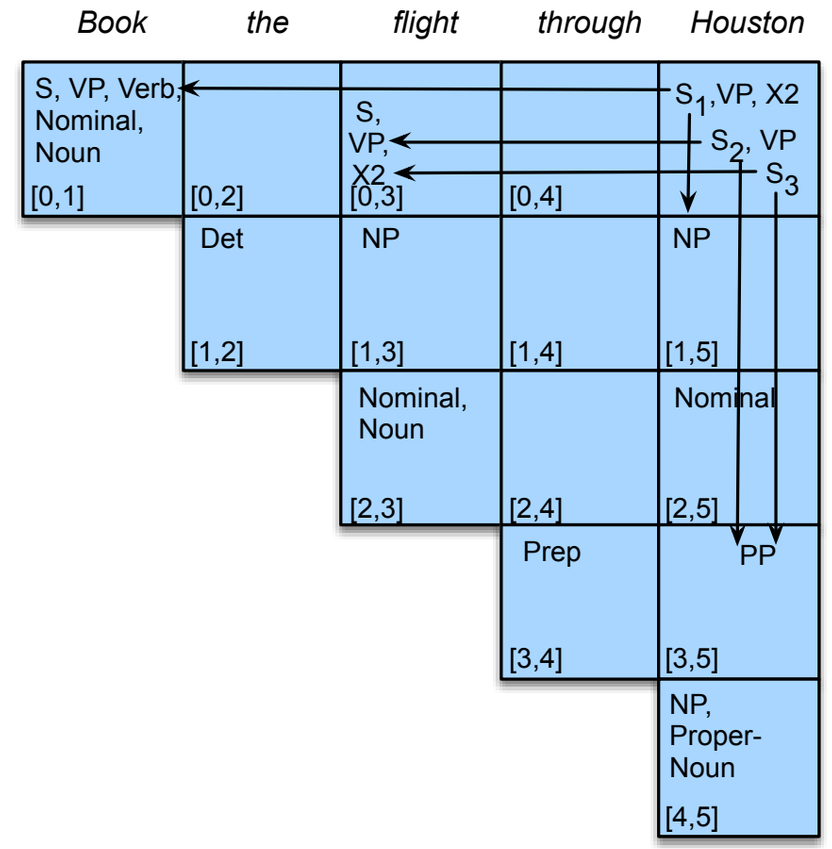
Book the flight, how? through Houston.

Can you do the S3?

Examples

The S3 in [0,5] points to the X2 (Verb NP) in [0,3] and the PP in [3,5].
This gives the parse Verb NP PP (also slightly wrong).

Book the flight. How? through Houston, e.g. while passing through Houston, etc.



OUTLINE

1) Syntax of natural language and formal grammars
2) Constituency Grammar: CFG
3) Syntactic Ambiguity and CFG Parsing
4) The CKY algorithm
5) Probabilistic CFGs: parsing and learning

Why go statistical?

The idea behind any statistical method:

- Sometimes we cannot have full knowledge about something.
- With enough knowledge, we can figure out its probability.
- This is better than saying nothing about it.

In the case of parsing:

- We want to account for **syntactic ambiguity** in a weighted way - i.e. to assign different probabilities to different possible parse trees, rather than just pick one tree which fits.
- Getting **likelihood measures** of grammaticality rather than just grammatical vs ungrammatical is more robust (like language modelling).

PCFG

- Probabilistic Context Free Grammars (PCFG's) were first introduced by Booth in 1969.

A tuple of 4 parameters: (N, Σ, R, S)

N a set of non-terminal symbols

Σ a set of terminal symbols (disjoint from N)

S a designated start symbol

R a set of production rules of the form $A \rightarrow \beta\,[p]$

for A and β as in a CFG

where *p* is a real value indicating the probability of β being used given A.

PCFG

- A PCFG is obtained from a CFG by augmenting each rule with a probability *p* .
- What is this probability? $A \rightarrow \beta\ [p]$
- The probability that the given non-terminal A will be expanded to the string β .
- In other words, the conditional probability of the string β given the non-terminal A.
- Other notations:

$$P(A \rightarrow \beta) \qquad P(A \rightarrow \beta|A) \qquad P(RHS|LHS)$$

- All possible expansions of a non-terminal add up to 1:

$$\Sigma_\beta P(A \rightarrow \beta) = 1$$

Computations

- A PCFG assigns a probability to each parse tree T of a sentence S. The tree with the higher probability is said to be the more likely one.
- The probability of a tree *T* of a sentence *S* is computed as follows:

$$P(T,S) = \Pi_{i=1}^{n}\ P(RHS_i|LHS_i)$$

- This is the product of the probabilities of all the *n* rules

$$LHS_i \rightarrow RHS_i$$

used to expand each of the *n* non-terminals of *T*.

- The most probable parse tree for a given *S* is chosen as follows:

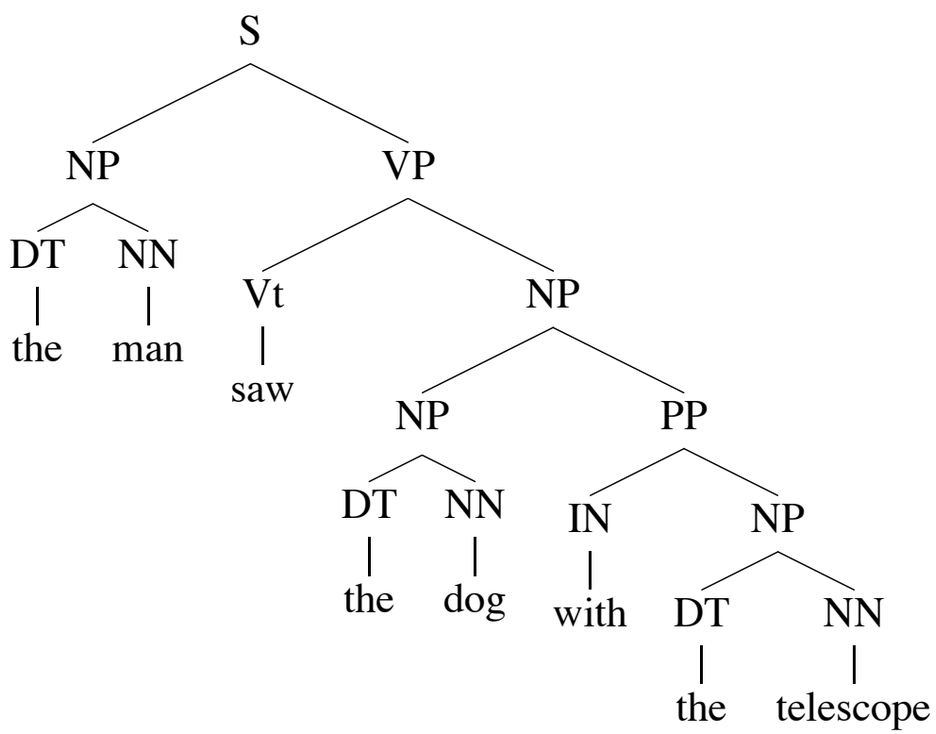
$$P(S) = argmax_T\, P(T)$$

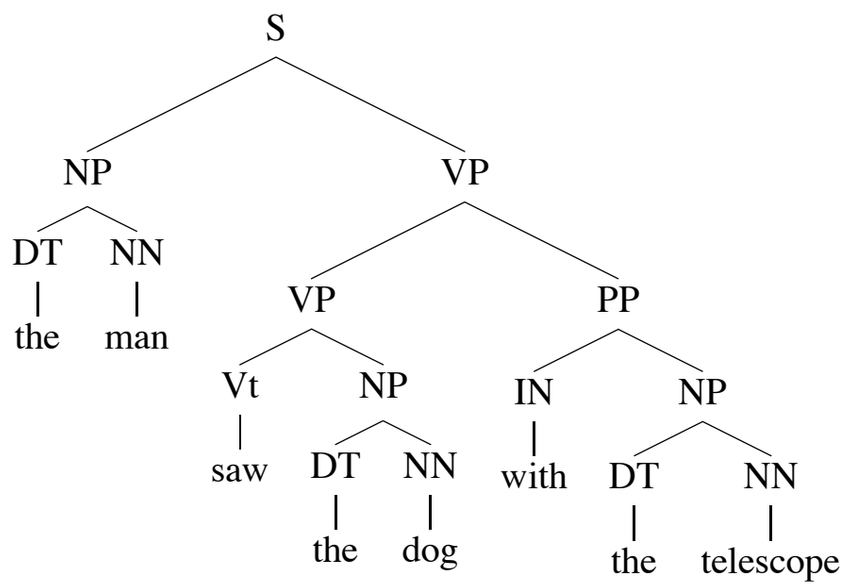
Example

- Consider the two parses of **“the man saw the dog with the telescope”.** Compute the probability of each parse tree.

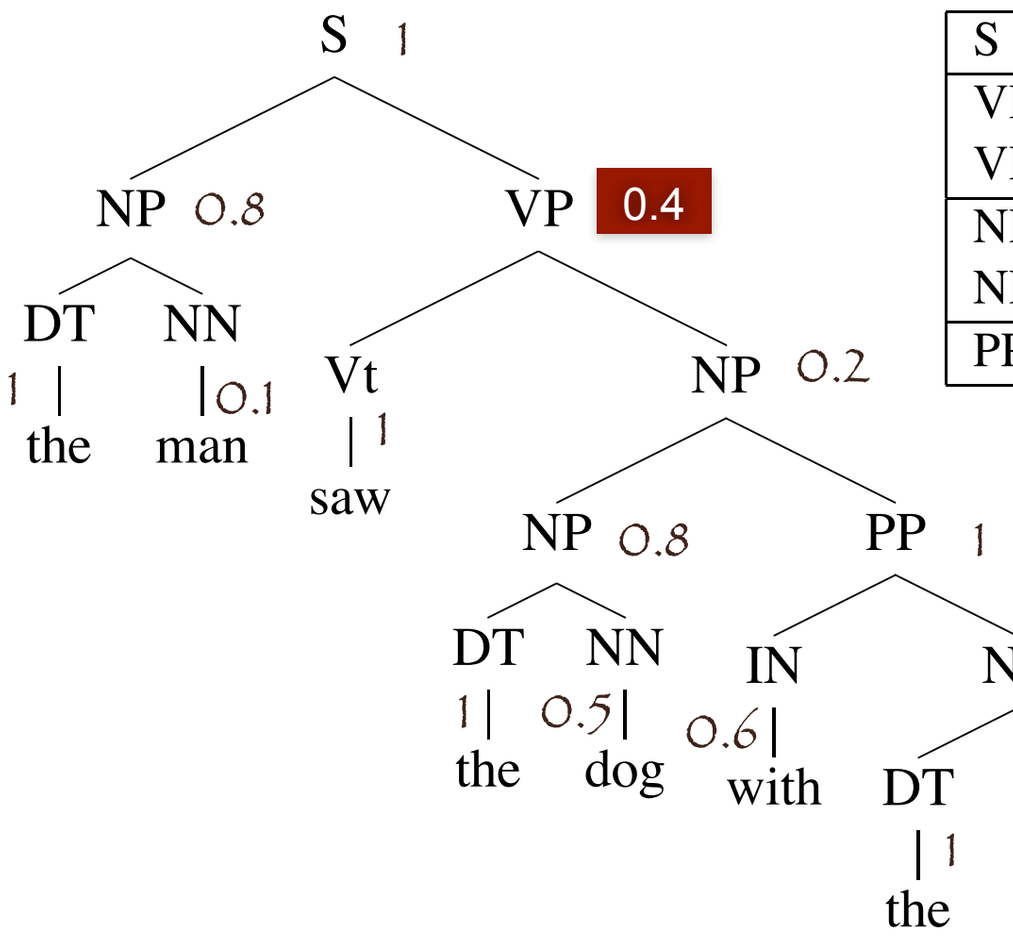
S	→	NP VP	1.0
VP	→	Vt NP	0.4
VP	→	VP PP	0.6
NP	→	DT NN	0.8
NP	→	NP PP	0.2
PP	→	IN NP	1.0

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





Example

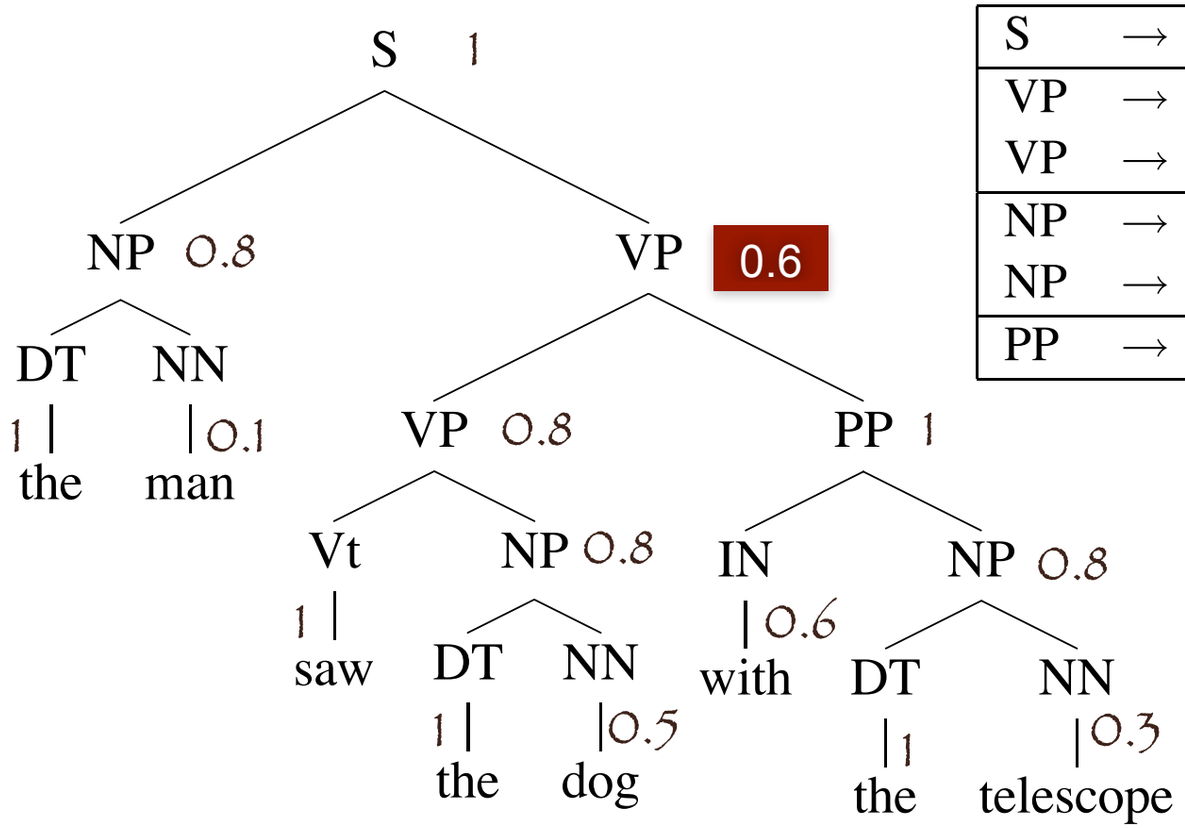


S	→	NP VP	1.0
VP	→	Vt NP	0.4
VP	→	VP PP	0.6
NP	→	DT NN	0.8
NP	→	NP PP	0.2
PP	→	IN NP	1.0

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

$$P(T,S) = 1\times0.8\times1\times0.1\times0.4\times1\times0.2\times0.8\times1\times1\times0.5\times0.6\times0.8\times1\times0.3 = 0.00036864$$

Example



S	→	NP	VP	1.0
VP	→	Vt	NP	0.4
VP	→	VP	PP	0.6
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

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

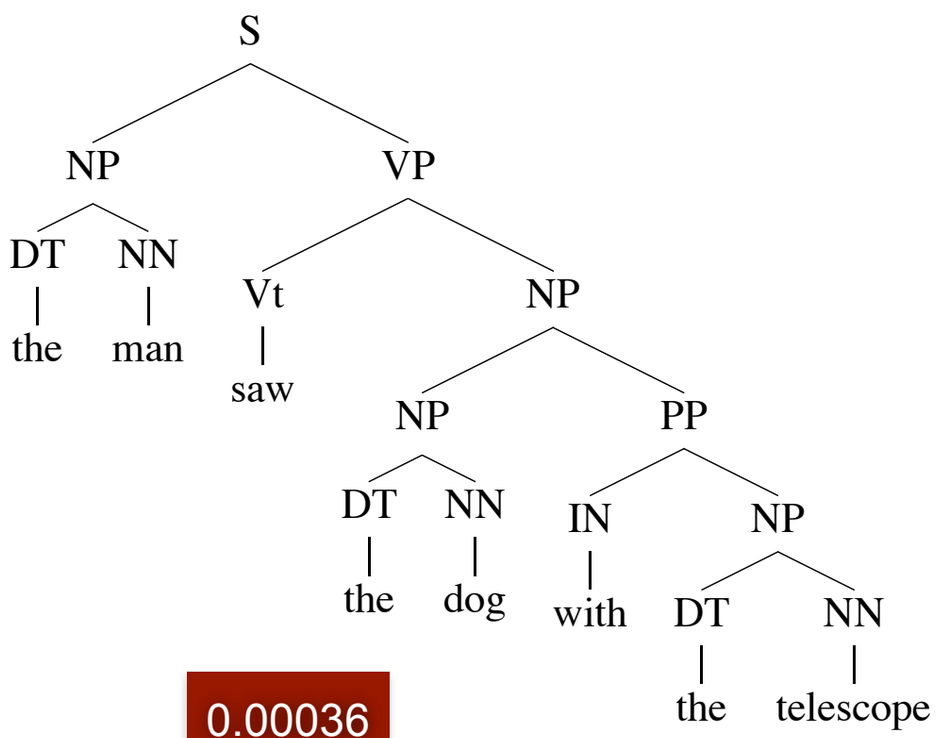
$$P(T,S) = 1\times0.8\times1\times0.1\times0.6\times0.8\times1\times0.8\times1\times0.5\times1\times0.6\times0.8\times1\times0.3 = 0.002211$$

Example

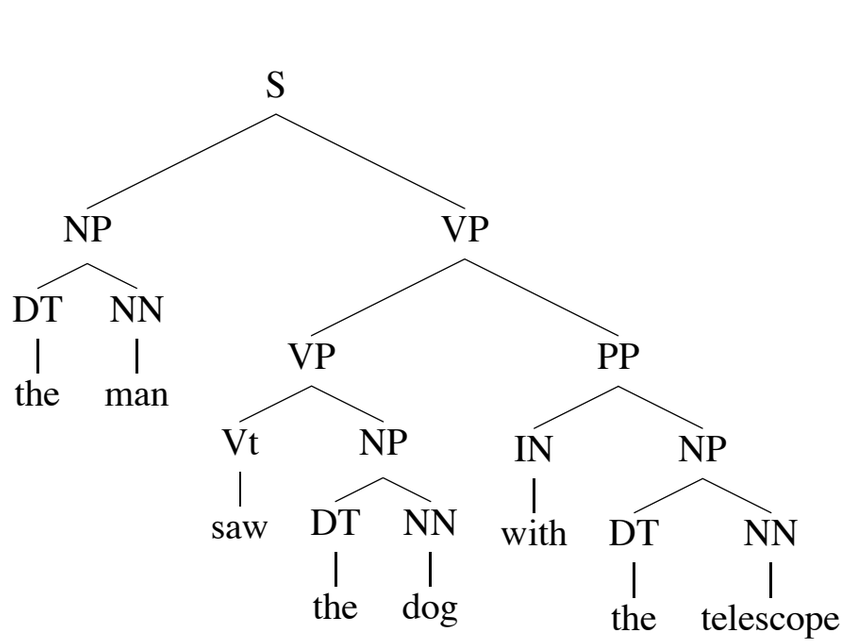
- Consider the two parses of **“the man saw the dog with the telescope”.** Compute the probability of each parse tree.

S	→	NP VP	1.0
VP	→	Vt NP	0.4
VP	→	VP PP	0.6
NP	→	DT NN	0.8
NP	→	NP PP	0.2
PP	→	IN NP	1.0

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



0.00036



0.0022

most probable

Some real statistics

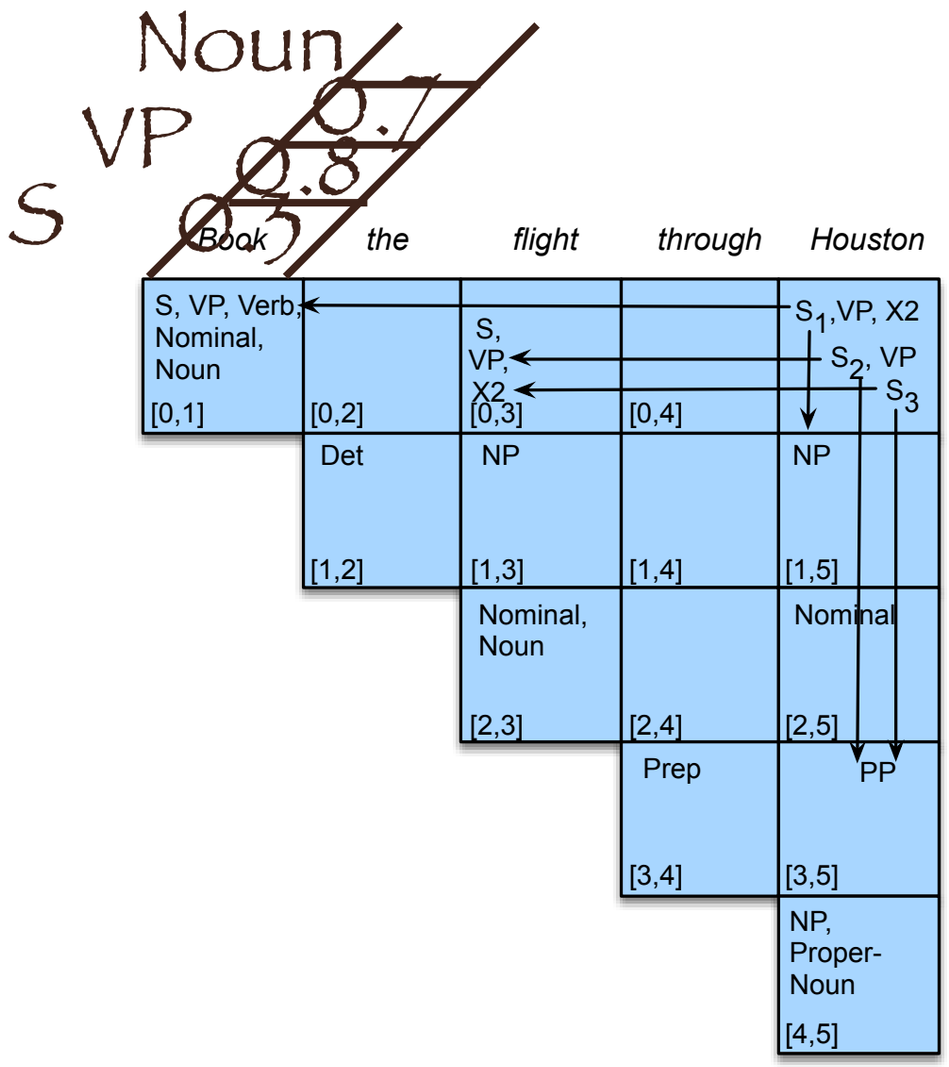
- Most frequent major categories of the Penn treebank, according to a CONLL shared task:

Label	Category	Proportion (%)	Example
NP	Noun Phrase	51	*The most frequently cancelled flight*
VP	Verb Phrase	20	*may not arrive*
PP	Prepositional Phrase	20	*to Houston*
ADVP	Adverbial Phrase	4	*earlier*
SBAR	Subordinate Clause	2	*that*
ADJP	Adjective Phrase	2	*late*

Probabilistic CKY

- Assign a 3rd dimension to each cell of matrix.
- This dimension corresponds to each non-terminal that can be put in the original cell.
- The value of the 3rd dimension cells contain a probability of that non-terminal.
- So we have an *(n+1)x(n+1)x k* matrix.
- A cell *[i,j,A]* therein is the probability of the constituent *A* that is between positions *i* to *j* of the input.

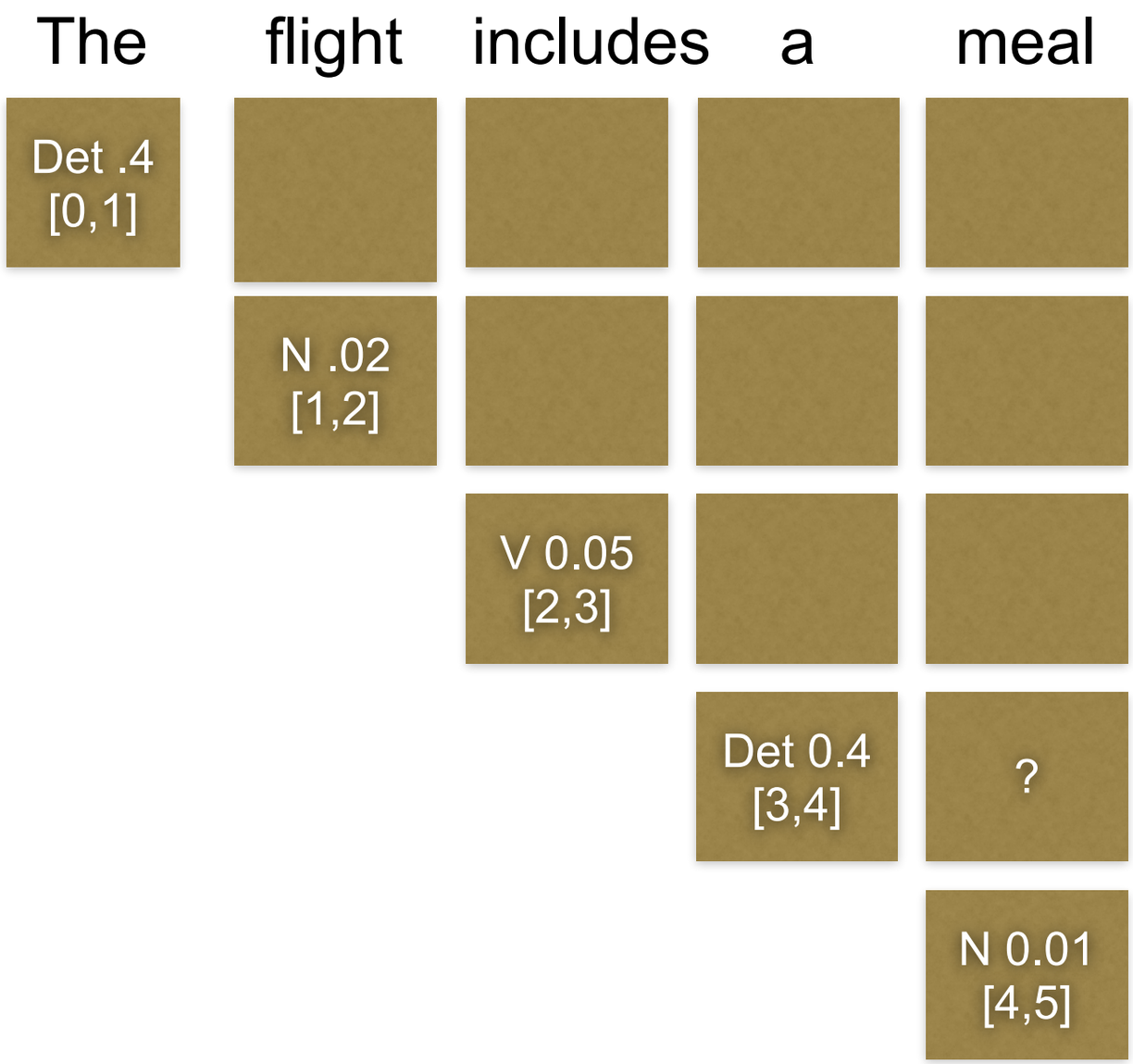
Probabilistic CKY



PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02

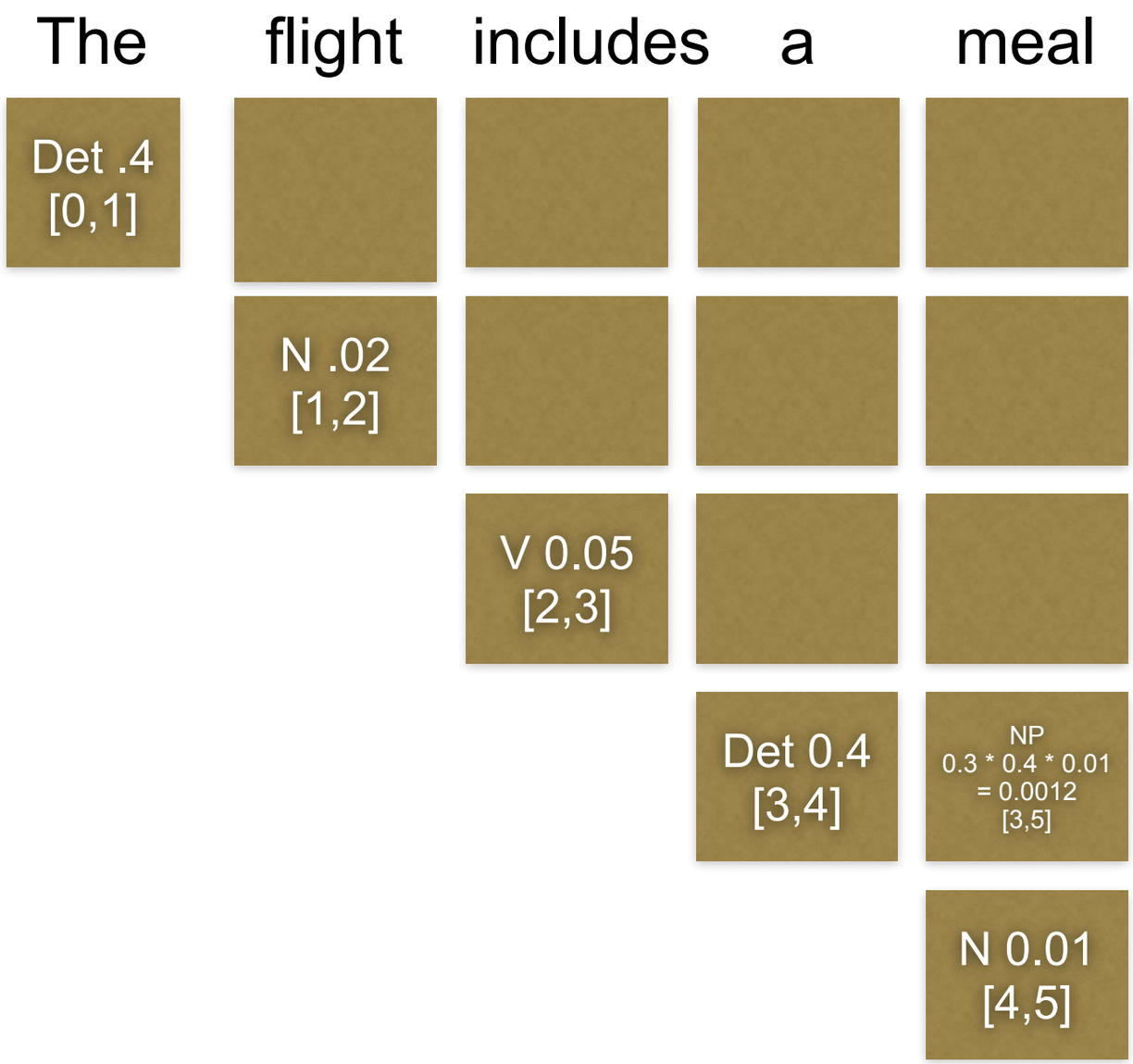


PCKY parse.

PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02



PCKY parse.

PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02

The	flight	includes	a	meal
Det .4 [0,1]	NP .3*.4 * .02= .0024 [0,2]			
	N .02 [1,2]	- [2,2]		
		V 0.05 [2,3]	- [2,4]	
			Det 0.4 [3,4]	NP 0.3 * 0.4 * 0.01 = 0.0012 [3,5]
				N 0.01 [4,5]

PCKY parse.

PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02

The	flight	includes	a	meal
Det .4 [0,1]	NP .3*.4 * .02= .0024 [0,2]	- [0,3]		
	N .02 [1,2]	- [2,2]	- [1,4]	
		V 0.05 [2,3]	- [2,4]	VP 0.2 *0.05 * 0.0012 = 0.000012 [2,5]
			Det 0.4 [3,4]	NP 0.3 * 0.4 * 0.01 = 0.0012 [3,5]
				N 0.01 [4,5]

PCKY parse.

PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02

The	flight	includes	a	meal
Det .4 [0,1]	NP .3*.4 * .02= .0024 [0,2]	- [0,3]	- [0,4]	
N .02 [1,2]	- [2,2]	- [1,4]	-	
	V 0.05 [2,3]	- [2,4]	VP 0.2 *0.05 * 0.0012 = 0.000012 [2,5]	
		Det 0.4 [3,4]	NP 0.3 * 0.4 * 0.01 = 0.0012 [3,5]	
			N 0.01 [4,5]	

PCKY parse.

PCFG

Example

S -> NP VP	0.8	Det -> the	0.4
NP -> Det N	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V -> includes	0.05	N -> flight	0.02

The	flight	includes	a	meal
Det .4 [0,1]	NP .3*.4 * .02= .0024 [0,2]	- [0,3]	- [0,4]	S 0.8 * .000012* .0024 =.00000002304 [0,5]
N .02 [1,2]	- [2,2]	- [1,4]	-	
	V 0.05 [2,3]	- [2,4]	VP 0.2 *0.05 * 0.0012 = 0.000012 [2,5]	
		Det 0.4 [3,4]	NP 0.3 * 0.4 * 0.01 = 0.0012 [3,5]	
			N 0.01 [4,5]	

PCKY parse.

Evaluating Constituency Parsers

Parse Eval:

• A metric that measures how much the constituents in the hypothesis parse tree look like the constituents in a gold standard parse tree.

• Gold standard: hand annotated corpora, like the Penn treebank.

$$\text{Recall} = \frac{\text{No. correct constituents in the hypothesis parse of s}}{\text{No. constituents in the gold standard parse of s}}$$

$$\text{Precision} = \frac{\text{No. correct constituents in the hypothesis parse of s}}{\text{No. of total constituents in the hypothesis parse of s}}$$

• Normal target measure: **F-Score**

Problems with PCFGs

Problem 1- Independence Assumption

PCFG's assume rules are applied independent of each other: "**the probability of a group of independent events is their multiplication**."

Not correct, e.g.:NP's that are sbj's are far more likely to be pronouns.

	pronoun	non pronoun
Subject	91%	9%
Object	34%	66%

Solution:

Conditionalise the probabilities on whether the non-terminal is in subject or object position, e.g.:

NP_sbj -> Pronoun [0.91] NP_obj -> Pronoun [0.34]
Splitting Non-Terminals

Problems with PCFGs

Problem 2- Lack of sensitivity to lexical facts

In English it is more likely that PP's attach themselves to NP's.
This will parse the following correctly:

"**fishermen caught tons of herring**",

PP "**of herring**" modifying NP "**tons**", via the rules:
VP->V NP, NP -> NP PP

But not the following:
"**workers dump sacks into bins**"

PP "**into bins**" modifying VP "**dump**", where the rule is:
VP->V NP PP

Solution: lexicalized grammars:

$$VP(dumped) \rightarrow VBD(dumped)\ NP(sacks)\ PP(into)$$

Treebanks and Grammar Extraction

- For automatically learning a grammar from <sentence,tree> pairs, we build a Treebank with **manual annotation**:
    - Use linguists to create trees by hand for each sentence.
- or, using **semi-automatic/assisted methods:**
    - 1- Use an automatic parser (based on manual rules or trained on other sources)
    - 2- Use linguists to hand-correct the parser
    - (still mainly a manual effort by human annotators)
- Example: Penn Treebank
- Produced from: Brown, ATIS, Wall Street Journal: English
- Other languages such as Arabic and Chinese

Treebanks and Grammar Extraction

- A treebank is a corpus whose **every sentence is paired with its parse tree.**
- The Penn treebank is a large treebank consisting of the corpora: WallStreet Journal, Brown, ATIS, Switchboard
- 1 million terminals, 1 million non-terminals, 17,500 rules
- Amongst these are 4,500 rules to expand VPs

Treebanks and Grammar Extraction

- Given the trees of a treebank, one can extract the rules used to generate the trees. This is called grammar extraction.

S -> NP VP .
NP -> DT JJ , JJ NN
VP -> VBD ADJP
ADJP -> JJ PP
…

DT -> that
JJ -> cold | empty
, -> ,
NN -> sky | fire
CC -> and
. -> .

…

Treebanks and Grammar Extraction

VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
VP → VB ADVP PP
VP → VB PP ADVP
VP → ADVP VB PP
VP → VBP PP PP PP PP PP ADVP PP

$VB \rightarrow arrive \mid have \mid wait$
$VBD \rightarrow was \mid said$

ADVP: adverbial phrase, e.g.
when: at noon,
where: in the corner,
how: in silence

This can lead to ad hoc rule:

The last rule is added to parse the following sentence:

This mostly happens because we *go from football in the fall to lifting in the winter to football again in the spring*.

What about PCFG's?
How to learn the probabilities of rules

1- Use a treebank:

- The probability of each expansion of a non-terminal is obtained by counting the number of times that expansion occurs in the corpus, then normalising it:

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)}$$

Learning the probabilities

2- (**For semi-supervised learning**) Parse your non-annotated corpus.

- For non-ambiguous sentences: increment a counter for every rule in the parse, then normalise.
- For ambiguous sentences: work incrementally
  - assign an equal prob to rules of the parser.
  - parse a sentence and compute a probability for it, use these to weight the counts and re-estimate the probabilities, until convergence.

Alternatives to constituency parsing: Logical Grammars and Dependency parsing

- In 1992 Pentus showed that the expressive power of the **logical grammar Lambek's calculus** is the same as Chomsky's context free grammars.
    - He also presented a translation between the formalism of generative grammars and that of logical grammars.
    - The two grammatical paradigms were finally united!
- **Dependency parsing** moved away from the notion of constituent and focused on syntactic dependencies, overcoming the limitation of CFGs, but with tractable algorithms.

Summary

- Formal grammars define a mechanism for generating all strings of a formal language which approximate natural languages.
- The notion of constituency is key to understanding them.
- CFGs are common grammars which define context-free languages, but these don't cover all of a natural language like English.
- We can parse automatically with a CFG through the CKY algorithm.
- We can add simple probabilities to the rule applications to get probabilistic CFG/CKY parsing.
- **<u>This week you will be drawing trees in the lab (unassessed).</u>**

Reading

- Jurafsky and Martin (3rd Ed) Chapters 12 (up to 12.5), 13 (up to 13.2) and Appendix C